THE BELL SYSTEM
TECHNICAL JOURNAL

VOLUME XXXVIII JULY 1959 NUMBER 4

Research Model for Time-Separation Integrated Communication

By H. E. VAUGHAN

(Manuscript received March 11, 1959)

A new communication system concept which is an important step toward an all-digital telecommunication plant is discussed. A research model, called ESSEX (Experimental Solid State Exchange), which combines remote line concentration, time-separation switching and PCM transmission is introduced to demonstrate the concept. The model, which uses solid state devices, works at the speed of a full-size system.

I. INTRODUCTION

A communication system requires channels for transmission of information and switching arrangements to interconnect the channels. At present, the transmission problem and switching problems are handled separately. Transmission channels may be divided into three classes: space separation, frequency separation and time separation. All have been in use for some time. Switching arrangements may be divided into the same three classes.[1] The space-separation class includes all telephone switching systems in use today. Frequency separation systems have been studied and are not economically feasible at this time. Exploratory switching systems in the time-separation class are being considered in this country and abroad.

This paper reviews the use of time-separation techniques for transmission systems and for switching systems, points out that the availability of solid state devices has revived interest in the subject and indicates that techniques using these devices are now feasible for both systems. It shows that an integrated communication system using these techniques is much more attractive than a combination of subsystems using time separation and presents a research model which demonstrates the technical feasibility of such an integrated system

The research model is primarily a digital system. In it, information-producing terminals — in our specific case, telephone subscribers' sets — in a small group are connected over voice-frequency cable pairs to a small switching unit remote from the central switching point. This unit connects them to a time-separation or multiplex channel group and converts the signals to digital form. The digital signals are transmitted and switched at various locations and are reconverted to original analog form at another small remote switching unit, which serves another group of terminals, or at the originating switching unit.

A laboratory model of this system has been built. It is known as ESSEX (Experimental Solid State Exchange) and, as its name implies, is built of solid state devices.

The following sections discuss the background for the experiment, outline the general plan, describe the laboratory model and give some results.

II. BACKGROUND

The common control type of switching system has two basic sections: (a) the switching network for interconnecting channels and (b) the common control section. Many proposals and experiments have been made which use electronics for the common control section.[2,3] They offer many advantages over the slower electromechanical control systems. Electronic components are at least one thousand times as fast as the mechanical ones now in use. Common control systems made of such devices[2,3] time-share the circuits, and thus can carry out their work with fewer devices. One control unit plus one spare can handle a very large switching system, and it can be made so that it is sufficiently flexible to handle new services.

The development of an electronic common control system is substantially complete. This system could be modified for use with a time-separation system of the type discussed herein; therefore, it will not be treated in this paper. The switching network is another story.

Existing and most proposed networks are in the space-separation class. Large numbers of switches are required to implement them. Substitution of electronic switches for electromechanical switches may afford indirect saving in the control and reduce the cost of the switches, but it does not reduce the number of switches. Space-separation networks require many switches or crosspoints per line. Time-separation switching networks, in which one physical path carries many conversations by time sharing, require something in the order of two switches per line. In addition, they require a few bits of memory per line to remember which switch is operated at what time interval. The switches, although fewer in number, must operate much faster than those used in a space-separation network. Present solid state switches are sufficiently fast that speed is no problem. Such a network can now be built. And a time-separation switching network is an important part of the research model.

The switching system mentioned above is only part of a communication system. It may represent about one-half of the cost. The other portion of the cost is for the transmission channels. In a telephone system, this is primarily copper cost, the cost of the cable pairs between subscribers and central office, and between offices. One way to reduce the amount of cable is to locate part of the central office in small pieces near to groups of subscribers.[4] These remote pieces of the switching network are called line concentrators. They provide switching so that a group of subscribers may share the use of cable conductors between the remote unit and the central switching point. The number of cable pairs required between the remote unit and the central point may be reduced by about 80 per cent. Thus, a reorganization and dispersion of part of the switching network can reduce the amount of cable required. Line concentration is another important part of the research model.

Additional saving of cable conductors can be achieved by the use of either frequency-separation or time-separation techniques for interoffice trunks.[5] Cost savings depend on the lengths of the cable runs and the cost of the terminal equipment for multiplexing. The advent of solid state devices is affording new opportunities for reducing the channel length needed to prove in multiplexed channels in place of individual space separated channels. One such method is time sharing of the cable conductors through the use of PCM (Pulse Code Modulation) transmission. This is another important part of the research model.

The parts mentioned above could be all considered and used in a communication system such as is shown in Fig. 1. In such a system voice-frequency signals would be time-switched at the concentrator and then changed back to voice frequency. For PCM transmission they would be converted to digital signals and then back again to voice at the central switching point; then again time-switched and changed back to voice, etc. This process is quite involved and, in fact, unnecessary. It can be simplified by removing all the transitions between time separation and space separation except those at the ends of the system, thus producing

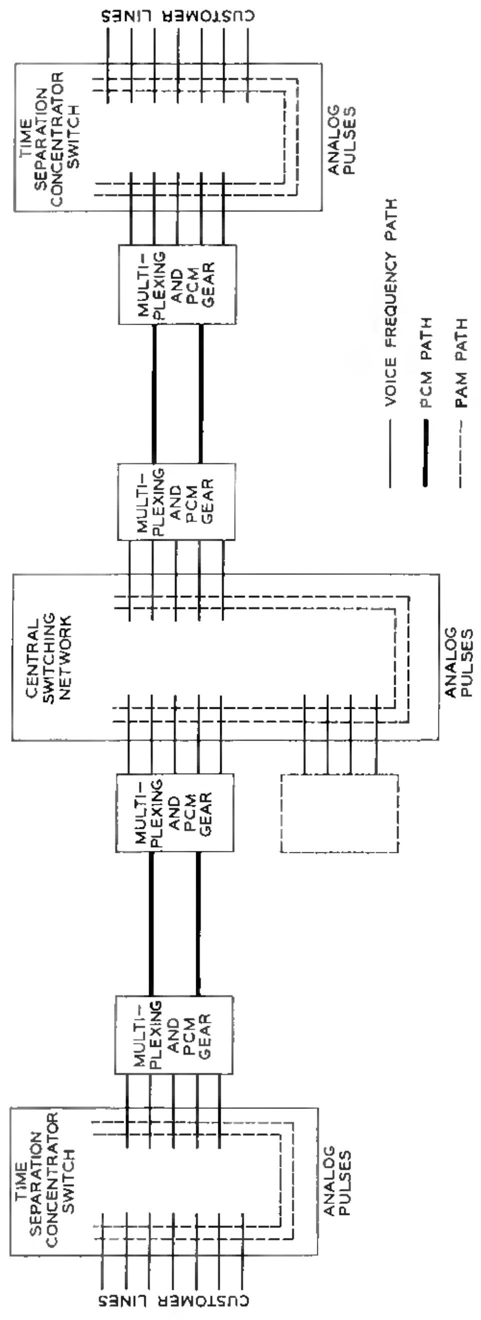

CUSTOMER LINES
TIME SEPARATION CONCENTRATOR SWITCH
ANALOG PULSES
MULTI-PLEXING AND PCM GEAR
MULTI-PLEXING AND PCM GEAR
CENTRAL SWITCHING NETWORK
ANALOG PULSES
MULTI-PLEXING AND PCM GEAR
MULTI-PLEXING AND PCM GEAR
TIME SEPARATION CONCENTRATOR SWITCH
ANALOG PULSES
CUSTOMER LINES
VOICE FREQUENCY PATH
PCM PATH
PAM PATH

Fig. 1 — Combination of time-separation switching and transmission systems.

a more economical system than that achieved by just building up subsystems. Such simplification leads to the system shown in Fig. 2—the ESSEX System. Voice-frequency signals from a group of subscriber lines are switched at a remote concentrator unit and converted to digital signals. The digital signals are transmitted, switched at one or more central switching points and handled as digital signals until they leave the system at another concentrator or *trunkor*, which is a converter for connecting to voice-frequency trunks.

ESSEX employs all the parts mentioned above and combines them in a manner which minimizes the amount of equipment and cable conductors required and provides a uniform-quality fixed-loss path between any two voice-frequency terminals, independent of the distance and the number of switching points between them.

The quality is fixed by the characteristics of the line filters, the amplitude range of the system and the noise. The voice is coded in digital form, and the pulses are retimed and reshaped at regular intervals. Thus, in the ideal case, the code at the distant end will be the same as that at the transmitting end independent of distance. It follows that length of the transmission path has no effect on the loss and quality, except as increased length increases the probability of errors in transmission due to interference from external sources and to timing irregularities. ESSEX provides analog-to-digital conversion as near to the subscriber as feasible and then operates as a digital system. In addition, the switches at the central switching point are simplified, since they handle digital signals only.

Time-separation techniques for transmission systems and for switching systems have been under investigation for many years. The potential advantages of an integrated digital communication system and increased demands for various speeds of digital transmission have spurred the efforts on the present research model. The use of solid state devices makes the system attractive. Such devices require less space and less power, and are fast enough to do the job. As the speed of the devices improves, more operations may be performed with the same number of devices or the same operations may be handled by fewer devices, and the picture will become even more attractive.

III. BASIC PLAN

The basic plan has a number of remote line concentrators, using time-separation switching and transmission, connected at a central point in a time- and space-separation switching network. Trunkors, which act as connecting and converting units for trunks, registers, etc., are similarly

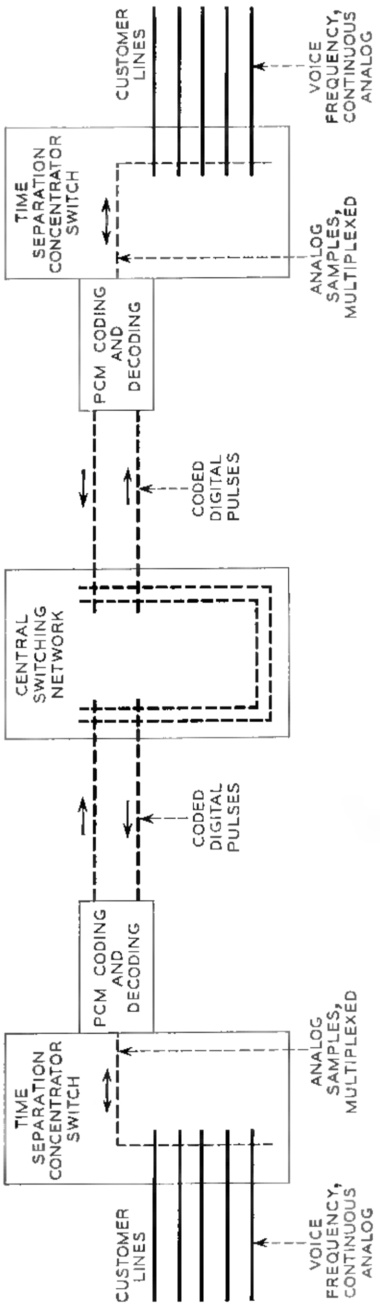

CUSTOMER LINES
VOICE FREQUENCY, CONTINUOUS ANALOG
TIME SEPARATION CONCENTRATOR SWITCH
ANALOG SAMPLES, MULTIPLEXED
PCM CODING AND DECODING
CODED DIGITAL PULSES
CENTRAL SWITCHING NETWORK
CODED DIGITAL PULSES
PCM CODING AND DECODING
ANALOG SAMPLES, MULTIPLEXED
TIME SEPARATION CONCENTRATOR SWITCH
CUSTOMER LINES
VOICE FREQUENCY, CONTINUOUS ANALOG

Fig. 2 — Integrated time-separation communication system.

connected. Each concentrator has some switching and control circuits located at the central switching point. The switches connect the four-wire transmission circuits from the concentrators to other concentrators, to trunkors or to routes to other offices. The control circuits have a memory which holds the information used to control both remote and central switches for the duration of a call. These circuits also control line scanning for supervision and handle programming for setting up and releasing calls. Each concentrator control unit is connected to the common control for the office.

Before going further into the system, it will be well to mention the sampling and transmission action. It is a well-known fact that, if a short time sample of a signal limited in frequency to $x/2$ cps is taken x times per second, transmitted and filtered, then any signal components in the band up to $x/2$ cps will be reproduced at the output of the filter.[6] In ESSEX, the sampling rate has been set at 8000 per second. Thus, the period between one sample of a message and the next sample is 125 microseconds, which is called a *frame*. The number of channels that can be inserted in a frame depends on the length of the sample and the guard space between samples. In ESSEX, each channel uses a time slot of 5.2 microseconds, so 24 channels are handled in a frame. Twenty-three channels are used for speech, and the 24th is used for supervisory functions. Each time slot has eight pulse positions, seven for the coded PCM signal and one for other functions. Thus, the pulse repetition rate on the four-wire digital transmission line is 1.536 mc. The four-wire system will use ordinary exchange cable pairs. Pulse regenerative repeaters are required every 6000 feet for transmitting pulses at the 1.536-mc rate in the case of 22-gauge paired cable. Closer repeater spacing may be required if the noise is greater than that now anticipated.

3.1 *Remote Concentrator*

A concentrator module consists of a *remote* unit and a *central* unit, as shown in Fig. 3. Let us consider the *remote* part of the line concentrator, shown on the left side of the figure, which is the starting point in the system. A maximum of 255 voice-frequency lines appear as inputs, and three cable pairs carry digital signals to and from the central unit. One pair, designated S, the send pair, takes PCM signals to the central unit. The second, the receive pair, R, brings PCM signals from the central unit. And the third, the control pair C, brings control words from the memory in the central unit. Each line requires a line circuit, which contains a gate and a filter. The line circuit, a plug-in package, is the lowest-order module in the system. These modules are added only as customers are connected to the concentrator. The ensemble of line-circuit packages makes up a two-wire bilateral switching stage controlled by a selector. The output of this stage is a two-wire time-separation PAM bus with 23 time slots or channels for use as links to the central office. A time diagram which may be helpful in visualizing some of the operation is shown on Fig. 4. The memory which controls the selection of the gates is located at the central point. The information from it is sent over the control pair C as eight-bit words in each time slot. These words pass through the selector to control the line gates. Each word designates a line gate number (LGN) and can select one of 255 gates.

The PAM two-wire bus must be converted to a four-wire bus so that the signals can be handled on a PCM basis. This conversion is accomplished by a circuit called a *time-division hybrid*. In brief, it permits a signal to pass from a line to the send bus or from a receive bus to a line, but *never* permits a direct connection from send to receive. PAM signals on the send bus are coded into seven-bit PCM signals and sent to the central point. Incoming PCM signals are decoded and presented as PAM signals to the two-wire bus and then to the voice-frequency line. Note that the line circuit is a passive circuit and that all the signal power needed is supplied by the common receiving amplifier in the receive bus. Timing signals necessary for the operation of this remote unit are generated by a local clock which is slaved to a master clock at the central switching point. Since both switching control and timing control signals originate at the central switching point, the remote unit is actually a

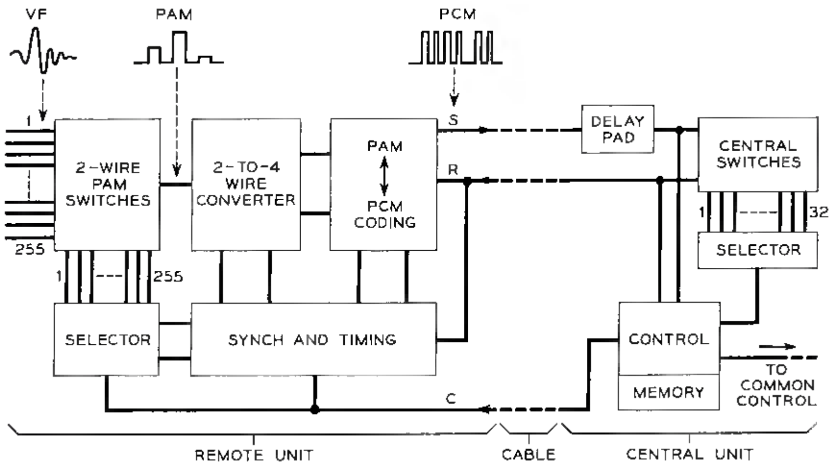


Fig. 3 — Line concentrator, showing both remote and central functions.

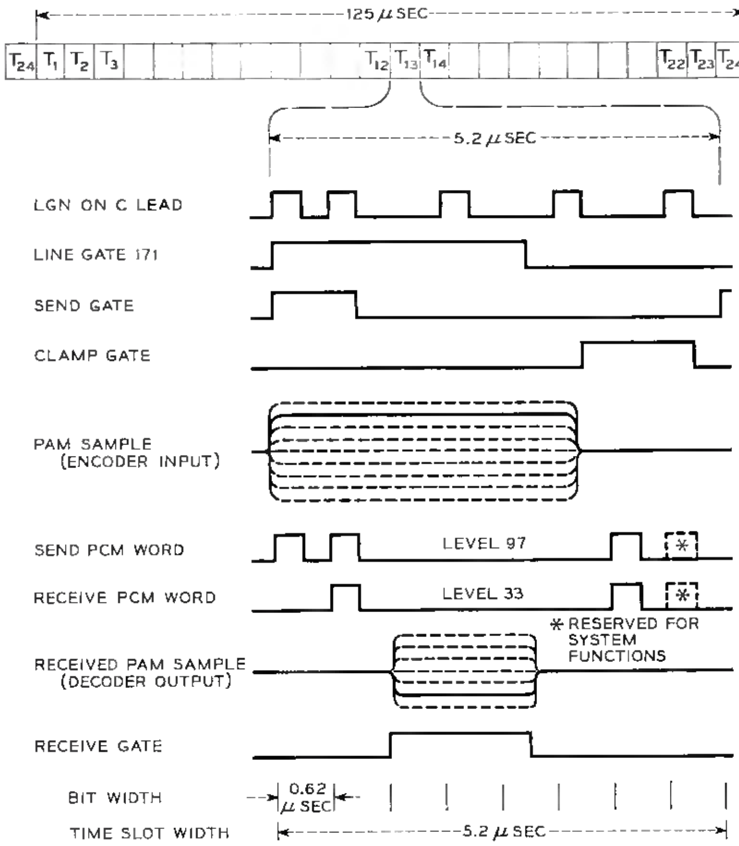


Fig. 4 — Timing diagram for remote concentrator.

slave. The problems of timing and synchronizing both remote and central units will be discussed in Section 3.4.

3.2 *Concentrator Central Unit*

The *central* unit of the concentrator module shown on the right side of Fig. 3 is made up of digital circuitry that includes the memory to control both remote and central switches, the central switches with their selector and a concentrator control unit. In addition, there is a delay pad and servo, which is discussed in Section 3.4. As mentioned above, the memory stores information which controls the operation of line gates. It also stores words to control the operation of the central switches associated with a particular concentrator unit and call progress information concerning the states of the calls being handled. For example, "on-hook" and "off-hook" conditions are recorded in this memory. The memory stores 24 bits of information for each time slot or channel, eight bits for the line gate, five bits for the central switch, eight bits for call progress marks, and three bits for checking. Each 24-bit word is read out every 125 microseconds; thus, the complete memory can be searched in this period to determine which channels are busy or idle or for any other pertinent information.

The central stage switches or junctor gates are simple digital AND gates which switch digital signals unilaterally. The switches handle low power, and thus the selector, which uses a five-bit input to mark one pair of 32 pairs of switches, is rather simple. The central switches for each concentrator are connected to the central switches of all other concentrators by junctors on a space-separation basis as in Fig. 5. Thus, each concentrator has access to all other concentrators, trunkors and junctors to other office modules over 32 separate paths in any of 23 time slots. A call from one concentrator to another must use the same time slot in each concentrator. The switching plan is really a four-stage network, one stage at each remote unit and one for each central con-

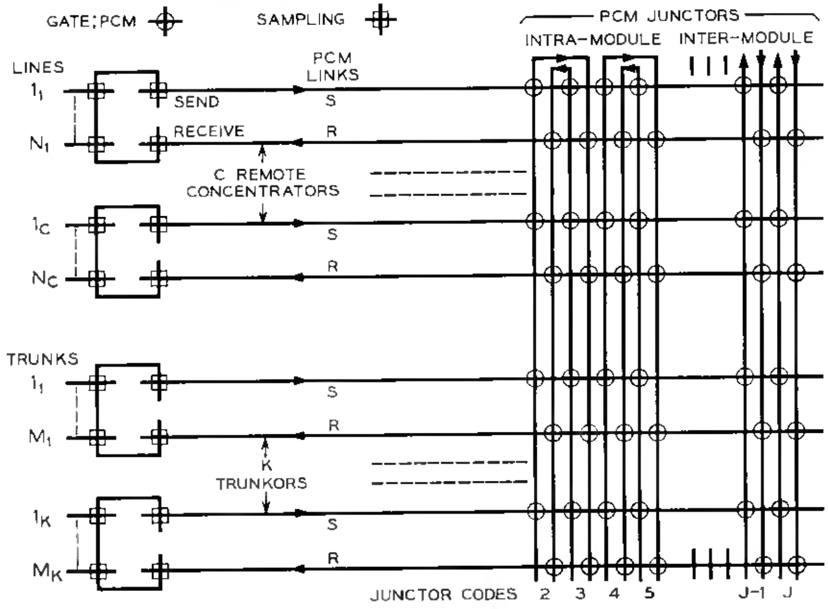


Fig. 5 — Switching plan.

centrator unit. Some blocking occurs due to the concentration to 23 channels and some due to time slot mismatch. Although a remote unit can handle up to 255 lines, about 115 lines, each submitting one tenth of an erlang, would load the 23 channels to 50 per cent of capacity. In the future, new services may use lines with much lower traffic, and then the large number of lines may be useful. If the call is to another module, the same number of stages are used, since only one switch is made in each central unit. The only difference is in the length of the junctor. Consider a call from a concentrator to a trunkor — a dial tone connection — and assume for the moment that there is no delay in the system. Information in the memory for concentrator A opens a gate in time slot 6 and opens the send and receive junctor gate pair 3 in the same time slot. Information in memory for the trunkor Z opens a gate in time slot 6 in the trunkor and send and receive junctor gate pair 3. Information then passes directly from A_6 to Z_6 and from Z_6 to A_6, and the operation is repeated 8000 times per second.

An important section of the central concentrator unit is the concentrator control, which works in cooperation with the common control. The division of responsibility between concentrator control units and common control is a field for further investigation, since the present division is based on judgment with limited knowledge of the problem. A detailed treatment of the organization and operation of the concentrator control will be given in a future paper. A simplified diagram of the control section is shown on Fig. 6. The most complex part is the logic which controls the generation, interpretation and modification of call progress marks. Some of these marks are operating orders to and from the common control. Supervisory information from the remote unit is held in the memory, and logical operations are performed on this information when necessary. Control of ringing, supervisory tones and answer indications are also handled in this section. Line scanning also is controlled here.

3.3 *Supervision, Dialing and Ringing*

Many auxiliary functions must be performed in order to use this transmission and switching system as a telephone system. Detection of "on-hook" and "off-hook" line conditions to determine the subscriber's wishes is done by scanning. The central control sends out a line designation in the 24th time slot, which is reserved for this purpose. This eight-bit word controls a transistor in the line package through the selector used to control the line gate. A combination of the pulse from the selector and current flowing in the subscriber's loop ("off-hook" condition) produces a pulse on a lead common to all such transistors. Thus, scanning requires little additional equipment in a time separation system. If the line is "off-hook", a "1" is returned to central in the eighth-bit position of the 23rd time slot on the *S* lead. Every fourth frame, a new number is sent to the remote unit, so that 255 lines are scanned in about one-eighth second. The result of the scan is stored in the call progress mark section of the memory if action is called for.

Switching networks using electronic crosspoints have a limited power-handling capacity; thus, it is necessary to use low-level tones. Ringing is done by sending tones in the voice band to actuate a tone ringer in the subset.[7] Ringing tone in the form of PCM signals is applied through a separate gate for each concentrator *R* lead, and audible ring in the same PCM form is applied through a separate gate for return to the originating end of the circuit (see Fig. 7). This arrangement permits full access on a time-separation basis to all 23 channels. It can be shown that this helps to reduce blocking. Busy tones or other tones in PCM form may be switched in the same way, and trunk splitting also can be

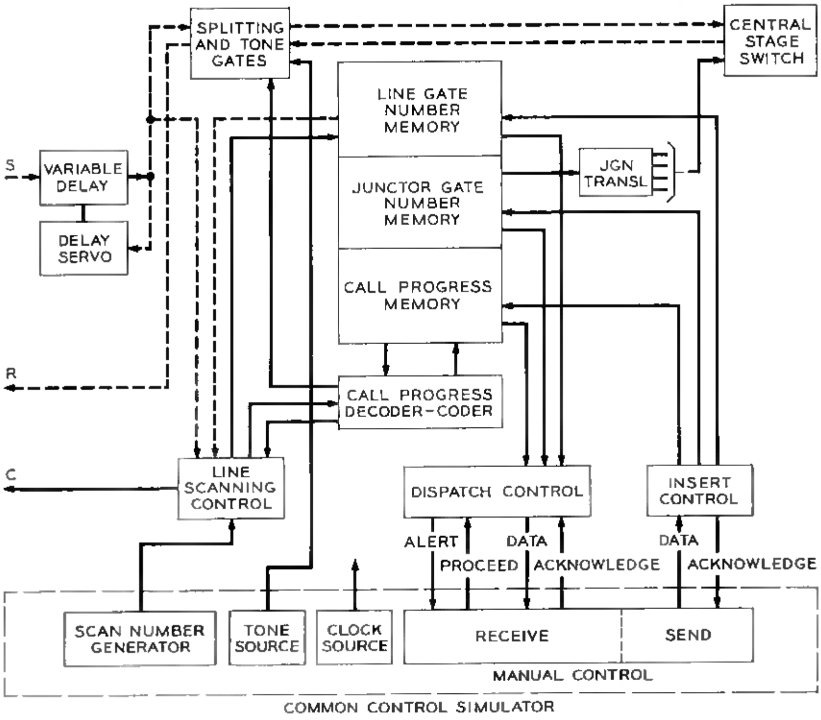


Fig. 6 — Line concentrator—block diagram of central unit.

taken care of in this fashion. This simple means for applying special tones is one more advantage of digital time-division switching.

Dialing signals are "in-band." Frequency-shift dialing with one frequency for "make" and another for "break" is used, and a form of multifrequency or pulse-coded signals may be employed. Registers connected to trunkors will be used to interpret the digits which will be assembled in the memory of the main common control. This is so similar to dialing methods in other electronic switching systems that no further detail will be given here.

3.4 *Delay, Synchronization and Timing*

The synchronization of a point-to-point four-wire PCM transmission system is straightforward. A clock times the sending end, and the receiving end is slaved to it. The same operation is used in the opposite direction. Synchronization between the two directions is not required. In the ESSEX system, which uses two-wire switching at the remote terminals operating under a central common control, over-all synchronization is necessary. It is a problem; unless all switches operate at the proper time, chaos will reign. The transmission delay, about 7 microseconds per mile for cable pairs, further complicates the problem. This problem was analyzed by Karnaugh, who has offered several solu-

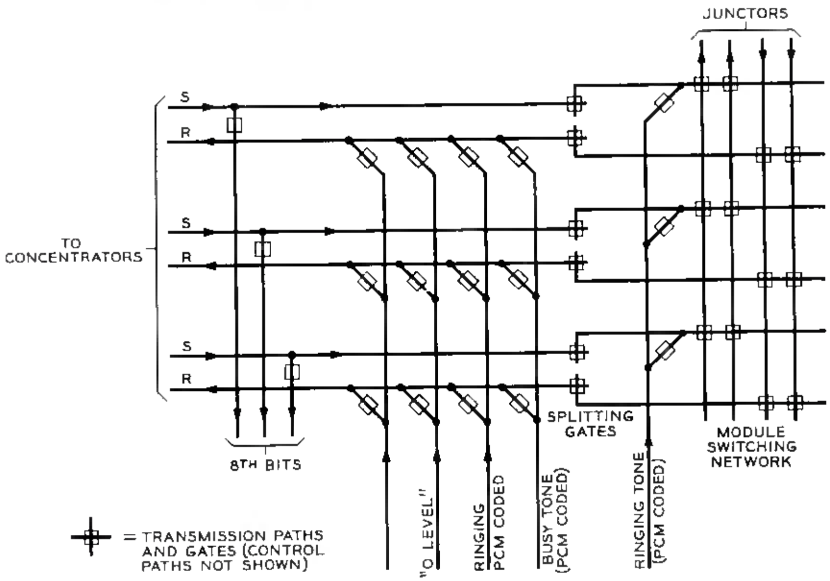


Fig. 7 — Switching of special tones in digital form.

tions.[8] One of these, the use of a delay pad, was adopted for use in the laboratory model.

Consider the complete concentrator shown in Fig. 3, which illustrates the delay-pad solution to the problem. Assume that control words are stored in the memory and that PCM words appear on the junctor pair, going in both directions at time τ. Just before τ, an eight-bit word is sent out to the remote unit and the central switch. The central switch closes at τ and permits the seven-bit word to go out over the R pair to the remote unit, so that it arrives there at $\tau + \alpha$, where α is the transmission delay. The eight-bit word on the C pair controls the line gate, so that it opens at $\tau + \alpha$ and the decoded sample that arrived on the R pair passes to the line. Now, in the opposite direction, a sample from the line passes to the coder and then to the S bus with some delay, d. The PCM representation of the sample arrives at the central switch, with an additional delay, α, at the time

$$t = \tau + \alpha + d + \alpha.$$

Every 125 microseconds after τ, the five-bit word again closes the central switch, and, if $\tau + 2\alpha + d = 125$ microseconds, the PCM signal would arrive just in time to pass through the junctor gate. However, α is dependent on the distance between the remote and central units, so the condition above is not satisfied. But it can be satisfied if a variable delay pad, x, is included in the send line, so that

$$\tau + 2\alpha + d + x = n(125 \text{ microseconds}).$$

A variable-length delay line provides the necessary delay x. Each concentrator, trunkor and intermodule junctor must be padded with such a delay to provide proper operation. Since the transmission time, α, varies slightly with temperature, a servo unit on the delay line automatically compensates for these small changes in α.

The clock at the remote unit is a crystal-controlled unit which is slaved to the master clock at the office module by monitoring the pulses on the C pair. Counting circuits are used to produce timing pulses at submultiples of the clock frequency. Once every 125 microseconds, a framing signal is sent in the 24th time slot on the R pair to the remote unit. When this signal is recognized, all counting circuits are checked, and, if out of frame, they are reframed.

3.5 *Modules*

The term "module" has been used in some of the preceding sections. It denotes a building block whose cost or complexity is significantly greater than the cost or complexity of connecting it to the system. ESSEX has a hierarchy of modules. The smallest one is the line package used to switch voice-frequency lines selectively to a group of PAM time-separated channels. It is a plug-in piece of hardware added at the time a line is put into service.

The next larger module is the concentrator, including both remote and central units. It is the basic multifunction unit in the model. The central office is built up of concentrators and trunkor modules and a common control unit. The whole switching and transmission array is made by running wires between such units. The proposal is to install sockets with junctor wires between them and to have the office grow by plugging switching units into these sockets.

The system as outlined so far permits the operation of only one switch in a time slot at the concentrator. For heavy intraconcentrator traffic, it is desirable to operate two switches simultaneously so that only one time slot is required. In this case, speech is switched directly in PAM form and does not pass to the central point. One way to operate two switches in one time slot is to provide extra line memory in the concentrator control, an extra control pair and an additional selector at the remote unit to control the second switch in a time slot. Such an arrangement could handle a maximum of 23 simultaneous calls between 46 customers in one remote concentrator. A concentrator module with these additions could then serve as a community dial office (CDO) or as a PBX with centralized control.

The largest module would be called the *modular center*. It would be made up of several concentrators and trunkors and use about 30 junctor pairs to serve between 2500 and 4000 lines, depending on the amount and characteristics of the traffic. Such a modular center could be located to minimize cable plant. Growth in an area could be handled by adding these central modules. For instance, a 10,000-line office would be made up of four modules, as shown in Fig. 8. These units might be interconnected by four-wire PCM junctors equipped with delay pads. With the present plan, each office module would have its own common control unit. Communication between the common control units in different modules would use the eighth-bit position of each time slot. A 192 kilopulse per second channel is proposed for this purpose. Office modules, distributed over an area, might use a single common office code, the same office code for each one until 10,000 numbers are used. This would help to conserve office codes.

The use of small central modules is only one way to handle central switching. Many valid arguments can be advanced to show that it is wasteful of common control equipment. There are many plans to have one common control serve several office modules, but these are beyond the scope of this paper.

3.6 *Use for Other Services*

Digital data signals in the voice-frequency band may be handled through line circuits just as they are now handled. High-speed baseband signals could be applied to the PCM channels through switches at the output of the encoder, and incoming data pulses may be taken out through switches ahead of the decoder. Since each channel handles eight bits in a time slot, one channel will handle 64,000 bits per second. If higher data rates are needed, more channels can be allocated for this purpose. The complete group could handle 23 × 64,000 bits per second.

Broader-band analog channels may be made available by changing the line package filter and by changing the sampling rate. If the address of a particular line terminal were stored in position n in the memory and again in position $(n + 12)$ on a modulo 24 basis, the sampling rate for the line would then be 2 × 8000, or 16,000 times per second. If it were stored in positions n, $(n + 6)$, $(n + 12)$ and $(n + 18)$ on a modulo 24 basis, the sampling rate would then be 32,000 times per second. Thus, by using several time slots in an ordered sequence for one line, the sampling rate may be increased, and a wider band may be provided. This is another type of flexibility.

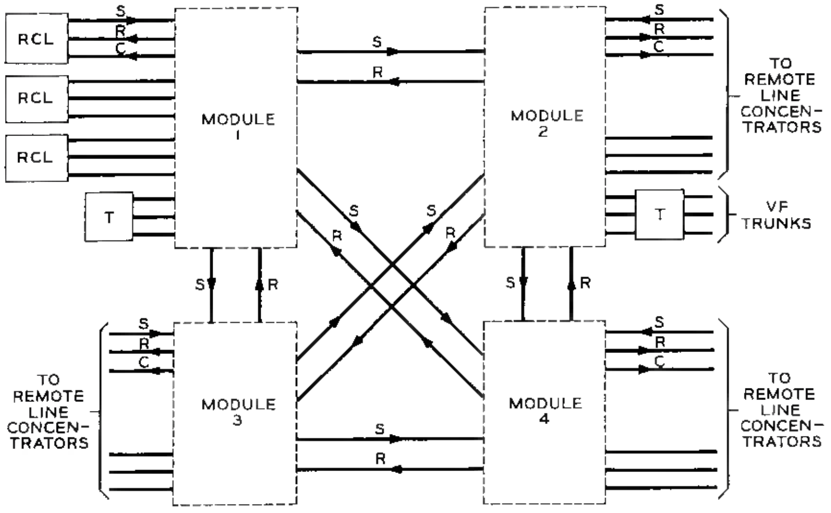


Fig. 8 — Interconnection of central switching modules.

IV. THE LABORATORY MODEL

The prime objective of the experiment is to demonstrate the technical feasibility of the system. Such research system experiments increase understanding of the operation of the system, unearth new problems caused by interaction of complex circuits which have been demonstrated individually, and provide the stimulation to invent new circuits and techniques to solve these problems.

Most research systems are highly skeletonized. This one is skeletonized only in the number of concentrators and trunkors. Two complete concentrators and one complete trunkor, along with a central clock, make up the model. Although each of these modules is capable of handling the maximum number of lines or trunks, the number of operating-line packages is limited to 12 per unit. However, a plugboard arrangement is provided, so that each package may be associated with any terminal on the selector.

Before discussing the model, it is appropriate to describe how it grew to its present state.

Initially, two partially equipped remote concentrators with a central memory were connected together, with no delay between them. Each unit contained a two-wire PAM selective switching stage, a time-division hybrid or two-wire to four-wire converter, and synchronizing and timing circuits. Tests were made on this phase of the system until the PCM equipment became available. In the next phase, the transmission path was opened, and PCM coding and decoding equipment complete with compressors and expandors were installed. These units introduced some delay, so the delay pads were added. The latest phase builds the system up to include two complete concentrators, a complete trunkor and an operator console to simulate many of the functions of the common control. This gradual evolution of the model has made it possible for one phase of the evolution to provide most of the environment for testing the circuits added in the next phase. It is a case where serial construction has saved a lot of work by reducing the amount of equipment needed to synthesize input-output gear that would have been needed to proceed in parallel on several parts at once.

4.1 *Layout of Model*

A layout of the laboratory model is shown in Fig. 9. The two racks at the left are a remote concentrator unit. The first rack contains the line selector, which takes the incoming serial eight bits of a word from the *C* lead, assembles them in parallel, selects a line gate and applies a sampling pulse to it. A group of 12-line gates is in the upper section of the rack, which also houses the plugboard that permits the interconnec-

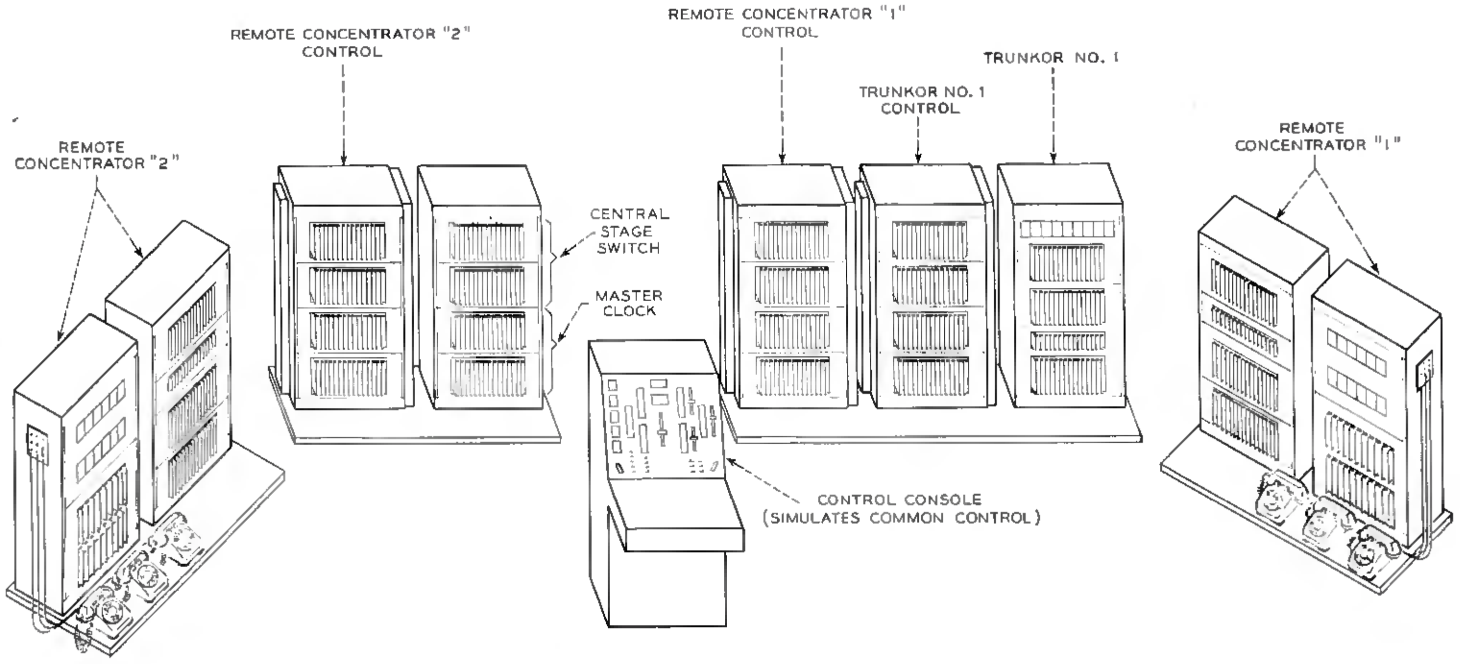


Fig. 9 — Layout of laboratory system.

tion of any package to any one of 255 selector terminals. The outputs of the gates are 23 time-separated channels on a two-wire PAM bus. This two-wire bus is connected to the second cabinet, which contains the time-division hybrid, encoders, decoders, synchronizing and timing circuits and other circuits common to the remote unit. The outputs from the second rack are three exchange-area type 22-gauge cable pairs, which handle digital signals at a 1.536-mc pulse rate. These cable pairs require a pulse regenerative repeater for each 6000 feet.

A second remote unit is located in the two racks at the right side of the figure.

The remote units are the only places where analog signals are handled. Since each of these units represents a small part of an office, the crosstalk problem is simplified, because the exposure to other circuits is minimized. This is an important feature in the organization of "single-wire-to-ground" switching systems.

The six racks in the center are all part of the office module. The third and fifth racks from the left are the control units for the two concentrators. Each one contains the delay pads, the three memory units, control circuits and logic circuits for a remote unit.

The sixth rack is the trunkor control unit, which is much the same as a concentrator control unit. The trunkor unit which converts from PCM to PAM and selectively switches voice frequency terminals for trunks, registers, etc., is located in the seventh rack.

The fourth rack has room for the printed circuit cards for a central stage switch serving 30 concentrators or trunkors. Each card holds the central switches and selector for one concentrator. It is presently equipped with only three switch cards, one for each of the two concentrators and one for the trunkor.

