T. I. Kirkpatrick
N. R. Clark

PERT as an Aid to Logic Design

Abstract: A new application is presented for PERT, the well-known statistical project-scheduling method. Using PERT, the logic designer could circumvent usually unrealistic worst-case criteria. He substitutes a formalized statistical method which determines (1) expected or most probable delays, (2) critical timing paths, (3) timing slack allowable between various inputs, and (4) probability of achieving an output by a certain time. From these data the designer can make a meaningful judgment regarding the reliability of his system. Significantly, he may achieve high reliability without being forced to resort to worst-case design.

Introduction

This paper provides a simple but useful means of statistically designing logic systems. The procedure is based on a widely used scheduling aid known as PERT and eliminates many of the drawbacks associated with designing to a worst-case criterion. Such valuable information as most probable time delay, critical paths, and the probability of any given delay are yielded by this application of PERT. Not only does the scheme require a minimum of effort and input data, but also it can be automated, since many computer libraries contain PERT programs.

The difficulty in using a worst-case technique is readily apparent. For instance, consider the problem of predicting the time that a signal will arrive at a given point after having traveled through several logic stages. Assuming all stages to be of the same type, the worst-case time is the worst-case delay of a single stage multiplied by the number of stages. The worst-case time will then differ from the nominal or average time in direct proportion to the number of stages being considered. It is obvious that not only would most practical systems fail under absolute worst-case conditions, but also that the assumption of such conditions is unrealistic. The probability is extremely low that all of the stages in a long logic chain will present a worst-case delay. Some allowance, then, should be made for this fact.

It is justifiable at this stage to ask the question "Can we make allowance for the element of probability without jeopardizing our design?" One design approach is simply to assume that over a long logic chain a nominal delay per stage may be counted on, relying on fast stages to average out the effect of slow stages. But in using this approach, there is definitely an involvement with chance. Even worse, no means is provided for quantitatively measuring the extent of that involvement. For instance, how do we answer the question "What is the probability that the delay will extend 50 ns beyond the nominal delay?" PERT can assign a definite probability to such an occurrence.

PERT was developed as a computer-implemented statistical aid in estimating end dates and critical paths for the scheduling of large projects. As a scheduling device, PERT requires as input three estimates concerning the length of time it will take to do a given job. These estimates are the most likely time, the shortest time, and the longest time. As output, PERT provides the probability of finishing the job at any given time. Thus, the application of PERT to logic timing is apparent. It is necessary only to obtain the three appropriate estimates of circuit delay, and then with the application of the PERT procedure, the probability of any given delay can be computed. Also, and equally important, timing slack and critical paths can be determined.

Procedure for logic design

In using the PERT approach, the logic designer would first assemble the basic data:

(1) The three delay figures for each type of logic block being used: a = shortest, m = most likely, and b = longest.

(2) A normal distribution curve in tabular form, such as that in Table 1.

From the above data the following procedure would then be used to predict whether an output will occur within a certain time following an input:

(1) Construct a logic diagram (Fig. 2) with the quantities a, m, b in each block. Since $t_{\text{off}} \neq t_{\text{on}}$ in general, there will be two sets of delay figures for each block. The set corresponding to the significant transition must be used at each stage.

(2) Compute the mean time delay and variance for each block according to the formulae

$$\text{mean: } t_e = \frac{a + 4m + b}{6}$$

$$\text{variance: } \sigma^2 = \left(\frac{b - a}{6}\right)^2.$$

Enter t_e and σ^2 in each block on the diagram.

(3) For each path through the logic diagram compute over-all mean, T_e, and variance, Σ^2, by summing the individual components. The critical path which determines the delay through the net is that path having the largest T_e.

(4) Compute the factor Z for each output (or internal node if desired) according to the formula

$$Z = \frac{T_{os} - T_e}{\sigma},$$

where T_{os} = Time by which output is required.
T_e = Critical path delay.

The quantity Z is the distance between the mean of the output delay distribution and the time by which output is required, in units of standard deviation.

(5) From the table for the normal distribution curve, read Pr(Z). "Pr" is defined as the probability that the output will arrive by T_{os}. If Z is negative (see Ref. 3, p. 166), then Pr($-Z$) = 1 − Pr(Z).

Those familiar with PERT will notice that our procedure is virtually identical to that used in the project management application. There are two differences, the first of which involves Step 1. The PERT chart has circles connected by paths, the circles representing completed tasks.[1,2,5] The paths represent the intervening time delay. In our case,

Table 1 Values of the standard normal distribution function.*

z	0	1	2	3	4	5	6	7	8	9
.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
.7	.7580	.7611	.7642	.7673	.7703	.7734	.7764	.7794	.7823	.7852
.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8395	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9278	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9430	.9441
1.6	.9452	.9463	.9474	.9484	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9648	.9556	.9664	.9671	.9678	.9686	.9693	.9700	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9762	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9874	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9990	.9993	.9995	.9997	.9998	.9998	.9999	.9999	1.0000

* From Ref. 1.

the blocks themselves represent the time delays. The interconnecting lines signify only signal flow, but have no significance with respect to delay.

The second difference involves the handling of OR functions, which do not occur in the usual PERT application. Wherever an OR is encountered, the *earliest* input t_e rather than the latest is used for critical path computations.

There is a further use to which this technique can be put. Suppose we have computed an output T_e over the critical path based on the assumption that all input signals to the net arrive simultaneously. Now we wish to know how much slack can be allowed between certain of these inputs.

The timing slack procedure is as follows.

(1) Using the over-all critical-path output T_e as a starting point, subtract the summation of the t_e for all the stages in the longest path back to the input under investigation. Do the same for the variances.

(2) If the path taken is *not* the critical path, there will be a finite, positive t_e left after performing the subtraction. This represents the maximum *average* slack t_s which can be tolerated without affecting the output time by causing this path to become critical.

(3) Assuming the input arrives at the latest time found in (2), the σ^2 left after performing the subtraction is the maximum that can be tolerated without affecting the output Pr calculation.

It is apparent that the σ^2 resulting from (3) can be positive, zero, or negative. One would normally expect it to be positive if similar circuits are used in all paths, but this is not guaranteed. So the question arises: "What is the meaning of positive slack coupled with negative variance?"

To answer this question, let us consider two cases:

Case No. 1: $t_{in} = t_s$, where t_{in} is defined as the input time to the path under investigation.

We can compute a Z' for the path, assuming some arbitrary input variance, σ_{in}^2. If $(\Sigma')^2$ is the summation of variances along the path, $Z' = (T_{os} - T_e)/\Sigma'$. This will yield a Pr' < Pr, where Pr' is the probability of achieving an output by T_{os} along the path. Note that Pr' < Pr even if $\sigma_{in}^2 = 0$.

Case No. 2: Pr' = Pr. This implies $Z' = Z$. We wish to solve for the t_{in} required to satisfy the condition. If T_e' is the mean output time required along the path under investigation in order for Pr' = Pr, then $Z' = Z = (T_{os} - T_e')/\Sigma'$.

Solving, then $T_e' = T_{os} - \Sigma' Z$.

Since $t_{in} = T_e - (T_e - t_s)$, we find

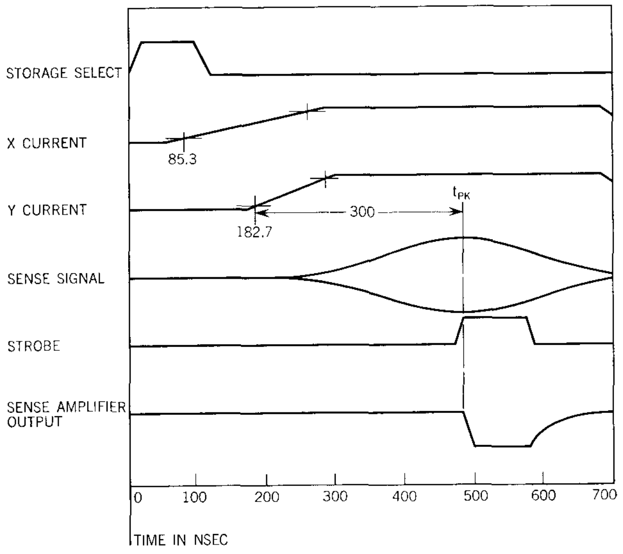


Figure 1 Timing chart for the read cycle.

$$t_{in} = T_{os} - T_e + t_s - \Sigma' Z.$$

The t_{in} limits are $-\infty < t_{in} < t_s$.