The arrangement for the junctor wiring is shown in Fig. 10. The upper half of this rack represents the *complete* central switching network for a 4000-line office module. The concentrator controls are connected by plugs, with only 420 wires being required to connect all 30 units, exclusive of power and clock pulses. The small size of this network demonstrates a major advantage afforded by PCM switching. Since the circuitry at the central module is all digital, the crosstalk in the wiring presents no formidable problem. The lower half of the fourth rack holds the master clock and timing circuits for the office module. Care must be taken in distributing timing pulses to the various control units to assure that timing pulses arrive at each one at the same time, plus or minus 10 millimicroseconds. This section also contains circuits to generate the supervisory control tones. These tones are switched, when needed, to the concentrator R lead under control of call progress marks in its memory.

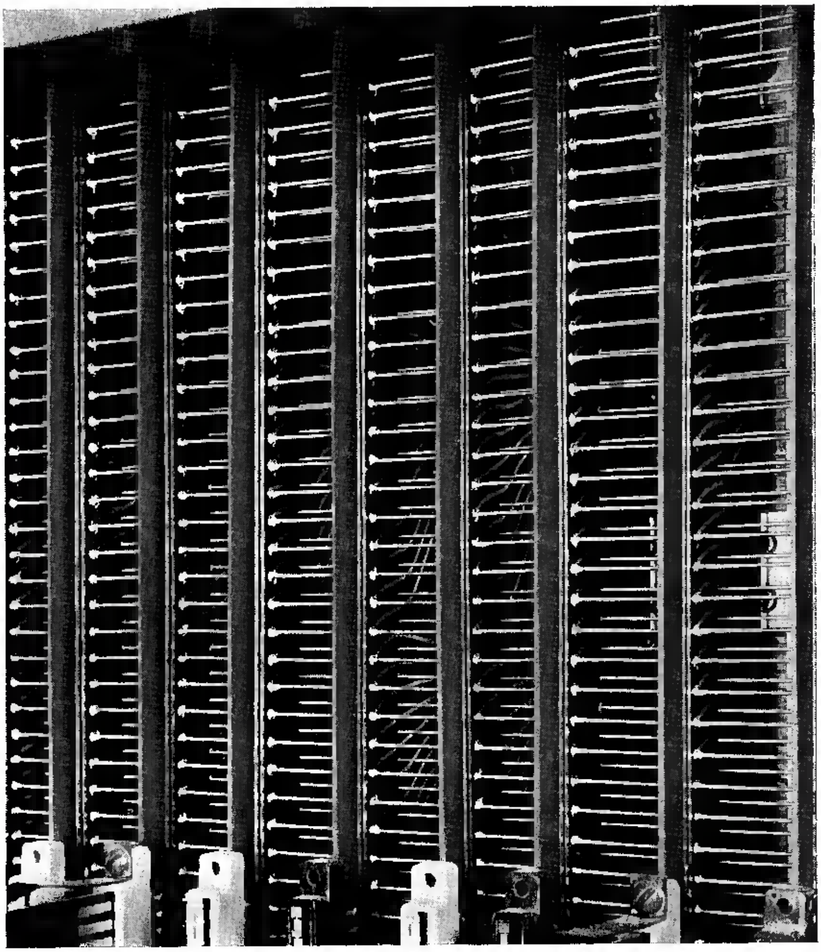


Fig. 10 — Junctor wiring at rear of connectors for central stage switches.

A control console takes the place of the office common control. It provides a visual display of calling number, called number, time slot number, etc. An operator manually performs operations on the console to interpret instructions from the concentrator controls and to issue orders to them for setting up and taking down calls. The console is located in front of the group of racks and is used for testing and demonstrating the system.

Fig. 11 is a photograph of a portion of the laboratory system. Power supplies for the system, not shown in the layout, are mounted in racks similar to the others.

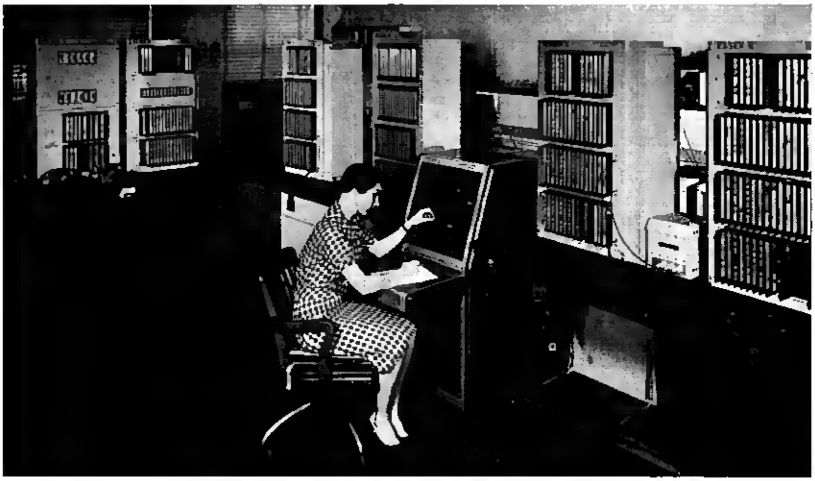


Fig. 11 — Section of the laboratory system.

4.2 *Circuitry*

Most of the circuitry is isochronous. Timing is furnished by a two-phase 1.536-mc clock. Rise and fall times in the order of 50 millimicroseconds are common.

The circuits in the model use commercially available transistors and diodes in conventional arrays . High-speed operation is achieved by use of clipping and clamping techniques, with a collector supply voltage much higher than the normal signal voltages. Since the emphasis is on the system, the circuits are not necessarily minimal. They are assembled on 5×8-inch wiring boards, which plug into sockets for convenience in testing and replacement. The boards or packages contain groups of basic building blocks, such as diode logic units, flip-flops, shift register stages, pulse amplifiers and blocking oscillators. The laboratory model uses about 4000 transistors and 12,000 diodes.

Most of the circuits use voltages in the range of -12 to $+12$ volts, with some collector supplies being as high as 25 volts. The total power drain for a remote concentrator is about 75 watts, with an additional $\frac{2}{3}$ watt for each operating subscriber set. A concentrator control unit requires about 50 watts.

Many unifunction circuits that probably arouse curiosity have been mentioned previously. The description of these would surely drag this paper too far into detail. It is planned to treat these details in two additional papers. However, it is unfair to leave the reader completely up in the air, so a few words are in order about some of these circuits.

The subject of the line gates has been investigated for some time, and

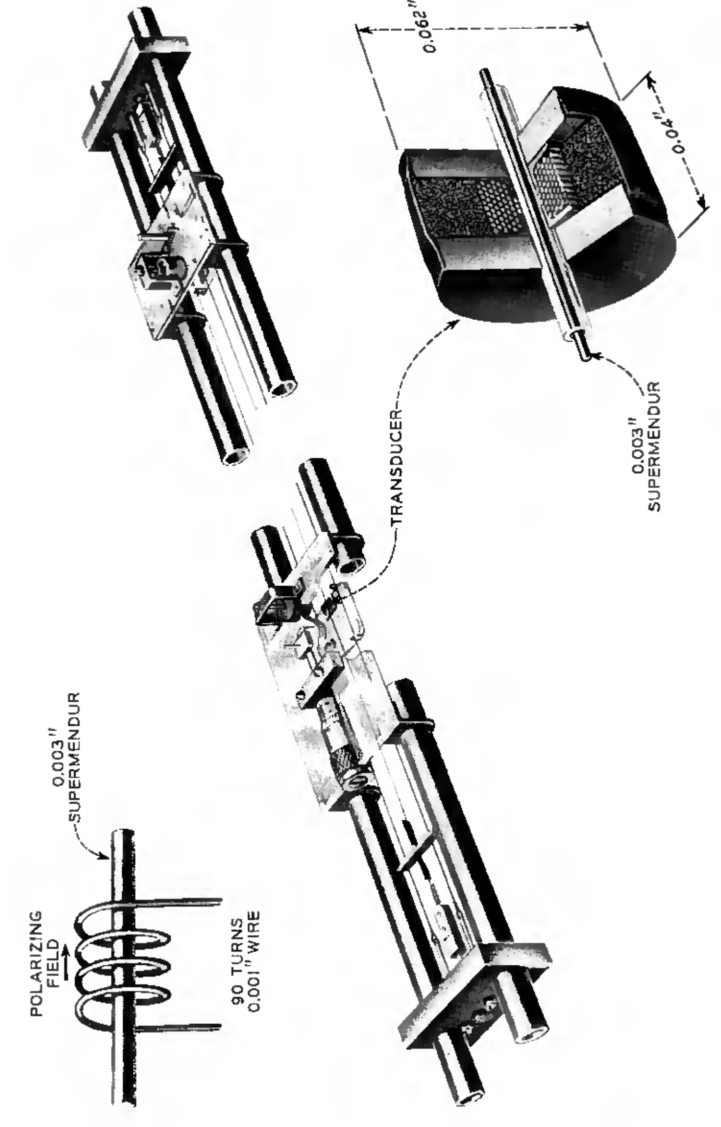


Fig. 12 — Magnetostrictive delay line.

has been published.[9] The time-division hybrid, a result of this experiment, is a circuit which permits a sample from a line gate to be passed to the "send" bus, held, stretched for coding, and then removed by a clamp so that there is no inter-time-slot crosstalk. Just after that sample passes through the common "send" gate and that gate is blocked, a common "receive" gate opens and passes an incoming PAM signal to the subscriber's line filter through his line gate, which is still open. This PAM signal is then dissipated by the subset which terminates the line filter. Approximately 123 microseconds later, the "send" gating operation is repeated. It is this time difference which provides the hybrid action, by preventing the receive signal from passing directly to the "send" bus.

The delay pads and the memories use magnetostrictive delay lines with transistor drivers and amplifiers. A typical line is shown in Fig. 12. The line, a 3-mil supermendur wire, is mounted so that the delay may be set manually any place in the range from a few microseconds to 125 microseconds. The servo unit for temperature correction is used only with the delay pads. It provides an automatic adjustment of ± 1.5 microseconds.

These lines have a wider bandwidth than those in common use. The pulses applied are baseband, and the pulse rate is 1.536 mc. The total loss in the two transducers and the line is about 50 db. The drive circuit uses two transistors, and the receiving amplifier uses four transistors, and the line with this associated circuitry is a delay unit with zero loss.

4.3 *Performance*

The transmission characteristic of a channel between two voice-frequency terminals, exclusive of the subscriber loops, shows a loss of 6 db $\pm$ 0.5 db from 100 to 3200 cps and 3 db additional loss at 75 and 3500 cps. A channel will handle up to 6 milliwatts of sine-wave power. The signal-to-noise ratio is about 30 db. As mentioned above, the characteristics are independent of the length of line between conversion points and of the number of switching points.

Two research models of remote concentrators without controllers have been operating satisfactorily for more than six months. This part of the model uses about 1800 transistors and more than 5000 diodes. The performance of the components has exceeded all expectations for a research model. The model, as outlined, complete with controllers, trunkor and control console, has been operating for two months with equally satisfactory results.

Facilities are available for making listening tests to compare straight-through wire connections with the PCM connection, and only a few people have been able to detect a difference between these conditions. The quantizing noise on signals seems to be unnoticeable. The low-level noise resulting from the indecision of the coders during silent periods seems to be more bothersome than the quantizing noise on the higher signals.

V. CONCLUSIONS

A communication system concept has been described which uses time-separation techniques for transmission channels and for switching. It is primarily a digital system. In principle, the use of digital transmission with regenerative repeaters would provide fixed low loss and fixed quality connections between remote line concentrators. It would provide flexibility for new services. It might be arranged in a modular manner to handle growth, and to facilitate manufacture and installation. A full-size office of this type would require less floor space than existing electromechanical systems. A laboratory model using solid state devices throughout has been built and tested. It demonstrates the technical feasibility of the concept and gives an indication of the number of components that might be needed for such a system.

VI. ACKNOWLEDGMENTS

The success of this experiment is due to the ingenuity and continuing efforts of D. B. James, J. D. Johannesen, M. Karnaugh, W. A. Malthaner, J. F. Müller, J. P. Runyon and many of their associates in the Systems Research Department of Bell Telephone Laboratories.

Basic designs of the coding, decoding and companding equipment were supplied by H. M. Straube, C. P. Villars and their associates in the Transmission Systems Development Department.

REFERENCES

1. Joel, A. E., Electronics in Telephone Switching Systems, B.S.T.J., **35**, September 1956, p. 91.
2. Malthaner, W. A. and Vaughan, H. E., An Automatic Telephone System Employing Magnetic Drum Memory, Proc. I.R.E., **41**, October 1953, p. 1341.
3. Ketchledge, R. W., An Introduction to the Bell System's First Electronic Switching Office, Proc. Eastern Joint Computer Conf., December 1957.
4. Joel, A. E., An Experimental Remote Controlled Line Concentrator, B.S.T.J., **35**, March 1956, p. 249.
5. Sumner, E. E., private communication.
6. Oliver, B. M., Pierce, J. R., and Shannon, C. E., Philosophy of PCM, Proc. I.R.E., **36**, November 1948, p. 1324.
7. Meacham, L. A., Power, J. R. and West, F., Tone Ringing and Pushbutton Calling, B.S.T.J., **37**, March 1958, p. 339.
8. Karnaugh, M., private communication.
9. James, D. B., Johannesen, J. D. and Myers, P. B., A Two-Transistor Gate for Time-Division Switching, I.R.E.-A.I.E.E. Transistor and Solid State Circuits Conf., February 1958.

Variable-Length Binary Encodings

By E. N. GILBERT and E. F. MOORE

(Manuscript received September 9, 1958)

This paper gives a theoretical treatment of several properties which describe certain variable-length binary encodings of the sort which could be used for the storage or transmission of information. Some of these, such as the prefix and finite delay properties, deal with the time delay with which circuits can be built to decipher the encodings. The self-synchronizing property deals with the ability of the deciphering circuits to get in phase automatically with the enciphering circuits. Exhaustive encodings have the property that all possible sequences of binary digits can occur as messages. Alphabetical-order encodings are those for which the alphabetical order of the letters is preserved as the numerical order of the binary codes, and would be of possible value for sorting of data or consultation of files or dictionaries. Various theorems are proved about the relationships between these properties, and also about their relationship to the average number of binary digits used to encode each letter of the original message.

I. INTRODUCTION

Table I gives three different encodings for representing the letters of the alphabet and the space symbol in binary form. These encodings have several special properties which are of some interest. First, each is a variable-length encoding; that is, the code for each letter is a sequence of binary digits, but the codes assigned to different letters are not all required to consist of the same number of binary digits. The first two of these encodings have the prefix property; that is, no one of the codes is a prefix of any other code of the same encoding. This property makes it easy to decipher a message, since it is only necessary to look at enough binary digits of the message until it agrees with one of the codes if it is desired to find the first letter of the deciphered message.

The first of these encodings, called the Huffman encoding, is constructed by the method given by Huffman,[1] and has the property of being a minimum-redundancy encoding; that is, among all variable-length binary encodings having the prefix property, this is an encoding

TABLE I

Letter	Probability	Huffman Code	Alphabetical Code	Special Code
Space	0.1859	000	00	00
A	0.0642	0100	0100	0100
B	0.0127	011111	010100	010100
C	0.0218	11111	010101	010101
D	0.0317	01011	01011	01011
E	0.1031	101	0110	0110
F	0.0208	001100	011100	011100
G	0.0152	011101	011101	011101
H	0.0467	1110	01111	01111
I	0.0575	1000	1000	1000
J	0.0008	0111001110	1001000	10001111111
K	0.0049	01110010	1001001	100100
L	0.0321	01010	100101	100101
M	0.0198	001101	10011	10011
N	0.0574	1001	1010	1010
O	0.0632	0110	1011	1011
P	0.0152	011110	110000	11000
Q	0.0008	0111001101	110001	110001111111
R	0.0484	1101	11001	11001
S	0.0514	1100	1101	1101
T	0.0796	0010	1110	1110
U	0.0228	11110	111100	111100
V	0.0083	0111000	111101	111101
W	0.0175	001110	111110	111110
X	0.0013	0111001100	1111110	1111101111111
Y	0.0164	001111	11111110	1111110
Z	0.0005	0111001111	11111111	11111101111111
Cost		4.1195	4.1978	

having the lowest possible cost (where the *cost* is defined as the average number of binary digits used per letter of the original message, assuming that the message is made up of letters independently chosen, each with the probability given).

The second of these encodings, called the alphabetical encoding, has the property that the alphabetical order of the letters corresponds to the numerical binary order of the codes. Among all such alphabetical-order-preserving binary encodings that are of variable length and have the prefix property, the one given has been constructed to have the lowest possible cost. It can be seen that the cost 4.1978 of the alphabetical encoding is quite close to the cost 4.1195 of the Huffman encoding, as compared to the cost 5 of the more conventional fixed-length encoding for the same alphabet, so that the alphabetical restriction adds surprisingly little expense to a variable-length encoding.

Part of this paper deals with the methods of constructing such best alphabetical encodings, and gives some theorems concerning their cost and their structure. However, this paper also includes theoretical results about various properties of variable-length binary encodings in general. The cost, the prefix property and unique decipherability have already been mentioned. The exhaustive property (roughly speaking, this permits all infinite binary sequences to occur as encoded messages) is also shown to be relevant, as is the finite delay property, which has to do with the amount of delay which must take place between receiving and deciphering the enciphered message. Various theorems are proved concerning the relationships of these properties to each other and to other properties. Some of these properties have also been considered by other authors.[1,2,3,4,5,6]

One property of special interest is the ability of certain variable-length encodings (but not of fixed-length encodings) to automatically synchronize the deciphering circuit with the enciphering circuit. This self-synchronizing property, while it has been previously mentioned,[5] is a little-known property which might have practical significance in that it would permit binary deciphering machines using variable-length encodings to be built without requiring any special synchronizing circuits or synchronizing pulses, such as are needed for fixed-length encodings. Thus, there may be cases where (despite some present opinions to the contrary) variable-length encodings lend themselves to simpler instrumentation than fixed-length encodings.

Since the probabilities given in Table I are derived from one of the tables of frequencies of letters in English text,[7] the encoding given should be reasonably efficient for encoding English words or phrases. The alphabetical property, together with the prefix property, implies that two such words or phrases could be compared for alphabetical order merely by putting the two entire phrases into a simple comparison circuit of the kind which would be used to compare binary numbers. If the two phrases begin with the same sequence of letters, the corresponding parts of their enciphered form would agree, and the outcome of the binary comparison would be determined by the comparison between the two binary codes corresponding to the first pair of letters which disagree.

Placing the space symbol before the letter A of the alphabet corresponds to the usual convention governing the filing of multiple-word entries in alphabetical order, although if it were desired also to include punctuation marks or numerals in the alphabet, the conventions are not so universal, and might not be of the sort which can easily be expressed in a binary encoding.

An alphabetical encoding might be used as a means of saving memory space needed for names or other alphabetical data that are to be sorted into alphabetical order on a data-processing machine or are to be stored in a file in alphabetical order. Similarly, it might be used for the words of a dictionary as a part of a language-translating machine, if it were desired to preserve the conventional alphabetical order of dictionaries. In addition to possible savings of memory space, it might be used to find entries in such a dictionary more quickly. Since the low redundancy of this encoding causes the digits 0 and 1 to be used with more nearly equal frequency and more nearly independently than in a fixed-length encoding, the binary numerical value associated with each word would increase more nearly as a linear function of distance progressed through the dictionary; hence, instead of searching for a given word by the method of successively halving the interval in which it is known to lie, linear interpolation (or some rough approximation to it which might be done by a simpler circuit) could be used to speed up convergence. However, for uses such as mentioned here, the particular alphabetical encoding given in Table I is not necessarily the optimum, since the frequencies of occurrence of letters in names or in dictionary entries are undoubtedly different than they are in connected English text. However, the methods given in this paper would enable such an encoding to be obtained for any given probability distribution.

II. TERMINOLOGY

We will use the word *letter* to refer to any symbol of some designated list, including even the space symbol of Table I. By an *alphabet* we will mean a set of letters. We will usually require each member of an alphabet to have associated with it a probability of occurrence, and we will also usually require that some linear ordering relationship (which we will call *alphabetical order*) be defined for the letters of this alphabet. So that we may call any subset of the letters of an alphabet a *subalphabet*, and may keep the same ordering and the same probabilities, we will require only that the sum of the probabilities be less than or equal to one. All of the alphabets considered in this paper have only a finite number n of letters, but it might be advisable to allow countably infinite alphabets in certain further theoretical extensions of this subject.

A *message* is a finite sequence of letters, or an infinite sequence $L_1L_2L_3 \cdots$ which extends infinitely only into the future, not into the past. We will consider a *source* which generates messages in which successive letters occur independently and with the given probabilities. However, in case the sum of the probabilities is less than one, we may imagine that the probabilities are proportionately increased just enough that their sum becomes one, so that the associated source is more realistic.

We distinguish between code and encoding, both of which are often called codes by other writers. A *code* is a finite sequence of binary digits. An *encoding* is a way of associating (or more formally, a function C which associates) a code C_i with each letter L_i of an alphabet.

The operation of *enciphering* (elsewhere often called encoding) constructs a sequence of binary digits which is made up of the code for the first letter of the message, followed immediately by the code for the second letter of the message, etc. Any message then produces a sequence of binary digits called the *enciphered message*. Any machine or circuit which does the operation of enciphering is called an *enciphering machine* or an *enciphering circuit*. The enciphered message of a finite message is obviously always finite.

An encoding will be said to be *uniquely decipherable* if, for each finite enciphered message, there exists exactly one original message which could have produced it. If an encoding is uniquely decipherable, then there is obviously a procedure for deciphering any finite enciphered message (by enumeration, for instance), and any machine or circuit capable of doing this will be called a *deciphering machine* or a *deciphering circuit*.

Following Huffman,[1] we define a *prefix* of any sequence Φ of binary digits to be any finite sequence which is either Φ itself or is obtainable by deleting all of the digits after a given point of Φ. For example, the prefixes of 10110 are 10110, 1011, 101, 10, 1, and the *null sequence*, which has no digits. We will say that an encoding C has the *prefix property* if no code of C is a prefix of any other code of C.

By a *presumed message* we will mean a finite or infinite sequence Φ of binary digits such that every prefix of Φ is a prefix of the enciphered form of some message. Then, at any given time while a presumed message is being sent into a deciphering machine, it is indistinguishable from a message, so it makes sense to allow presumed messages as well as messages to be the class of sequences which can be sent into a deciphering machine.

III. THE ENCODING THEOREM FOR ALPHABETICAL ENCODINGS

Consider a discrete source S which uses the alphabet: space, A, B, $\cdots$, Z (any other linearly ordered alphabet will also serve). An encoding of blocks of N letters into binary sequences will be called an *alphabetical encoding* if it is uniquely decipherable and the codes for the blocks in alphabetical (dictionary) order are themselves in numerical order. Here the codes are imagined to be prefixed by binary points to convert them into numbers in binary form. The alphabetical encoding of Table I is a case with $N = 1$. It is a natural question to ask if a restriction to alphabetical encodings may not be severe for some sources S. In particular, are the results of Shannon's encoding theorem (Ref. 8, Theorem 9) still obtainable with alphabetical encodings?

Shannon proved that the output of a discrete source having entropy H bits per character can be enciphered in a uniquely decipherable manner into a sequence of binary digits so that the average number of digits used per character exceeds H by an arbitrarily small amount. Shannon's construction encodes blocks of N source characters into binary sequences, using a cost (average number of binary digits per character) H_N' which satisfies

$$NG_N \leqq NH_N' \leqq NG_N + 1. \tag{1}$$

Here, NG_N is Shannon's notation for the information contained in a block of N characters produced by the source; i.e.,

$$NG_N = -\sum_i p_i \log p_i , \tag{2}$$

in which the p_i are the N-gram probabilities of the source. Then, since

$$\lim_{N\to\infty} G_N = H,$$

Shannon's theorem

$$\lim_{N\to\infty} H_N' = H \tag{3}$$

follows from (1). Since NG_N must be a lower bound on the average number of digits used to encode a block of N characters by any means whatever, (1) shows that Shannon's construction is not far from the best possible one for block encoding. We now give a similar theorem for alphabetical encoding.

Theorem 1: Let S be a source producing messages which may be ordered (alphabetically). Let G_N be computed from the N-gram probabilities p_i of S by (2). There exists a uniquely decipherable alphabetical encoding of blocks of N characters of S into sequences of binary digits for which the cost, H_N, satisfies

$$NG_N \leqq NH_N \leqq NG_N + 2. \tag{4}$$

By picking N large enough, H_N may be made arbitrarily close to the entropy H of S in bits per character.

Proof: The proof is adapted from Shannon's[8] proof of his Theorem 9. Let all possible blocks of N source characters be listed in alphabetical order, and let p_i denote the probability of the ith block in the list (recall that Shannon lists his blocks in order of probability rather than alphabetically). Let m_i be the integer for which

$$2^{-m_i} \leqq p_i < 2^{1-m_i}.$$

Also, let numbers $A_1, A_2, A_3, \cdots$ be defined by

$$\begin{aligned} A_1 &= \frac{p_1}{2}, \\ A_2 &= p_1 + \frac{p_2}{2}, \\ \vdots & \qquad \vdots \\ A_i &= (p_1 + \cdots + p_{i-1}) + \frac{p_i}{2}. \end{aligned} \tag{5}$$

Note that

$$0 \leqq A_1 \leqq A_2 \leqq \cdots \leqq 1.$$

We now construct an alphabetical encoding. The code for the ith block will be the first $m_i + 1$ digits of the binary expansion of the number A_i. In Shannon's encoding this same block has a code formed by expanding a (different) number to m_i places. Then our scheme uses only one more digit than does Shannon's for each block, $NH_N = NH'_N + 1$, and (4) follows from (1). It remains now to show that our encoding is uniquely decipherable; i.e., that the sequence of letters generated by S may be reconstructed from the binary digits.

It suffices to prove that our construction produces a list of codes which have the prefix property. Then the enciphered message produced by each block of N letters may be deciphered as soon as all its digits have been received.

To prove that our list has the prefix property, consider any two blocks of letters, say the ith and the jth with $i < j$. By (5),

$$A_j \geqq A_i + \frac{p_j}{2} + \frac{p_i}{2},$$

and

$$A_j \geqq A_i + 2^{-1-m_j} + 2^{-1-m_i}. \tag{6}$$

If $p_i \leqq p_j$, then $m_i \geqq m_j$; but, by (6), the jth code cannot be identically the same as the first $1 + m_j$ places of the ith code. Similarly, if $p_i \geqq p_j$ the ith code cannot be a prefix of the jth code. Thus, the prefix property, and the theorem, follow.

Except in the case of an alphabet having only one letter, the prefix property is sufficient to insure unique decipherability, but it is not necessary. For example, the list 0, 01, 11 does not have the prefix property; still it could be used. In a received message 00001111 ⋯ there would be no doubt about the first three 0's, and the fourth 0 would be recognized as 01 or not according to whether an odd or even number of 1's followed it.

However, by a *best alphabetical encoding* we will mean an encoding which has the lowest cost among all alphabetical encodings which have the prefix property. This insistence upon the prefix property will make it possible for us to prove Theorems 2 through 5 and give constructive methods for finding these best alphabetical encodings.

If we use the construction just described to design an alphabetical encoding of English with $N = 1$, we obtain a cost of 5.75 digits per character. As guaranteed by the theorem, this cost is less than $G_1 + 2 = 6.08$. However, we could have done better by simply assigning a five-digit code to each letter. The encoding can be much improved by

Table II

Letter	Code	Shortened Code
Space	0001	000
A	00110	001
B	01000001	010000
C	0100011	010001

deleting some digits which are obviously not needed. For example, the first few codes are those listed in Table II. Clearly the code 00110 for A is too long. As soon as the prefix 001 is received, A is the only possibility. The final digits 10 may be deleted. Similarly, the other codes may be shortened, as indicated in Table II, until no code can lose a final digit without becoming a prefix for some other code. The cost is thereby reduced to 4.44 digits per character.

A different encoding is obtainable using the same sort of construction but with

$$A_i = \sum_{j=1}^{i-1} 2^{-m_j} + 2^{-m_i-1}.$$

The same proof can be used, since (6) still holds. Since the code lengths are again the numbers $m_i + 1$, the new encoding will have the same cost. The numbers A_j can now be computed with ease directly in the binary system, and much of the arithmetic needed for the first construction may be avoided. However, the kind of shortening used in Table II does not work as well with the new encoding. All codes (as numbers) are now less than

$$\sum_i 2^{-m_i}.$$

This number need not be near 1 (typically it is about $\frac{3}{4}$). The codes are then cramped together in a range smaller than (0,1) and cannot be shortened as much. For the case of the English source with $N = 1$, the new encoding can only be shortened to cost 5.02 digits per letter.

IV. ENCODING TRICKS

The simple construction just given does not produce the best encoding, i.e., the one with least cost. The best encoding can always be found by a systematic, although long, calculation which is described in the next section. Here we list a few tricks whereby the problem of finding the best encoding may be simplified and, in some cases, solved.

We will describe these results in terms of encoding single letters into binary form; however, it is to be understood that blocks of N letters may always be considered the single letters of a larger alphabet. By a *prefix set* of an encoding we will mean the set of all letters which have codes beginning with a given prefix. For example, in the Huffman encoding of Table I the prefix 011 has the prefix set consisting of letters B, G, J, K, P, Q, V, X and Z. In an alphabetical encoding every prefix set must consist of all letters lying between some two fixed letters in the alphabet.

The tricks to be described enable one to prove that certain collections of letters must be prefix sets in any best alphabetical encoding. Whenever a prefix set is known the encoding problem can then be reduced as follows to one for a smaller alphabet.

Theorem 2: In a best alphabetical encoding let S be a prefix set for a prefix π. Construct a shorter alphabet by replacing the letters of S by a single new letter, L^1, occupying their place in alphabetical order and having as its probability the sum of their probabilities. A best encoding of the new alphabet gives L^1 the code π and gives every other letter its old code.

Proof: Let $C(L)$ denote the code for letter L in the original best encoding. Suppose, contrary to the theorem, that the new problem had a better solution in which L, L^1 had codes $C^1(L)$ and $C^1(L^1)$. One would then obtain a better solution of the original problem by encoding L into $C^1(L)$. The code for a letter M in the prefix class would be $C(M)$ with the prefix π changed to $C^1(L^1)$.

Huffman's encoding scheme uses a result similar to Theorem II for nonalphabetical encodings. The two letters of lowest probability must form a prefix set, and his result is used again and again, until there are only two letters left and the problem is solved. When the encoding must be alphabetical one cannot always find a prefix set easily. Some results in this direction are given by the following theorems. The symbols L_1, L_2, $\cdots$ are used to represent the letters of the alphabet in order; p_1, p_2, $\cdots$ will be their probabilities; $C(L_1)$, $C(L_2)$, $\cdots$ will be their codes in the encoding C and N_1, N_2, $\cdots$ will be the numbers of binary digits in their codes. Also, if Φ is any code or any prefix, $N(\Phi)$ will be used to represent the number of binary digits in Φ.

An encoding will be said to be *exhaustive* if it encodes an alphabet of two or more letters in a uniquely decipherable manner and, for every infinite sequence $x = x_1x_2x_3 \cdots$ of binary digits, there is some message which can be enciphered as x; or if it encodes an alphabet of one letter by using the null sequence.

Theorem 3: Every best alphabetical encoding is exhaustive.

Proof: Consider an encoding of an alphabet having two or more letters which is alphabetical and has the prefix property, but is not exhaustive. It will be shown that it is not a best encoding. Let x be an infinite sequence of binary digits such that no message can be encoded as x. If any code of the encoding is a prefix of x, remove it from x, and, after a finite number of repetitions of this process, an x will be obtained which has no one of the codes for a prefix. Let Φ be the greatest prefix of x which is also a prefix of any one of the codes. Let C_i be some code of which Φ is a prefix. We will use $\Phi 0$ to represent the sequence Φ followed by 0. Then either $\Phi 0$ is a prefix of C_i and $\Phi 1$ is a prefix of x, and $\Phi 1$ is not a prefix of any code of this encoding; or else $\Phi 1$ is a prefix of C_i and $\Phi 0$ is a prefix of x, and $\Phi 0$ is not a prefix of any code of this encoding. Without loss of generality, we assume the second one of these alternatives. Then consider the new encoding which agrees with the old one for all codes not having Φ as a prefix, but which has a code $\Phi\theta$ in place of each code of the form $\Phi 1\theta$. The new encoding has a lower cost than the old one, is still alphabetical and still has the prefix property. Hence the original encoding was not a best alphabetical encoding.

Lemma 1: Let π be a prefix. In a best alphabetical encoding, if there is a code with prefix $\pi 0$ there is one with prefix $\pi 1$. Conversely, if there is a code with prefix $\pi 1$, there is one with prefix $\pi 0$.

Proof: If $\pi 0$ is a prefix, then by Theorem 3 the sequence $\pi 111 \cdots$ must have some code C_i as a prefix. But by the prefix property, C_i cannot be a prefix of $\pi 0$; hence, C_i has prefix $\pi 1$. The converse is proved similarly.

Lemma 2: Let L_a be the letter of lowest probability. In a best alphabetical encoding, L_a, together with one of L_{a+1} or L_{a-1} must form a prefix set.

Proof: Suppose $C(L_a)$ ends in 0, say $C(L_a) = \pi 0$, where π stands for some prefix. By Lemma 1, $\pi 1$ is a prefix of $C(L_{a+1})$. If $C(L_{a+1}) = \pi 1$, we have the desired result. If not, $\pi 10$ must be a prefix of $C(L_{a+1})$. By Lemma 1 there exist codes with prefix $\pi 11$. A better encoding (and hence a contradiction) may be had by the following changes: Lengthen $C(L_a)$ from $\pi 0$ to $\pi 00$. Change all codes of the form $\pi 10\psi$ to $\pi 01\psi$. Shorten all codes of the form $\pi 11\psi$ to $\pi 1\psi$. Since the last change applies to at least one letter (of higher probability than L_a), there is a net decrease in cost.

The proof in the other case [$C(L_a)$ ending in 1] is similar. If, as is the case of the probabilities of Table I, the least probable letter is at the end of the alphabet, then this letter has only one neighboring letter and must form a prefix set with it. Thus, as a first step in Table I, we can write

$$C(\mathrm{Y}) = \pi(\mathrm{Y,Z})0,$$
$$C(\mathrm{Z}) = \pi(\mathrm{Y,Z})1,$$

where $\pi(\mathrm{Y,Z})$ is some unknown prefix. Then, using Theorem 2, the problem is reduced to an encoding for a 26-letter alphabet in which Y and Z have been replaced by a single letter $L(\mathrm{Y,Z})$ of probability 0.0169. When this new problem is solved, $\pi(\mathrm{Y,Z})$ will be found as the code for $L(\mathrm{Y,Z})$. The new least probable letter is J or Q, both with the same probability 0.0008; J, for example, can be in a prefix set with either I or K, but Lemma 2 gives no clue for deciding which one. One might hope that one can always pick the less probable neighbor, K in this case. However, it is easy to find counter-examples which disprove this conjecture. A weaker, but true, theorem is the following one.

Theorem 4: Let L_a be the letter of lowest probability. Suppose that

$$p_{a+1} > p_a + p_{a-1}\,. \tag{7}$$

Then L_a and L_{a-1} must form a prefix set in any best alphabetical encoding. Similarly, if $p_{a-1} > p_a + p_{a+1}$, L_a and L_{a+1} must form a prefix set.

Proof: Suppose (7) holds but that L_a and L_{a-1} do not form a prefix set. Then, by Lemma 2, L_a and L_{a+1} form a prefix set. The codes for L_a and L_{a+1} must be of the form

$$C(L_a) = \pi 0,$$
$$C(L_{a+1}) = \pi 1$$

for some prefix π. The code $C(L_{a-1})$ must end in 1, say $C(L_{a-1}) = \rho 1$.

For, if $C(L_{a-1})$ were $\rho 0$, Lemma 1 would show that some code has prefix $\rho 1$ and hence must stand for a letter between L_{a-1} and L_a in the alphabetical order, an impossibility. Lemma 1 now shows that some other letters have prefix $\rho 0$.

We consider two cases determined by the numbers $N(\pi)$ and $N(\rho)$ of digits in π and ρ:

Case 1 — $N(\pi) < N(\rho)$. An improved encoding can be made by changing $C(L_a)$ from $\pi 0$ to $\pi 01$, $C(L_{a-1})$ from $\rho 1$ to $\pi 00$ and all codes of the form $\rho 0\psi$ to $\rho\psi$. The last change, a shortening, affects some codes and so offsets the lengthening of the least probable code.

Case 2 — $N(\rho) \leqq N(\pi)$. An improvement can be made by shortening $C(L_{a+1})$ from $\pi 1$ to π while changing $C(L_a)$ from $\pi 0$ to $\rho 11$ and $C(L_{a-1})$ from $\rho 1$ to $\rho 10$. That there is a net decrease in cost follows from (7).

The other half of the theorem is proved in a similar way.

Applying Theorem 4 to our reduced problem of Table I, we obtain further reductions, producing new letters L(J,K) and L(P,Q) with probabilities 0.0057 and 0.0160. Now the lowest-probability letter has become X, and we need another kind of theorem.

Theorem 5: If L_i and L_j ($i < j$) are two letters both of probability exceeding $p_{i+1} + p_{i+2} + \dots + p_{j-1}$, then the intervening letters L_{i+1}, L_{i+2}, $\dots$, L_{j-1} form a prefix set in any best alphabetical encoding.

Proof: Let π denote the greatest common prefix of $C(L_i)$ and $C(L_j)$, i.e., a prefix such that $\pi 0$ is a prefix of $C(L_i)$ while $\pi 1$ is a prefix of $C(L_j)$. The intervening letters have either $\pi 0$ or $\pi 1$ as prefixes. Supposing that there are some intervening letters with prefix $\pi 0$, we assert that *the intervening letters with prefix $\pi 0$ form a prefix set.* To prove this assertion, let the intervening letters with prefix $\pi 0$ be $L_{i+1}, \dots L_c$, where $C(L_{c+1})$ has prefix $\pi 1$. Let $\pi 0\rho$ denote the greatest common prefix of $C(L_i)$ and $C(L_c)$. Then $C(L_c)$ must have prefix $\pi 0\rho 1$; otherwise, by Lemma 1, L_{c+1} would have prefix $\pi 0\rho$, and hence $\pi 0$. Also, $C(L_i)$ has prefix $\pi 0\rho 0$; otherwise, $\pi 0\rho 1$ would be a greater common prefix than $\pi 0\rho$. The assertion requires only that we prove that $C(L_{i+1})$ has prefix $\pi 0\rho 1$, for then the letters in question and no others have this prefix. If, on the contrary, $C(L_{i+1})$ has prefix $\pi 0\rho 0$, find the greatest common prefix $\pi 0\rho 0\sigma$ such that $\pi 0\rho 0\sigma 0$ is a prefix of $C(L_i)$ and $\pi 0\rho 0\sigma 1$ is a prefix of $C(L_{i+1})$. Now shorten all codes of the form $\pi 0\rho 0\sigma 0\psi$ to $\pi 0\rho 0\sigma\psi$ and lengthen all other codes $\pi 0\rho\psi$ to $\pi 0\rho 1\psi$. The shortened codes include the one for L_i, which has more probability than the total probability of all the lengthened codes. The assertion is now proved, and likewise *intervening letters with prefix $\pi 1$ form a prefix set.*

By our two assertions, each of $C(L_{i+1}), \dots, C(L_{j-1})$ has one of two

prefixes, which we may call $\pi 0 \rho 1$ and $\pi 1 \tau 0$, while $\pi 0 \rho 0$ is a prefix of $C(L_i)$ and $\pi 1 \tau 1$ is a prefix of $C(L_j)$. Again, one proves the theorem by making changes which put the intervening letters into a single prefix set. There are two cases:
Case 1 — $N(\pi 0 \rho) \leqq N(\pi 1 \tau)$. Lengthen codes $\pi 0 \rho 1 \psi$ to $\pi 0 \rho 10 \psi$. Change codes $\pi 1 \tau 0 \psi$ to $\pi 0 \rho 11 \psi$. Shorten all codes $\pi 1 \tau 1 \psi$ to $\pi 1 \tau \psi$. The intervening letters now form a prefix set with prefix $\pi 0 \rho 1$ and the new encoding has smaller cost.
Case 2 — $N(\pi 1 \tau) \leqq N(\pi 0 \rho)$. By changes similar to those of Case 1, one may reduce the cost by making the intervening letters into a prefix set with prefix $\pi 1 \tau 0$.

Applying Theorem 5 to Table I, we now recognize new prefix sets and reduce the problem by introducing new letters L(F,G) and L(U,V, W,X,Y,Z) of probabilities 0.0360 and 0.0668. Now L(J,K) becomes the least probable letter, Theorem 4 applies, and we form a new letter L(J,K,L) of probability 0.0378. Next, Theorem 4 applies to letter B, and we form a new letter L(B,C) of probability 0.0345. Again we are at an impasse.

Theorem 6: If $p_1 < p_3$, then L_1 and L_2 form a prefix set in any best alphabetical encoding. Similarly, if L_n is the last letter of the alphabet, L_{n-1} and L_n must form a prefix set if $p_n < p_{n-2}$.

Proof: If $p_1 < p_3$ and L_1 and L_2 are not a prefix set, then $C(L_1)$, $C(L_2)$ and $C(L_3)$ may be shown to have the forms $\pi 0$, $\pi 1 \rho 0$ and $\pi 1 \rho 1 \psi$. Then one could improve the encoding by changing $C(L_1)$ to $\pi 00$, $C(L_2)$ to $\pi 01$ and all codes $\pi 1 \rho 1 \psi$ to $\pi 1 \rho \psi$.

This theorem provides no further reduction of our example. Note, however, that it might have been applied following the creation of L(Y,Z) to prove that X,Y,Z, forms a prefix set. This information is helpful when we must add the final digits to the prefix π(U,V, . . . , Z) to form the codes for U, . . . , Z. Using Huffman's encoding method, we find, disregarding questions of alphabetical order, the best way of encoding four letters which have probabilities in the same ratio as our letters U,V,W and L(X,Y,Z). The solution gives each letter two digits. Then, an equally good alphabetical encoding gives these letters the code 00, 01, 10, 11. We now know parts of the codes sought, as summarized in Table III. The unknown prefixes π(B,C), . . . are to be determined by finding a best alphabetical encoding of the 17-letter alphabet listed in Table IV.

Again we might try a Huffman encoding for Table IV. However, we note in advance that M and L(P,Q) are much less probable than their neighbors. Then a Huffman encoding will give these letters such long

TABLE III

Letter	Code
B	π(B,C)0
C	π(B,C)1
F	π(F,G)0
G	π(F,G)1
J	π(J,K,L)00
K	π(J,K,L)01
L	π(J,K,L)1
P	π(P,Q)0
Q	π(P,Q)1
U	π(U, $\cdots$, Z)00
V	π(U, $\cdots$, Z)01
W	π(U, $\cdots$, Z)10
X	π(U, $\cdots$, Z)110
Y	π(U, $\cdots$, Z)1110
Z	π(U, $\cdots$, Z)1111

codes that there will be no alphabetical encoding which uses the same length codes for every letter. To circumvent this difficulty we use Lemma 2, first on L(P,Q) and next on M, and conclude that L(P,Q) must form a prefix set with O or R and M must form a prefix set with L(J,K,L) or N. There are then four new alphabets to consider, and we have constructed Huffman encodings for each one. The one with smallest cost is the one in which J,K,L,M and P,Q,R were made into new letters. The numbers of digits for the letters in Table IV which this Huffman encoding required are listed. We next look for an alphabetical encoding in which the same numbers of digits is used. Such an encoding actually

TABLE IV

Letter	Probability	Number of Digits
Space	0.1859	2
A	0.0642	4
L(B,C)	0.0345	5
D	0.0317	5
E	0.1031	4
L(F,G)	0.0360	5
H	0.0467	5
I	0.0575	4
L(J,K,L)	0.0378	5
M	0.0198	5
N	0.0574	4
O	0.0632	4
L(P,Q)	0.0160	5
R	0.0484	5
S	0.0514	4
T	0.0796	4
L(U, $\cdots$,Z)	0.0668	4

exists, and so we obtain the best alphabetical encoding shown in Table I. It must be admitted that we were somewhat lucky to be able to reduce the problem to one in which one of the best possible encodings, disregarding alphabetical order, includes an alphabetical encoding. Undoubtedly, minor changes in the probabilities in Table I might make the problem much harder. In the next section we give an encoding method which will apply in all cases.

V. THE GENERAL ALPHABETIZING ALGORITHM

The method which will be used in general builds up the best alphabetical encoding for the entire alphabet by first making best alphabetical encodings for certain subalphabets. In particular, the subalphabets which will be considered will be only those which might form a prefix set in some alphabetical binary encoding of the whole alphabet. Since only those sets of letters consisting exactly of all those letters which lie between some pair of letters can serve as a prefix set, we will call such a set an *allowable* subalphabet.

We will denote the allowable subalphabet consisting of all of those letters which follow L_i in the alphabet (including L_i itself) and which precede L_j (again including L_j itself) by (L_i, L_j). When referring to the ordinary English alphabet of Table I we will use the symbol ⋕ for the space symbol. Thus, (⋕,B) will be the subalphabet containing the three symbols space, A and B, and (A,A) will be used to denote the subalphabet containing only the letter A.

If it were desired to find an optimum encoding satisfying certain kinds of restrictions other than the alphabetical one, different allowable subalphabets could be used, with the rest of the algorithm remaining analogous. This method of building up an encoding by combining encodings for subalphabets is analogous to the method used by Huffman,[1] except that he was able to organize his algorithms such that no subalphabets were used except those which actually occurred as prefix sets in his final encoding. However, we consider all allowable subalphabets, including some which are not actually used as part of the final encoding.

The term cost of an encoding has been used to refer to the average number of binary digits per letter of transmitted message, that is, $\sum_i p_i N_i$. Since, in the algorithm to be described, we will be constructing an encoding for each allowable subalphabet, we will also use the corresponding sum for each subalphabet. But, since the probabilities p_i do not even add up to 1 for proper subalphabets, the sum $\sum_i p_i N_i$ does not correspond exactly to a cost of transmitting messages, and so the corresponding sum will be called a *partial cost*.

The algorithm to be described takes place in n stages, where n is the number of letters in the alphabet. At the kth stage, the best alphabetical binary encoding for each k-letter allowable subalphabet will be constructed and its partial cost will be computed. For $k = 1$, each subalphabet of the form (L_i, L_i) will be encoded by the trivial encoding which encodes L_i with the null sequence; it has cost 0, since the number of digits in the null sequence is zero. For $k = 2$, each subalphabet of the form (L_i, L_{i+1}) will be encoded by letting the code for L_i be 0 and the code for L_{i+1} be 1. The partial cost of this encoding is $p_i + p_{i+1}$. In general, the kth stage of the algorithm, in which it is desired to find the best alphabetical binary encoding for each subalphabet of the form (L_i, L_{i+k-1}) and its partial cost, proceeds by making use of the codes and the partial costs computed in the previous stages.

For each j between $i + 1$ and $i + k - 1$, we can define a binary alphabetical encoding as follows: Let $C_i, C_{i+1}, \ldots C_{j-1}$ be the codes for $L_i, L_{i+1}, \ldots L_{j-1}$ given by the (previously constructed) best alphabetical encoding for (L_i, L_{j-1}), and let $C'_j, C'_{j+1}, \ldots, C'_{i+k-1}$ be the codes for $L_j, L_{j+1}, \ldots, L_{i+k-1}$ given by the (previously constructed) best alphabetical encoding for (L_j, L_{i+k-1}). Then the new encoding for $L_i, L_{i+1}, \ldots, L_{j-1}, L_j, L_{j+1}, \ldots, L_{i+k-1}$ will be $0C_i, 0C_{i+1}, \ldots, 0C_{j-1}, 1C'_j, 1C'_{j+1}, \ldots, 1C'_{i+k-1}$. Such an encoding can be defined for each j, and the encoding is exhaustive. It follows from Theorem 2 that the best encoding for this subalphabet is given by one of the $k - 1$ such encodings which can be obtained for the $k - 1$ different values of j. The partial cost of such an encoding made up out of two subencodings is the sum of the partial costs of the two subencodings plus $p_i + p_{i+1} + \ldots + p_{i+k-1}$. To perform the algorithm it will not be necessary to construct all of these encodings, but only to compute enough to decide which one of the $k - 1$ different encodings has the lowest partial cost. This is done by taking the sums of each of the $k - 1$ pairs of partial costs of subencodings and constructing the best encoding only.

After the kth stage of this algorithm has been completed for $k = 1, 2, \ldots, n$, the final encoding obtained is the best alphabetical encoding for the entire original alphabet, and the final partial cost obtained is the cost of this best alphabetical encoding.

If the above algorithm were performed on a digital computer, the length of time required to do the calculation would be proportional to n^3. The innermost inductive loop of the computer program would perform the operation mentioned above of computing sums of pairs of partial costs, and this would be done $k - 1$ times in the process of encoding each one of the subalphabets considered in the kth stage. But, since there are $n - (k - 1)$ different allowable subalphabets to be encoded in the kth stage, there are $(k - 1)\,[n - (k - 1)]$ steps to be done in the kth stage. To find the total number of operations done in all of the stages, we sum, and find that

$$\sum_{k=1}^{n} (k - 1)[n - (k - 1)] = \frac{(n^3 - n)}{6},$$

which is an identity which can be verified by mathematical induction.

VI. PROPERTIES OF EXHAUSTIVE ENCODINGS

We have already shown (Theorem 3) that every best alphabetical encoding is exhaustive. Another reason for considering exhaustive encodings to be of some general interest is given by the following theorem.

Theorem 7: The Huffman binary encoding of any alphabet is exhaustive.

Proof: We prove by induction that each of the encodings for prefix sets arrived at during the steps of the algorithm of Huffman[1] is an exhaustive encoding. If this holds for the first k encodings constructed during this algorithm, consider the prefix set L encoded at the $(k + 1)$th step. Let $x = x_1x_2x_3 \ldots$ be any infinite sequence of binary digits. It suffices to show that there is some letter whose code is a prefix of x. The set L was made by combining two previous prefix sets of letters, L' and L'', and it was encoded by prefixing the codes from their previous encodings by 0 and 1 respectively. Let L' be the set whose codes were prefixed by x_1. Then if L' is a single letter, x_1 is its code, and hence its code is a prefix of x. But if L' is a prefix set, then its previous encoding is exhaustive by inductive hypothesis, and hence there is a letter L‴ whose previous code is a prefix of $x_2x_3 \ldots$. Then the new code for L‴ is a prefix of x.

Several of the properties of exhaustive encodings will be considered, since both the Huffman encoding and the best alphabetical encoding are exhaustive, and it seems likely that exhaustive encodings might arise from other types of optimizing problems. For instance, the shortening procedure used in Table II was essentially a way of making the encoding more nearly exhaustive.

Lemma 3: Whenever an encoding C has the property that for any infinite sequence $x = x_1x_2x_3 \ldots$ there is a code of C which is a prefix of x, then

$$\sum_{i=1}^{n} 2^{-N_i} \geqq 1, \tag{8}$$

and equality holds if and only if C has the prefix property.

Proof: Consider the set P of all finite sequences x having length exactly k, where k is some fixed integer longer than the longest code of C. Then the property assumed in the hypothesis implies that each element of P has at least one of the codes for a prefix. But P has exactly 2^k elements, and for each code of length N_i there are 2^{k-N_i} elements of P of which it is a prefix. Hence,

$$\sum_{i=1}^{n} 2^{k-N_i} \geqq 2^k,$$

which is equivalent to (8), and equality holds if and only if no element of P has two different codes for a prefix. However, the occurrence of two different codes which are prefixes of the same sequence is exactly equivalent to having one of the two codes be a prefix of the other.

Theorem 8: Every exhaustive binary encoding has the prefix property and satisfies

$$\sum_{i=1}^{n} 2^{-N_i} = 1. \tag{9}$$

Proof: By Lemma 3 and the definition of exhaustive, (8) holds, but, by McMillan,[3] unique decipherability implies

$$\sum_{i=1}^{n} 2^{-N_i} \leqq 1. \tag{10}$$

Then we combine (8) and (10) to obtain (9). But, by Lemma 3, this implies the prefix property.

Lemma 4: For any exhaustive encoding of an alphabet, and any prefix Φ of this encoding, the new encoding of the prefix-set subalphabet which associates the new code θ with each letter whose original code was $\Phi\theta$ is an exhaustive encoding of this subalphabet.

Proof: Given any x, to find a letter whose new code is a prefix of x we consider the letter L whose original code was a prefix of Φx. Then, by the prefix property, the original code of L cannot be a prefix of Φ, and thus the original code of L is of the form $\Phi\theta$. Hence, L is in the subalphabet, its new code is θ, and θ is a prefix of x. To complete the proof that the new encoding is exhaustive, note that it has the prefix property because the original encoding does. Hence, the new encoding is either the trivial encoding (of a one-letter alphabet) or is uniquely decipherable.

Lemma 5: For any exhaustive binary encoding of an alphabet having n letters, the total number of prefixes is $2n - 1$.

Lemma 6: In any exhaustive binary encoding of an alphabet having n letters, none of the codes consist of more than $n - 1$ digits.

Each of the last two lemmas associates a number with each exhaustive encoding, and they can be proved by induction on the number of letters in the alphabet. The number associated with each exhaustive encoding is represented in terms of the number associated with each of the two encodings that are constructed as described in Lemma 4 for the subalphabet having the prefix 0 and the subalphabet having the prefix 1.

Theorem 9: The cost of the Huffman encoding of an alphabet is a continuous function of the probabilities of the letters.

Theorem 10: The cost of the best alphabetical encoding of an alphabet is a continuous function of the probabilities of the letters.

The last two theorems will be proved together, enclosing in parentheses the changes which convert the proof of Theorem 9 into a proof for Theorem 10. In fact, what will be proved are the slightly stronger theorems: For two alphabets A and A^* having the same n letters, if p_i is the probability of the ith letter of A, p_i^* is the probability of the ith letter of A^*, and if k and k^* are the costs of the Huffman encoding (best alphabetical encoding) for A and A^*, then

$$|k - k^*| \leqq (n - 1) \sum_{i=1}^{n} |p_i - p_i^*|. \tag{11}$$

If we let B be the right member of inequality (11) and let k' be the cost of using the Huffman (best alphabetical) encoding of A^* as an encoding for A, then, by Lemma 6 and the definition of cost, we can conclude that $|k' - k^*| \leqq B$ and, since from the definition of k we can conclude that $k \leqq k'$, we can combine these to obtain $k^* - k \leqq B$. By a similar argument involving the use of k'', the cost of using the Huffman (best alphabetical) encoding of A as an encoding for A^*, we obtain $k - k^* \leqq B$. Combining these, we obtain (11).

Theorem 11: The Huffman encoding for a given alphabet has a cost which is less than or equal to that of any uniquely decipherable encoding for that alphabet.

Proof: This proof is essentially that of McMillan.[3] Let us consider any uniquely decipherable encoding C. We will construct a new encoding C' which has the same cost as C, and which has the prefix property. However, by its method of contruction, the Huffman encoding has a cost which is less than or equal to that of any encoding having the prefix property, completing the proof of the theorem. Let N_i be the number of digits in the code which C associates with the ith letter of the alphabet. Let the letters of the alphabet be renumbered in such a way that $N_i \leq N_{i+1}$. Then, as in the encoding theorem (Theorem 1 of this paper, or Theorem 9 of Shannon,[8] we let

$$A_i = \sum_{j=1}^{i-1} 2^{-N_j},$$

and we define C' to be the encoding which associates with the ith letter the code C_i' obtained by truncating A_i after N_i digits. Then it follows that the digits truncated were 0's, and hence that each C_i' agrees numerically with the corresponding A_i. By (10), each of the A_i is less than 1. To show that C' has the prefix property, we assume that C_i' is a prefix of C_j'. Then $i < j$, by the renumbering. However, $A_{i+1} = A_i + 2^{-N_i}$, and hence $A_j \geqq A_i + 2^{-N_i}$. Thus, A_j cannot agree with the first N_i places of A_i. Hence, the first N_i digits of C_j' are different from those of C_i'.

Theorem 12: If A_n is the number of exhaustive binary alphabetical encodings for an alphabet having n letters, $A_1 = A_2 = 1$, and for $n \geqq 3$ we have

$$A_n = \frac{(2n-3)!2}{(n-2)!\,n!}. \tag{12}$$

Theorem 13: If T_n is the total number of exhaustive binary encodings for an alphabet having n letters, $T_1 = 1$, $T_2 = 2$ and, for $n \geqq 3$, we have

$$T_n = \frac{(2n-3)!2}{(n-2)!}. \tag{13}$$

These theorems show how rapidly A_n and T_n increase with increasing n. Since, by Theorem 3, A_n would be the number of encodings to consider if it were desired to find the best alphabetical encoding by enumeration, Theorem 12 shows that the methods already given in this paper (even the general alphabetizing algorithm) are much faster than exhaustive enumeration. Similarly, Theorem 7 and Theorem 8 show how much slower exhaustive enumeration is than the algorithm given by Huffman.[1]

Each of the A_n alphabetical encodings may be converted into $n!$ of the T_n encodings by permuting its codes in all possible ways. It follows that $T_n = n!A_n$, and it suffices to prove Theorem 12. Consider for $n \geqq 2$ an exhaustive alphabetical encoding of n letters. Some number $k = 1, \ldots, n-1$ of these letters has a code with prefix 0. These k codes, each with its leading digit 0 removed, have been shown (Lemma 4) to form one of the A_k exhaustive alphabetical encodings of k letters. Similarly, the remaining $n - k$ codes, minus their leading digits 1, form one of the A_{n-k} exhaustive alphabetical encodings of $n - k$ letters. Thus, if $n \geqq 2$,

$$A_n = \sum_{k=1}^{n-1} A_k A_{n-k}, \tag{14}$$

while $A_1 = 1$. To solve (14), construct the generating function $a(x) = A_1x + A_2x^2 + A_3x^3 + \ldots$. By (14), $a(x) = x + a^2(x)$; i.e.,

$$a(x) = \tfrac{1}{2}(1 - \sqrt{1 - 4x}). \tag{15}$$

The negative sign of the square root is needed to make $a(0) = 0$. The series for $a(x)$ is obtained using the binomial theorem with power $\frac{1}{2}$. The coefficient of x^n (which is A_n) has the expression (12).

VII. ENCODINGS WITHOUT THE PREFIX PROPERTY

So far in this paper very little has been said about encodings without the prefix property. For instance, we restricted the best alphabetical encoding to be the encoding having the lowest cost among all alphabetical order-preserving encodings having the prefix property. However, in view of the fact that the special encoding given in Table I is an alphabetical encoding and has cost 4.1801, it appears to be advantageous to dispense with the prefix property requirement. However, not very much is known about the properties of encodings lacking the prefix property, and, in fact, it is not known whether the special encoding given in Table I can be further improved or not. In fact, it was not constructed on the basis of any general procedure, but was found by a heuristic method. The next few paragraphs will give a few results which we have found about encodings without the prefix property, but will also give some examples of the difficulties which it is possible to get into when using such encodings.

It should be noted that a message which begins with the letter Y in the special encoding cannot be deciphered as soon as the Y has been received, but it is necessary to wait for further received digits in order to distinguish it from a Z. In particular, in the case of the message enciphered as 11111101111110 it is necessary to wait for the 14th received binary digit before the first letter can be deciphered.

In general, we will say that the *delay of a presumed message* is d if it is necessary to wait for the receipt of the first d binary digits before the first transmitted letter can be recognized. We will say that the *delay of an encoding* is d if d is the least upper bound of the delays of all presumed messages of that encoding. We will say that an encoding has the *finite delay property* if the delay of that encoding is finite. For instance, the special encoding of Table I has the finite delay property, and in fact has delay 14.

Theorem 14: If an encoding C has infinite delay, then there exists a presumed message of C which has infinite delay.

Proof: Given an encoding C with infinite delay, there exists an infinite sequence of presumed messages $M_1, M_2, M_3, \ldots$ such that M_i has delay at least i. Then either the set of those presumed messages M_i whose first binary digit is 0 or the set whose first binary digit is 1 is an infinite set. We thus can choose an infinite subsequence of presumed messages $M_1, M_2, M_3, \ldots$ such that M_i has delay at least i and such that all of the messages agree on the first binary digit. Proceeding by induction, we can choose at the kth step a subsequence of presumed messages which all agree on the first k digits. Then the infinite presumed message whose kth binary digit is the kth binary digit of all presumed messages remaining after the kth inductive step is a presumed message, and has infinite delay.

For an encoding to be useful in practice, it seems likely that it must have the finite delay property. This would permit a deciphering machine to be built having only a finite amount of memory, and it would permit two-way communication (as in telephony) to be almost instantaneous. However, in delayed communication systems (common in telegraphy) for which a tape is used for storing messages, this tape might be used to provide the unbounded amounts of memory needed to decipher an infinite delay encoding.

To investigate further the problems of designing an optimal-cost encoding of any sort (such as an alphabetical-order encoding), without requiring it to have the prefix property, it should be remarked that the problem is finite, but not necessarily easy to attack. That is, given an alphabet in which all of the letters have positive probability, and given a constant K, there are only a finite number of encodings of this alphabet which have a cost less than K. For if m is the smallest of the probabilities, there are not more than K/m digits in the longest code of any such encoding, and there are only a finite number of encodings of an n-letter alphabet in which each code has length less than K/m. However, this number would be astronomically large for any alphabet of reasonable size.

One particular way of generating encodings which will be used in a few examples below is of some general interest. The *reversal* of an encoding C is a new encoding (which will be called C^* for the remainder of this paper) which is obtained by letting the code for each letter be written in the reverse order. This interchanges the direction of increasing time, and changes many of the properties of the encoding, but it does preserve unique decipherability.

Table V demonstrates many of the properties and complications of encodings, contrasting the one having the prefix property with three other encodings lacking this property. Each of the four encodings shown

TABLE V

Letter	Probability	First Code	Second Code	Third Code	Fourth Code
A	0.330	000	00	00	00
B	0.005	001	001	0011	00111
C	0.330	01	10	01	01
D	0.005	10	101	0111	01111
E	0.330	11	11	10	10
	Cost	2.335	2.01	2.02	2.03

preserves alphabetical order, and each is uniquely decipherable. The first encoding has the prefix property, and in fact is the best alphabetical encoding in the sense used in this paper. However, it has an appreciably higher cost than either of the other three encodings, none of which has the prefix property. The reversals of each of the last three encodings have the prefix property, but the reversal of the first encoding does not.

The second encoding of Table V has the lowest possible cost of any uniquely decipherable binary encoding by Theorem 11, since it is the reversal of a Huffman encoding. However, the second encoding has infinite delay, since the presumed message 001111 . . . has infinite delay. Furthermore, the second encoding, although it preserves the alphabetical order of individual letters, does not preserve the alphabetical order of words made up out of these letters. For instance, the enciphered form of CE is a larger binary number than the enciphered form of DA, although the latter occurs later in alphabetical order. The property of preserving alphabetical order of all words will be called the *strong alphabetical property*, and it has already been shown that alphabetical encodings having the prefix property have the strong alphabetical property. However, both the alphabetical encoding and the special encoding of Table I have the strong alphabetical property, and all of the encodings of Table V except the second encoding have the strong alphabetical property. There would be very little to be gained by employing an alphabetical order encoding for sorting or dictionary purposes unless it had the strong alphabetical property.

The third encoding lacks these defects of the second encoding, but it has a special one of its own, about which more will be said in the next section. This defect has to do with synchronizing, and it can be explained in this case by the observation that every code of the third encoding has an even number of binary digits. Thus, if the deciphering circuit starts up while it is out of phase, it can never get back in phase. The two phases correspond to the odd-numbered and the even-numbered binary digits, and the deciphering machine, if it is out of phase, would never get back in. In this case, where there are certain codes which cannot occur, the defect could be remedied by designing the circuit to additionally change phase if it ever receives a code 1011 or 1111, but this adds an extra complication to the circuit. However, the first and second encodings have the property that each of them will automatically get back in synchronism with probability 1, without the addition of any other codes or any other special features to the circuit.

The fourth encoding has none of these defects, and since its cost is so near to the least possible, it would undoubtedly be a reasonably good choice as a solution, if this particular alphabet had arisen in an actual practical problem.

So far in this paper, each example of an encoding with the finite delay property has had a delay equal to $N_{\max}$, where $N_{\max}$ is the number of digits of the longest code of the encoding. This result does not hold in general, as is illustrated by Table VI. The fifth encoding has $N_{\max} = 6$, but it has delay 8.

Table VI

Letter	Fifth Code	Sixth Code
W	00	00
X	001	01
Y	101	10
Z	110101	11

The encodings having the finite delay property but not the prefix property, such as the special encoding of Table I and the fifth encoding of Table VI, provide counterexamples which contradict Remark II of Schützenberger (Ref. 5, page 55) and provide the example which is asked for in the sentence following Remark I of the same paper.

As an alternative to the above method of expressing quantitatively the finite delay property, we may make the following definitions for use later in this paper. We will say that the *excess delay of a presumed message* is e if it is necessary to wait for the receipt of e binary digits beyond the end of the first transmitted letter of the presumed message before this first letter can be recognized. We will say that the *excess delay of an encoding* is e if e is the least upper bound of the delays of all presumed messages of the encoding.

If d is the delay of an encoding, e is its excess delay, and $N_{\min}$ and $N_{\max}$ are, respectively, the minimum and maximum numbers of digits of any codes of the encoding, then we obviously have $e + N_{\min} \leqq d \leqq e + N_{\max}$. Then an encoding has the finite delay property if and only if the excess delay of that encoding is finite. Also, an encoding has the prefix property if and only if the excess delay of that encoding is 0.

VIII. SELF-SYNCHRONIZING PROPERTIES

Problems of how to make a transmitting device and a receiving device become and remain synchronized with each other are important in the engineering design of many kinds of systems. Since the encodings discussed in this paper are variable-length, it might seem that the synchronizing problem for enciphering and deciphering circuits would be especially difficult. However, the synchronizing problem is very simple for many variable-length binary encodings, because of a particularly favorable property which they possess. These remarks can best be illustrated by an example. Suppose that (using the alphabetical encoding of Table I as an example) a message beginning 1110011110100111000 . . . is received, and we wish to observe how a deciphering circuit would decipher it. Since the encoding has the prefix property, the deciphering circuit should first find a code which is a prefix of this message, and then decode this to obtain the first letter T of this message. Proceeding with

Table VII

```
  :     T         :         H         :     A     :           T   :   #       :
    1   1   1   0   0   1   1   1   1   0   1   0   0   1   1   1   0   0   0 ...
      :         R         :       T       :         M         :       I       :
```

the remaining part, it then finds the letter H, and then the rest of the deciphered version shown in the first line of Table VII, where the symbol ":" is used to mark the divisions between those sequences of binary digits which were deciphered as individual letters.

Next suppose that the same sequence of digits had been received, but that the deciphering circuit was not in synchronism with the enciphering circuit. In particular, suppose that, when the deciphering circuit was first turned on, it was in the state that it would be in if it were partly through the operation of deciphering some letter, and that the initial 1 of the message was interpreted as the last digit of this letter. This deciphering is indicated on the third line of Table VII. Once again, the symbol ":" has been used to mark the divisions between letters. Then these two decipherings are out of phase (i.e., out of synchronism) with one another at the beginning of the message, but at the end of the received message they are in phase with each other, as is indicated by the fact that the ":" symbols align with each other at the right end of Table VII. This means that the deciphering circuit would have automatically become synchronized, without any special synchronizing circuits or synchronizing pulses being necessary. It was, of course, necessary for at least two of the codes of the encoding to end in the same sequence of digits, but this is very likely to happen for any variable-length encoding, unless special efforts are made to prevent it.

However, if we had been using a fixed-length encoding, such as the sixth encoding of Table VI, in which all of the codes have a fixed length k, there would be exactly k different phases in which the deciphering circuit might find itself, and the circuit could never make a transition between them. No pair of different codes can end in exactly the same sequence of digits, and so no two of these phases can become synchronized. Each of these phases will have all of the codes ending after j digit times, and after $k + j$, $2k + j$, etc., where j is the remainder obtained on dividing the position of the symbol ":" by k, and hence j can take on k different possible values.

Also, even in the case of variable-length encodings, if all of the code lengths are divisible by some integer k, then there will be at least k different phases. For if the position of one occurrence of the symbol ":" has remainder j when divided by k, the position of all other occurrences of the symbol ":" in this phase of decipherment will have the same remainder.

The above remarks apply strictly to exhaustive encodings, but may not apply where there are certain sequences of digits which can never occur. For if such a sequence of digits does occur, this may be used by the circuit as a special indication that it is out of phase, and hence it may be possible to build auxiliary circuits which can cause resynchronization, even when a fixed-length encoding is used. So a more complete treatment of synchronization would allow such auxiliary circuits, but here we will consider only self-synchronization, which is carried out inherently by the same means as is used for deciphering.

To speak more precisely about the self-syncrhonizing properties, we will make some definitions. Given any encoding C and any

finite sequences x and y such that x is not the enciphered form (with respect to encoding C) of any message, and xy is a presumed message, (16)

if z is a finite sequence of binary digits such that both xyz and yz are complete enciphered messages, we will say that z is a *synchronizing sequence for x and y*. As an example, we have seen in Table VII that 01111010011100 is a synchronizing sequence for 1 and 110.

Given any uniquely decipherable encoding C which has some codes of length more than 1, exactly one of the three statements given below will hold:

i. For all (16), there is no z such that z is a synchronizing sequence for x and y. The encoding C will then be said to be *never-self-synchronizing.*

ii. For each (16), there is a z which is a synchronizing sequence for x and y. The encoding C will then be said to be *completely self-synchronizing.*

iii. For some (16), there is a synchronizing sequence for x and y, but for other (16), there is no synchronizing sequence for x and y. The encoding C will then be said to be *partially self-synchronizing.*

Furthermore, we will define a sequence z to be a *universal synchronizing sequence* for the encoding C if, for all (16), this same sequence z is a synchronizing sequence for x and y.

Theorem 15: Given an exhaustive encoding C, then C is completely self-synchronizing if and only if there exists a z which is a universal synchronizing sequence for C.

Proof: A universal synchronizing sequence clearly satisfies the conditions of the definition of completely self-synchronizing, so it remains only to construct a universal synchronizing sequence, given that there is a synchronizing sequence for each finite sequences x and y. By the exhaustive property, there is a code consisting entirely of 0's. We will assume that there are k 0's in this code. We will construct our z by starting with $N_{\max}$ 0's, where $N_{\max}$ is the length of the longest code of C; after this, there are only k different phases in which the circuit could be. Then we find a synchronizing sequence for two of these phases (for instance, a synchronizing sequence for 00 and 0), and put this next after our sequence. Next we put on the sequence of $N_{\max}$ 0's again. There are now at most $k - 1$ phases to synchronize, and, adding on sequences for these one at a time, we eventually construct our desired universal synchronizing sequence.

The alphabetical encoding of Table I can be shown by Theorem 15 to be completely self-synchronizing, since the sequence 010001011 is a universal synchronizing sequence for this encoding. The message AD has this sequence as its enciphered form. In addition, there are many other short universal synchronizing sequences for this encoding, such as the enciphered forms of ⋕Y, AY, BD, BY, EY, HI, ID, JO, JU, MW, NY, OW, PO, PU, TY, etc. Since just these digraphs listed here occur as about three per cent of all digraphs in connected English text,[9] it can be seen that, if English text were transmitted by use of this encoding, it would be quite likely to synchronize itself very quickly.

In fact, it is easy to see that any exhaustive encoding which is completely self-synchronizing will synchronize itself with probability 1 if the messages sent have the successive letters independently chosen with any given set of probabilities, assuming only that all of these probabilities are positive numbers. This will occur since the probability of a universal synchronizing sequence occurring at any given time is positive, and, if we wait long enough, this will have happened with probability 1.

The fact that this occurs with probability 1 does not make it quite certain to occur, and, in fact, it is possible to choose arbitrarily long sequences of English words which do not contain a universal synchronizing sequence. An example of such a sequence for the alphabetical encoding of Table I is

CHECK # SYNCHRONISM # OF # LONG # FILTHY
CHUCKLE # HEH # HEH # HEH # HEH $\cdots$.

But such a sequence is extremely unlikely to continue indefinitely in any practical communication system or record-keeping system. Also, slight complications of the encoding could permit certain sequences which are certain to occur in English text (such as a period followed by a space symbol) to be universal synchronizing sequences.

One quality which might be worth comparing for various proposed encodings under consideration for possible use might be the average speed with which they synchronize themselves, when carrying typical traffic. This speed could be calculated from a sufficiently good knowledge of the statistics of the traffic, but it could more easily be measured experimentally, either by the use of actual enciphering and deciphering circuits, or by simulating their behavior on a digital computer.

The synchronization problem occurs not only when the equipment is first turned on, but also in transmission systems for which there is a noisy channel. For if some digits of a message encoded in a variable-length encoding are changed, the change may cause the circuits to get out of synchronism by the change of a short code into the prefix of a long one, or *vice versa*. Also, of course, temporary malfunctions of the enciphering or deciphering circuit themselves might cause them to get out of phase.

It may be of interest to enumerate the known results about combinations of synchronizing properties and lengths of the codes of exhaustive encodings.

If an exhaustive encoding has a fixed length (all codes having length the same integer k), then it must be

$$\text{never-self-synchronizing.} \tag{17}$$

If an exhaustive encoding has all the lengths of its codes divisible by some integer $k > 1$, but these lengths are not all equal to k, then it must be one of the following:

never-self-synchronizing, (18)

partially self-synchronizing. (19)

If an exhaustive encoding has the greatest common divisor of the lengths of its codes equal to 1, then it must be one of the following:

completely self-synchronizing, (20)

partially self-synchronizing, (21)

never-self-synchronizing. (22)

Of the above six cases, (17), (19) and (20) occur very much more commonly than the others. In fact, it is very difficult to construct examples of the other three, unless you deliberately set out to do so. The following theorems will give indications of the fact that cases (18) and (22) are hard to obtain.

Theorem 16: Given an exhaustive encoding which is never-self-synchronizing, if we let

$$Q = \sum N_i 2^{-N_i}, \tag{23}$$

then Q will always be an integer.

It can be seen that, in the case of a fixed-length code, Q will be the length. However, no one of the exhaustive encodings (except those having fixed length) listed so far in this paper has an integer value for Q. Rather than give the full details of a rigorous proof of Theorem 16, only the main ideas involved will be explained. The sum Q is the average length of the codes obtained by deciphering a presumed message, if the presumed message was obtained by choosing 0's and 1's as successive digits by independent choices having probability one-half. If we put such a random presumed message into the deciphering circuit, we have several different phases in which it may be deciphered. By the never-self-synchronizing property no two of these phases can ever come together.

Let H be the set of all prefixes of the presumed message. Then two of these prefixes will be said to be of the same phase if they are of the form θ and $\theta\Phi$, where Φ is the enciphered form of a complete message. The set H is subdivided by the equivalence relation "being of the same phase" into B distinct sets, where B is the number of phases. By symmetry, the probability that any two given members of H will be of the same phase is equal, and, since each phase occurs with equal probability and the sum of all of them is 1, each phase occurs with probability $1/B$, where B is the number of phases. However, Q was the expected difference in length between a given member of H and its next longer member; hence, we will have $Q = B$.

Theorem 17: Given an exhaustive encoding C, C is never-self-synchronizing if and only if its reversal C^* has the prefix property.

Suppose that C is not never-self-synchronizing. By the definition of synchronizing sequence, there exist finite sequences x, y and z such that x is not the enciphered form of a message, but yz is the enciphered form of message m_1 and xyz is the enciphered form of message m_2.

For some values of n the last n letters of m_1 may agree with the last n letters of m_2. But, by the fact that x is not the enciphered form of a message, there is a largest value of n for which this is true. Let this largest value be n', and let the letters which are $n' + 1$ from the end of m_1 and m_2, respectively, be called L_1 and L_2. Then $C(L_1)$ and $C(L_2)$ are both suffixes of the same message (the previous part of xyz), and hence the reversed form of one of them is a prefix of the reversed form of the other.

The converse follows more readily, since, if θ and $\theta\Phi$ are both codes of C^*, then the reversed form of θ is a synchronizing sequence for the reversed form of Φ and the null sequence.

To return to the problem of which of cases (17) through (22) can occur, it can easily be shown by the use of Theorems 16 and 17 that, among all exhaustive encodings in which not all codes are of the same length, the only ones which are never-self-synchronizing and have fewer than 16 letters in their alphabet are the encoding which encodes a nine-letter alphabet by using the list of codes (000, 0010, 0011, 01, 100, 1010, 1011, 110, 111), and the reversal of this encoding. This encoding is due to Schützenberger.[5]

This provides an example showing that case (22) can occur. That (21) can occur is shown by an encoding (derived from the above by composition) using the list of codes (000000, 0000010, 0000011, 00001, 000100, 0001010, 0001011, 000110, 000111, 0010, 0011, 01, 100, 1010, 1011, 110, 111).

It is also possible to construct an example of case (18), but the one we have found is too complicated to be worth presenting here.

IX. ONE REALIZATION FOR ENCIPHERING AND DECIPHERING CIRCUITS

Some reluctance to use variable-length encodings has been based on the opinion[10,11] that it is hard to build circuits to encipher or decipher

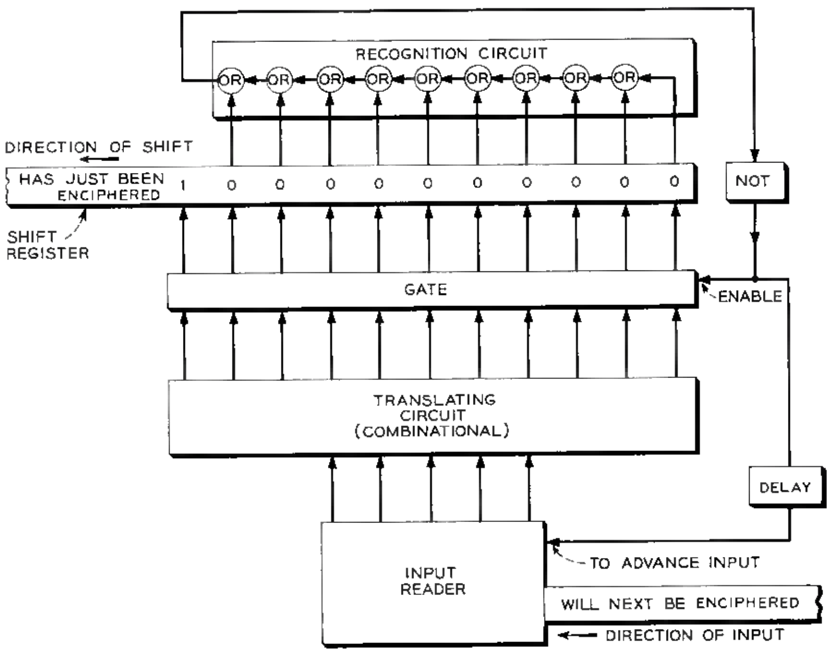


Fig. 1 — Block diagram of enciphering circuit.

them. Descriptions will be given below for one circuit for doing each of these, using principally just a shift register and a combinational translating circuit. Since using any code requires having a combinational translating circuit, and since presumably most devices using coded alphabetical information are likely to cause it to pass through a shift register, the kind of circuit described below would add very little complexity to such machines, and would automatically give them the self-synchronizing property, in the case of most variable-length binary encodings.

The enciphering circuit, shown in Fig. 1, contains a shift register containing the words "HAS JUST BEEN ENCIPHERED" followed by a binary digit 1 and a string of zeros as long as the longest code which can occur in the variable-length encoding. We will assume that it is in such a state as to have the zeros as shown, although it can easily be seen that it will get into this state if it starts in any other condition.

The circuit of Fig. 1 also contains an input reader (which can for concreteness be thought of as a punched paper tape reader, although it could be a buffer or other input device), which can read in one letter at a time whenever it is given a pulse on the lead labelled "to advance input".

The recognition circuit, which consists of a multiple-input OR circuit followed by a negation circuit, gives an output whenever there are as many binary zeros present as there are in the illustration. This sends a signal to enable the gate, letting the code corresponding to the next letter be read into the locations previously occupied by the 1 and all of the zeros. However, the translating circuit, which translates the letters into this encoding, instead of being designed to give directly the original variable-length encoding, gives an encoding which differs from it by having an extra "1" added to the end of each code. The output of the recognition circuit also goes to advance the input, reading in the next letter to be converted, after passing through a delay sufficient to be sure that the gate is now no longer enabled. This delay prevents the letter being translated from changing while it is being gated into the output shift register.

As soon as the new code has been read into the shift register, it begins to be shifted along to the left in Fig. 1. The 1 at the end of the code serves to mark the end of the code during this shifting, but it will be eliminated from the enciphered form of the message. The shift register is connected so that, when it is shifted, a 0 appears at the right end. As soon as the 1 passes beyond the end of the recognition circuit, there will be only zeros present, and hence the recognition circuit will again recognize the end of a letter and repeat the cycle as given above.

Instead of having a counter or a special sequential circuit to keep track of where the current letter ends, this has been done here by adding a single binary digit to the code and adding one to the length of the required shift register.

Similarly, an analogous scheme can be used to decipher from a variable length code into any other representation for letters, by using one special position in the shift register, as shown in Fig. 2. This deciphering circuit can be built only for encodings having the finite delay property, although the enciphering circuit of Fig. 1 can be used for any binary encoding.

The shift register into which the digits to be deciphered are shifted is divided into two halves, which will be called the left half and the right half. The right half has e digit positions, where e is the excess delay of the encoding. The left half has $N_{\max} + 1$ digit positions, with the extra 1 being used to mark the end of those digits which already have been deciphered.

At the beginning of the cycle we will assume that the left half of the shift register has just been cleared to the state shown in Fig. 2, that is, to contain $N_{\max}$ 0's followed by a 1. Next, the digits of the message to be enciphered shift toward the left. Since the 1 precedes them, it marks clearly how many of these digits have been shifted into the left half. As soon as all of the digits of the code of the first letter of the message have been shifted into the left half, the translating circuit will then give its outputs. It gives the translated codes for the letter, as well as giving another output, w, which equals 1 only when the complete first letter is present. The translating circuit makes use of the inputs from only the left half of the shift register, ignoring the digits in the right half, unless the code C present in the left half is a code which is also a prefix of another code. It makes use only of those digits from the right half which are necessary to distinguish between this code and the partially shifted-in code of which it is a prefix. It gives the output $w = 1$ whenever the entire code for the first letter of the enciphered message has been shifted over into the left half and, whenever only a prefix of the code of the first letter is there, the output w will equal zero.

This output w will then cause the left half to clear back to its original state, and, after a delay sufficient to allow the output to be received, it gives the "to advance output" signal to the output punch or buffer.

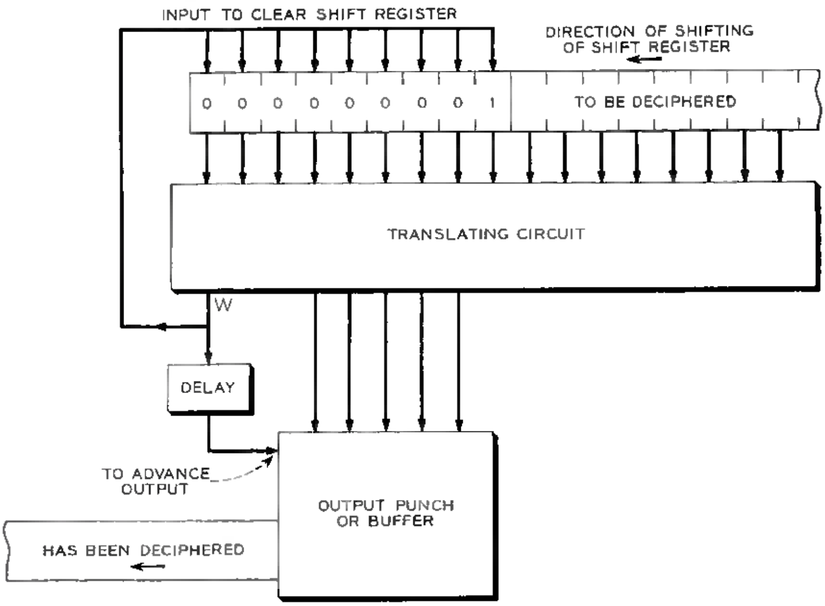


Fig. 2 — Block diagram of deciphering circuit.

The deciphering circuit then repeats the above cycle for the next letter of the message.

The translating circuit of this deciphering circuit must give the appropriate outputs whenever the complete code for the first letter is present in the left half of the shift register, and must give $w = 1$ in these cases. It must also be designed to give the output $w = 0$ whenever an incomplete prefix of the first letter is present, but, since in general there may be many states of the shift register which do not correspond to either a letter or a prefix, there may be many "don't cares" occurring in the design of this translating circuit, which will permit it to be simpler than a completely specified function having this many inputs.

The time delay between the receipt of the beginning of an N-digit code for a letter and the actual sending of this letter to the output punch or buffer will be $N + e$, which may sometimes be slightly longer than the delay d of the message. However, the circuit for doing the deciphering in the minimum time would be more complicated, in that it would not always clear the shift register to the same state, so it is not presented here.

However, in the enciphering circuit given in Fig. 1 there is only a delay of one digit time, while the message is shifted through the one extra stage at the left end of the shift register. Hence, neither of these two circuits operates in quite the minimum possible time, since speed has been sacrificed for simplicity of construction.

X. FURTHER PROBLEMS

There are many further problems suggested by the ideas discussed in this paper, and which we have not been able to solve. Are there any binary encodings which satisfy (9) other than the exhaustive encodings and their reversals? Are there any encodings C which satisfy (9) and such that both C and its reversal C^* have the finite delay property without both C and C^* having the prefix property? Given an encoding which is uniquely decipherable but which does not possess the finite delay property, does the set of presumed messages having infinite delay always form a finite set? Does it always form a set of measure zero? Is there a simple polynomial in $N_{\max}$ and n which will be an upper bound to the delay of any encoding having the finite delay property? Are the encodings for which the algorithm of Sardinas and Patterson[2] fails to terminate precisely the same as the encodings having infinite delay? Given any encoding having infinite delay, is there a Turing machine (perhaps having several tapes and several reading and writing heads on each) which can decipher any K-digit message in a length of time which is less than a constant times K?

XI. ACKNOWLEDGMENTS

We would like to express our thanks to T. H. Crowley for suggesting Theorem 7, and to S. P. Lloyd for suggesting to us some of the complications of encodings which do not have the prefix property.

REFERENCES

1. Huffman, D. A., A Method for the Construction of Minimum-Redundancy Codes, Proc. I.R.E., **40,** September 1952, p. 1098.
2. Sardinas, A. A. and Patterson, G. W., A Necessary and Sufficient Condition for Unique Decomposition of Encoded Messages, I.R.E. Conv. Rec., 1953, Part 8, p. 104.
3. McMillan, B., Two Inequalities Implied by Unique Decipherability, I.R.E. Trans., **IT-2,** December 1956, p. 115.
4. Mandelbrot, B., On Recurrent Noise Limiting Coding, Proceedings of the Symposium on Information Networks, Polytechnic Institute of Brooklyn, April 1954, p. 205.
5. Schützenberger, M. P., On an Application of Semi-Groups Methods to Some Problems in Coding, I.R.E. Trans, **IT-2**, September 1956, p. 47.
6. Michel, W. S., Fleckenstein, W. O. and Kretzmer, E. R., A Coded Facsimile System, I.R.E.-Wescon Conv. Rec., 1957, Part 2, p. 84.
7. Dewey, G., *Relativ Frequency of English Speech Sounds*, Cambridge Univ. Press, Cambridge, Eng., 1923, p. 185.
8. Shannon, C. E., A Mathematical Theory of Communication, B.S.T.J., **27,** July 1948, p. 379; October 1948, p. 623.
9. Pratt, F., *Secret and Urgent*, Doubleday & Co., Garden City, N. Y., 1939.
10. Brooks, F. P., Ph.D. thesis, Harvard Univ., May 1956.
11. Brooks, F. P., Multi-Case Binary Codes for Non-Uniform Character Distributions, I.R.E. Conv. Rec., 1957, Part 2, p. 63.

Recurrent Codes: Easily Mechanized, Burst-Correcting, Binary Codes

By D. W. HAGELBARGER

(Manuscript received April 14, 1959)

A class of codes capable of correcting multiple errors is described. Some of these codes can be implemented with considerably less hardware than was needed for previous multiple error-correcting codes. A general method is shown for constructing a code of redundancy 1/b that will correct error bursts of Kb or fewer digits (K and b integers). The logical design of the encoder and decoder, as well as the guard space requirement of good digits between bursts of errors, is described.

I. INTRODUCTION

In adapting the existing telephone network to high-speed digital data transmission, an error control problem arises. Most of the circuits were designed primarily for voice-type signals and considerable attention was given to control of thermal noise. Because of the high redundancy of speech, impulse noise on the lines is usually not even noticed by the telephone users, and hence has not been a serious problem. On the other hand, high-speed digital data (especially numerical data) contain little redundancy, and the noise pulses may resemble the signal pulses and thus cause errors.

The deliberate introduction of redundancy to detect and correct transmission errors has been used for some time. Early systems[1] used repetition of characters and duplication of channels. There were two schemes which sent pictures of the characters using raster scans. By the late 1930's a radio telegraph system[2] using a 3-out-of-7 code for error detection had been patented and telephone apparatus using 2-out-of-5 codes[3] was being designed.

Most, if not all, of the recent work on error-detecting or error-correcting codes stems from Hamming's Systematic Parity Check codes.[4] These codes will correct a single error per block of digits. Since then, much work has been done on codes for multiple errors (see Refs. 5 through 16). The assumption has usually been made that the errors are statistically independent. On many communication channels, however, the errors are not independent but tend to come in groups. For example, a lightning stroke may knock out several adjacent telegraph pulses. These groups of errors are called "bursts". Codes for detecting and correcting bursts have been proposed by Abramson,[17] Gilbert,[18] Hamming[13] and Meyer.[13] The class of codes described here differs in that the block structure has been minimized; the resulting symmetry allows very simple mechanization and also simplifies the synchronization problem. Since the block size is small, the codes also fit very naturally into systems where the data must be accepted and delivered continuously, rather than in batches.

II. EXAMPLE

Before describing the general recurrent code we will give a particularly simple example. Assume that we wish to correct bursts of length six or less. The simple code has every other digit a check digit, giving a redundancy of one-half. The encoder is illustrated in Fig. 1. It consists of a shift register of length seven. The data digits enter from the left (Position 1) and are shifted through the register before being transmitted. For each shift we generate a check digit so that the parity (number of 1's) of the check digit and the data digits in the first and fourth positions of the shift register is even (zero or two). This check digit is transmitted before the data digit in the seventh position. Then a shift is made; the data digit which was in Position 7 is transmitted, and a new check digit is calculated. This process is illustrated in Fig. 2; the successive lines are one shift time apart. During this time interval one new data digit is accepted by the encoder and two digits, one data, one check, are transmitted.

The decoder is shown in Fig. 3. The received code enters the switch where the alternating data and check digits are separated, the check digits going to the lower shift register and the data digits to the upper

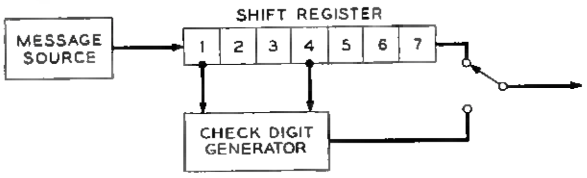


Fig. 1 — Encoder.

TIME	DATA BITS	ENCODER SHIFT REG $D_1 + D_4 \equiv C$ (MOD 2) 1	2	3	4	5	6	7	C	TRANSMITTED CODE
0	0 1 1 0	0	1	1	0	1	0	0	0	
1	0 1 1	0	0	1	1	0	1	0	1	0 0
2	0 1	1	0	0	1	1	0	1	0	0 1 0 0
3	0	1	1	0	0	1	1	0	1	1 0 0 1 0 0
4		0	1	1	0	0	1	1	0	0 1 1 0 0 1 0 0

Fig. 2 — Timing chart of digits moving through the encoder.

one. There are two copies of the parity circuits, R and S, each one checking the parity relation imposed by the encoder. The decoding rule is:

> Whenever both R and S fail, change the data digit in Position 4 ($0 \rightarrow 1$, $1 \rightarrow 0$) while shifting it to Position 5. If only one parity check fails, make no change.

The corrected data digits are available at Position 5 of the data shift register. In a burst of length six or less, there can be at most three data digits and three check digits wrong. The parity relation used in encoding involves digits which are spread far enough apart so that no burst of six or less will affect more than a single digit in any one parity group.

After any burst of length six or less, a 20-digit errorless message is enough to completely refill the decoder shift registers. Hence, the next burst can be corrected without interference from a previous burst, if there is a "clean" message 19 or more digits long between bursts.

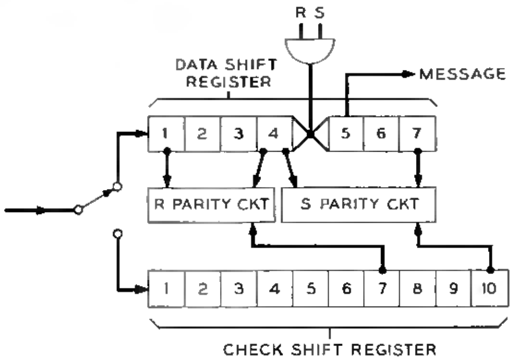


Fig. 3 — Decoder.

TABLE I

Burst Length	Data Shift Register Length	Check Shift Register Length	Guard Space Between Bursts
4	5	7	13
6	7	10	19
8	9	13	25
10	11	16	31

If the message is to be retransmitted the check digits can be corrected at Position 10 of the check digit shift register (Fig. 3). The rule is:

Whenever parity check R holds and parity check S fails, change the digit in Position 10 of the check digit register.

A data digit error cannot cause parity R to hold and parity S to fail, because this would require an error in Position 7 of the data register, and, since all data digit errors are corrected in going from Position 4 to Position 5, this cannot happen.

This particular coding scheme fails first for a burst of length seven, consisting of a check digit and another check digit six digits later. When just these two check digits are wrong, the decoder assumes that a data digit is wrong and changes it.

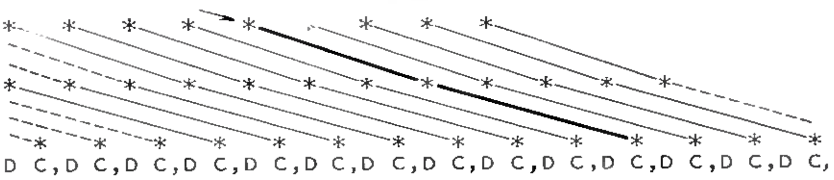
A demonstration device using this code has been built. It consists of a punched tape reader, an encoder, a transmission line, a decoder and a tape printer. The circuitry uses relays and can be operated fast, slow or one step at a time. Digits in the encoder, transmission line and decoder are displayed on lamps. The transmission line has switches for inserting errors in the encoded message. Other lamps are used to indicate parity check failures. An auxiliary circuit can be used for detecting overlong bursts, flashing a yellow lamp whenever the decoder has a burst and locking up a red lamp whenever a detectable overlong burst occurs.

There is an extension of this code to correct bursts of any even length. We merely spread out the parity check so that the burst can only effect one term in any parity group. To correct bursts of length $2K$ or less requires an encoding shift register of length $2K + 1$. The decoding data digit register has the same length, $2K + 1$, and the check digit register must be $3K + 1$. The parity checks involve digits K apart in the registers. A clean message of length $6K + 1$ will always be sufficient separation between bursts. Table I shows typical values.

III. GENERAL RECURRENT CODE

The general (binary) recurrent code is constructed as follows:

We are given a *message* to be transmitted consisting of *data* digits.


D C, D C, D C, D C, D C, D C, D C, D C, D C, D C, D C, D C, D C, D C, D C,

$P_1 = P_D$	1	0	0	1	0	0	0
$P_2 = P_C$	0	0	0	0	0	0	1

Fig. 4 — Portion of encoded message.

We will add to this *check* digits to form the *encoded message.* The encoded message is divided into blocks of length b. One position in the block is assigned to be a check digit† and the rest are data digits. Since the block must have at least one data digit, the shortest block length is $b = 2$. (This is the value used in the example of Section II.) The data digits are loaded into the data digit positions in the order received. The check digit is determined by a parity relation applied once for each block. This parity relation extends over a selected set of the digits in p consecutive blocks. (To be useful against bursts, p must be at least 2 and usually will be larger.) Fig. 4 shows a portion of the encoded message from the example of Section II. The data and check bits are indicated by D's and C's; the blocks are marked off with commas and the parity relation is shown by lines having *'s over the digits in a given *parity group*. Note that p for this example is 7; that is, any one of the parity groups extends over seven blocks. Every parity group has three digits in it, two data and one check; thus, each data digit is in two parity groups and each check digit is in only one parity group.

We will denote which digits enter into the parity relation by b binary words of p digits each. We call these the *parity words* and label them $P_1, P_2, \cdots, P_b$. Consider p consecutive blocks. We form P_1 by observing the first position of each block; if the digit is in this parity group we write 1, otherwise 0. Then P_2 depends on the second positions of these p blocks, and so on. This is illustrated in Fig. 4. We have used the parity group indicated by the arrow and written the parity words so that the digits fall under the corresponding blocks. Thus P_1 has 1's in the first and fourth blocks and P_2 has a 1 only in the seventh block. (The numbering of digits and blocks here and in Fig. 4 is from right to left. Fig. 4 can be considered a "snapshot" of the encoded message on the trans-

† Codes with more than one check digit per block are possible, but it can be shown that, for a given efficiency and burst-correcting capability, they always require more complicated encoding and decoding equipment than would the equivalent code with one check digit per block.

mission line. This is different from the numbering method used below, where the digits are considered to be flowing past a fixed point, and are labeled with numbers which increase with time.) With this notation, there is a simple correspondence between the 1's in the parity words and the connections to the shift registers of the encoder and decoder.

Another way to describe these codes is to think of the digits leaving the encoder as being numbered serially; if X_i is the ith digit, then the parity relation is given by a recurrent equation of the form:

$$X_{r+kb} \oplus X_{s+kb} \oplus \cdots \oplus X_{w+kb} = \text{constant},$$

where the constant is 0 or 1, $\oplus$ means sum modulo 2, k takes successive integral values, b is the block length, and r, s, $\cdots$, w denote which digits enter into the parity group.

If we order the terms so that $r < s < \cdots < w$, then

$$p - 2 < \frac{w - r}{b} < p.$$

The requirements that every position in the block must be represented at least once implies that each of the integers 0, 1, $\cdots$, $(b - 1)$ must occur at least once as a remainder upon dividing r, s, $\cdots$, w by b.

For the example of Section II we have:

$$X_{1+2k} \oplus X_{8+2k} \oplus X_{14+2k} = 0.$$

IV. BURSTS

Consider an encoded message flowing through a communication channel. A burst occurs and some of the digits of the message are changed. We will describe the burst pattern by a binary word having a 1 for each changed digit and a 0 for each correct digit. We require that the first and last digits of the word both be 1's, since there is no point in including correct digits which are outside of the bursts. The length of the burst is the number of digits in the word. There are 2^{l-2} different burst patterns of length l and 2^{l-1} different burst patterns of length l or less. The latter are the odd binary numbers having l or fewer digits. For example, the eight burst patterns of length 4 or less are: 1, 11, 101, 111, 1001, 1011, 1101 and 1111.

The effect of a burst on the encoded message depends on the phase of the burst pattern with respect to the block structure; hence, we will have to consider $b2^{l-1}$ different possible bursts of length l or less if the code has a block length b. We will indicate particular bursts either by showing a portion of the encoded message with the erroneous digits marked with *'s or by listing the erroneous digits with superscripts to indicate which block a particular digit occupies. For example,

$$\cdots DC,\ \overset{*}{D}\overset{*}{C},\ D\overset{*}{C},\ \overset{*}{D}C,\ \cdots$$

is the same as $D^2C^2C^1D^0$.

V. SYNDROMES

At the decoder we will always have a circuit for checking the parity relation imposed by the encoder. This circuit gives an output once for each block received. If the parity check fails, the output is a 1; otherwise, it is a 0. In a practical transmission system, there are usually no errors and the check circuit has an output of all 0's. When a burst of errors does occur, the check circuit will give a pattern of 1's and 0's. This pattern will be used to identify the burst, and hence we shall call it the *syndrome*. Since the syndromes occur immersed in a string of 0's, only binary words having 1's on both ends (odd numbers) can be used as syndromes and, as above, there are 2^{k-1} possible different syndromes with k or fewer digits.

Our first problem is to choose the parity relation in such a fashion that each burst of length l or less has a distinct syndrome.† A further problem is to choose the parity relation so that, given a distinct syndrome, the correction of the burst which caused it is easily mechanized. That is, we want a systematic scheme for correcting a burst, given the corresponding syndrome, which is much better than having a table of all possible syndromes with the corresponding bursts.

The procedure for calculating the syndrome corresponding to a given burst is as follows:

$$\cdots D\ C, D\ C, D_3\overset{*}{C}, \overset{*}{D}_2C, \overset{*}{D}_1\overset{*}{C}, \overset{*}{D}_0C, D\ C, D\ C,\ \cdots$$

$$\begin{array}{lcccccccccc}
P_D{}^0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & & & \\
P_C{}^1 & & 0 & 0 & 0 & 0 & 0 & 0 & 1 & & \\
P_D{}^1 & & 1 & 0 & 0 & 1 & 0 & 0 & 0 & & \\
P_D{}^2 & & & 1 & 0 & 0 & 1 & 0 & 0 & 0 & \\
P_C{}^3 & & & & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
\text{Syndrome:} & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1
\end{array}$$

† If our code has a block length b which is a power of 2, then it is possible for $b2^{l-1} = 2^{k-1}$. A *close-packed* recurrent code is one which has all possible syndromes of k or fewer digits in a one-to-one correspondence with all possible bursts of length l or less. Since this is of more mathematical than practical interest, examples are deferred to Appendix I.

Number the blocks, starting with 0 for the first block containing an error, and continue the numbering far enough to include all blocks having errors in the burst under consideration. (The direction of numbering should be such that increasing numbers represent digits received at later times.) Write the parity word for the error in block number 0. Under this write the parity words for any other errors in this block. Now write the parity words for the other errors, shifting each one (in the direction of increasing time) the number of places equal to the block number in which the error occurs. The syndrome is the sum modulo 2 of these parity words. In the sample above we use superscripts on the parity words to indicate in which block each error occurred. This shows the syndrome for the indicated burst of length 6, using the code of Section II. Note that there is never more than a single 1 in any column. By spreading out the parity words with 0's sufficiently to prevent any interaction of errors (in a burst not larger than the design maximum), we allow the use of a simple circuit for recognizing the parity words and correcting the errors one at a time. In other words, the decoder for our example of Section II is simple because we have a single circuit for correcting a data-digit error, which is time-shared by every data digit. Each data digit is compared with four other digits, all far enough away from each other so that not more than one of these five digits can be in error due to an allowable burst. Under these circumstances, it is an easy matter to decide if the particular data digit needs correcting.

VI. LOWER REDUNDANCY CODES

We can apply the same technique, spreading out the parity words with 0's so as to avoid interactions between errors in a burst, to make the redundancy as low as we wish and still have a relatively simple correcting mechanism. For instance, if we desire a code with a redundancy of one-quarter good for bursts of length 4, we choose the following parity words (the digits are spaced to emphasize the method of forming):

P_A	111	000	000	0
P_B	000	101	000	0
P_C	000	000	110	0
P_D	000	000	000	1

Note that placing the code 100 to the extreme right allowed us to shorten the parity words by dropping the last two columns, which were all 0's. In assigning the parity words, the groups of 1's should be arranged to go from upper left to lower right as shown above. That is, the order of the groups of 1's in the parity words should agree with the order of the digits in the block structure. If this is not done extra columns of 0's must be inserted in the array of parity words, which would mean more shift register stages in the encoder and decoder. The difficulty is illustrated as follows:

	Code I	*Code II*
P_A	111 000 000 0	000 111 000 0
P_B	000 101 000 0	101 000 000 0

Burst: $\cdots A\overset{*}{B}CD\overset{*}{A}B \cdots$

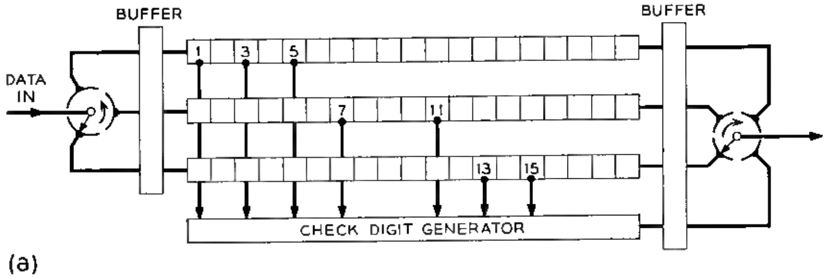
$P_A{}^0$	1110000000	0001110000
$P_B{}^1$	0001010000	1010000000
Syndrome:	1110101	10011
	[O.K.]	[N.G.]

(The burst $\cdots \overset{*}{A}BCDA\overset{*}{B} \cdots$ is not allowed, since the above codes are for bursts of length 4 or less.) If we wish to make a code of the same redundancy good for bursts of length 8 or less, we form the parity words by inserting 0's between each of the digits of the above code, giving:

P_A	10101	000000	000000	00
P_B	00000	010001	000000	00
P_C	00000	000000	010100	00
P_D	00000	000000	000000	01

Fig. 5(a) shows an encoder for this code. The data digits enter from the left side and are cyclically switched to the three shift registers by the input commutator. The buffers allow the shift registers to be stepped together. A check digit, calculated from the parity of the indicated positions of the shift registers is transmitted with the digits from the last positions of the registers by the output commutator. Note the correspondence between the parity words and the shift registers. The top register comes from P_A, the second from P_B, and so on. Each digit of a parity word becomes a stage of shift register. The stages representing 1's are connected to the parity circuit; those representing 0's are not. If P_D, which has a single 1 at the right end, is assigned to the check digit, no shift register is required at the encoder for this parity word. (However, one is needed in the decoder.)

At the decoder, Fig. 5(b), the incoming digits are commutated to the four shift registers (synchronization must be maintained so that the digits get in the proper registers.) As each block arrives at the decoder, the parity relation is checked; if it fails, a 1 is put in the syndrome register at the bottom of the figure. Suppose that a digit in the top register is wrong; it will cause parity failures as it goes through Positions 1, 3 and 5. When it is in Position 5 the syndrome register will have 1's in Positions R, S and T. This will enable the AND circuit, which will correct the error as it shifts from Position 5 to Position 6. In a similar manner, an error in the second register is corrected between Positions 11 and 12, and an error in the third register between Positions 17 and 18. Whenever a 1 reaches Position T of the syndrome register, any 1's in Positions R and S are cleared on the next shift. (If for some reason it should be desirable to correct the check digits rather than discard them, this can be done by adding the extension to the bottom register shown dotted.) Because of the way the taps for the parity circuit are spaced,



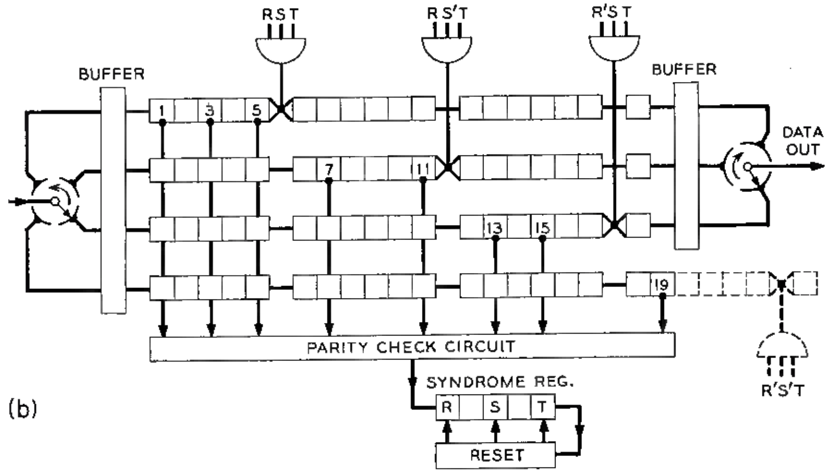


Fig. 5 — (a) Encoder; (b) decoder.

any register can correct two adjacent errors, and the system is good against bursts of length 8 or less.

It takes 92 digits entering the decoder to completely refill all the registers (including the syndrome register). Thus, a guard space of 91 good digits between bursts is sufficient to assure that there is no interaction between bursts.

In general, a procedure for constructing a code of redundancy $1/b$ (block length b) is as follows:

> Take the first b binary numbers and let L be the number of digits in the largest one. Form each of these numbers to a L-digit word by adding zeros to the *right* end. Now form a square array with b rows and b columns. The entries in the array are L-digit words. Put the above-formed words along the main diagonal, with the word having a single 1 going in the lower right corner. The order of the other words on the diagonal is arbitrary. Fill in all remaining words with zeros. Now replace the right-hand column of L-digit words with single-digit words; 1 in the bottom row, 0 elsewhere. (Strike out the $(L - 1)$-digit columns from the right.)

The rows of this array are the parity words of the desired code. The order from top to bottom is the ("snapshot") order of occurrence of the corresponding digits in the block structure of the message.* This code will correct all bursts of length b or less. To make a code good for burst of length Kb or less, add $K - 1$ zeros between each adjacent pair of digits of the above parity words.

If the odd binary numbers were not increased to L digits as above, certain otherwise allowable bursts could cause syndromes which would be incorrectly interpreted by the decoder. For instance, 001 and 110 might add to form 001110. The procedure given prevents this type of difficulty.

VII. SHIFT REGISTER AND GUARD SPACE REQUIREMENTS

If we design codes by the method indicated in the previous section, the method is regular enough to allow us to give formulas for the shift register stages and guard space.†

Definitions:

$$b,\ l,\ k,\ L(b) \text{ are positive integers.}$$

* The bottom row is the parity word for the digit that is transmitted first in any block. See the direction of rotation of the output commutator in Fig. 5(a).

† If the block length is not a power of 2 it may be possible to save a little from these calculations by taking advantage of the fact that one or more of the odd numbers is not used. For example, the burst-length 3, redundancy one-third decoder shift registers can be reduced from 24 to 21 stages.

The block length is b with one check digit; hence, the redundancy is $1/b$. The code is to be usable against bursts of length l or less, where $l = kb$. $L(b)$ is the smallest integer such that

$$L(b) \geqq 1 + \log_2 b.$$

Thus,

b	2	3	4	5	6	7	8	9	10
$L(b)$	2	3	3	4	4	4	4	5	5

The encoder will have $b - 1$ registers, each of length

$$\left(\frac{l}{b}\right) L(b)(b - 1) + 1$$

or

$$(b - 1)^2 \left(\frac{l}{b}\right) L(b) + b - 1$$

stages. The decoder will have

$$l(b - 1)L(b) + 2b + \left(\frac{l}{b}\right) L(b) - l$$

stages.

The guard space can be shown to be

$$blL(b) + b - l - 1.$$

Table II shows some typical values. (Efficiency is 1-redundancy.)

VIII. DETECTION OF OVERLONG BURSTS

Any error-correcting system fails when the errors get beyond its correcting capabilities. In some cases, it is desirable to detect that a burst has occurred which the system cannot correct. In Section II we mentioned that the code for that example failed on a burst consisting of two check-digit errors exactly six apart. The effect of this burst is to cause the decoder to change the data digit which is common to the parity groups containing these check digits. This is a fundamental difficulty of the particular code and cannot be avoided, since such a burst converts our encoded message to what looks like a different encoded message with a single data-digit error. In general, we cannot correct or even detect bursts which convert one encoded message to another encoded message, or to another encoded message modified by an allowable burst (assuming we still wish to correct allowable bursts). For the example of Section II there are:

four bursts of length 9,
two bursts of length 8,
one burst of length 7,

which convert one encoded message to another encoded message modified by an allowable burst. These seven bursts are not detectable, but all other bursts of length 9 or less are detectable and, of course, all bursts of length 6 or less are correctable. In connection with the code demonstrator described in Section II, a circuit which detects overlong bursts by monitoring the sequence of failures of the two parity circuits in Fig. 3 has been built. All 512 bursts of length 9 or less have been tried on it, and it rings the alarm on all the uncorrected bursts except the seven listed above.

In the code of Section II the syndrome for a single data error is the same as the syndrome for a pair of check-digit errors six apart. To improve the error-detection capabilities of our code, we can always change the parity words so that the first occurrence of two bursts giving the same syndrome involves bursts longer than the correctable one.

As an example, consider the code given by the parity words

$$P_D = 1001001000000,$$

$$P_C = 0000000001001.$$

This particular code permits a very simple overlong-burst-detection scheme. It corrects or detects all bursts of length 13 or less and corrects all bursts of length 6 or less. The encoder and decoder are shown in

TABLE II

Block	Efficiency (in per cent)	Burst Length	Shift Register Stages		Guard Space
			Encoder	Decoder†	
2	50	2	3	8	7
		4	5	12	13
		6	7	16	19
		8	9	20	25
		10	11	24	31
3	67	3	14	24	26
		6	26	42	50
		9	38	60	74
4	75	4	30	43	47
		8	57	78	91
5	80	5	68	89	99
		10	132	168	194

† The decoder here is different from the one for Table I; this one has a syndrome register.

Fig. 6. Note that, in the process of correcting an allowable burst, the three parity-check circuits R, S and T cannot have either of the failure patterns 010 or 101. It happens that one or the other of these patterns occurs for every burst of length 13 or less that is not corrected by the decoder.

The decoder here shows an alternative arrangement compared to Fig. 5. Instead of the syndrome shift register, we have three copies of the parity check circuit shifted so that Position 7 of the data register is the only one common to all three. If we were to make a circuit similar to Fig. 5, we would save the S and T parity circuits, the last five stages of the data register and the last six stages of the check register in exchange for a seven-stage syndrome register with its reset circuit.

IX. SYNCHRONIZATION

In general, there are two synchronization problems with any binary code, bit synchronization and block synchronization. For purposes of

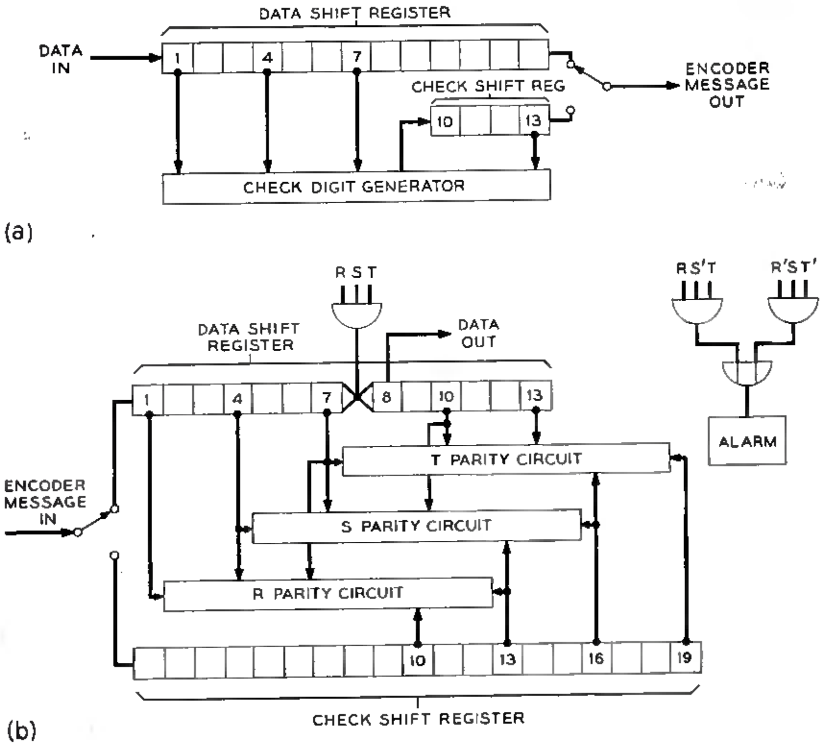


Fig. 6 — (a) Encoder; (b) decoder.

Table III — Impossible Codes

		Number of Terms	
		ODD	EVEN
Number of Ones	ODD	00000···	0000···, 11111···
	EVEN	11111···	

bit synchronization, it is desirable to have transitions from 0 to 1 or 1 to 0 at some minimum rate. By proper choice of the number of terms in the parity relation and also whether the parity is odd or even we can prevent either of the codes 00000··· or 11111··· from occurring in the encoded message, regardless of what the input to the encoder may be. It is probably also desirable to exclude these codes as possible encoded messages since they are the most likely messages to be put out by an encoder with something stuck on or off. Table III shows which codes cannot occur.

Table IV

Redundancy	Burst	Syndrome
$\frac{1}{2}$	A^0	111
	B^0	001
	A^0B^0	110
	B^1A^0	101
$\frac{1}{4}$	A^0	1100
	B^0	0111
	C^0	1101
	D^0	1001
	A^0B^0	1011
	B^0C^0	1010
	C^0D^0	0100
	D^1A^0	1111
$\frac{1}{8}$	A^0	01100
	B^0	10111
	C^0	11100
	D^0	01101
	E^0	10010
	F^0	11101
	G^0	11001
	H^0	10011
	A^0B^0	11011
	B^0C^0	01011
	C^0D^0	10001
	D^0E^0	11111
	E^0F^0	01111
	F^0G^0	00100
	G^0H^0	01010
	H^1A^0	10101

With the redundancy one-half code, the block synchronization problem is minimized, since there are only two phases that the decoder can have. One possibility is to make a decoder with two equal shift registers and two copies of the error-correcting circuit, one wired in each of the possible phases. The fact that the wrong parity circuits fail half of the time can be used to tell which phase to use.

X. ACKNOWLEDGMENT

I wish to thank E. F. Moore for many helpful suggestions. R. V. Anderson built the demonstration device described in Section II.

APPENDIX

Examples of Close-Packed Codes

All of the examples known to date are for burst length two. It is not too hard to show that there is no close-packed code of redundancy one-half good for burst length three (see Table IV).

REFERENCES

1. Storch, P., Evolution of Long Distance Type-Printing Traffic by Wire and Radio, Elek. Z., **55,** 1934, pp. 109; 141.
2. Moore, J. B. and Mathes, R. E., U. S. Patent 2,183,147.
3. Holbrook, B. D., U. S. Patent 2,317,191.
4. Hamming, R. W., Error Detecting and Error Correcting Codes, B.S.T.J., **29,** 1950, p. 147.
5. Plotkin, M., Binary Codes with Specified Minimum Distance, Research Div. Rep. 51-20, Moore School of Electrical Engineering, Univ. of Pennsylvania, 1951.
6. Golay, M. J. E., Binary Coding, I.R.E. Trans., **PGIT-4,** 1954, p. 23.
7. Elias, P., Error-Free Coding, I.R.E. Trans., **PGIT-4,** 1954, p. 29; Coding for Noisy Channels, I.R.E. Conv. Rec., 1955, Part 4, p. 37.
8. Muller, D. E., Application of Boolean Algebra to Switching Circuit Design and to Error Detection, I.R.E. Trans., **EC-3,** 1954, p. 6.
9. Reed, I. S., A Class of Multiple-Error-Correcting Codes and the Decoding Scheme, I.R.E. Trans., **PGIT-4,** 1954, p. 38.
10. Slepian, D., A Class of Binary Signalling Alphabets, B.S.T.J., **35,** 1956, p. 203.
11. Lloyd, S. P., Binary Block Coding, B.S.T.J., **36,** 1957, p. 517.
12. Ulrich, W., Non-Binary Error Correction Codes, B.S.T.J., **36,** 1957, p. 1341.
13. Brown, A. B. and Meyers, S. T., Evaluation of Some Error Correction Methods Applicable to Digital Data Transmission, I.R.E. Conv. Rec., 1958, Part 4, p. 37.
14. Green, J. H., Jr., and SanSoucie, R. L., An Error-Correcting Encoder and Decoder of High Efficiency, Proc. I.R.E., **46,** 1958, p. 1741.
15. Kautz, W. H., A Class of Multiple-Error-Correcting Codes, Stanford Research Institute, 1958.
16. Sacks, G. E., Multiple Error Correction by Means of Parity Checks, I.R.E. Trans., **IT-4,** 1958, p. 145.
17. Abramson, N. M., A Class of Systematic Codes for Nonindependent Errors, Stanford Electronics Labs., Tech. Rep. No. 51, 1958.
18. Gilbert, E. N., A Problem in Binary Coding, Symposium on Combinatorial Designs and Analysis, Amer. Math. Soc., 1958 (to be published).

Representation of Switching Circuits by Binary-Decision Programs

By C. Y. LEE

(Manuscript received May 8, 1958)

A binary-decision program is a program consisting of a string of two-address conditional transfer instructions. The paper shows the relationship between switching circuits and binary-decision programs and gives a set of simple rules by which one can transform binary-decision programs to switching circuits. It then shows that, in regard to the computation of switching functions, binary-decision programming representation is superior to the usual Boolean representation.

I. INTRODUCTION

In his 1938 paper,[1] Shannon showed how relay switching circuits can be represented by the language of symbolic logic and designed and manipulated according to the rules of Boolean algebra. This far-reaching step provided an algebraic language for a systematic treatment of switching and logical design problems and provided a root system from which new art can grow and flourish.

We may want to know, however, if there might not be other ways of representing switching functions and circuits, and to compare such representations with the algebraic representation of Shannon. In this paper we will give a new representation of switching circuits, and will call this representation a "binary-decision program."

Binary-decision programs, as the reader will see, are not algebraic in nature. They are, therefore, less easily manipulated. A switching circuit may be simplified not by simplifying its binary-decision program, but by essentially finding for it a better binary-decision program. A good binary-decision program generally means one which is well-knit and makes efficient use of subroutines; it is good in the sense then that a computer program is good. Binary-decision programs do not seek out series-parallel circuits, but are more suited for representing circuits with a large number of transfers. In these respects, binary decision programs therefore differ very greatly from the usual Boolean representation.

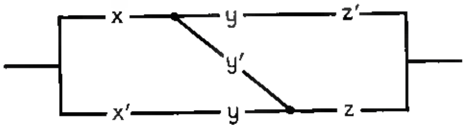


Fig. 1 — Typical switching circuit.

The characteristic which sets binary decision programs still further apart from Boolean representation and gave this study its initial stimulation is in the computation of switching functions. It is here that we will give direct evidence of the superiority of binary-decision programs.

II. STRUCTURE OF BINARY-DECISION PROGRAMS

A binary decision program is based on a single instruction

$$T \quad x;\, A,\, B.$$

This instruction says that, if the variable x is 0, take the next instruction from program address A, and if x is 1, take the next instruction from address B. Every binary-decision program is made up of a sequence of instructions of this kind.

Take, for example, the switching circuit shown in Fig. 1. This circuit is described exactly by the following binary-decision program:

1. $T \quad x;\, 2,\, 4.$
2. $T \quad y;\, \theta,\, 3.$
3. $T \quad z;\, \theta,\, I.$
4. $T \quad y;\, 3,\, 5.$
5. $T \quad z;\, I,\, \theta.$

The program is actually a sequential description of the possible events that may occur. We begin at program address 1 by examining the variable x. If x should be 0, we go to address 2 and examine y. If y is 0, we go to address θ; otherwise we go to address 3, and so forth. The symbols θ and I indicate whether the circuit output is 0 or 1. From a computer viewpoint, they can be the exit addresses once the circuit output value is known.

III. CONSTRUCTION OF SWITCHING CIRCUITS FROM BINARY-DECISION PROGRAMS

The question that we will consider here is this: Suppose the logical requirements of a switching circuit are given, when would it be possible to design the circuit according to the following procedure:

$$\begin{tikzcd} \text{Logical requirements} \arrow[r] & \text{Binary decision program} \arrow[r] & \text{Switching circuit} \end{tikzcd} \ ?$$

In various examples we have tried, this approach has given us a fresher look at things and, in several instances, has given us rather good circuits. The process of going from binary-decision programs to switching circuits is very well defined, so that how good a circuit we get depends entirely upon how good a binary-decision program we can write. Roughly speaking, if a problem has a fairly sizable set of logical requirements to begin with, it would call for a well-organized array of subroutines in the binary-decision program, which are called in as the need arises. Nevertheless, there are many exceptions, and it is very hard to say where ingenuity ends and routine process begins.

Let us now state the rules on how a switching circuit can be constructed from a binary-decision program.

Rule 1. Each address of the binary-decision program corresponds to a node of the circuit.

Rule 2. If at address A the instruction is $T\quad x;\ B,\ C$, a variable x' should be connected between nodes A and B and a variable x should be connected between nodes A and C.

Rule 3. The node corresponding to address 1 is the input node. The node corresponding to address I is the output node.

A simple change in Rule 3 will enable us to get the negative of the switching circuit. This is done by making the output node the node corresponding to address θ rather than to address I.

We will illustrate our procedure by two examples.

3.1 *Example 1*

We wish to design a circuit with six switching variables, a,b,c and x,y,z. Let M be the binary number abc and N be the binary number xyz. Then the output is to be 1 whenever $M \geqq N$.

The problem says that two binary numbers M and N are to be compared; these two numbers may be compared one bit at a time. We may compare the most significant bits a and x first. If $a = 0$ and $x = 1$, then $M < N$, so that there is no output. If $a = 1$ and $x = 0$, then $M > N$, and the output will be 1. If $a = 0$ and $x = 0$ or $a = 1$ and $x = 1$, the comparison process must be continued to the next pair of bits.

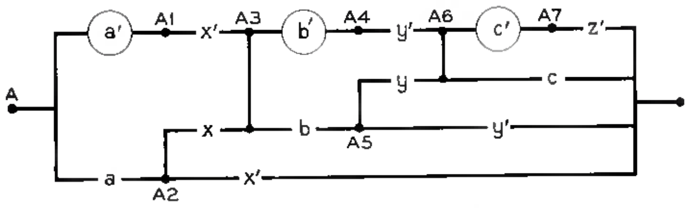


Fig. 2 — Switching circuit for Example 1.

The program proceeds by first comparing the pair of variables a and x. Depending on the values of a and x, b and y are compared next and, lastly, c and z, if necessary. The program instructions are

Address	*Instruction*
A	$T \quad a; A1, A2$
$A1$	$T \quad x; A3, \theta$
$A2$	$T \quad x; I, A3$
$A3$	$T \quad b; A4, A5$
$A4$	$T \quad y; A6, \theta$
$A5$	$T \quad y; I, A6$
$A6$	$T \quad c; A7, I$
$A7$	$T \quad z; I, \theta.$

Following the three rules, we begin with the first instruction and let A correspond to the input node of the circuit. A branch labeled a' leads to the node $A1$ and a branch labeled a leads to the node $A2$. $A1$ and $A2$ thus become internal nodes of the circuit. From $A1$ we need to put down only the branch x' leading to internal node $A3$, since a branch labeled x would give no output. Continuing in this way, we get all the paths from the input to the output; the addresses tell us where the interconnections between these paths are to be made. The circuit for this example is given in Fig. 2. From this circuit it can be seen that the three circled variables are superfluous and can be deleted.

3.2 *Example 2*

We wish to design a switching circuit with eight variables, a,b,c,d and w,x,y,z. Let L be the number of the variables a,b,c,d which are in the 0 state and let R be the number of the variables w,x,y,z which are in the 0 state. Then the output is to be 1 whenever $L \geqq R$.

The problem tells us that, if all of the variables a,b,c,d are 0, the output would be 1 regardless of what w,x,y,z are; if exactly three of the variables a,b,c,d are 0, then the output would be 1 whenever three or fewer of the variables w,x,y,z are 0; and so forth. To program this problem, we will therefore begin with the following subroutines:

S1. The output is to be 1 whenever three or fewer of the variables w,x,y,z are 0.

S2. The output is to be 1 whenever two or fewer of the variables w,x,y,z are 0.

S3. The output is to be 1 whenever one or fewer of the variables w,x,y,z is 0.

S4. The output is to be 1 whenever none of the variables w,x,y,z is 0.

The program is then completed by counting the number L of the variables a,b,c,d which are 0. If L is 4, the output is made 1 directly; if L is 3,2,1 or 0, the program enters subroutine $S1$, $S2$, $S3$ or $S4$ respectively.

We will begin with the subroutine programs. The subroutine $S1$ is a successive scan of the states of the variables w,x,y,z:

Address	*Instruction*
$S1$	$T \quad w; S11, I$
$S11$	$T \quad x; S12, I$
$S12$	$T \quad y; S13, I$
$S13$	$T \quad z; \theta, I.$

The subroutine $S2$ can be written likewise. A moment's reflection will show, however, that $S2$ can make use of a portion of the instructions of $S1$. Similarly, $S3$ can make use of $S2$ and $S4$ can make use of $S3$. The subroutine programs come out to be:

Address	*Instruction*
$S1$	$T \quad w; S11, I$
$S11$	$T \quad x; S12, I$
$S12$	$T \quad y; S13, I$
$S13$	$T \quad z; \theta, I$
$S2$	$T \quad w; S21, S11$
$S21$	$T \quad x; S22, S12$
$S22$	$T \quad y; \theta, S13$
$S3$	$T \quad w; S31, S21$
$S31$	$T \quad x; \theta, S22$
$S4$	$T \quad w: \theta, S31.$

The main program which evaluates L and selects the appropriate subroutine is

Address	*Instruction*
A	$T \quad a; A1, A2$
$A1$	$T \quad b; A3, A4$
$A3$	$T \quad c; A5, A6$
$A5$	$T \quad d; I, S1$
$A6$	$T \quad d; S1, S2$
$A4$	$T \quad c; A6, A7$
$A7$	$T \quad d; S2, S3$
$A2$	$T \quad b; A4, A8$
$A8$	$T \quad c; A7, A9$
$A9$	$T \quad d; S3, S4.$

Following the three rules of construction, the final circuit is given in Fig. 3. To show how the subroutine circuits can be combined in stages, the circuit for $S1$ is given in Fig. 4 and the combined circuit for $S1$ and $S2$ is given in Fig. 5.

Generally, we find that the switching circuits constructed from binary-decision programs have several distinct characteristics. The construction does not distinguish among series-parallel, bridge and nonplanar circuits, but it is restricted to unidirectional flow of current in any

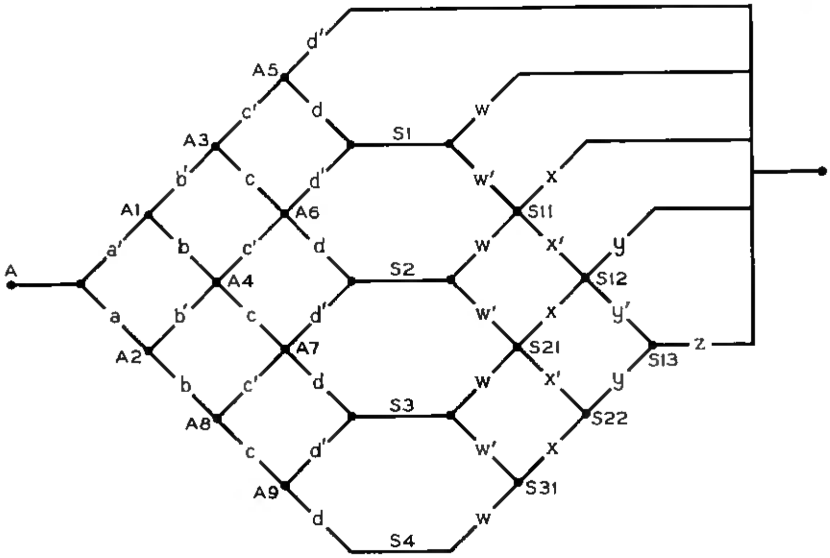


Fig. 3 — Switching circuit for Example 2.

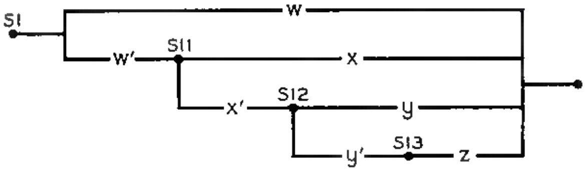


Fig. 4 — Circuit for S1 of Example 2.

branch. Because of the transfer characteristic of the program instruction, the program in most cases gives bridge or nonplanar circuits and very rarely gives series-parallel circuits. Again, because of the transfer characteristic of the instruction, the procedure tends to give circuits having a large number of transfers, causing unnecessary appearance of variables in the circuits. On the other hand, the presence of the transfers prevents sneak paths, which are often a source of worry.

IV. COMPUTATION OF SWITCHING FUNCTION BY BINARY-DECISION PROGRAMS

The problem that we wish to consider here is this: Suppose, in carrying out a complicated task, a complex decision depending on many variables is to be made and made repeatedly. Question: What procedure should one follow so as to arrive at the decision quickly and without having to go through a large amount of computation?

To make the problem more tractable, let us say that the decision function is a switching function of n variables. The problem is to find a good procedure for the computation of this function.

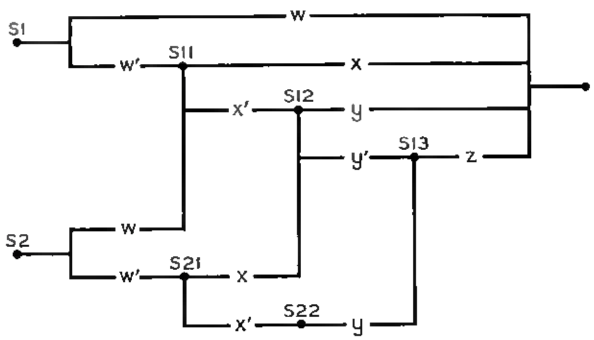


Fig. 5 — Combined circuit for S1 and S2 of Example 2.

The first question one should ask is, perhaps, what are the choices? If the switching function is, say,

$$x(y'z \vee yz') \vee x'yz,$$

what alternatives in computation are there?

There are indeed many alternatives. One may, for instance, carry out the following direct computation:

1. y AND z', store result in location a;
2. y' AND z, store result in location b;
3. (contents of a) OR (contents of b), store result in location a;
4. x AND (contents of a), store result in location a;
5. x' AND y, store result in location b;
6. z AND (contents of b), store result in location b;
7. (contents of a) OR (contents of b), store answer in location b.

Or, one may go back to the switching function itself and rewrite it as

$$(y \vee z)\, [x \oplus (yz)],$$

where $\oplus$ (called SUM) stands for addition modulo 2. A direct computation now becomes:

1. y OR z, store result in location a;
2. y AND z, store result in location b;
3. x SUM (contents of b), store result in location b;
4. (contents of a) AND (contents of b), store answer in location b.

We have done in four steps what took seven above.

Finally, we may write for this switching function a binary-decision program:

1. $T \quad x; 2, 4.$
2. $T \quad y; \theta, 3.$
3. $T \quad z; \theta, I.$
4. $T \quad y; 3, 5.$
5. $T \quad z; I, \theta.$

This computation scheme differs from the other two in one major aspect, namely, the number of steps one needs to go through in the binary-decision program before one gets an answer depends upon the initial values of the variables. For example, we will arrive at an answer in two or three steps according as $(xyz) = (001)$ or $(xyz) = (110)$. To compare this program with the second procedure (with AND, OR and SUM), therefore, let us sum over all eight combinations of the three variables. This yields a total of 22 steps for the binary-decision program, and 32 for the AND-OR-SUM procedure. On the other hand, the binary-decision program is longer by one instruction.

The questions that we want to answer here are these:

1. For an arbitrary switching function of n variables what is the order of magnitude of the number of instructions in its binary-decision program?

2. Comparing specifically the binary-decision program approach with the AND-OR-SUM procedure, which will in general need fewer instructions and which will need less time to execute?

Questions of this nature but pertaining to the number of relay contacts or electronic components have been studied by Shannon[2] and Muller.[3] As is the case with their investigations, we are not able to answer these questions for individual functions but our answers apply to an overwhelming fraction of switching functions of n variables.

4.1 *Computation by Binary-Decision Programs.*

Let f be a switching function of n variables, and let $\mu(f)$ be a number such that no binary-decision program representing f has fewer than $\mu(f)$ instructions. We will let μ_n be the smallest number of instructions sufficient to represent any switching function of n variables. That is,

$$\mu_n = \max\{\mu(f) \mid f \in F(n)\},$$

where $F(n)$ is the set of all switching functions of n variables. Then

Lemma 1: $\mu_n > 2^n/2n$.

Proof: Let $N(n,p)$ denote the number of possible binary-decision programs involving n variables with p instructions. Since each instruction can be chosen from at most np^2 instructions, it follows that $N(n,p) \leqq (np^2)^p$ and, in particular, $N(n,\mu_n) \leqq (n\mu_n{}^2)^{\mu_n}$.

Now suppose $\mu_n \leqq 2^n/2n$. Then

$$N(n,\mu_n) \leqq \left(n\,\frac{2^{2n}}{4n^2}\right)^{2^n/2n} = \left(\frac{2^{2n}}{4n}\right)^{2^n/2n} < 2^{2^n},$$

and the lemma follows.

To find an upper bound for μ_n requires an interesting subroutine technique. Let a set of programs each of which computes a switching function be called a *library*. Let $L(n)$ be the library of programs which represent all 2^{2^n} switching functions of n variables. Then

Lemma 2: The library $L(n)$ can be written so that it contains not more than 2^{2^n} instructions.

Proof: For $n = 2$, the 16 functions of two variables are

1. θ.
2. I.
3. x_1.
4. x_2.
5. x_1'.
6. x_2'.
7. $x_1 x_2$.
8. $x_1' x_2$.
9. $x_1 x_2'$.
10. $x_1' x_2'$.
11. $x_1 \vee x_2$.
12. $x_1' \vee x_2$.
13. $x_1 \vee x_2'$.
14. $x_1' \vee x_2'$.
15. $x_1' x_2 \vee x_1 x_2'$.
16. $x_1 x_2 \vee x_1' x_2'$.

Now $L(2)$ can be the following program:

1. $T \quad x_1 ; \theta, \theta$.
2. $T \quad x_1 ; I, I$.
3. $T \quad x_1 ; \theta, I$.
4. $T \quad x_2 ; \theta, I$.
5. $T \quad x_1 ; I, \theta$.
6. $T \quad x_2 ; I, \theta$.
7. $T \quad x_1 ; \theta, 4$.
8. $T \quad x_1 ; 4, \theta$.
9. $T \quad x_1 ; \theta, 6$.
10. $T \quad x_1 ; 6, \theta$.
11. $T \quad x_1 ; 4, I$.
12. $T \quad x_1 ; I, 4$.
13. $T \quad x_1 ; 6, I$.
14. $T \quad x_1 ; I, 6$.
15. $T \quad x_1 ; 4, 6$.
16. $T \quad x_1 ; 6, 4$.

Therefore, $L(2)$ can be written in exactly 2^{2^2} instructions.

Now suppose the lemma is true for all n, $2 \leqq n \leqq m$. Consider $n = m + 1$. The library $L(m)$ by hypothesis has not more than 2^{2^m} instructions and covers all functions of m variables among the $2^{2^{m+1}}$ functions of $m + 1$ variables. For each of the other $2^{2^{m+1}} - 2^{2^m}$ functions of $m + 1$ variables, the program can be written

$$T \quad x_{m+1} ; A, B,$$

where A and B refer to addresses in the library $L(m)$. Hence $L(m + 1)$ can be written with not more than

$$(2^{2^{m+1}} - 2^{2^m}) + 2^{2^m} = 2^{2^{m+1}}$$

instructions. This proves the lemma.

Using Lemma 2 and combining it with Lemma 1, we have

Theorem 1: For all n,

$$\frac{1}{2}\frac{2^n}{n} < \mu_n < 4\frac{2^n}{n} - 1.$$

Proof: The lower bound was given by Lemma 1. To get the upper bound, let us write $n = (n - j) + j$, where j may vary from 0 to n. Let f be a switching function of n variables. Then f may be expanded about $n - j$ of its variables in its canonical expansion:

$$f(x_1, x_2, \cdots, x_n) = x_1' x_2' \cdots x_j' f(0,0, \cdots, 0, x_{j+1}, \cdots, x_n) \vee \\ \cdots \vee x_1 x_2 \cdots x_j f(1,1, \cdots, 1, x_{j+1}, \cdots, x_n).$$

Now, we may write a program with exactly

$$1 + 2 + 2^2 + \cdots + 2^{j-1} = 2^j - 1$$

instructions to give all the 2^j functions of j variables

$$x_1'x_2'x_3' \cdots x_j', \cdots, x_1x_2x_3 \cdots x_j .$$

Also, by Lemma 2, we may construct a library $L(n - j)$ with not more than $2^{2^{n-j}}$ instructions. Hence f can be programmed with not more than $2^{2^{n-j}} + 2^j - 1$ instructions. Now set

$$j = n - [\log_2(n - \log_2 n)],$$

where $[x]$ denotes the largest integer less than or equal to x. Then

$$2^j = \frac{2^n}{2^{[\log_2(n-\log_2 n)]}} \leqq 2\,\frac{2^n}{n - \log_2 n} \leqq 3\,\frac{2^n}{n} \quad \text{for} \quad n \geqq 4.$$

Therefore, for $n \geqq 4$,

$$\mu_n \leqq \min \{2^{2^{n-j}} + 2^j - 1 \mid 4 \leqq j \leqq n\} \leqq 4\,\frac{2^n}{n} - 1.$$

Now, by direct computation, we find $\mu_1 = 1$, $\mu_2 \leqq 4$ and $\mu_3 \leqq 6$. Hence for all n, we have $\mu_n \leqq 4\,(2^n/n) - 1$, and the theorem follows.

Theorem 2: Given any ϵ, $0 < \epsilon < 1$, a fraction $1 - 2^{-\epsilon 2^n}$ of switching functions of n variables will need at least $2^n(2n)^{-1}(1 - \epsilon)$ binary-decision program instructions to program.

Proof: The number of possible binary-decision programs with not more than $2^n(2n)^{-1}\,(1 - \epsilon)$ instructions cannot exceed

$$\left\{2n\left[\frac{2^n}{2n}\,(1 - \epsilon)\right]^2\right\}^{2^n(2n)^{-1}(1-\epsilon)},$$

which is less than $2^{2^n(1-\epsilon)}$. Therefore, the fraction of switching functions of n variables which need not more than $2^n(2n)^{-1}(1 - \epsilon)$ instructions to program cannot exceed $2^{-\epsilon 2^n}$. Hence the rest must need at least $2^n(2n)^{-1}$ $(1 - \epsilon)$ instructions to program and the proof follows.

The procedure outlined in the proof of Theorem 1 yields for each switching function of n variables a binary-decision program. We will call this program the *normal* binary-decision program for that function. A close examination of this procedure will show that the number of program instructions executed in the computation of a particular value of any switching function of n variables never exceeds n. That is, in the computation no variable is examined more than once. Therefore, we have

Corollary 1: The number of instructions which has to be executed in the normal binary-decision program for the computation of each value of any switching function of n variables never exceeds n.

These results together give us a fairly good idea of how efficient it is to compute switching functions with binary-decision programs. For $n = 20$, for instance, practically all switching functions need more than 25,000 instructions to program, although none needs more than 200,000 instructions. The number of instructions that one needs to go through to compute a single value is never more than 20, however. We want now to compare these results with the AND-OR-SUM procedure mentioned earlier.

4.2 *An Alternative Procedure*

Before we consider the AND-OR-SUM procedure illustrated previously, it might be well for us to show why this particular procedure is chosen for comparison. A switching function is commonly written in terms of its variables and their complements connected by AND and OR. Besides AND and OR, there are eight other binary operations, denoted by $/$, $\downarrow$, $\oplus$, $\leftrightarrow$, $\supset$, $\subset$, $\not\supset$, and $\not\subset$, where we have called $\oplus$ the SUM operation. These can be written in terms of AND and OR operation:

$$\begin{array}{ll} x / y = x' \vee y'. & x \supset y = x' \vee y. \\ x \downarrow y = x'y'. & x \subset y = x \vee y'. \\ x \oplus y = x'y \vee xy'. & x \not\supset y = xy'. \\ x \leftrightarrow y = xy \vee x'y'. & x \not\subset y = x'y. \end{array}$$

In order not to be restrictive with our alternative computational procedure, let us be allowed to use any of these 10 operations in a computation. The first thing we wish to show is that we lose nothing by throwing away seven of these operations. In order to do this, let us call any switching function expression involving the variables and their complements, in which any of the 10 operations may appear, a *binary expression*. Let us also say that two binary expressions are *equivalent* if they represent the same function. Then

Theorem 3: Let $\bar{f}$ be a binary expression with r operations. Then there is an equivalent binary expression $\bar{g}$ having r or fewer operations such that the only binary operations appearing in the expression $\bar{g}$ are AND, OR and SUM.

For example, the expression

$$(x' \not\supset y) \downarrow [(z' \oplus u')/w']$$

is equivalent to the expression

$$(x \vee y)\,[w'(z \oplus u)]$$

in which no operation other than AND, OR and SUM appears.

To prove this theorem, we note that, if $\bar{g}$ is any binary-expression of r operations, $r \geqq 1$, then $\bar{g}$ is expressible as

$$\bar{g} = \bar{h} * \bar{k},$$

where $\bar{h}$ and $\bar{k}$ are binary-expressions each of $(r - 1)$ or fewer operations and $*$ is one of the 10 binary operations. Also,

$$\bar{g}' = (\bar{h} * \bar{k})' = \bar{h} *' \bar{k},$$

where $*'$ is again one of the 10 binary operations. Therefore, if $\bar{g}$ is any binary-expression with r operations, then its complement expression $\bar{g}'$ requires not more than r operations.

Going back to Theorem 3, we note that the theorem is true for $r = 1$. Now suppose that the theorem is true for all r, $1 < r \leqq R$. Let $\bar{f}$ be an expression with $R + 1$ operations. Then $\bar{f}$ is expressible in the form

$$\bar{f} = \bar{g} * \bar{h},$$

where $\bar{g}$ and $\bar{h}$ are expressions each with not more than R operations and $*$ is one of the 10 binary operations. Since for the seven operations $/$, $\downarrow$, $\leftrightarrow$, $\supset$, $\subset$, $\not\supset$ and $\not\subset$ we have $\bar{g}/\bar{h} = \bar{g}' \vee \bar{h}'$, $\bar{g} \downarrow \bar{h} = \bar{g}'\,\bar{h}'$, $\bar{g} \leftrightarrow \bar{h} = \bar{g}' \oplus \bar{h}$, $\bar{g} \supset \bar{h} = \bar{g}' \vee \bar{h}$, $\bar{g} \subset \bar{h} = \bar{g} \vee \bar{h}'$, $\bar{g} \not\supset \bar{h} = \bar{g}\,\bar{h}'$ and $\bar{g} \not\subset \bar{h} = \bar{g}'\,\bar{h}$, it follows that $\bar{f}$ can be expressed as

$$\bar{f} = \bar{k} *' \bar{m},$$

where $*'$ is one of the three operations AND, OR or SUM, $\bar{k}$ is either $\bar{g}$ or its complement $\bar{g}'$ and $\bar{m}$ is either $\bar{h}$ or its complement $\bar{h}'$. The theorem now follows from the previous assertion and the induction hypothesis.

Because of Theorem 3, we may as well consider only those operations AND, OR and SUM in our computation procedure. To make specific comparisons possible, let us define a Boolean program as one that is made up of three kinds of Boolean instructions:

AND	$A, B, C,$
OR	$A, B, C,$
SUM	$A, B, C,$

Here A, B or C refers to one of $4n$ possible locations in which values of the n variables and their complements as well as intermediate results may be stored. We have reserved $2n$ locations for the storage of intermediate results; this is sufficient for computing the most complex of functions of n variables. To standardize location reference we will use locations $1, 2, \cdots, n$ for storing variable values; $1', 2', \cdots, n'$ for storing complement values; and $1'', 2'', \cdots, (2n)''$ for storing intermediate results. The Boolean program for the function

$$[x \oplus (yz)]\,(y \vee z),$$

for example, can now be written

```
1. OR     2, 3, 1″;
2. AND    2, 3, 2″;
3. SUM    1, 2″, 2″;
4. AND    1″, 2″, 2″;
```

where x, y and z values are stored in locations 1, 2 and 3 respectively; x', y' and z' in locations $1'$, $2'$ and $3'$ respectively; and the final result is in $2''$.

To each function f, let $\upsilon(f)$ be a number such that no Boolean program which computes f can have fewer than $\upsilon(f)$ instructions. Let

$$\upsilon_n = \max\,\{\upsilon(f) \mid f \in F(n)\},$$

where, as before, $F(n)$ is the set of all switching functions of n variables. Then

Lemma 3:

$$\upsilon_n \geqq \frac{2^n}{3 \log_2 n + 8}.$$

Proof: Let $M(n,p)$ be the number of possible Boolean programs with p instructions. Then

$$M(n,p) \leqq [3(4n)^3]^p,$$

and

$$M(n,\upsilon_n) \leqq (192n^3)^{\upsilon_n}.$$

Suppose

$$\upsilon_n < \frac{2^n}{3 \log_2 n + 8}.$$

Then

$$M(n,\upsilon_n) \leqq (192n^3)^{2^n(3\log_2 n+8)^{-1}} < (192n^3)^{2^n(\log_2 192n^3)^{-1}} = 2^{2^n},$$

and the proof follows.

Lemma 3 affords us a comparison of Boolean programs and binary-decision programs. Combining Theorem 2 and Lemma 3, we see that, for $n \geqq 64$, the inequality

$$\mu_n < \upsilon_n$$

strictly holds, and it most probably holds for much smaller values of n as well. In fact, we see that, for large n, v_n is bigger than μ_n by an order of magnitude. It therefore follows that Boolean programs are in general much longer than binary-decision programs. Moreover, since for each computation every instruction in a Boolean program has to be executed, the difference in speed of computation becomes indeed astronomical.

It has been amply clear that, although Boolean representation of switching circuits has been the foundation on which switching theory had been built, the inherent limitations in the Boolean language seem to be difficult hurdles to surmount. Boolean representation is algebraic and highly systematic, but so inflexible that it is powerless against all but series-parallel circuits. Moreover, as this paper shows, it is extremely inefficient as an instrument for computation. Binary-decision programming is our attempt of a way to get beyond these limitations. It works well for computation. Further studies will be required to find efficient ways of minimizing binary-decision programs and to make binary-decision programming an instrument for circuit synthesis.

REFERENCES

1. Shannon, C. E., A Symbolic Analysis of Relay and Switching Circuits, A.I.E.E. Trans., **57,** 1938, p. 713.
2. Shannon, C. E., The Synthesis of Two-Terminal Switching Circuits, B.S.T.J., **28,** January 1949, p. 59.
3. Muller, D. E., Complexity in Electronic Switching Circuits, I.R.E. Trans., **EC-5,** 1956, p. 15.

A Method of Coding Television Signals Based on Edge Detection

By BELA JULESZ

(Manuscript received February 5, 1959)

A method is described for transmitting digitalized video signals to reduce channel capacity from that needed for standard PCM. This method takes advantage of the inability of the human eye to notice the exact amplitude and shape of short brightness transients. The transmitted information consists of the amplitudes and times of occurrence of the "edge" points of video signals. These selected samples are coarsely quantized if they belong to high-frequency regions, and the receiver then interpolates straight lines between the samples. The system was simulated on the IBM 704 computer. The processed pictures and obtained channel-capacity savings are presented.

I. INTRODUCTION

There is an increasing trend in the communication field to utilize the physiological and psychological properties of the ultimate receiver — the human observer. Some of these properties were applied many years ago in establishing television transmission standards — for example, visual acuity and flicker-fusion frequency thresholds. The development of information theory made this trend even more apparent, particularly in Shannon's first coding problem, where he posed the question of finding an optimum code for a continuous information source when the fidelity criterion of the receiver was given.

Unfortunately the fidelity criteria of human observers are not known. This lack of knowledge is particularly apparent in visual processes, even though in this field the challenge of possible channel-capacity saving is tempting. From a theoretical standpoint, the solution of the first coding problem must be postponed until enough psychological data are collected. But, from a practical point of view, it is possible to overcome this barrier. Instead of searching for human fidelity criteria, we can proceed in the following simpler way.

First, we take the present television pictures of toll quality as a standard. Then we process the signals of the television source by performing some reasonable operations which reduce information rate, and compare by mere inspection the resulting pictures with the standard pictures. If, for a well-chosen class of different pictures representing most of the possible cases, the results of statistical preference tests do not discriminate between the processed pictures and the standard toll quality pictures, we can regard the obtained rate of information as a practical upper bound. If we choose a processing which codes the continuous source in binary digits, and assume an error-free binary channel for transmission (e.g., PCM as a good approximation), we can ensure that no further picture quality degradation will occur. Thus, the viewer will get the same quality of pictures he was accustomed to seeing, but with less channel capacity.

As we see from the above considerations, the search for an optimum code becomes a trial-and-error procedure. The problem now is to find reasonable operations for the processing. They should be based on psychological facts or hypotheses and should not be too complicated for realization. In the last few years several ideas have been tried out along these lines, with more or less success.[1,2,3] The complexity of the required instrumentation limited or prevented a thorough investigation of these ideas. However, the rapid development of general-purpose digital computers has made it possible to test new ideas without actually building equipment. We can simulate any system on a computer by writing a program which converts the general-purpose computer to a special-purpose computer. Special input and output transducers convert the input pictures to sequences of digitalized numbers and, after processing them, reconvert the output of the computer to pictures. Such equipment was developed by and is used now in the Visual and Acoustical Research Department of Bell Telephone Laboratories as a valuable research tool.[4,5] For the processing we use an IBM 704 computer. Although at present we cannot perform the simulations of television coding schemes in real time on the existing computers, we can evaluate many aspects of a system's performance without building it. Thus, it is possible to compare systems and choose the best one before actual realizations.

II. PROPERTIES OF TELEVISION SIGNALS AND OF THE HUMAN RECEIVER RELEVANT TO EDGE DEFINITION

This paper describes a system which transmits only certain points of a television signal, depending on some given signal properties, and, after reception, interpolates between the points according to a given law. Several similar systems are described which differ only in the criteria by which the transmitted points are selected and coded, and in the function used for the interpolation.[6,7] In our case these criteria were chosen to match some properties of both the usual television pictures and vision.

Television waveforms, in contrast to acoustical signals, include fast transients followed by horizontal or slowly changing sections, and are relatively poor in damped oscillations. Because of this, a recent system[6] which transmits only the extremals of acoustical signals and interpolates the output at the receiver according to a given law is not suitable for television. We tried out this system by simulating it on the computer and in the pictures on the left in Fig. 6 results are shown — that systems which perform well for acoustics may not work for vision. Furthermore, there is experimental evidence that the human eye is not very sensitive to the exact amplitude and shape of sudden brightness changes, but is able to locate the starting and ending points of these brightness transients fairly accurately. (The meaning of this property will be made clearer by quantitative results explained in the course of this paper.) Because of these properties of the source and of the ultimate human receiver, we chose to transmit only the end points of the brightness transients. Provided the standard horizontal scanning technique is used, it is quite simple to give a mathematical criterion for selecting such "edge" points.

To locate an edge it seems natural to require that some combinations of the first and higher order derivatives of the input signal should comprise an extremum. Now, according to the sampling theorem, the least rate of discrete sampling points which determine a band-limited signal (limited to bandwidth W) must occur at the Nyquist rate ($2W$). These samples are enough to determine also derivatives of any order. If $u(t)$ is the continuous band-limited input signal and is sampled at Nyquist rate, which yields u_i ($i = \cdots -2, -1, 0, 1, 2, \cdots$) samples, the samples u_i' of the derivative signal $u'(t)$ are given by the following linear transformation:

$$\mathbf{u}' = \mathbf{A}\mathbf{u}, \tag{1}$$

where

$$\mathbf{u} = (u_1, u_2 \cdots u_n); \qquad \mathbf{u}' = (u_1', u_2' \cdots u_n')$$

and the elements of the transformation matrix $\mathbf{A}$ are

$$A_{mm} = 0; \qquad A_{mn} = 2W \frac{(-1)^{m-n}}{m - n}.$$

For the processing on digital computers we get the input data in sampled and quantized form. As we see from (1), to compute only a first deriva-

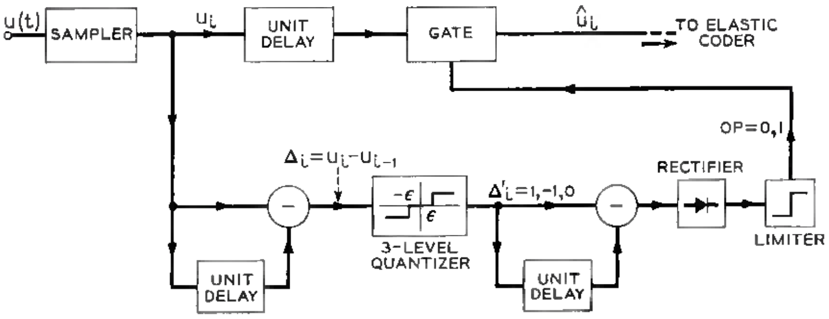


Fig. 1 — Actual system used to transmit only certain points of television signal.

tive would require a vast amount of computations, taking into account all sample values of the input signal. We can reduce the number of operations to a few subtractions if we introduce in place of the derivatives differences between sequential samples. If we now define an edge in terms of differences, we get a new system which resembles the previous one superficially, but in the microstructure (i.e., in the determination of a point within one Nyquist interval) the systems may differ considerably. We must not forget that, on account of human vernier acuity, ambiguities within one Nyquist interval $1/(2W)$ may be clearly visible. Because we cannot decide by mere speculation which of these systems will prove to be superior, we investigate the simpler one first.

III. DEFINITION OF AN EDGE POINT

The actual system is given in Fig. 1. The band-limited video signal $u(t)$ is sampled at Nyquist rate, and the difference ($\Delta_i = u_i - u_{i-1}$) between sequential samples is computed. A three-level quantizer with decision levels ϵ and $-\epsilon$, and with representative levels 1, -1 and 0 performs the following quantization:

$$\begin{aligned} \Delta_i' &= 1, && \text{if } \Delta_i \geqq \epsilon, \\ \Delta_i' &= 0, && \text{if } |\Delta_i| < \epsilon, \\ \Delta_i' &= -1, && \text{if } \Delta_i \leqq -\epsilon. \end{aligned} \tag{2}$$

Here the ϵ decision level has to be set experimentally. If it is too small, the operation will be affected by trivially fine structure; if it is too large, the fine details in the picture will be lost.

Now we define a sample point as an edge point ($\hat{u}_i$) if the quantized left- and right-hand differences (Δ_{i-1}' and Δ_i') of that point belong to one of the six cases, given by (3) and shown in Fig. 2(a):

$$\begin{aligned} &\Delta_{i-1}'\Delta_i' < 0, \\ &\Delta_{i-1}' = 0 \text{ and } \Delta_i' \neq 0, \\ &\Delta_{i-1}' \neq 0 \text{ and } \Delta_i' = 0; \end{aligned} \tag{3}$$

that is (in a more efficient notation),

$$\Delta_{i-1}' \neq \Delta_i'.$$

These cases refer to the local maxima or minima of the differences and to the end points of horizontal sections, provided the changes are above the ϵ threshold. Sample points on monotonic increasing, decreasing or horizontal sections will be omitted. These nontransmitted samples thus fall in the next three cases, shown in Fig. 2(b).

To select (from the nine possible cases) the six cases which correspond to an edge point, we have to perform the operations indicated in Fig. 1.

The output of the quantizer is again delayed one sample period and subtracted from its undelayed form to obtain the difference of the quantized left- and right-hand differences. After the second subtraction we get $OP = \Delta_i' - \Delta_{i-1}'$, which is nonzero for samples for which we want to define an edge and is zero otherwise.

The OP signal after full-wave rectification and limitation operates as a gating pulse and specifies which samples have to be transmitted.

As the result of these operations we get samples at an irregular rate, the average of which is substantially less than the Nyquist rate. This average rate depends on the picture material and on the ϵ threshold. Because of the irregular occurrences of the selected samples we also must

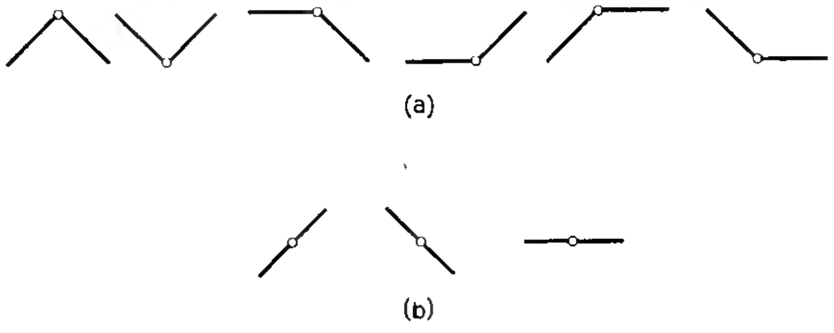


Fig. 2 — (a) Six cases in which quantized left- and right-hand differences are not equal; (b) monotonic increasing, decreasing and horizontal sections.

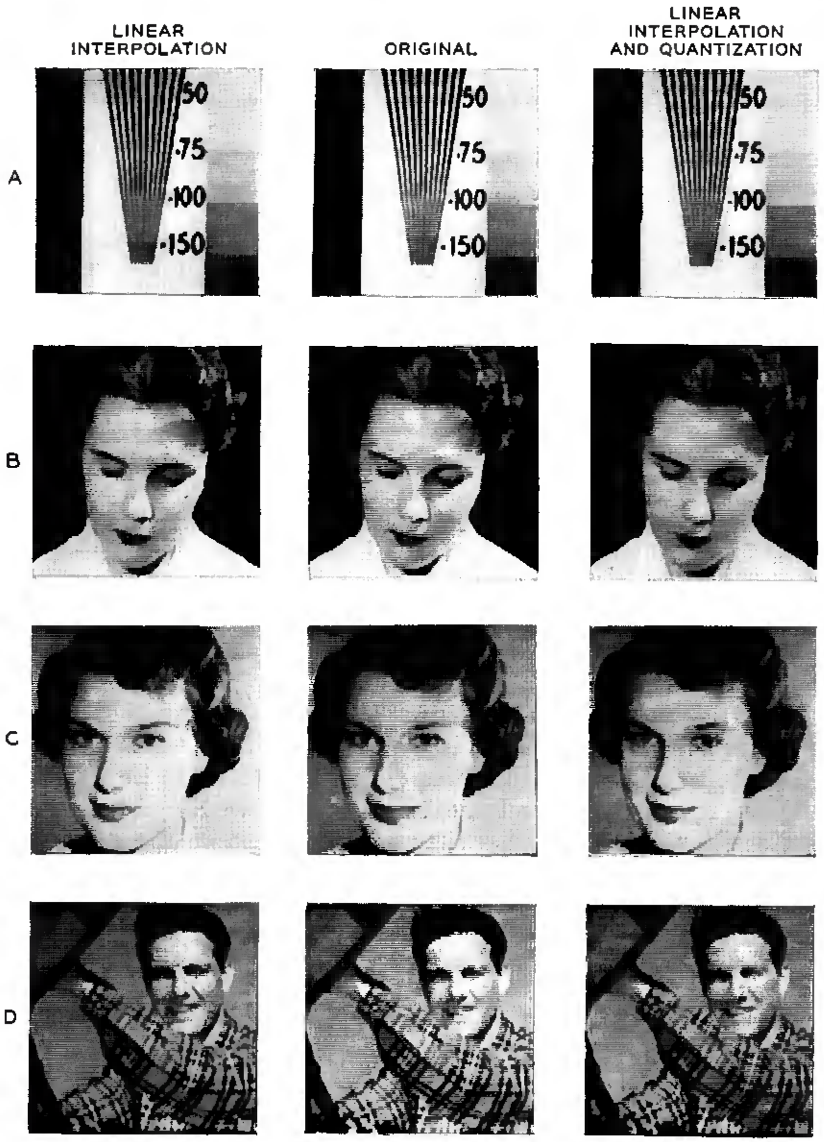


Fig. 3 — Picture material before and after processing (finer threshold setting).

specify their positions in time. That requires additional information to be transmitted; thus, the saving in required channel capacity is less than the ratio between the Nyquist rate and the average rate of the edge points. The net saving depends considerably on the coding schemes we apply to transmit the values and locations of the chosen samples, as will be discussed later.

IV. INTERPOLATION

After we specify the criteria for selecting the samples, we have to decide on an appropriate interpolation function. Because the selected samples occur less frequently than at Nyquist rate, they can be considered independently of each other. This means that there is no preferred curve connecting the selected samples from a mathematical point of view. From a psychological standpoint, the eye is not sensitive to the exact shape of a short transient, and thus the simplest choice in that region is a linear interpolation function. Furthermore, the longer monotonic increasing or decreasing sections between two edge points can be convex or concave and, in the average case, the best interpolation is again the linear one.

V. COMPUTER SIMULATION

According to the above considerations a program was written to determine the edge points by using the criteria given in (3) and to interpolate straight lines between them.[8] The program also provided the statistics of the distribution of the distances between adjacent edge points. The time fluctuation of the selected sample rate also was recorded.

The picture material before processing but quantized in time and amplitude is shown in Fig. 3 (middle column). The picture consists of 100 lines, each containing 120 picture elements. For synchronization and blanking we used 15 picture elements in every line and the complete first line; thus the number of picture points is $99 \times 105 = 10{,}395$. This resolution corresponds to a television picture $\frac{1}{25}$ the area of the present standards. That means that the given pictures have to be observed from five times greater distance to get the usual resolution. If we take four times picture height as the usual viewing distance for standard television, the presented pictures have to be judged from a distance of 20 times picture height. The reason for the choice of this coarser resolution was a compromise between acceptable picture quality and computer storage capacity. The amplitudes were quantized into 10 bits (1024 levels) between the white and blacker-than-black levels, and into 9 bits

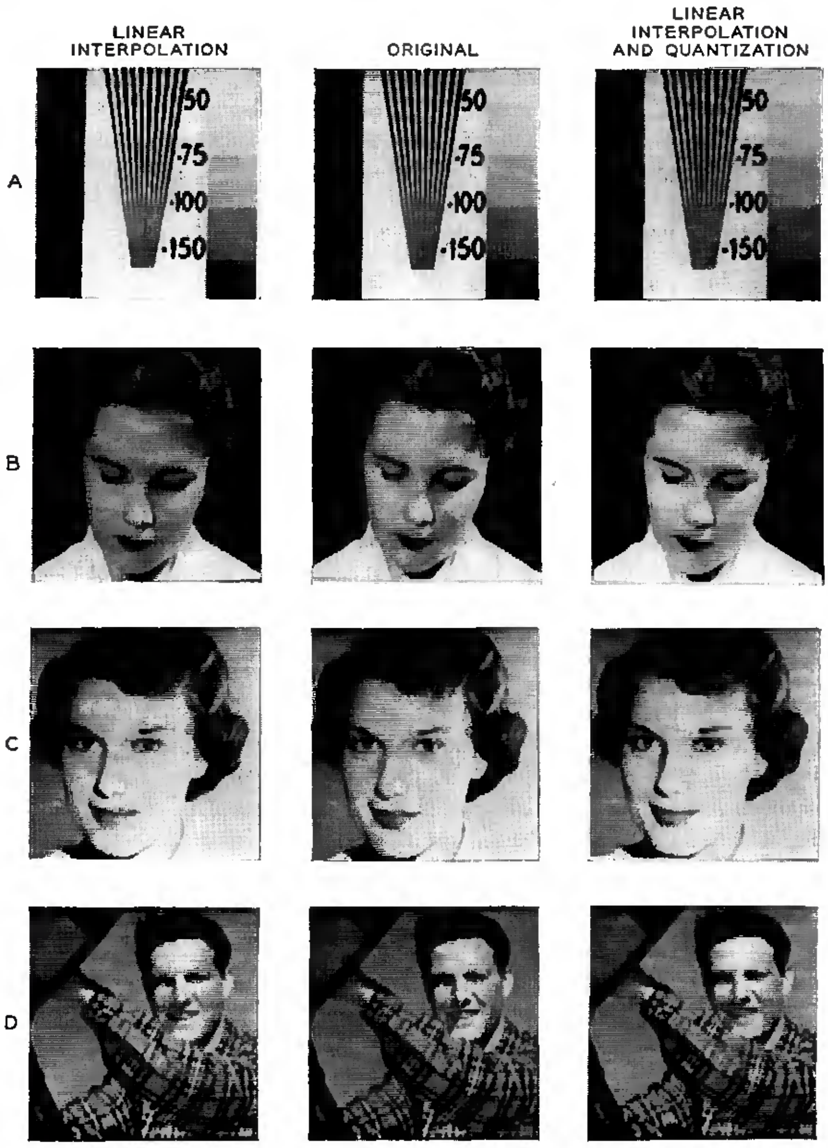


Fig. 4 — Picture material before and after processing (coarser threshold setting).

(512 levels) between the white and black levels, although 7 bits are enough for excellent quality.[1] The sampling and quantization were performed by an analog-to-digital converter, which could perform the opposite operations as well. A slow-speed scanning system converted the pictures into electrical signals and back to pictures. The sampled and quantized signals were put on a tape which served as an input to the IBM 704. The processed output of the computer was written on tape too, and the same devices in reversed operation converted it into pictures. The pictures which are designated as "original" (Figs. 3 and 4, middle columns) went through all these devices, but the program of the computer was such that it copied the input tape unchanged onto an output tape.

After we tried the processing with several ϵ threshold values we got the surprising result that, although the number of selected samples increased with decreasing ϵ values, the over-all appearance became worse. The most apparent defects were at vertical edges. The explanation of this effect is as follows: With decreasing ϵ thresholds the positioning of an edge point at the endings of horizontal sections becomes very sensitive. A little change in slope can shift the edge points several Nyquist intervals apart (see points ϵ_1 in Fig. 5). At a vertical edge each slope of the transients differs slightly from the one in the lines above (a small amount of added noise has the same effect), giving a very annoy-

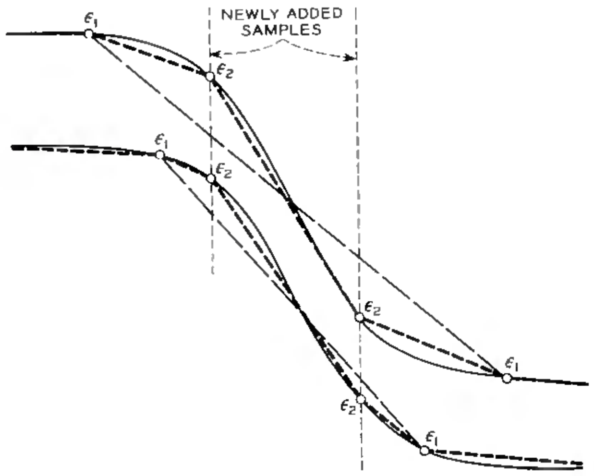


Fig. 5 — Slight change in slope (as in upper curve) moves edge points ϵ_1 several Nyquist intervals apart.

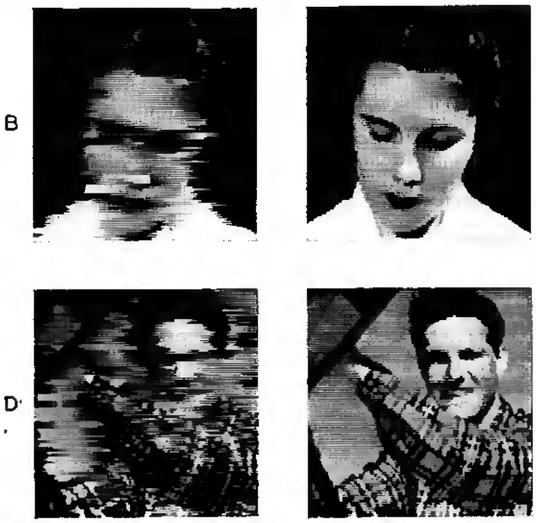


Fig 6 — (left) Distorted pictures show that systems which perform well for acoustics may not work for vision; (right) output of system with only one fine threshold.

ing fuzziness to the vertical edge. The right-hand pictures of Fig. 6 show the output of this system with a fine threshold setting ($\epsilon = 3.6$ per cent) and the above-mentioned defects are clearly visible. If we increase the ϵ threshold, this sensitiveness to edge positioning decreases, but the quality of the pictures also decreases. The reason for this is that, by taking increased threshold settings, we get fewer selected samples, and thus fine details in the pictures will be lost. With small threshold values, we get phase errors in edge positioning. Therefore, ϵ can be neither too small nor too large, and even the best compromise does not ensure adequate picture quality.

VI. OBJECTIONABLE SENSITIVITY TO EDGE POSITIONING AND ITS CORRECTION

There is a way to get rid of this annoying fuzziness and still be able to choose a value of ϵ that is small enough. If we take two threshold values (ϵ_1, ϵ_2) such that $\epsilon_1 \ll \epsilon_2$, we get two sets of edge points. We then take the union of these two sets. In most cases the set of edge points determined by the finer threshold contains the set determined by the coarser threshold, and thus does not increase the number of selected points. In the few cases when that is not the case, the additional points help to cure the sensitivity to small slope changes. In Fig. 5 we see that the edge points determined by ϵ_2 remain fixed in subsequent lines, and the phasing errors due to the edge points given by ϵ_1 have no effect on the over-all interpolation.

We determined the number of selected samples for this system using the picture material shown in the middle column of Fig. 3. We chose $\epsilon_2 = 10$ per cent (of the peak-to-peak value between black and white) and $\epsilon_1 = 3.6$ or 5 per cent. The increase of selected samples due to the additional samples determined by ϵ_2 depended on the picture material and was small (less than 11 per cent for pictures A, B, C and 17 per cent for picture D). The ratio of the selected samples to the total number of samples is given in Table I in per cent.

To simplify the design of coding devices, we limited the maximal distance to 16 Nyquist intervals. If, after determining an edge point and scanning further from left to right 16 Nyquist intervals, we did not find a next edge point, we selected a new sample 17 Nyquist intervals away from the previously selected sample. The frequency of occurrence of such a case is very small; thus, the increase due to these newly selected points is negligible.

The foregoing process gives good results in nearly every case. In exceptional cases, the pictures leave something to be desired. The reason for this and its correction are discussed next.

VII. THE "TUNNEL EFFECT" AND ITS CORRECTION

The system discussed above selects the edge points by analyzing the quantized differences according to (3). If the difference between subsequent samples is less than the ϵ_1 threshold, we do not transmit any sample. Now the pictures may contain hill- or valley-like sections with slopes so mild that the left- and right-hand differences around the maximum or minimum are less than ϵ_1, and thus we do not select these maximum or minimum points for transmission. The linear interpolation between the subsequent edge points looks like a tunnel, and if $t_j - t_i$

TABLE I — RATIO OF SELECTED SAMPLES TO TOTAL (PER CENT)

Scene	System Setting	
	$\epsilon_1 = 3.6$ per cent; $\epsilon_2 = 10$ per cent	$\epsilon_1 = 5$ per cent; $\epsilon_2 = 10$ per cent
A	32.9	29.1
B	30.1	24.0
C	34.9	28.3
D	47.0	42.0

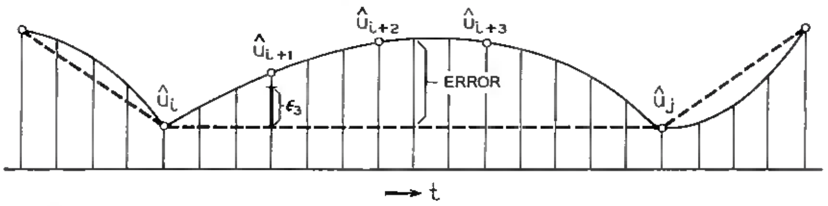


Fig. 7 — The tunnel effect between edge points $\hat{u}_j$ and $\hat{u}_i$ and its correction with ϵ_3 and $\tau = 3$.

(the distance between edge points $\hat{u}_j$ and $\hat{u}_i$) is long enough, large errors can be committed (see Fig. 7). It is possible to correct this effect in many ways. We used the following procedure: We subtracted the original picture from the processed one to give the interpolation errors. A new threshold value ϵ_3 was chosen. Whenever the error exceeded this threshold at time, t_k, the routine searched for the closest edge points left and right. If the distance, τ, on both sides was equal to or more than three Nyquist intervals, the routine selected the sample, u_k, for transmission; if the distance was less, no additional samples were selected. Thus we left the errors uncorrected for short sections (less than six Nyquist intervals long), utilizing the same psychological effect; i.e., the eye is not sensitive to the exact value of brightness changes in short times. This last manipulation improved the picture quality further. The number of selected points in this system is given in Table II. The distribution of the distance between the edge points for scene B is given in Fig. 8. Here, P_i is the frequency of the distances between subsequent edge points, and the index refers to the distances in Nyquist intervals.

By comparing Table I with Table II we see that the tunnel effect occurs very seldom, and that the increase in transmitted samples is negligible. The pictures obtained by this variant are shown in the left columns of Figs. 3 and 4. Fig. 3 corresponds to the finer threshold setting ($\epsilon_1 = 3.6$ per cent, $\epsilon_2 = 10$ per cent, $\epsilon_3 = 5$ per cent, $\tau = 3$ Nyquist

Table II — Ratio of Selected Samples to Total (Per Cent)

Scene	System Setting	
	$\epsilon_1 = 3.6$ per cent; $\epsilon_2 = 10$ per cent; $\epsilon_3 = 5$ per cent; $\tau = 3$	$\epsilon_1 = 5$ per cent; $\epsilon_2 = 10$ per cent; $\epsilon_3 = 7.2$ per cent; $\tau = 3$
A	32.9	29.2
B	31.3	25.3
C	35.8	29.3
D	47.3	42.4

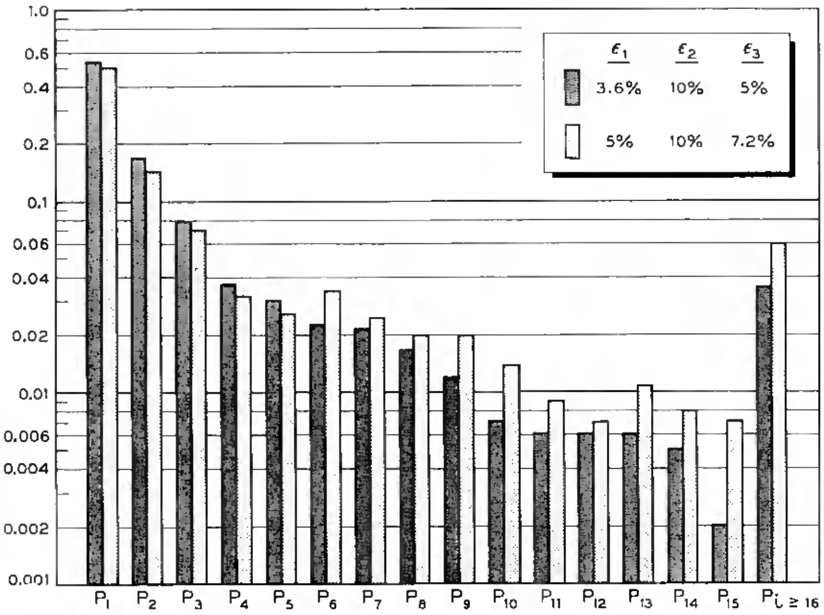


Fig. 8 — Distribution of the distance between subsequent edge points in scene B.

intervals); Fig. 4 to the coarser setting ($\epsilon_1 = 5$ per cent, $\epsilon_2 = 10$ per cent, $\epsilon_3 = 7.2$ per cent, $\tau = 3$ Nyquist intervals).

As we see, minor modifications in the program parameters improve the appearance of the pictures considerably. The statistics of the selected points for scene B are given in Fig. 8. Another advantage of choosing linear interpolation becomes apparent. As we add new points to the original edge points according to some different criterion, we need not label them separately because, in the case of linear interpolation, every received sample can be treated equally.

VIII. EVALUATION OF PROCESSED PICTURES

In Figs. 3 and 4 the left columns show the pictures processed according to the last variant. This variant, which includes edge determination by fine and coarse thresholds and tunnel-effect correction, we shall refer to simply as "linear interpolation." As we see, small changes in the threshold greatly affect the number of selected samples and the picture quality. If we decrease the thresholds, the number of selected samples reaches an asymptotic value which, depending on the picture material, is considerably higher than those obtained by $\epsilon_1 = 3.6$ per cent. (For example, in scene B the asymptotic ratio of selected samples to the total is 49.3 per cent.) Nevertheless, the improvement of the processed picture is very slight if $\epsilon_1 < 3.6$ per cent. So, the setting $\epsilon_1 = 3.6$ per cent, $\epsilon_2 = 10$ per cent, $\epsilon_3 = 5$ per cent, $\tau = 3$ Nyquist intervals presents a good compromise between picture quality and information savings.

The pictures taken with settings $\epsilon_1 = 5$ per cent, $\epsilon_2 = 10$ per cent, $\epsilon_3 = 7.2$ per cent, $\tau = 3$ Nyquist intervals could be taken as another compromise, with an emphasis on economy rather than on quality.

The reason that the picture quality does not improve much with decreasing thresholds can be explained simply: The sensitivity of the eye to phase errors in locating the edge points within one Nyquist interval is the major cause of the deteriorated appearance of the processed pictures. This ambiguity within one Nyquist interval will not improve much as we set the thresholds finer. One way to get better results would be to specify the location of the selected edge points more accurately than one Nyquist interval. Even a modest oversampling of a factor of two would be advantageous. In the next section it will be apparent that this operation will not increase the required channel capacity by more than 12 per cent in the worst case (scene D) but would be beneficial in locating the edge points within one-half Nyquist interval.

The above-mentioned phase errors are most disturbing on the vertical edges of scene A and on the outline of the face in scene B. For more detailed material this effect is much less objectionable.

IX. COARSE QUANTIZATION OF FAST-TRANSIENT REGIONS

Some recent work has exploited the same psychological phenomenon (that is, the insensitivity of the eye to the amplitude and shape of sudden brightness changes) from a different approach.[9,10,11] These authors quantized the amplitude of the samples in the region of fast transients into fewer levels than for the rest of the picture. We can add this feature advantageously to the linear interpolation between the edges. The obtained benefits are complementary: For pictures with many fast transients the number of selected samples is large, but these are just the samples which can be quantized in fewer number of bits. On the contrary, for pictures with fewer details (thus with fewer fast transients) we have to specify the selected samples more accurately, at least by 7-bit quantization.

We incorporated this feature in the linear interpolation system in the following way: The selected samples were divided in two categories. The first category contained those edge points which were no more than

TABLE III — RATIO OF COARSELY QUANTIZED SAMPLES TO SELECTED SAMPLES

Scene	System Setting	
	ϵ_1 = 3.6 per cent; ϵ_2 = 10 per cent; ϵ_3 = 5 per cent; $\tau = 3$; $t = 2$	ϵ_1 = 5 per cent; ϵ_2 = 10 per cent; ϵ_3 = 7.2 per cent; $\tau = 3$; $t = 2$
A	0.71	0.63
B	0.52	0.44
C	0.59	0.54
D	0.77	0.72

two Nyquist intervals from the left and the right neighboring edge points. These points thus belonged to high-frequency regions and were quantized coarsely. In the experiments, 3- and 4-bit (8- and 16-level) quantization was tried out. The remaining edge points which were the end points or inner points of low-frequency regions had to be quantized into finer steps. We used here 9 bits (512 levels), as in the linear interpolation system, but 7 bits would probably be very satisfactory. A 3-bit quantization for the fast-transient region turned out to be very noticeable, but 4-bit quantization gives quite satisfactory results, as the right columns of Fig. 3 and 4 show. The ratio of the coarsely quantized samples to the selected samples is given in Table III. Here t is the parameter which defines the fast-transient regions and, as mentioned, was set for two Nyquist intervals. According to this setting, edge points falling in regions which contained oscillation higher than half of the maximum frequency of the signal were coarsely quantized. We might have increased t even further from a psychological point of view, but the additional reduction in channel capacity would have been slight. In the following section we evaluate the obtained statistics and give the channel-capacity figures for possible coding schemes.

X. CODING AND AMOUNT OF CHANNEL-CAPACITY SAVINGS

After the processing of the pictures, the second step is a subjective evaluation of them. Provided we accept the obtained picture quality, the next step is to evaluate the information content and the obtained channel-capacity savings. Information theory enables us to get a theoretical lower bound of the information content of the processed pictures, but to realize it even approximately requires very involved coders an decoders. Therefore, we also make computations with simpler coding devices. Such devices do not make use of the obtained statistics of distances between edge points, but regard all possible distances as equally probable. Because the greatest distance between selected samples is restricted to 16 Nyquist intervals, 4 bits are required to specify the location of a selected sample from its previous neighbors. To describe the amplitude of the selected sample, 7 bits are adequate. Thus, 11 bits are required to specify the location and amplitude of a selected sample point. In conventional systems, 7 bits are enough to specify the amplitude of samples occurring at Nyquist rate.

Aside from the foregoing, the saving in the transmitted information obviously would be the ratio of the selected samples to the total number of samples. Because of the foregoing, the saving is diminished in the ratio of 11/7. If N is the total number of samples and N' is the number of selected samples, then the average rate of information is

$$R = 11 \frac{N'}{N} \text{ bits/sample.} \tag{4}$$

The coder contains a time-variable buffer storage to smooth out the incoming signals, which arrive at an irregular rate, and to transmit them on the channel at a constant average rate. At the receiver, the inverse elastic operation is performed in the decoder. If $N'/N < 7/11$, we get a saving in information rate over the conventional Nyquist rate sampling.

The rate of information is computed for the linear interpolation system with and without quantization. In the quantized case the required rate is

$$R_q = 11 \frac{N' - N^*}{N} + 8 \frac{N^*}{N} \text{ bits/sample,} \tag{5}$$

where N^* is the number of coarsely quantized samples.

If we take advantage of the highly peaked distribution curve of the distance between selected samples, and use a Shannon-Fano code to encode them, the rate of information for the linear interpolation system without and with quantization is as follows:

$$R_m = \frac{7 + H_X}{d}, \tag{6}$$

$$R_{mq} = \frac{4 + H_X}{d} k + \frac{7 + H_X}{d} (1 - k), \tag{7}$$

where

$$d = \frac{N}{N'}, \qquad k = \frac{N^*}{N}$$

TABLE IV — INFORMATION RATE FOR PROCESSED PICTURES (BITS PER SAMPLE)

Scene	System Setting							
	$\epsilon_1 = 3.6$ per cent; $\epsilon_2 = 10$ per cent; $\epsilon_3 = 5$ per cent; $\tau = 3$; $l = 2$				$\epsilon_1 = 5$ per cent; $\epsilon_2 = 10$ per cent; $\epsilon_3 = 7.2$ per cent; $\tau = 3$; $l = 2$			
	R	R_q	R_m	R_{mq}	R	R_q	R_m	R_{mq}
A	3.62	2.92	2.92	2.22	3.21	2.66	2.65	2.01
B	3.44	2.96	2.94	2.45	2.78	2.45	2.43	2.34
C	3.94	3.31	3.27	2.64	3.23	2.75	2.76	2.28
D	5.20	4.10	4.08	2.99	4.66	3.76	3.72	2.80

and

$$H_X = -\sum_{i=1}^{16} P_{Xi} \log_2 P_{Xi} \, .$$

The P_{Xi} are the frequencies of the distances between extremals of a given picture, X; R, R_q, R_m and R_{mq} are tabulated for different scenes and system settings in Table IV. The obtained information reduction is considerable, and it is an advantageous situation that the smallest entropy values, H_X, are obtained for the most involved pictures which require the most selected samples.

If we use statistical coding (e.g., Shannon-Fano or Huffman codes), we have to use the same code for all scenes. If we choose the code according to a scene Y, and we have to encode a different scene X, the expected code length in binary digits will be approximately

$$L_{XY} = -\sum_{i=1}^{16} P_{Xi} \log_2 P_{Yi} \, , \tag{8}$$

where L_{XY} is always greater than $L_{YY} = H_Y$. To see how these values compare with the entropies, we computed them for the 16 possible combinations of scenes using the finer threshold settings. Table V shows L_{XY}, which is not very sensitive to Y (i.e., to the particular code used).

TABLE V — EXPECTED CODE LENGTH L_{XY} IN BITS

Scene		Y			
		A	B	C	D
X	A	1.88	1.99	1.95	1.99
	B	2.50	2.37	2.41	2.51
	C	2.20	2.16	2.13	2.20
	D	1.71	1.75	1.70	1.65

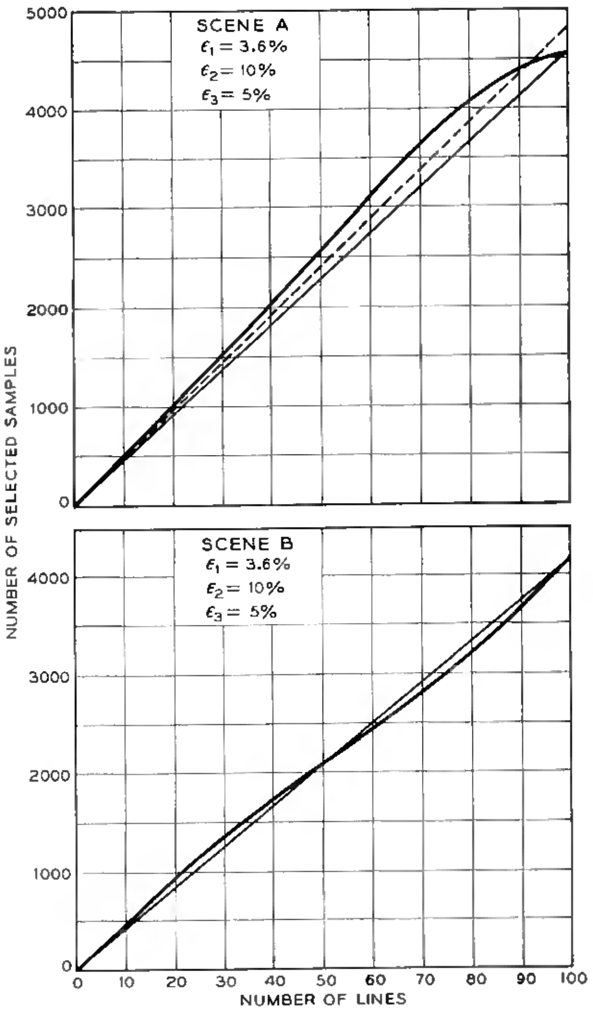


Fig. 9 — Fluctuation in time of number of selected samples at input of the buffer storage.

The channel capacity has to be enough to transmit all possible picture material. Because we do not know whether the selected few pictures are good representatives of all possible entertainment pictures, we cannot state theoretically anything definite about channel capacity, but we hope that the results are close to reality. If we regard the pictures as a whole, instead of as a 25th portion, and look at them from the usual four times picture height instead of from the distance of 20 times picture height, we can get an impression of how the quality would look for the most crowded scenes.

The picture materials used were not far from the stationary case; i.e., entropy values calculated from statistics gathered across different lines of a picture did not fluctuate much.

XI. BUFFER-STORAGE REQUIREMENTS

In Fig. 9 we show how the number of selected samples fluctuates in time at the input of the buffer storage (curved lines). If we read out the data at a constant rate, we get a straight-line representation as a function of time. If we choose this constant rate at the output as the average rate of the input, the straight line starts at the origin and hits the input curve at the end point. The maximal difference between the input and output curves gives a good estimate of the buffer-capacity requirements. The curves for scenes A and B are shown for the fine setting of the linear interpolation system without quantization or statistical coding. The coordinates are equivalent to time and are specified in terms of the number of scanned lines. The abscissae are the number of selected samples at the input and output of the quantizer, with the synchronization signals added. The requirement in storage capacity is about one scanning line (120 samples) for scene B and about four scanning lines for scene A. If an increased output rate (dashed straight line) is used, the storage-capacity requirement can be reduced.

XII. SUMMARY

The above-described experiments used the inability of the eye to notice the exact amplitude and shape of short brightness transients. By using straight-line interpolation between edge points and coarse quantization of edge points in fast-transient regions, we can transmit information at a rate of 3 bits per sample or less for the given scenes and shown picture quality. If we take the present 7 bits per sample rate as a reference, the greatest possible saving for scene D is $7/2.99 = 2.3$ times, and for scene B it is 2.9 times. Naturally, with practical buffer-storage size we cannot average out the differences in information rate for the different scenes, and we have to match the channel to the worst case.

If we use additional information to specify the location of edge points within a Nyquist interval, the quality of the pictures will greatly improve.

The obtained savings are modest and close to the figures achieved by other authors. Probably the results are interesting more because of what they reveal of visual perception than because of their immediate engineering applicability.

XIII. ACKNOWLEDGMENTS

The author is very much indebted to R. E. Graham and J. L. Kelly, Jr. for many helpful suggestions.

REFERENCES

1. Goodall, W. M., Television by Pulse Code Modulation, B.S.T.J., **30**, January 1951, p. 33.
2. Harrison, C. W., Experiments with Linear Prediction in Television, B.S.T.J., **31**, July 1952, p. 754.
3. Oliver, B. M., Efficient Coding, B.S.T.J., **31**, July 1952, p. 724.
4. David, E. E., Jr., Mathews, M. V. and McDonald, H. S., Experiments with Speech Using Digital Computer Simulation, I.R.E.-Wescon Conv. Rec. (Audio), August 1958, p. 3.
5. Graham, R. E. and Kelly, J. L., Jr., A Computer Simulation Chain for Research on Picture Coding, I.R.E.-Wescon Conv. Rec. (Computer Applications), August 1958, p. 41.
6. Mathews, M. V., unpublished manuscript.
7. Youngblood, W. A., Picture Processing, Quart. Prog. Rep., M.I.T. Res. Lab. of Elect., January 1958, p. 95.
8. Julesz, B., unpublished manuscripts.
9. Graham, R. E., Predictive Quantizing of Television Signals, I.R.E.-Wescon Conv. Rec. (Electronic Computers), August 1958, p. 147.
10. Kretzmer, E. R., Reduced Alphabet Representation of Television Signals, I.R.E. Conv. Rec., Part IV, 1956, p. 140.
11. Schreiber, W. F. and Knapp, C. F., TV Bandwidth Reduction by Digital Coding, I.R.E. Conv. Rec. (Information Theory), 1958, p. 88.

Equilibrium Delay Distribution for One Channel With Constant Holding Time, Poisson Input and Random Service

By PAUL J. BURKE

(Manuscript received February 19, 1959)

The equilibrium delay distribution is found for a single-server queueing system with Poisson input, random service and constant holding time. Curves are presented for various occupancy levels, and these are compared with their queued-service constant-holding-time and random-service exponential-holding-time counterparts.

In many situations involving waiting lines — for example, when customers are being served at a bargain counter in a crowded store — the ideal queue discipline (service order) of first-come first-served is not achieved. Instead, the service order tends to be at least somewhat random, and the probability of long delays is thereby increased over what it would be for the strict first-come first-served discipline. Unfortunately for the analyst, when the queue discipline is somewhere between order-of-arrival and random — as is often the case in practice — the problem of calculating the delay distribution seems to be intractable. If the service order is assumed to be actually random, however, then this problem can sometimes be solved, and the delay distribution thus found is useful as a kind of bound on the distributions to be expected in those cases where the queue discipline deviates from order-of-arrival service toward randomness.

The term "bound" as used here does not mean a bounding function in the strict sense. Actually, the delay distributions for models which differ only in queue discipline will cross each other, and hence no individual distribution can be a true bound for a family of such distributions. However, the longer delays are generally of more interest in waiting-line problems than are the shorter ones, and it is true that, other things being equal, the probability of sufficiently long delays is greater for random service than for order-of-arrival service.

Although the assumption of random service does not provide this type of bound in all situations, as would the assumption that the service order is last-come first-served, it is clearly more realistic than the latter assumption in many cases and will provide a closer bound, or approximation, to the actual delay distribution in these cases.

In addition to its usefulness as a boundary case, a queueing model which involves random service and constant holding times is of direct interest in certain telephone switching applications. For example, it is approximated, under some circumstances, at the marker connectors of the No. 5 Crossbar system, and it may have further application in electronic switching systems.

Of all possible holding-time distributions, the exponential and constant distributions have been studied most intensively in connection with queueing systems. The delay distribution was first obtained for a queueing system with constant holding times at least as early as 1909 by Erlang (Ref. 1, pp. 133–137). A solution to the delay problem for exponential holding times was published in 1917 by the same author (Ref. 1, p. 138–155). In both of these cases the service was order-of-arrival (queued) rather than random. The first attempt to obtain a delay distribution when the service order was random was published in 1942 by Mellor,[2] but this was not completely correct. A correct formulation of a random-service problem was obtained in 1946 by Vaulot,[3] the solution to which was given by Pollaczek.[4] The same problem also was solved by Palm.[5] A method for computing the delay distribution was published by Riordan in 1953.[6] The random-service problem solved by Vaulot and Pollaczek involved an exponential holding-time distribution. The present study is apparently the first attempt to combine the queue discipline of random service with holding times which are constant.

The model considered here is characterized by the following:

i. *Random input* — the probability that a call will arrive during any infinitesimal interval of length dt is proportional to dt within infinitesimals of higher order, and is independent of the state of the system, arrival times of previous calls or any other conditions whatever. It is equivalent to say that the call arrivals constitute a Poisson process.

ii. *Constant holding times* — the service time of each call is the same constant, taken here to be unity.

iii. *Random service* — if there are n calls waiting for service at the instant of a completion of service, the probability that any particular one of the calls will be served next is $1/n$. The server is never idle when there are calls waiting to be served.

iv. *No defections* — all calls wait in the system until they are served.

v. *Statistical equilibrium* — the distribution of the number of calls in the system is stationary, i.e., independent of time. Under the above assumptions, this will be the case when the arrival rate is less than one call per service interval and the system is given enough time to "settle down." Mathematically, the condition is assured by the assumption that the initial distribution of the number of calls in the system is the equilibrium distribution.

The over-all delay distribution is obtained below by decomposing it into a weighted sum of conditional delay distributions, depending on the state of the system at the epoch (instant) of the first departure (completion of service) following the arrival of the call. It suffices to define the state of the system at the departure epochs as the number of calls remaining in the system. (The call just completing service is not counted.) Each delay consists of two parts. The first part of the delay is the time from the arrival of the call in question until the first departure epoch following the arrival. The second part is the time from this departure epoch until the call in question gains service. The first part has a continuous distribution over the interval [0 — 1]; the second part is distributed over the nonnegative integers.

Thus, in Fig. 1 the call that arrives at time a_2 suffers a delay $d_1 - a_2$, which may vary from zero to a full holding time, until the first departure after its arrival. At time d_1 the call that arrived at a_2 will surely gain service, since it is the only call in the system at that time, and hence the integral part of the delay for this call will be zero units of time with probability one. In contrast, the call which arrives at a_3 will have to compete for service at d_2 with another call, and hence the integral part of its delay will have a probability of one-half to be zero and an equal probability to be greater than zero. In general, the integral part of the delay will have a (discrete) probability distribution that depends

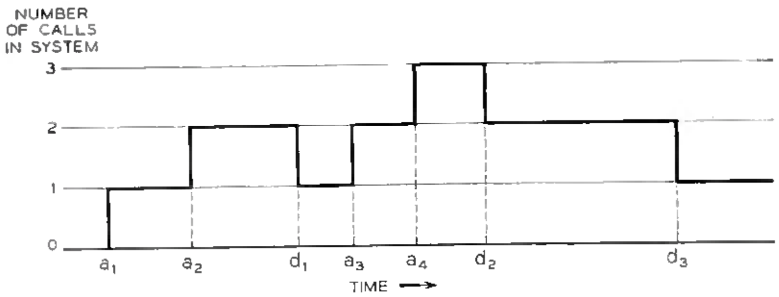


Fig. 1 — Number of calls in the system as a function of time.

on the number of calls in the system at the first opportunity that the call in question has to gain service. The determination of this probability distribution is the major portion of the task of evaluating the over-all delay distribution.

It might be pointed out that the equilibrium state probabilities are the same whether they are considered over the whole process or only over the set of discrete instants of time consisting of the departure epochs. This is shown by the fact that the generating function for the state probabilities given by Kendall,[7] which refers to the departure epochs, is the same, when specialized to constant holding times, as that of Crommelin[8] specialized to one server; and that the latter refers to the entire process.

What is needed now is the probability that a call, conditional on its being delayed, will be one of n calls remaining in the system at the first departure epoch after its arrival. It turns out that the latter probability is the same as the unconditional probability of $n - 1$ calls in the system, as shown by the following argument.

An arriving delayed call will be one of n calls in the system just after the next departure following its arrival in one of $n + 1$ mutually exclusive ways: there were k calls in the system at the last previous departure epoch before the arrival of the call in question and $n - k$ other calls arrived during the service interval, $k = 1, \cdots, n$; or there were zero calls in the system at the last previous departure epoch and $n - 1$ other calls (besides the call being served) arrived during the service interval. If $\Pr\{n \mid \lambda\}$ represents the desired probability and $P_k(\lambda)$ represents the unconditional probability of k calls in the system when λ is the arrival rate,

$$\Pr\{n \mid \lambda\} = [P_0(\lambda) + P_1(\lambda)]\, p(n-1, \lambda) + \cdots + P_n(\lambda)\, p(0,\lambda), \qquad (1)$$

where $p(k,\lambda)$, is an individual Poisson term, i.e., the probability of k arrivals during a service-time interval. However, (1) is exactly Crommelin's equation for P_{n-1} in the one-server case. Therefore

$$\Pr\{n \mid \lambda\} = P_{n-1}(\lambda). \qquad (2)$$

With the dependence on λ suppressed hereafter, let the conditional probability that the delay is not greater than t, given that the delayed call is one of n calls waiting for service at the first post-arrival departure epoch, be denoted $G(t \mid n)$. Let the resultant delay distribution for delayed calls be denoted $F(t)$. Then

$$F(t) = \sum_{n=1}^{\infty} P_{n-1}\, G(t \mid n). \qquad (3)$$

It remains to evaluate $G(t \mid n)$.

Owing to the queue discipline of random service, the delay suffered by a call between the instant of its arrival and the first post-arrival departure epoch is independent of the delay subsequent to this epoch. Also, it is known that the initial delay of a fractional part of a holding time is uniformly distributed over a service interval, owing to the property of random arrivals. Furthermore, conditioning on the number in the system at the first post-arrival departure epoch does not affect the independence property or the uniform distribution of the fractional delay (as would, for example, conditioning on the number in the system at the instant of arrival). Let the delay be represented by T, the integral part of T by T' and the fractional part by T''. Similarly, let the quantity t be decomposed into t' and t''. Then

$$\begin{aligned} G(t \mid n) &= \Pr\{T \leqq t \mid n\} \\ &= \Pr\{T' < t' \mid n\} + \Pr\{T' = t' \mid n\} \Pr\{T'' \leqq t''\} \end{aligned} \tag{4}$$

because of the independence of the integral and fractional parts of the delay. Or

$$G(t \mid n) = \sum_{i=0}^{t'-1} \Pr\{T' = i \mid n\} + t'' \Pr\{T' = t' \mid n\} \tag{5}$$

because of the uniform distribution of T''.

It is not difficult to write a formula for $\Pr\{T' = i \mid n\}$. First,

$$\Pr\{T' = 0 \mid n\} = \frac{1}{n},$$

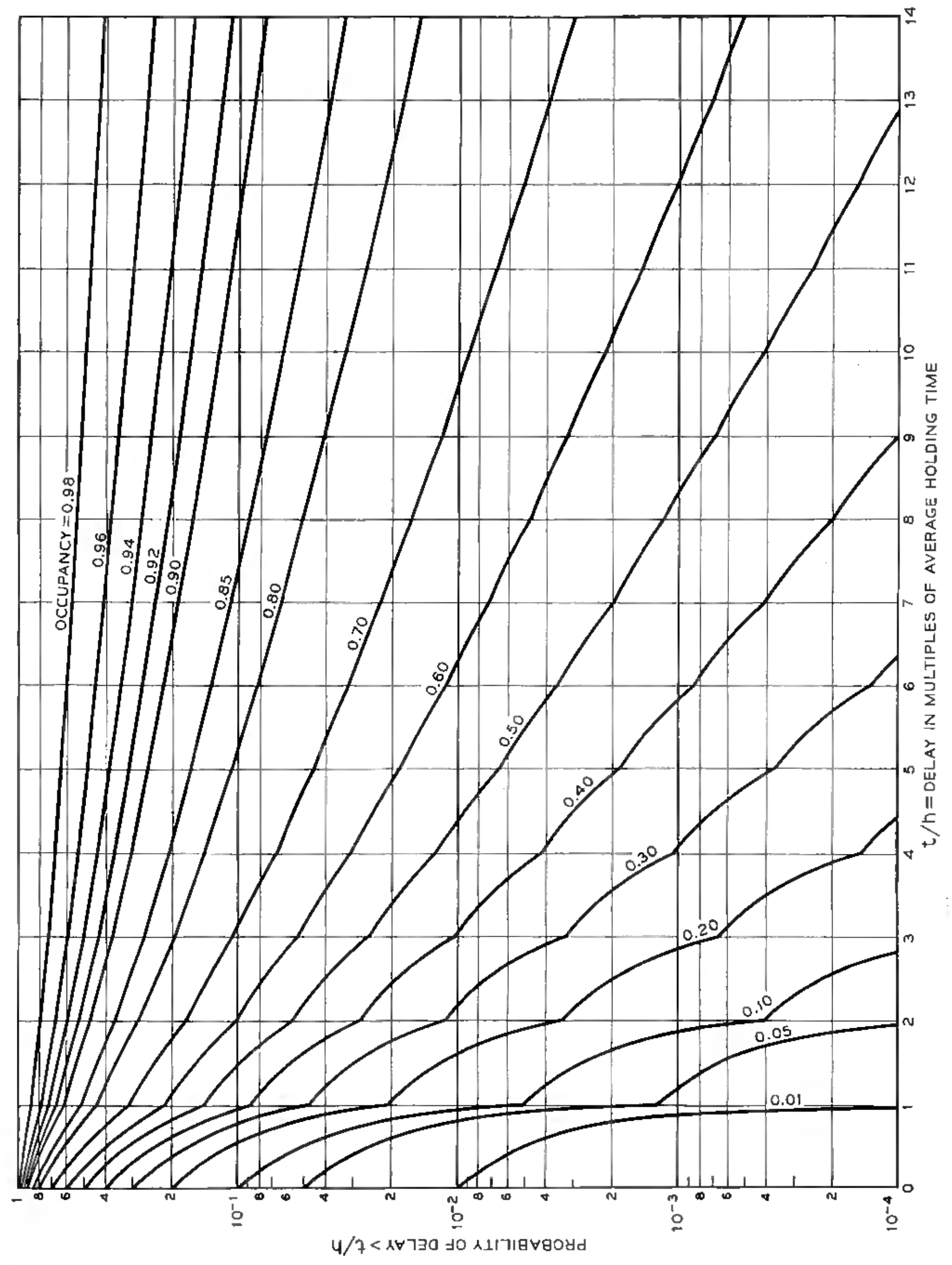
since, at the first post-arrival departure epoch, the delayed call is one of n calls equally likely to be served. Also

$$\Pr\{T' = 1 \mid n\} = \left(1 - \frac{1}{n}\right) \sum_{j_1=0}^{\infty} p(j_1) \frac{1}{n - 1 + j_1},$$

where $p(j_1)$ represents the Poisson probability of j_1 arrivals in a service interval, since, if the delayed call is not served at the first opportunity, any number of calls from zero upward may arrive during the next complete service interval. Extending this reasoning, one has for $i > 0$,

$$\begin{aligned} \Pr\{T' = i \mid n\} = \left(1 - \frac{1}{n}\right) \sum_{j_1 + \cdots + j_k \geqq k-n+1} \prod_{k=1}^{i-1} p(j_k) \\ \cdot \left[1 - \frac{1}{n - k + \sum_1^k j_r}\right] \frac{p(j_i)}{n - i + \sum_1^i j_r}. \end{aligned} \tag{6}$$

Although (6) can be written down directly, it is more convenient



OCCUPANCY = 0.98
0.96
0.94
0.92
0.90
0.85
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.05
0.01
t/h = DELAY IN MULTIPLES OF AVERAGE HOLDING TIME
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
PROBABILITY OF DELAY > t/h
1
8
6
4
2
10^{-1}
10^{-2}
10^{-3}
10^{-4}

computationally to use a recursive formula for the probabilities. (This was pointed out to the author by W. S. Hayward, Jr.) Let

$$\Pr\{T' = i \mid n\} = Q_i(n).$$

Then

$$Q_0(n) = \frac{1}{n}$$

and

$$Q_i(n) = [1 - Q_0(n)] \sum_{j=0}^{\infty} p(j) Q_{i-1}(n + j - 1). \tag{7}$$

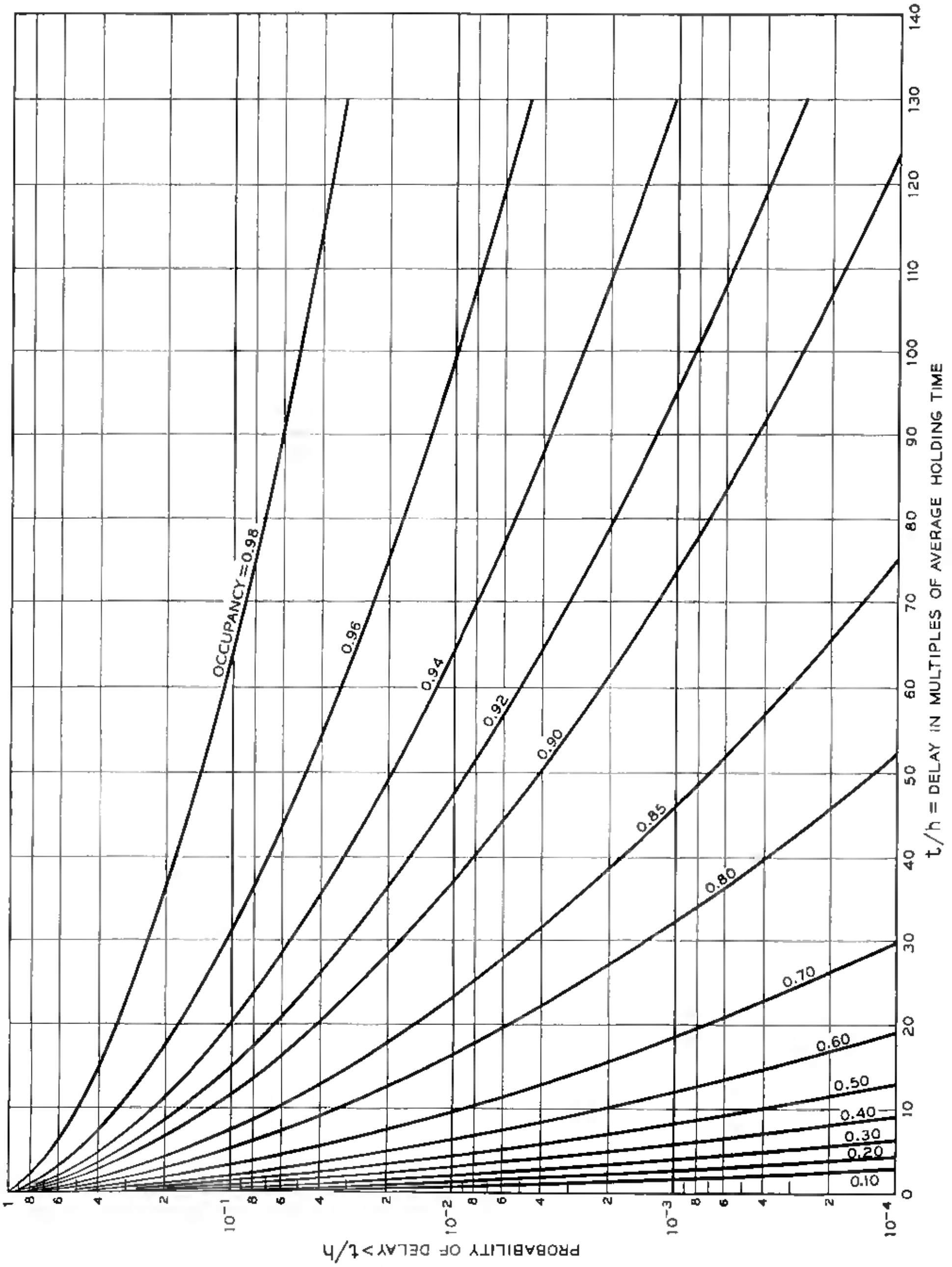
It is clear that (6) is the solution of (7). The delay distribution, $F(t)$, is obtained by substituting (6) into (5) and the latter into (3). The values of P_{n-1} necessary for evaluating (3) are obtained recursively from (1) and (2), together with the relation $P_0 = 1 - \lambda$.

The results of the calculations are shown as falling distributions of delays for all calls. That is, $\lambda[1 - F(t)]$ is plotted rather than $F(t)$, in keeping with custom in the field of delay theory. The distributions are shown in Fig. 2 for delays up to 14 holding times and, in Fig. 3 on a compressed scale, for delays up to 130 holding times.

As an example of the use of the curves, suppose a single marker whose holding time is 0.1 second serves calls at random and that it is desired to limit the probability of a delay greater than 2 seconds at this marker to 0.0001. It is required to find the permissible occupancy. Since a delay of 20 holding times is involved, Fig. 3 should be consulted. On this chart, the occupancy for a probability equal to 0.0001 of delay $t/h = 20$ is found to be just above 0.60, roughly 0.61. Thus, the marker can handle a random input averaging 6.1 calls per second.

In some applications in which service is not precisely order-of-arrival, it may be presumed that the delay distribution will lie between those for random and queued service. In such cases, the delay distributions will fall in a band bounded by random service and queued-service (Crommelin) curves. Examples of such bands are shown on Fig. 4. It should be noted that the bounding curves for any occupancy must cross, since the average delay is independent of the queue discipline.

It is of some interest also to compare the random-service delay distributions for constant and exponential holding times. It is conjectured that a pair of such curves for a given occupancy defines a band containing all random-service delay distributions, for that occupany, where the holding-time distribution is of gamma type in which the coefficient of


OCCUPANCY = 0.98
0.96
0.94
0.92
0.90
0.85
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
t/h = DELAY IN MULTIPLES OF AVERAGE HOLDING TIME
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
PROBABILITY OF DELAY > t/h
1
8
6
4
2
10^{-1}
10^{-2}
10^{-3}
10^{-4}

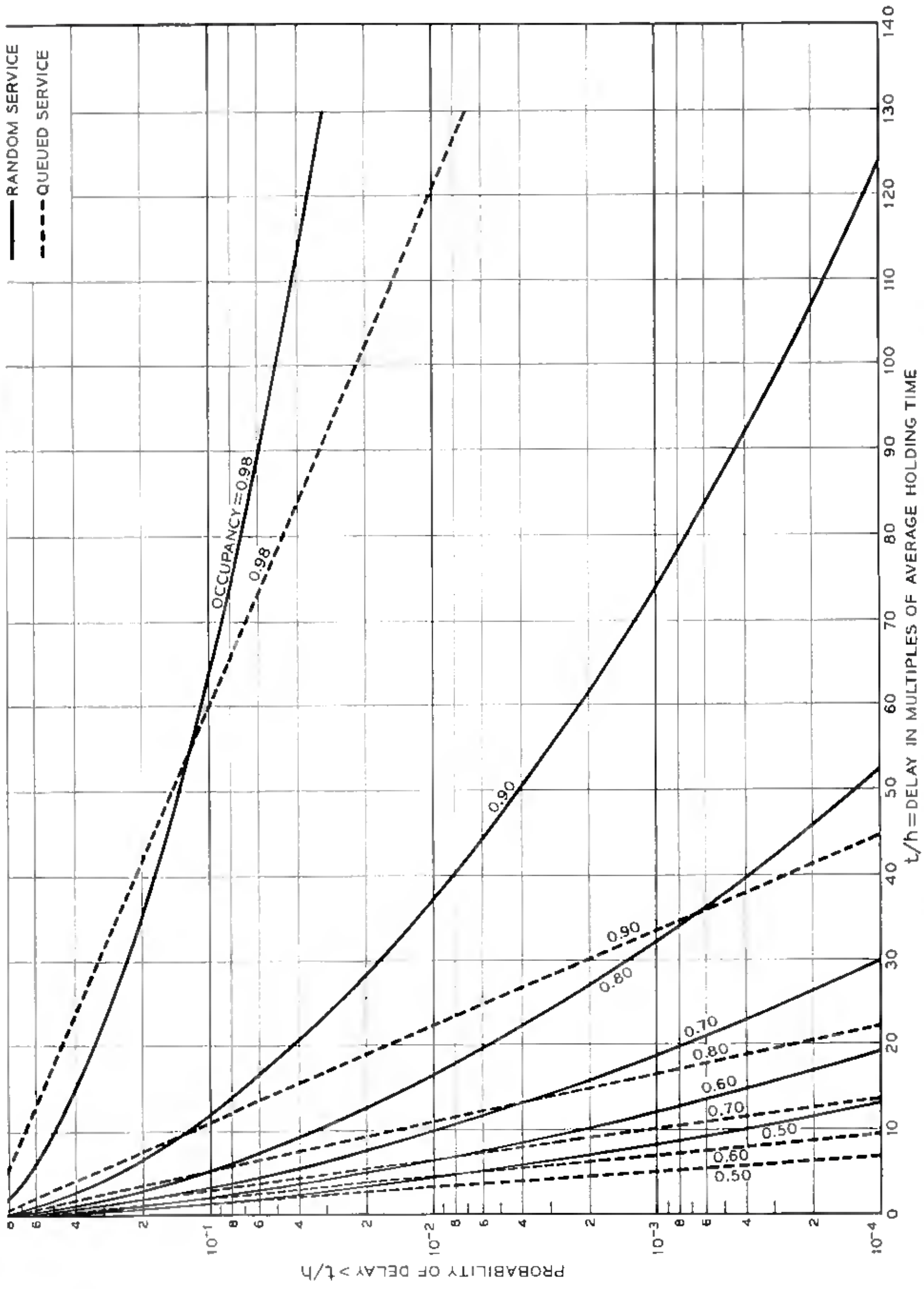
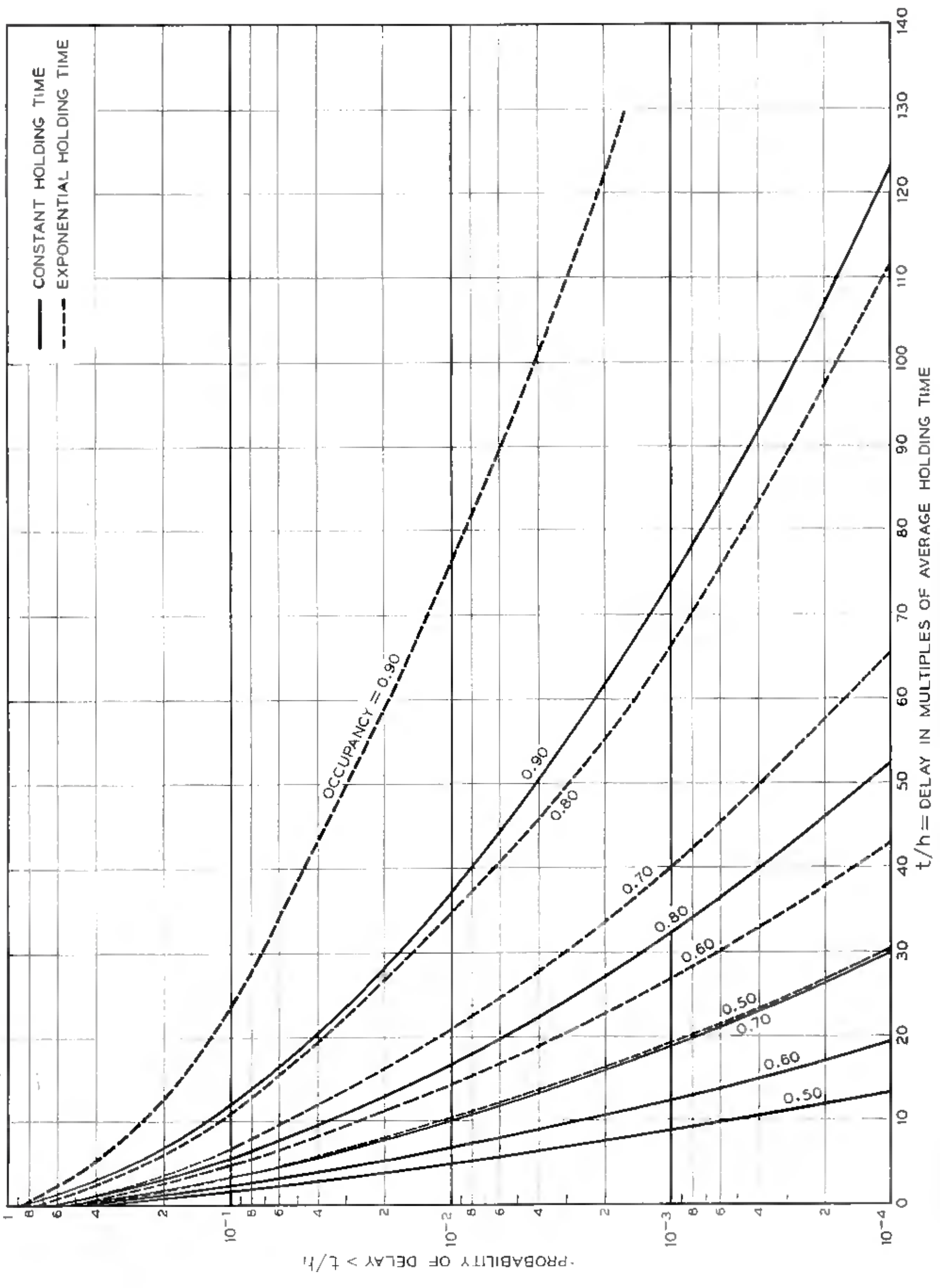


Fig. 4 — Comparison of random and queued service, with constant holding times.


CONSTANT HOLDING TIME
EXPONENTIAL HOLDING TIME
OCCUPANCY = 0.90
0.90
0.80
0.70
0.80
0.60
0.50
0.70
0.60
0.50
t/h = DELAY IN MULTIPLES OF AVERAGE HOLDING TIME
PROBABILITY OF DELAY > t/h
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
1
8
6
4
2
10^{-1}
10^{-2}
10^{-3}
10^{-4}

variation is not greater than unity. (In particular, the χ^2 distributions with two or more degrees of freedom are of this type.) Several such pairs of curves are shown on Fig. 5. (The exponential-holding-time curves are based on Wilkinson.[9]) Here, of course, the curves do not cross — the exponential-holding-time curves always (except at $t = 0$) lie above their constant-holding-time counterparts.

ACKNOWLEDGMENT

The extensive computations required for Figs. 2 through 5 were programmed by Miss Catherine Lennon, to whom the author expresses his gratitude.

REFERENCES

1. Erlang, A. K., *The Life and Works of A. K. Erlang*, The Copenhagen Telephone Co., Copenhagen, 1948.
2. Mellor, S. D., Delayed Call Formulae When Calls Are Served in a Random Order, P.O.E.E.J., **35,** 1942, p. 53.
3. Vaulot, E., Delais d'attente des appels téléphoniques traités au hazard, Comptes Rend., **222,** 1946, p. 268.
4. Pollaczek, T., La loi d'attente des appels téléphoniques, Comptes Rend., **222,** 1946, p. 353.
5. Palm, C., Väntetider Vid Slumpvis Avverkad Kö, Tekniska Meddelanden Från Kungl, Telegrafstyrelsen, Specialnummer för Teletrafikteknik, Stockholm, 1946, p. 70. (Translated in Tele, English Edition, No. 1, 1957, p. 68.)
6. Riordan, J., Delay Curves for Calls Served at Random, B.S.T.J., **32,** 1953, p. 100.
7. Kendall, D. G., Stochastic Processes Occurring in the Theory of Queues and Their Analysis by the Method of Imbedded Markov Chains, Ann. Math. Stat., **24,** 1953, p. 338.
8. Crommelin, C. D., Delay Probability Formulae When the Holding Times Are Constant, P.O.E.E.J., **25,** 1932, p. 41.
9. Wilkinson, R. I., Delayed Exponential Calls Served in Random Order, B.S.T.J. **32,** 1953, p. 360.

Evaluation of Solderless Wrapped Connections for Central Office Use

By S. J. ELLIOTT

(Manuscript received September 10, 1958)

In the development of solderless wrapped connections for telephone central office applications, the general reliability objective has been that the connections should remain mechanically secure and electrically stable during manufacture, shipment and installation and for 40 years thereafter in actual service. Destructive mechanical tests have been used to evaluate the mechanical properties of the connections. Combinations of elevated temperatures and mechanical disturbances have been used to accelerate the aging processes that tend to cause electrical instability. The results of such tests have provided considerable assurance that properly designed and properly made solderless wrapped connections will perform satisfactorily for 40 years in central office service.

I. INTRODUCTION

Since 1952, when the solderless wrapped connection first was described publicly,[1] its use in telephone central offices has grown steadily. Several hundred million solderless connections are being wrapped annually in the Bell System today, and the growth is continuing.

The tangible and immediate results of solderless wrapping have been gratifying. For example, the use of solderless wrapping has reduced manufacturing costs by speeding up many wiring operations and by reducing troubles caused by wire clippings and solder splashes. Furthermore, since solderless wrapping avoids the risk of damaging heat-sensitive insulation by soldering operations, it has made widespread use of plastic-insulated wire practicable, and this is leading to substantial additional savings.

In the end, however, these savings could be illusory if the use of solderless wrapped connections degraded telephone service or increased the maintenance effort required in the telephone plant. The laboratory evaluation of solderless wrapped connections has been continued, therefore, in order to assess the risk of deterioration in service and to provide guidance for the design of connections that are most likely to be reliable. This work has revealed certain limitations of solderless wrapped connections, but, at the same time, it has provided considerable assurance that properly designed and properly made connections will be reliable in central office service.

Many persons have inquired about the methods used for evaluating the capabilities of solderless wrapped connections and about the results that have been obtained since publication of earlier articles.[2,3] Since the inquiries continue undiminished year after year, it appears that there is sufficient interest in solderless wrapping to warrant another paper on the subject. An attempt is made here, therefore, to bring the story on evaluation up to date.

1.1 *Description of Solderless Wrapped Connection*

A solderless wrapped connection is made by wrapping a wire tightly around a terminal, and the connection is held together thereafter by the elastic stresses left in the two members. A typical solderless wrapped connection is shown in Fig. 1.

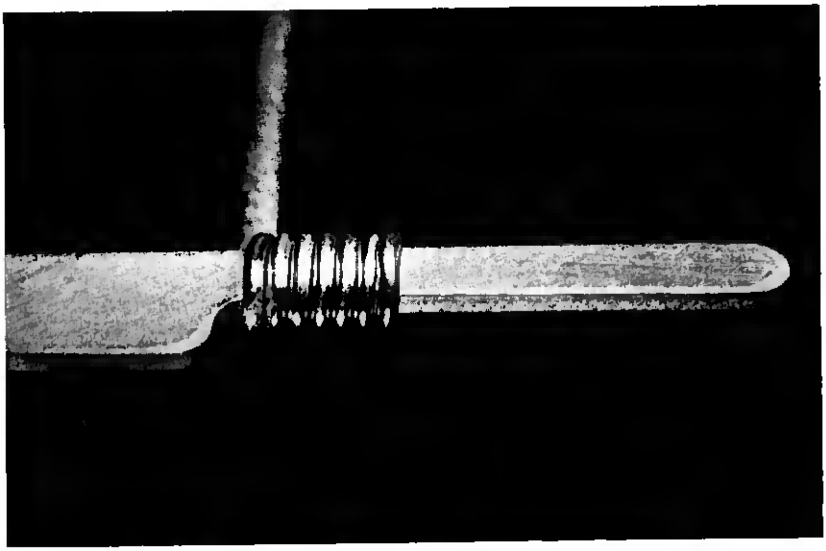


Fig. 1 — Typical solderless wrapped connection.

For Bell System applications, a minimum of five turns of wire is specified when No. 22 gauge wire is used, and a minimum of six turns is specified when No. 24 gauge wire is used. The wire should be closely wound on the terminal, but turns of wire should not overlap.

Only solid copper wire has been approved for solderless wrapping. The use of stranded wire presents a number of difficulties and has not been investigated in detail.

1.2 *Reliability Objectives*

The general reliability objective in the development of solderless wrapped connections has been that the connections should remain mechanically secure and electrically stable during manufacture, shipment and installation and for 40 years thereafter in telephone central office service.

The mechanical security objective cannot be defined in absolute terms, for too little is known about the magnitudes and distributions of the disturbances to which wires are subjected in service. As a comparative objective, however, solderless wrapping should not increase the occurrence of broken wires and loose connections beyond the levels that now prevail with soldered connections.

The electrical stability objective can be stated more specifically. Not more than one connection in 10,000 should exhibit resistance fluctuations greater than 0.1 ohm in service. The electrical noise produced by resistance fluctuations of this magnitude is considered to be at the threshold of transmission impairment in the most critical transmission circuits now in service.

II. ELECTRICAL MEASUREMENTS

The method used for checking the electrical stability of solderless wrapped connections has been described fully by Van Horn.[2] In essence, it consists of a resistance measurement made by the familiar voltmeter-ammeter method. The voltage drop across the connection is measured with a moving coil millivoltmeter while a direct current of 0.1 ampere flows through the connection.

The measurement of interest is the *variation* of resistance when the connection is disturbed mechanically. The disturbance is created by plucking the terminal — that is, by slowly deflecting the free end of the terminal a predetermined distance and then releasing it abruptly, allowing terminal and connection to vibrate freely. The resistance variation

(ΔR) is the difference between the maximum and minimum resistance values observed during the disturbance.

Although the moving-coil millivoltmeter is too sluggish to follow rapid fluctuations of resistance, the measurement is surprisingly sensitive. In a series of measurements made by an appropriate electrical noise meter while connections were subjected to vibration of the sort encountered in service, the measured noise levels consistently were lower than the levels calculated from the results of the simple ΔR measurements. It was concluded that the simple ΔR measurement would be adequate for routine development tests and, since it could be made far more rapidly than any of the more refined measurements that had been explored, it was adopted.

2.1 *Analysis of ΔR Measurements*

Before reviewing the accelerated aging tests that have been employed in the evaluation of solderless wrapped connections, it is necessary to describe the method that will be used to summarize the results. This method, a product of hindsight rather than foresight, has been used for only a short time in development of the solderless wrapped connection.

A major problem in the evaluation, of course, has been to demonstrate by means of comparatively small samples that the probability of ΔR exceeding 0.1 ohm is no more than one in 10,000. The expense of testing large enough samples to determine the frequency distribution of ΔR in every case would have been prohibitive. Over a period of years, however, a moderately large number of tests were made on a few particular types of connections. The cumulative sample sizes in those cases, although still small compared with 10,000, were large enough to warrant attempts at curve fitting.

As measured by the chi-square test, the expression that seems to fit the observed distributions best is obtained by first grouping the ΔR data in cells as follows:

Cell Number, x	ΔR (Ohms)	Number of Connections in Cell
0	$0 \leqq \Delta R \leqq 0.001$	n_0
1	$0.001 < \Delta R \leqq 0.01$	n_1
2	$0.01 < \Delta R \leqq 0.1$	n_2
3	$0.1 < \Delta R \leqq 1.0$	n_3
...	...	...
etc.	etc.	etc.

The arithmetic mean of the grouped distribution then is calculated as follows:

$$\begin{aligned} m &= \frac{0n_0 + 1n_1 + 2n_2 + 3n_3 + \cdots}{n_0 + n_1 + n_2 + n_3 + \cdots} \\ &= \frac{n_1 + 2n_2 + 3n_3 + \cdots}{N}, \end{aligned} \tag{1}$$

where N is the total number of connections in the sample.

Once the mean, m, has been calculated, the probability of finding a connection in cell x can be expressed as

$$P_x = \frac{m^x e^{-m}}{x!}. \tag{2}$$

Bearing in mind that $0! = 1$, the probabilities associated with the first few cells become

$$\begin{aligned} P_0 &= e^{-m}, \\ P_1 &= mP_0, \\ P_2 &= \frac{mP_1}{2}, \\ P_3 &= \frac{mP_2}{3}. \end{aligned} \tag{3}$$

The reader may recognize that (2), the general expression for P_x, is the same as the expression for the Poisson distribution. The physical interpretation, however, is different. In the Poisson distribution, P_x is the probability that x defectives will occur in a random sample of size N if m is the expected average number of defectives in a sample of that size. As a description of the ΔR distribution, however, P_x is the probability that ΔR, for a single connection chosen at random, will fall between the limits defined by the cell number x. For a random sample of N connections, then, the number of connections expected to fall in cell number x will be NP_x.

In general, the agreement between (2) and observed ΔR distributions has been best for the more stable types of connections — types in which high values of ΔR rarely occur and for which m is small. These, of course, are the types of connections that are desirable. For less stable types of connections — the types that would be rejected as unreliable — the

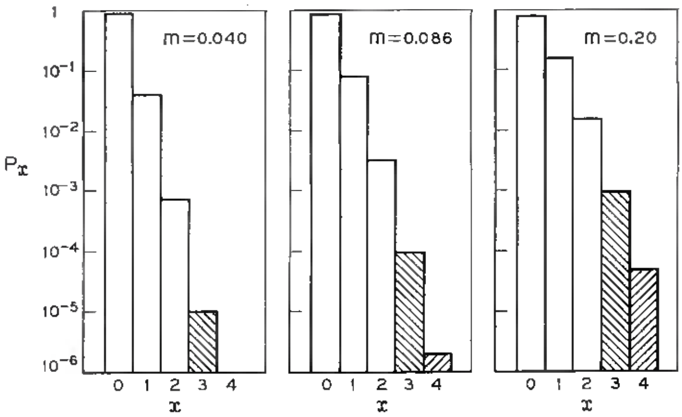


Fig. 2 — Examples of the ΔR distribution defined by (2). Shading indicates cells for which ΔR exceeds 0.1 ohm.

agreement has been poorer. The dividing line between good agreement and poor agreement appears to be somewhere in the vicinity of $m = 0.3$.

The mean, m, is a convenient statistic to use in summarizing the results of a group of ΔR measurements, and it will be used for that purpose in the discussion of accelerated aging tests. Qualitatively, low values of m indicate stable connections and high values indicate unstable connections. Quantitatively, in those cases where m is less than about 0.3, m defines the observed ΔR distribution very effectively.

The distribution corresponding to $m = 0.086$ (shown in Fig. 2) is of particular interest. It represents the case for which there is one chance in 10,000 that ΔR will exceed 0.1 ohm. Consequently, a value of $m = 0.086$ *in service* is the maximum value that will satisfy the stability objective for central office use of solderless wrapped connections in the Bell System.

One more point needs to be made before proceeding to the discussion of accelerated aging tests. The variance of the distribution described by (2) is equal to the mean, m. It follows, then, that the standard deviation is equal to the square root of m. This relationship will be used later in developing criteria for deciding whether or not connections are stable enough to be approved.

III. ACCELERATED AGING TESTS

Fairly early in the development of solderless wrapped connections, it was concluded that electrical instability, if it occurred at all, would result principally from relaxation of stresses in the wire and terminal — the stresses that hold the connection together. Since that time, all aging tests employed in the evaluation of solderless wrapped connections have included elevated temperatures to accelerate the relaxation process.

The work of Mason[2,3] and others indicated that the stress in copper wire (measured with respect to its value one day after a connection was wrapped) would relax about 40 per cent in 40 years at room temperature. To allow for some error in the room temperature prediction, and to allow for the fact that temperatures in enclosed equipments may approach 55°C in some cases, it has been assumed that the stresses in solderless connections wrapped with copper wire may relax as much as 50 per cent under central office conditions. That has been the degree of stress relaxation aimed at in the various accelerated aging tests.

One of the early tests consisted simply of baking the connections for three hours at 175°C. (Mason had shown that the stresses in connections wrapped with copper wire would relax 50 per cent in about $2\frac{1}{2}$ hours at 175°C, and the extra half-hour was needed to bring the specimens from room temperature up to the oven temperature.) At the end of the heat treatment, the connections were cooled to room temperature and then were checked for electrical instability. The general experience was that any solderless wrapped connection which was stable before the heat treatment still would be stable after the heat treatment.

Although such results were encouraging, they were, at the same time, disconcerting. For example, some of the connections that behaved so well in the 175°C test were wrapped on terminals that scarcely could be considered good mechanical structures for supporting stresses over long periods. The twin-wire terminal of the wire-spring relay, in particular, fell in the questionable category. At that time it consisted simply of two parallel nickel silver wires which were bonded together by being dipped in soft solder and then serrated. The parallel wires by themselves did not have sufficient torsional stiffness to support the stresses that are required to hold a solderless wrapped connection together; and soft solder, because of its cold flow properties, is a notoriously poor material for supporting stresses. It was doubtful that the soft solder, at room temperature, could maintain the stresses long enough at levels high enough for solid state diffusion to occur. Since 175°C was not far below the softening temperature of the solder, there was at least a suspicion that something akin to a soldered joint had been produced in the accelerated aging test.

There were questions, also, about the metallurgical behavior of the copper wire at the elevated temperature. It was known, for example, that recrystallization is more likely to occur in copper at 175°C than at lower temperatures.

In short, it was suspected that a temperature as high as 175°C might do more than accelerate the aging process: it might alter the physical nature of the aging process itself. If this were so, then the 175°C test might give a false picture of the aging that would occur in actual service

3.1 *105°C Aging Test*

It was decided, therefore, that some exploratory aging tests should be run at a substantially lower temperature. For several reasons, a temperature of 105°C was chosen. It was well below the softening temperature of tin-lead solders; it was low enough so that there should be little likelihood of recrystallization occurring in the copper wire; yet it was high enough to produce the desired stress relaxation in a few months. Mason's work had indicated that the stresses in the wrapped connections would relax 50 per cent in about 150 days at 105°C, so the test period was set at 150 days, or approximately five months.

As compared with the desired service life of 40 years, the three-hour 175°C test represented a 100,000:1 acceleration of the stress relaxation process. Since a five-month 105°C test would represent an acceleration of only 100:1, it was expected to be a far more realistic aging test.

Five months is a long time, however, to wait for test results. It was decided, therefore, that the test connections should be measured periodically during the aging test, so that any instability would be detected as soon as it occurred. It is important to note that this decision introduced two more changes in the accelerated aging test: (a) it subjected the connections to temperature cycling, for they were removed from the oven and allowed to cool whenever they were measured, and (b) it subjected the connections to mechanical disturbances while they were being aged, for the terminals were plucked whenever resistance variations were measured.

The first connections subjected to the 105°C aging test were wrapped on the solder-dipped twin-wire terminals described earlier. Connections of that type had survived the 175°C test without showing any evidence of instability. In the 105°C test, however, they soon began to exhibit measurable resistance fluctuations, and they grew more and more unstable as the test continued. The history of that first group of connections is plotted in terms of the statistic m in Fig. 3. It was evident that the 105°C aging test with periodic cycling and measurement was more severe than the undisturbed 175°C test.

Subsequently an experiment was performed to compare the relative effects of temperature, temperature cycling and mechanical disturbance. One group of connections was aged at 105°C, cooled to room temperature once each week and measured (plucked) while at room temperature

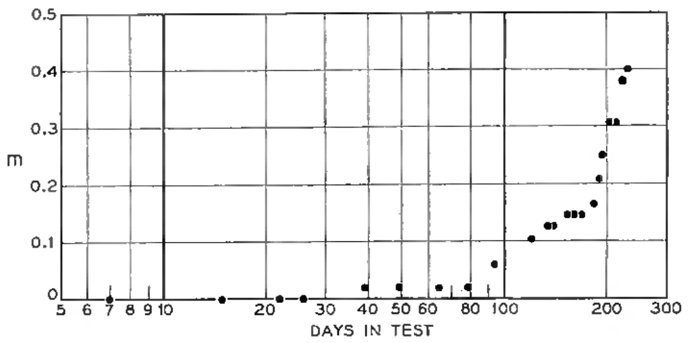


Fig. 3 — Effect of 105°C accelerated aging test on solderless connections wrapped with No. 24 gauge copper wire on solder-dipped twin wire terminals of wire-spring relay. Sample size 48.

on alternate weeks. At the end of 150 days, the value of m for that group was 2.1.

A second group of similar connections was aged at 105°C and cooled to room temperature once each week, but was not disturbed by measurements. After 150 days, m was 0.2.

A third group was aged continuously at 105°C without either temperature cycling or mechanical disturbance. After 150 days, m was 0.04.

A fourth group was maintained continuously at room temperature but measured every two weeks. After 150 days, m was 0.02.

Three important inferences were drawn from the results of this experiment:

i. Although solid state diffusion may occur during undisturbed aging of solderless wrapped connections, thus tending to offset the detrimental effects of stress relaxation, repeated mechanical disturbances during the aging process can impede diffusion and can lead to unstable connections.

ii. Since connections may be disturbed from time to time in service, accelerated aging tests should include periodic mechanical disturbances.

iii. Temperature cycling alone can produce a form of mechanical disturbance if the temperature coefficients of expansion of wire and terminal are not equal.

Another related experiment was performed to study the effects of the frequency with which connections were disturbed during the 105°C aging test. The more frequently the connections were disturbed, the sooner they became unstable. A few of the test results are shown in Fig. 4 to illustrate the pattern.

Long before these two experiments were completed, it had been neces-

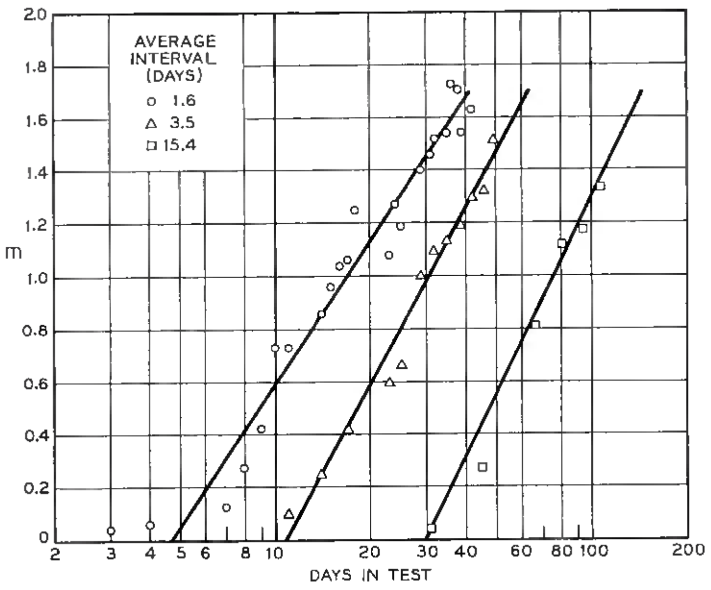


Fig. 4 — Effects of varying the frequency of disturbance in 105°C accelerated aging test. Connections wrapped with No. 24 gauge tinned copper wire on 0.010 × 0.062-inch flat nickel silver terminals. Sample size 48 in each case.

sary to standardize an accelerated aging test so that the development of suitable terminals could proceed without further delay. The 150-day 105°C test was adopted, primarily because it was the only aging test up to that time that had revealed highly significant differences among various types of connections — differences that usually were consistent with qualitative analyses of the various mechanical structures. The connections were cooled to room temperature once each week, and their resistance variations were measured at room temperature every other week. Each connection was plucked two or three times during the ΔR measurement. Eventually, a plucking amplitude of $\frac{1}{32}$ inch was standardized.

A standard procedure for the preparation of test specimens also evolved gradually. It has become the usual practice now, in preparing each group of test connections, to use two wrapping bits (one that wraps more tightly than the average bit, and one that wraps less tightly), two wrapping tools (one power-driven and the other manually operated), two grades of wire (one comparatively hard and the other comparatively soft) and two operators. All 16 combinations of the four parameters now are included at least twice in each test.

3.2 *Results of 105°C Accelerated Aging Test*

Altogether, more than 11,000 solderless wrapped connections have been subjected to the 105°C accelerated aging test. Most of the tests have been aimed at the problem of finding terminal configurations which would yield reliable connections and which, at the same time, could be manufactured economically. The results of a number of those tests are summarized in Table I, along with the results of a screening test that is described in Section 3.4.

Fig. 5 shows the results that were obtained with No. 24 gauge tinned copper wire wrapped on terminals punched from several widely used thicknesses of nickel silver stock. The performance of the thinner terminals was considered to be unsatisfactory. Various types of longitudinal embossing were investigated to find means for improving the thin terminals, and the form shown in Fig. 6 finally was standardized. Although this form is not necessarily optimum, it is a convenient form to manufacture; and it has behaved well in the accelerated aging test, as shown in Table I.

TABLE I — RESULTS OF ACCELERATED AGING TESTS ON SOLDERLESS CONNECTIONS WRAPPED WITH TINNED COPPER WIRE

Type of Terminal	Material	Finish	105°C Aging Test				175°C Screening Test			
			No. 22 Wire		No. 24 Wire		No. 22 Wire		No. 24 Wire	
			N	*m*	*N*	*m*	*N*	*m*	*N*	*m*
Rectangular										
0.045 × 0.045	Brass	Electrotin	136	0	296	0	168	0	328	0.009
0.0319 × 0.062	Brass	Electrotin	152	0.026	152	0.020	160	0	160	0
0.0319 × 0.062	Nickel Silver	None	152	0	104	0	976	0.15	2050	0.078
0.0253 × 0.062	Nickel Silver	None	—	—	24	0.053	105	0.37	93	0.13
0.0201 × 0.062	Nickel Silver	None	—	—	144	0.084	116	0.88	342	0.16
0.0159 × 0.062	Nickel Silver	None	—	—	24	1.2	—	—	121	0.38
0.0126 × 0.062	Nickel Silver	None	—	—	24	1.8	—	—	203	0.40
0.010 × 0.062	Nickel Silver	None	—	—	96	2.1	—	—	1164	0.42
Embossed										
0.0253 × 0.062	Nickel Silver	None	—	—	—	—	136	0.31	137	0.007
0.0201 × 0.062	Nickel Silver	None	—	—	64	0	—	—	274	0.16
0.0159 × 0.062	Nickel Silver	None	—	—	—	—	—	—	131	0.084
0.0126 × 0.062	Nickel Silver	None	—	—	64	0	—	—	131	0.015
0.010 × 0.062	Nickel Silver	None	—	—	64	0	—	—	180	0.11
Wire-Spring Relay										
Single Wire	Nickel Silver	None	144	0.12	288	0	—	—	144	0.010
Twin Wire	Nickel Silver	None	144	0.021	513	0.006	—	—	749	0.064

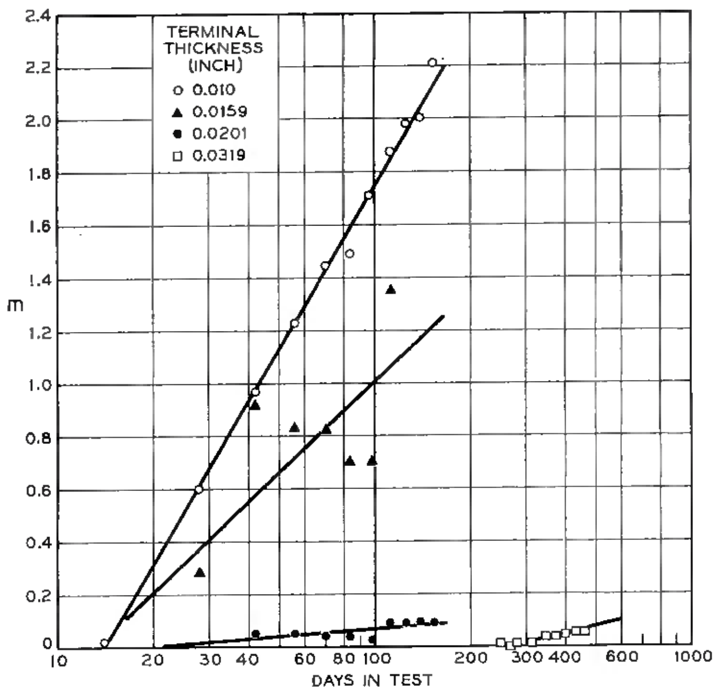
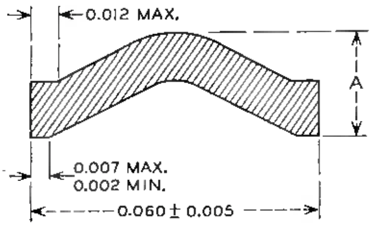


Fig. 5 — Results of 105°C accelerated aging tests. All terminals flat nickel silver, 0.062 inch wide. Connections wrapped with No. 24 gauge tinned copper wire.

The early single-wire terminal of the wire-spring relay was formed from a round wire by flattening, serrating on one side and solder-dipping. Its performance was unsatisfactory, but it was improved simply by omitting the solder. Its present form is shown in Fig. 7.

The twin-wire terminal of the wire-spring relay was more of a problem. Many designs were conceived and investigated, but most of the



NOMINAL THICKNESS OF STOCK (INCH)	DIMENSION A (INCH)
0.010	0.021 ± 0.002
0.013	0.023 ± 0.002
0.016	0.025 ± 0.002
0.020	0.026 ± 0.002

Fig. 6 — Embossing approved for Bell System terminals.

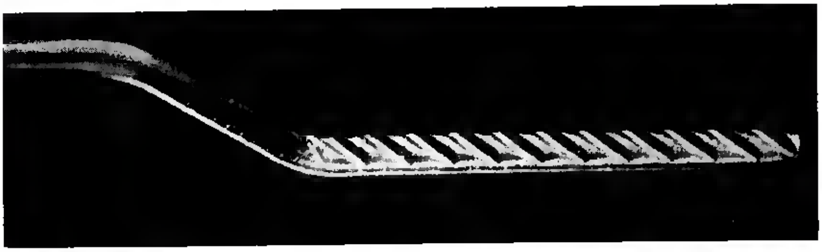


Fig. 7 — Single-wire terminal of wire-spring relay.

designs that showed promise from the solderless wrapping standpoint were objectionable from the manufacturing standpoint. In the end, a compromise design was adopted. The twin wires are twisted tightly together, then they are coined in a closed die which has a trapezoidal cross section. The resulting terminal is shown in Fig. 8.

Although most of the 105°C tests have been made with No. 24 gauge wire, a number of tests have been made also with No. 22 gauge wire. As shown in Table I, the results indicate that No. 22 gauge connections on the heavier terminals are as stable as No. 24 gauge connections, but that No. 22 gauge connections on the lighter terminals are inferior.

The aging tests of No. 26 gauge connections are not completed yet. The preliminary results indicate, however, that No. 26 gauge connections, even when wrapped with as many as nine turns of wire, are less stable than six-turn No. 24 gauge connections on the same types of terminals.

For certain types of wiring in the Bell System, it has been standard practice to use untinned copper wire. Solderless connections wrapped with untinned wire, however, tend to deteriorate sooner than connections wrapped with tinned wire. Fig. 9 illustrates results obtained with untinned wire.

The data that have been presented so far should be sufficient to provide a bird's-eye view of the results that have been obtained in the

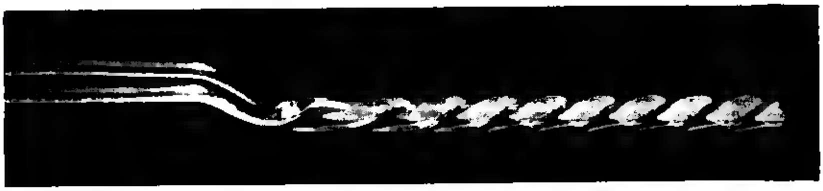


Fig. 8 — Twin-wire terminal of wire-spring relay.

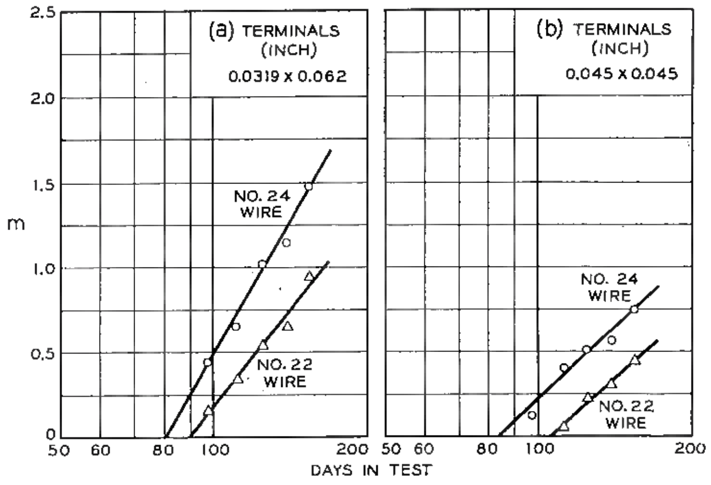


Fig. 9 — Effect of 105°C accelerated aging test on connections wrapped with untinned copper wire on electrotinned brass terminals. Sample size 32 in each case.

105°C aging test. Other alleys and byways have been explored, but the picture would not be sharpened perceptibly by presenting more details. It is more pertinent at this point to consider what use can be made of the test results.

3.3 *Criterion for Acceptance (105°C Aging Test)*

The purpose of the accelerated aging test, of course, is to distinguish between those types of solderless wrapped connections which are likely to satisfy the 40-year stability objective in service and those types which are not likely to satisfy the objective. The test results supply a reasonably clear picture of the relative instability of the several types of connections, but this by itself is not enough. Somewhere on the instability scale the development engineer eventually must draw a line and say, at least to himself, "I will approve the use of types of connections that fall below this line; I will not approve types that fall above it." In other words, he must establish a criterion for acceptance.

The criterion for acceptance has not remained static as the development of solderless wrapped connections progressed. Instead, it has been revised several times as the aging test evolved and as the information obtained from aging tests expanded. Its present form has considerably more meaning and is more convenient to use than some of the earlier forms.

The criterion for acceptance is based upon two premises: (a) that the 105°C accelerated aging test does, in fact, produce at least as much instability as 40 years of central office service will produce and (b) that (2) is an adequate description of the ΔR distribution. Although final confirmation will not be available until about 1990, evidence that these premises are valid is increasing year after year.

It was stated previously that the value of m should not exceed 0.086 *in service* if the resistance variation of not more than one connection in 10,000 is to exceed 0.1 ohm. If the 105°C test is an adequate simulation of service conditions, then those types of connections which have values of m below 0.086 in the 105°C test should be acceptable. This would be the criterion for acceptance if very large samples were tested. Where small samples are tested, however, it is prudent to make allowance for sampling errors.

As stated earlier, the variance of the distribution described by (2) is equal to m. Consequently, the standard deviation of the distribution is equal to the square root of m. If a large number of samples of size N are drawn from a population whose true mean is m', then roughly five per cent of the sample means can be expected to fall below the value

$$m_{0.05} = m' - 1.645\sqrt{\frac{m'}{N}}. \tag{4}$$

A reasonable acceptance level, then, is

$$\begin{aligned} m_{0.05} &= 0.086 - 1.645\sqrt{\frac{0.086}{N}} \\ &= 0.086 - \frac{0.482}{\sqrt{N}}. \end{aligned} \tag{5}$$

In other words, the stability of the connections will be considered acceptable if the mean, m, of a sample of N connections is less than the value of $m_{0.05}$ given by (5). The acceptance level is plotted as a function of N in Fig. 10.

As an aid to decision making, it also is possible to set up a criterion for rejection. If a large number of samples of size N are drawn from a population whose true mean is m', then roughly 95 per cent of the sample means can be expected to fall below the value

$$m_{0.95} = m' + 1.645\sqrt{\frac{m'}{N}}. \tag{6}$$

A reasonable rejection level for the case $m' = 0.086$, then, is

$$m_{0.95} = 0.086 + \frac{0.482}{\sqrt{N}}. \tag{7}$$

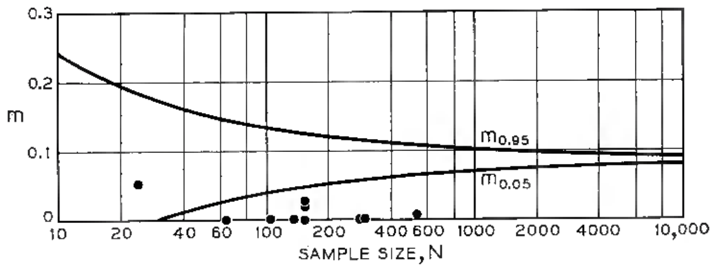


Fig. 10 — Criteria for acceptance and rejection in 105°C accelerated aging test. Points are results of 105°C tests on approved types of solderless wrapped connections.

In other words, the stability of the connections will be considered unacceptable if the mean, m, of a sample of N connections exceeds the value of $m_{0.95}$ given by (7). Further testing of that particular type of connection is not very likely to be profitable, so it might as well be rejected.

Establishing a rejection limit that does not coincide with the acceptance limit provides a zone of uncertainty in which the connections are neither clearly acceptable nor clearly unacceptable. It recognizes the possibility that further testing of a marginal type of connection might demonstrate that the type is acceptable. If the value of being able to approve that type of connection outweighs the cost of further testing, then it may be worthwhile to continue.

On the average, the acceptance criterion of (5) is conservative. Not only does it provide margin for moderate sampling errors, it also provides some margin, on the average, for an error in the basic premise that the 105°C test adequately simulates aging in service.

At the same time, the form of (5) is helpful to the experimenter, for it tells him the minimum sample size that can be used as the basis for acceptance. By setting $m_{0.05}$ equal to zero in (5), the corresponding minimum sample size is found to be 32. If he tests 32 connections and m turns out to be zero, he is entitled to approve the connections without further testing. With a smaller sample, he would not be entitled to approve them even though m turned out to be zero.

From a practical standpoint, it is prudent to test a larger sample when approval is needed in the shortest possible time. If ΔR for even a single connection exceeds 0.001 ohm in a sample of 32, then the connections cannot be approved, the test has to be expanded, and the final decision is delayed. A practical compromise is to test a sample large enough so that four connections could exceed 0.001 ohm slightly without causing m to exceed $m_{0.05}$. If $x = 1$ for the four connections, then $m = 4/N$ and, from (5), the corresponding sample size is 104.

3.4 *175°C Screening Test*

Although the 105°C test appears to be quite effective in distinguishing between stable connections and unstable connections, its slowness is a practical disadvantage. In cases where a number of alternate terminal designs are being considered, for example, and where it is desirable to concentrate development effort on a few of the most promising alternates, the five-month waiting period can be extremely inconvenient. There is need, therefore, for a quick screening test that will serve to identify those types of connections that are likely to pass the 105°C aging test.

Early experience with the 105°C test suggested that addition of mechanical disturbances to the original three-hour undisturbed 175°C test might make it capable of detecting potentially unstable connections. Within limits, this proved to be so.

In the screening test that eventually was standardized, the solderless wrapped connections are baked in an oven for three hours at 175°C. The three-hour period includes the warming up period of the oven and fixtures, which amounts to about one-half hour with the equipment that has been used. During the entire three-hour period, each terminal is plucked automatically once per minute. The plucking mechanism deflects the free end of each terminal $\frac{1}{16}$ inch, then releases it abruptly, allowing the terminal and lead wire to vibrate freely. The opposite end of the terminal is supported rigidly, of course, and the unsupported length (on which the connection is wrapped) is $\frac{19}{32}$ inch.

At the end of the 175°C heat treatment, the connection is cooled to room temperature, and its resistance variation is measured as described previously except that (a) the terminal is plucked only once during the measurement instead of two or three times, and (b) it is plucked $\frac{1}{16}$ inch instead of $\frac{1}{32}$ inch.

In the ordinary cases, where there is no soft solder in the connection, the correlation between the results of this test and the results of the 105°C test has been reasonably good. The principal discrepancy is that the 175°C test is not as sensitive as the 105°C test; that is, the spread between stable and unstable connections tends to be smaller in the 175°C test than in the 105°C test. This can be seen in Table I.

Despite its limitations, the 175°C screening test has been extremely useful in conserving testing effort and in speeding up various phases of the development program. Altogether, more than 16,000 solderless wrapped connections have been subjected to the 175°C test. On a few occasions, final approval of particular types of connections has been based upon results of the 175°C test, although the usual practice is to withhold final approval until the 105°C test is completed.

3.5 *Criterion for Acceptance (175°C Screening Test)*

Empirically, it is possible to define acceptance and rejection limits for the 175°C test that will correspond roughly to the limits for the 105°C test. Although such limits actually have not been used in the past, it appears that limits based upon $m' = 0.22$ for the 175°C test would have led to essentially the same decisions that were made on the basis of 105°C test results. For $m' = 0.22$, the limits for the 175°C test would be

$$m_{0.05} = 0.22 - \frac{0.77}{\sqrt{N}} \tag{8}$$

and

$$m_{0.95} = 0.22 + \frac{0.77}{\sqrt{N}}. \tag{9}$$

These limits are shown in Fig. 11, along with observed values for various types of connections that have been subjected to both the 175°C and the 105°C tests. The symbols used for the values observed in the 175°C test indicate how similar types of connections fared in the 105°C test.

As indicated previously, the limits for the 175°C test can serve as guides for making decisions. If m for a particular type of connection is

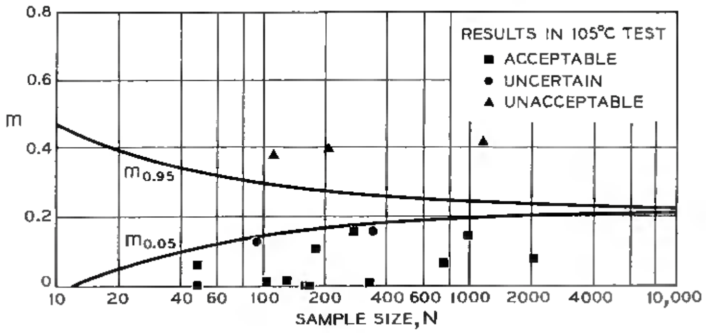


Fig. 11 — Criteria for acceptance and rejection in 175°C screening test. Points are results of 175°C tests, but symbols indicate how similar connections behaved in 105°C tests.

less than $m_{0.05}$, it probably is worthwhile to make tests at 105°C. If m is greater than $m_{0.95}$, it probably is not worthwhile. If m falls between $m_{0.05}$ and $m_{0.95}$, the decision will have to be based upon additional considerations.

The sample size for which $m_{0.05} = 0$ is about 13. The sample size for which $m_{0.05} = 4/N$ is about 41. A sample of 13, then, is the smallest sample that should be tested, and 41 is a more realistic minimum.

IV. MECHANICAL TESTS

Certain standard mechanical tests are made regularly to qualify wrapping bits for use in production, and the same tests have been made to qualify the bits used in the evaluation studies. Sample connections are wrapped on specified types of terminals, then two different types of tests are made on the sample connections. In the first test, the force required to strip the connection off of the terminal is measured. It is required that this stripping force be at least 3000 grams and that the median for any subgroup of five connections be at least 4200 grams. The purpose of this test is to provide assurance that the bit is capable of wrapping tight connections.

In the second test, the wire is unwrapped from the terminal, the unwrapping force being applied to the wire at least one-half inch from the terminal without the wire being restrained from twisting or bending back upon itself. It is required that the connection be capable of being unwrapped in this fashion without the wire breaking. The purpose of this test is to provide assurance that the bit does not wrap too tightly — so tightly that the wire would be weakened excessively.

Assuming that the connections have been wrapped with qualified bits, it is of interest to consider what might happen to them subsequently in service. In general, this reduces to a consideration of mechanical treatments that would tend to loosen the connection and break the wire.

The principal types of mechanical treatment that could loosen a wrapped connection are (a) squeezing the sides of the connection, (b) pushing or pulling on the body of the connection and, of course, (c) unwrapping the connection.

Although the squeezing forces that would be required to loosen connections have not been measured, it is evident that enough force could be exerted with a pair of pliers to damage a connection. Wiremen and maintenance men have been cautioned, therefore, not to squeeze the connections — either with pliers or with test clips.

The force required to slide the connection along the terminal ordi-

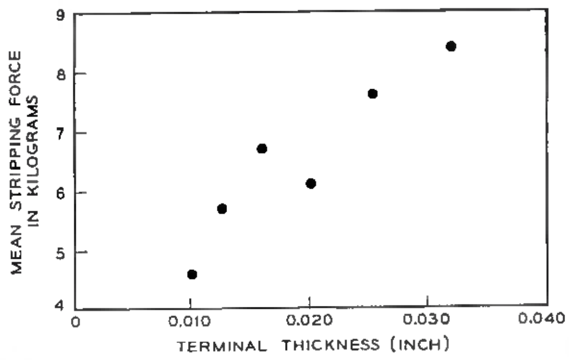


Fig. 12 — Effect of terminal thickness on stripping force. Connections wrapped with No. 24 gauge tinned copper wire on flat nickel silver terminals 0.062 inch wide. Sample size 100.

narily is well above the minimum requirement of 3000 grams, ranging up to more than 10,000 grams in some cases. In general, the heavier terminals tend to give higher stripping forces than do the lighter terminals, and the heavier gauges of wire tend to give higher values than do the lighter gauges. Figs. 12 and 13 illustrate the relationships.

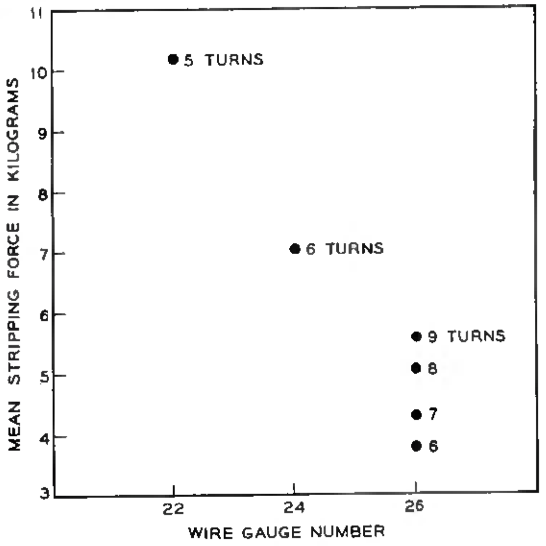


Fig. 13 — Effect of wire size on stripping force. Connections wrapped with tinned copper wire on 0.0319 × 0.062-inch nickel silver terminals. Sample size 50 to 175.

Ordinarily there is little risk that a connection will be unwrapped in normal wiring or maintenance operations — unless, of course, the wire is unwrapped deliberately to remove the connection. It is possible, nevertheless, to unwind a connection by pulling steadily on the lead wire in the direction parallel to, and in line with, the terminal axis. On the average, a force of about 825 grams is sufficient to unwrap one-half turn of a No. 24 gauge connection on a 0.0319 × 0.062-inch terminal, and a force of about 2300 grams will unwrap one full turn. The standard deviations are about 15 per cent of these values.

The principal types of mechanical treatment that are liable to break wires in service are (a) tension alone, (b) repeated bending alone and (c) repeated bending combined with tension. A number of tests have been made to compare the effects of these treatments on wires connected to terminals by solderless wrapping with the effects on wires connected by soldering.

The results of a few tensile tests are summarized in Table II. When the wire was pulled radially (perpendicular to the terminal axis), almost the full breaking strength of the wire was realized with the soldered connections. The breaking strength with the solderless wrapped connections was about eight per cent lower, however, because the wire was indented where it had been wrapped around the corners of the rectangular terminal.

It is interesting to note that, even with soldered connections, the full breaking strength of the wire was not realized when the wire was pulled axially. The soldered joint was broken in stages, and, in most cases, the wire was completely unwrapped from the terminal before the force reached the breaking strength of the wire. With the flat terminals, the ultimate strength of the solderless wrapped connections was significantly lower than that of the soldered connections. With the wire-spring relay terminals, on the other hand, the differences between the solderless wrapped and soldered connections were trivial. The serrations of the

TABLE II — MEAN ULTIMATE STRENGTH (IN GRAMS) OF CONNECTIONS MADE WITH NO. 24 GAUGE COPPER WIRE

Type of Terminal	Direction of Pull on Wire	Solderless Wrapped Connections (6 Turns)	Wrapped and Soldered Connections (1¼ to 2 Turns)
0.0319 × 0.062	Axial	2330	3030
0.0319 × 0.062	Radial	4814	5205
Single Wire	Axial	4450	4444
Twin Wire	Axial	4527	4512

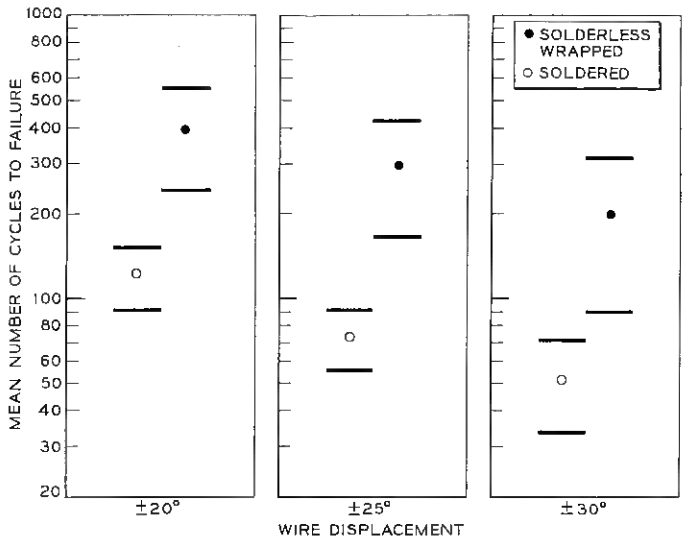


Fig. 14 — Effects of repeated bending without tension. Connections made with No. 24 gauge tinned copper wire on single-wire terminals of wire-spring relay. Sample size 16.

single-wire terminal and the irregularities of the twisted and coined twin-wire terminal were as effective as soldering in locking the wrapped wire in place.

The effects of repeated bending without tension are illustrated in Fig. 14. The test was performed by bending the wire back and forth through the angles indicated until the wire broke. The bending moments were large enough to exceed the elastic limit of the copper wire. In Fig. 14 conventional $\pm 3\sigma$ control limits (or confidence limits) are shown with each plotted point as a simple, graphic way of indicating that the performance of the solderless wrapped connections was significantly better, on the average, than that of the soldered connections.

Vibration tests, of course, provide another method for measuring the effects of repeated bending without tension. Fig. 15 shows the results of a vibration test performed by the Western Electric Company on small equipment units wired with local cables. The $\pm 3\sigma$ control limits in this case are based upon the observed breakage ("fraction defective") of the soldered wires. The fact that the observed breakage of the solderless wrapped wires consistently fell below the lower control limit for the soldered wires indicates that the performance of the solderless wrapped wires was superior on the average.

The effects of repeated bending combined with tension are shown in Fig. 16. In this test, the wire was kept under tension continuously while it was bent from its starting position perpendicular to the terminal axis to a second position parallel to the terminal axis and then back to its starting position. This cycle was repeated, always in the same direction from the starting position, until the wire broke. In Fig. 16, the $\pm 3\sigma$ control limits indicate again that the performance of the solderless wrapped wires was better, on the average, than the performance of the soldered wires.

From the results of tests such as those which have been described, it has been concluded that the solderless wrapped connection satisfies the mechanical security objectives in part, but not completely. It is more vulnerable to loosening by axial pull on the wire than is the soldered connection, so this limitation should be recognized in authorizing applications of solderless wrapping. In its ability to withstand vibration and repeated bending of the lead wire, however, the solderless wrapped connection appears to be fully as good as, and probably better than, the soldered connection.

V. APPROVED COMBINATIONS OF TERMINALS AND WIRE

Present approvals of specific combinations of terminals and wire are based very largely upon the results of the accelerated aging tests that

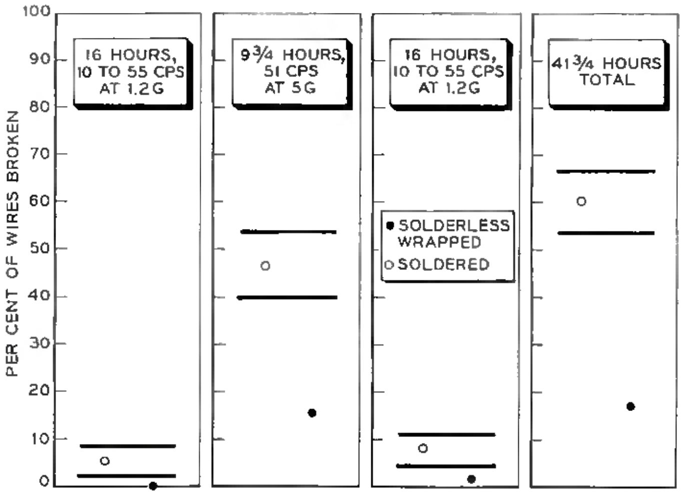


Fig. 15 — Results of vibration test on small equipment unit wired with local cable. Sample size 550 for each type of connection.

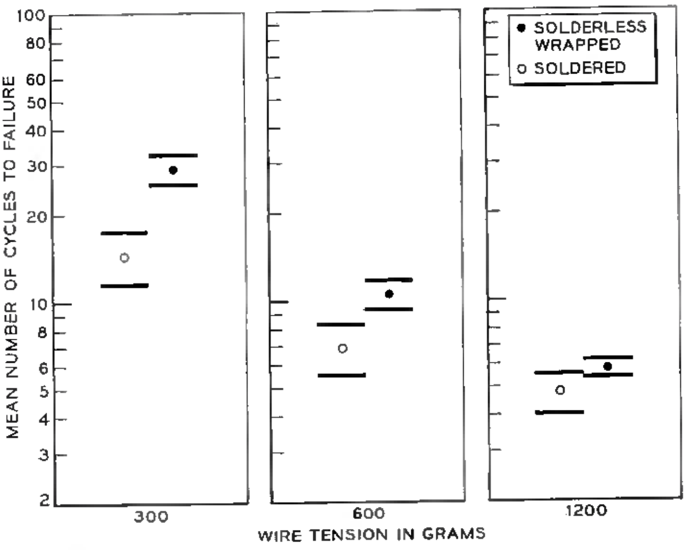


Fig. 16 — Effects of repeated bending with wire under tension. Connections made with No. 24 gauge wire on single-wire terminals of wire-spring relay. Sample size 25 in each case.

have been described, upon the ability of the manufacturer to manufacture the terminals economically and upon a limiting dimension of a wrapping bit that is used widely.

The limiting dimension is the minimum diameter of the terminal hole in the bit used for wrapping No. 24 gauge wire. In order to be sure that No. 24 gauge wire can be wrapped on any approved terminal, the maximum dimensions of the cross-section are limited so that the terminal will pass through a circular opening 0.073 inch in diameter.

Solderless wrapping with No. 22 gauge tinned copper wire is approved on terminals of rectangular cross section whose nominal thickness is at least 0.030 inch and whose minimum diagonal exceeds 0.061 inch.

Solderless wrapping with No. 24 gauge tinned copper wire is approved on (a) terminals of rectangular cross section whose nominal thickness is at least 0.025 inch and whose minimum diagonal exceeds 0.059 inch, (b) embossed terminals of the form shown in Fig. 6 punched and formed from flat stock whose nominal thickness is at least 0.010 but less than 0.025 inch and (c) the wire-spring relay terminals in Figs. 7 and 8.

Approved terminal materials include nickel silver, brass, phosphor bronze and silicon copper. The best terminal materials for solderless wrapping have a high modulus of elasticity, a low rate of stress relaxation and a temperature coefficient of expansion near that of the wire.

In general, a copper flash plus an electrotinned finish is required on brass, phosphor bronze and silicon copper terminals, or they may be punched from either tin-coated or solder-coated stock. No finish is required on nickel silver terminals, although any of the foregoing finishes is permissible and is specified in some cases to facilitate soldering. Solder dipping is not approved, because of the risk of obtaining abnormally thick coatings from time to time and the undesirable effect that this could have upon the stability of connections wrapped on such terminals.

The use of untinned copper wire has been approved only in cases where the required service life of the connections is substantially less than 40 years. Furthermore, such approvals are limited to the heavy terminals which are approved for use with No. 22 gauge wire.

There have been exceptions to some of the standards described in the preceding paragraphs. Solderless wrapping on solder-dipped terminals and on thin, flat terminals was approved on a limited basis early in the development program. Those connections, however, are in circuits where trouble, if it should occur, would be detected automatically, could be located quickly and could be corrected easily by soldering the defective connection. For future production, those terminals are being brought into agreement with present standards.

VI. PERFORMANCE IN SERVICE

The first field trial of solderless wrapped connections was installed in 1950, and limited use of solderless wrapping in regular manufacture began a few years later. During the period 1950 to 1958, the service performance of solderless wrapped connections appears to have been highly satisfactory.

A two-year survey of 411,000 solderless wrapped connections in five central offices showed a lower wire breakage rate than would have been expected with soldered connections, and the difference was great enough to be considered statistically significant.

There has been no report of solderless connections being pulled off of terminals in service or of being partially unwrapped, and inspection of about 20,000 connections in two central offices revealed no sign of partial unwrapping.

The resistance variations of 135 connections were measured after two years in service, and 160 more were measured after three and one-half years of service. The highest value of ΔR observed was 0.0004 ohm, so the value of m for the 295 connections still was zero.

VII. CONCLUSION

The laboratory studies and field experience to date have provided considerable assurance that properly designed and properly made solderless wrapped connections will perform satisfactorily for 40 years in central office service. The use of solderless wrapping is being expanded, therefore, in the Bell System. Suitable terminals now are being provided on many types of telephone apparatus. Design changes have been authorized to provide suitable terminals on a number of additional types of apparatus, although the changes have not been introduced yet in manufacture. And design changes to provide suitable terminals on still other types of apparatus are being studied.

Several hundred million solderless connections are being wrapped each year in the Bell System now. The number will grow, but it is difficult to predict what the saturation level will be. Inevitably, the solderless wrapped connection is in competition with soldered connections, clinched connections, welded connections and all the other types. In the end, the choice of connections for any particular application is likely to be an economic choice — based not only upon the cost of the labor involved in making a connection, but upon many other factors as well. The cost of modifying terminals, for example, is an obstacle to solderless wrapping on existing apparatus. In some cases, the cost of modifying the terminals of a particular type of apparatus outweighs the potential savings of solderless wrapping. It is not likely, therefore, that solderless wrapping ever will displace other types of connections completely.

The present program for central office equipment in the Bell System calls for modification of the terminals of existing types of apparatus in those cases where the preparation expense clearly is outweighed by the direct savings from solderless wrapping and the indirect savings from the use of plastic insulated wire, which is made practicable by solderless wrapping. In other cases, modification of terminals will be deferred until present manufacturing tools wear out and have to be replaced.

In designing new types of apparatus — especially switching apparatus — the trend is to provide terminals suitable for solderless wrapping at the start. Furthermore, every effort is made to arrange the terminals in modular arrays that will facilitate automatic wiring by machines. These two steps are opening a door to economical use of automatic wiring in manufacture. It seems quite probable, therefore, that telephone switching systems of tomorrow will be well populated with solderless wrapped connections.

VIII. ACKNOWLEDGMENTS

Many persons, working in several closely knit teams, have participated in the development of solderless wrapped connections. Each of those persons has played an important part in the development. In particular, the author wishes to acknowledge the contributions of H. L. Coyne and R. H. Van Horn, who supervised the several phases of the evaluation program.

REFERENCES

1. Keller, A. C., A New General Purpose Relay for Telephone Switching Systems, B.S.T.J., **31,** November 1952, p. 1023; Trans. A.I.E.E., **71,** Part 1, January 1953, p. 413.
2. McRae, J. W., Mallina, R. F., Mason, W. P., Osmer, T. F. and Van Horn, R. H., Solderless Wrapped Connections, B.S.T.J., **32,** January 1953, p. 523.
3. Mason, W. P. and Anderson, O. L., Stress Systems in the Solderless Wrapped Connection and Their Permanence, B.S.T.J., **33,** September 1954, p. 1093.

Recent Monographs of Bell System Technical Papers Not Published in This Journal*

Babington, W., see Weissmann, G. F.

Bemski, G.

Recombination Properties of Gold in Silicon, Monograph 3190.

Benson, K. E., see Lovell, L. C.

Benson, K. E., see Wernick, J. H.

Birdsall, H. A., see McMahon, W.

Buck, T. M. and McKim, F. S.

Certain Chemical Treatments and Ambient Atmospheres on Surface Properties of Silicon, Monograph 3191.

Camilli, C. T., see McMahon, W.

Clogston, A. M.

Structure of the Metastable State of Mn^{++} and Fe^{+++}, Monograph 3077.

Collins, R. J. and Thomas, D. G.

Photoconduction and Surface Effects with Zinc Oxide Crystals, Monograph 3192.

Conrad, D. A., see Flugge, W.

Derick, L., see Frosch, C. J.

Douglass, D. C. and McCall, D. W.

Diffusion in Paraffin Hydrocarbons, Monograph 3079.

* Copies of these monographs may be obtained on request to the Publication Department, Bell Telephone Laboratories, Inc., 463 West Street, New York 14, N. Y. The numbers of the monographs should be given in all requests.

Early, J. M.

Structure-Determined Gain-Band Product of Junction Triode Transistors, Monograph 3193.

Early, J. M., see Warner, R. M., Jr.

Ferrell, E. B.

Plotting Experimental Data; and Control Charts for Log-Normal Universes, Monograph 3194.

Flugge, W. and Conrad, D. A.

Thermal Singularities for Cylindrical Shells, Monograph 3195.

Frosch, C. J. and Derick, L.

Diffusion Control in Silicon by Carrier Gas Composition, Monograph 3196.

Fuller, C. S., see Wolfstirn, K.

Geller, S., see Wernick, J. H.

Gibbons, D. F.

Thermal Expansion of Some Crystals with the Diamond Structure, Monograph 3197.

Hrostowski, H. J. and Kaiser, R. H.

Absorption Spectrum of Arsenic Doped Silicon, Monograph 3198.

Johnson, G. R., see McMahon, W.

Kaiser, R. H., see Hrostowski, H. J.

Knab, E. D.

Connections Made to Printed Circuit Boards Through Resistance Fusing, Monograph 3200.

Loman, G. T., see Warner, R. M., Jr.

LOVELL, L. C., WERNICK, J. H. and BENSON, K. E.

Dislocation Etch Pits in Tellurium and Apatite, Monograph 3088.

MATREYEK, W., see Winslow, F. H.

MCCALL, D. W., see Douglass, D. C.

MCKIM, F. S., see Buck, T. M.

MCMAHON, W., BIRDSALL, H. A., JOHNSON, G. R. AND CAMILLI, C. T.

Degradation Studies of Polyethylene Terephthalate, Monograph 3250.

RAISBECK, G.

Minimum-Loss Two-Conductor Transmission Lines, Monograph 3202.

RIGTERINK, M. D.

Ceramic Electrical Insulating Materials, Monograph 3203.

THOMAS, D. G., see Collins, R. J.

WAHL, A. J.

Analysis of Base Resistance for Alloy Junction Transistors, Monograph 3204.

WARNER, R. M., JR., EARLY, J. M. AND LOMAN, G. T.

Characteristics and Performance of a Diffused-Base Germanium Oscillator Transistor, Monograph 3205.

WEISSMANN, G. F. AND BABINGTON, W.

A High-Damping Magnesium Alloy for Missile Aplications, Monograph 3206.

WERNICK, J. H., GELLER, S. and BENSON, K. E.

Constitution of a Semiconducting Pseudo-Quaternary System, Monograph 3081.

WERNICK, J. H., see Lovell, L. C.

WINSLOW, F. H., MATREYEK, W. and YAGER, W. A.

Carbonization of Vinyl Polymers, Monograph 3207.

WOLFF, P. A.

Theory of Plasma Resonance in Solids, Monograph 3208.

WOLFSTIRN, K. and FULLER, C. S.

Comparison of Radio-Copper and Hole Concentration in Germanium. Monograph 3209.

YAGER, W. A., see Winslow, F. H.

Contributors to This Issue

Paul J. Burke, B.S., 1940, College of the City of New York; Ed.M., 1950, Harvard University; Faculty, Vineland, N. J., High School, 1943–45; Cooperative Test Service, 1945–48; Educational Testing Service, 1948–49; Columbia University, 1950–53; Bell Telephone Laboratories, 1953—. He has been engaged in telephone traffic studies. Member Operations Research Society of America; Institute of Mathematical Statistics, American Statistical Association, American Association for the Advancement of Science, Harvard Engineering Society, Phi Beta Kappa, Phi Delta Kappa.

Stanley J. Elliott, B.S.E.E., 1937, and M.S.E.E., 1938, University of California; Bell Telephone Laboratories, 1938—. Mr. Elliott was first engaged in fundamental contact studies. During World War II he took part in development work on airborne radars, fire control radar and other military projects. He later turned to development of glass-enclosed switches and relays and wire-spring relays. He was a coordinator for Bell Laboratories with outside suppliers in development of apparatus for guided missile systems. In 1953 he was named Switching Apparatus Engineer with responsibility for work on central office apparatus and some military devices. In April, 1959 he was appointed Military Systems Development Engineer. Member A.I.E.E., I.R.E., American Society for Quality Control, Phi Beta Kappa, Sigma Xi, Eta Kappa Nu, Tau Beta Pi.

E. N. Gilbert, B.S., 1943, Queens College; Ph.D., 1948, Massachusetts Institute of Technology; M.I.T. Radiation Laboratory, 1944–46; Bell Telephone Laboratories, 1948—. Mr. Gilbert has been engaged in studies of information theory and switching theory. Recipient M.I.T. Applied Mathematics Fellowship, 1946–48. Member American Mathematical Society.

David W. Hagelbarger, A.B., 1942, Hiram College; Ph.D., 1947, California Institute of Technology; faculty, University of Michigan, 1946–49; Bell Telephone Laboratories, 1949—. His first work was with the electron dynamics group on development of microwave tubes. Since 1952 he has specialized in special purpose computers, automata and switching research. He recently developed a new error-correcting code. Member I.R.E., American Physical Society, Sigma Xi.

BELA JULESZ, Dipl. in Electrical Engineering, 1950, Budapest (Hungary) Technical University; Candidate in Technical Sciences, 1956, Hungarian Academy of Sciences; Bell Telephone Laboratories, 1956—. He is engaged in studies of systems for reducing television bandwidth, with emphasis on problems of pattern recognition. Member I.R.E.

C. Y. LEE, B.E.E., 1947, Cornell University; M.S.E.E., 1949 and Ph.D., 1954, University of Washington; John McMullen Regional Scholar at Cornell, 1944–47; instructor, electrical engineering, University of Washington, 1948–51; Bell Telephone Laboratories, 1952—. Mr. Lee has been engaged in mathematical research in switching systems development. He was a visiting member of the Institute for Advanced Study in the School of Mathematics during the academic year 1957–58. Member Sigma Xi, Eta Kappa Nu.

EDWARD F. MOORE, B.S., 1947, Virginia Polytechnic Institute; M.S., 1949, and Ph.D., 1950, Brown University; assistant professor, University of Illinois, 1950–51; Bell Telephone Laboratories, 1951—. He has been engaged in mathematical research on switching circuit design, automata theory and information theory. Member American Mathematical Society, Association for Computing Machinery, Association for Symbolic Logic, I.R.E., Sigma Xi, Phi Kappa Phi.

H. EARLE VAUGHAN, B.S. in C.E., 1933, Cooper Union; Bell Telephone Laboratories, 1928—. He was first engaged in work on voice operated devices and studies of the effects of speech and noise on voice frequency signaling systems. During World War II he worked on anti-aircraft computers, fire-control radar and other military projects. He was later concerned with digital techniques and systems and electronic switching systems. He was appointed Switching Systems Research Engineer in 1955 and Director of Systems Research in 1957. Senior Member I.R.E.