If t_{in} is found to be negative, and if Pr' = Pr is a rigid requirement, then a new effective Critical Path might have to be defined.

Example in ferrite core system design

It will be useful to go through an example of the application of this technique. The example that we selected is an actual case which occurred in the design of a 2-μs coincident-current ferrite-core storage system.

We desired to know if a 520-ns access time could be achieved. Absolute worst-case calculations indicated that it could not. However, we felt that this was not reasonable because of the large number of circuit elements.

Figure 1 shows the timing chart for the read cycle. The fixed delay of 300 ns between the 10% point of the Y current and the strobe pulse is determined empirically. It depends on a number of parameters that are beyond the scope of this paper. It will suffice to say that it is fixed by the averaged core-array characteristics. A timing adjustment is provided to bring the strobe pulse precisely to 300 ns. The timing points to be determined then are the X and Y drive current turn-on times relative to the storage-select pulse. We also wish to know the amount of stagger between the two drive pulses.

In calculating the delays we refer to the X-dimension timing logic in Fig. 2, where each block contains values for a, m, b, t_e, and σ^2. The pre-driver delay figures are for the emitter-driven case where the base signal is "solid" well before the emitter. The driver delay figures are for

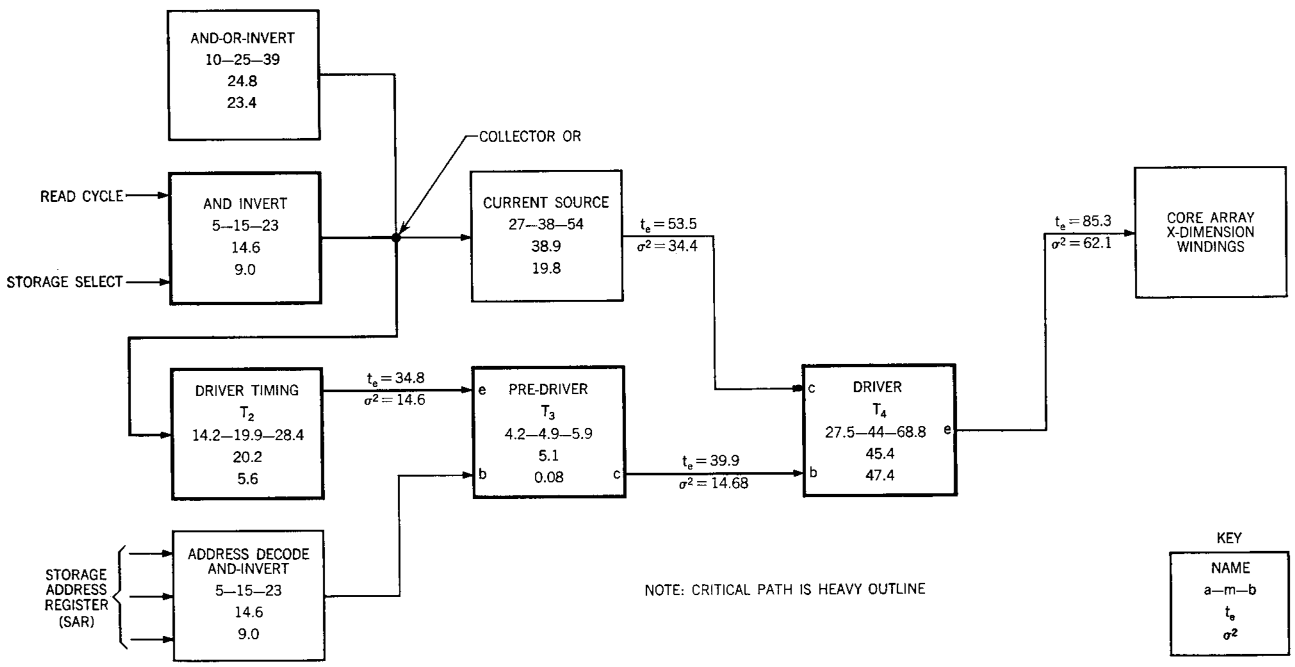
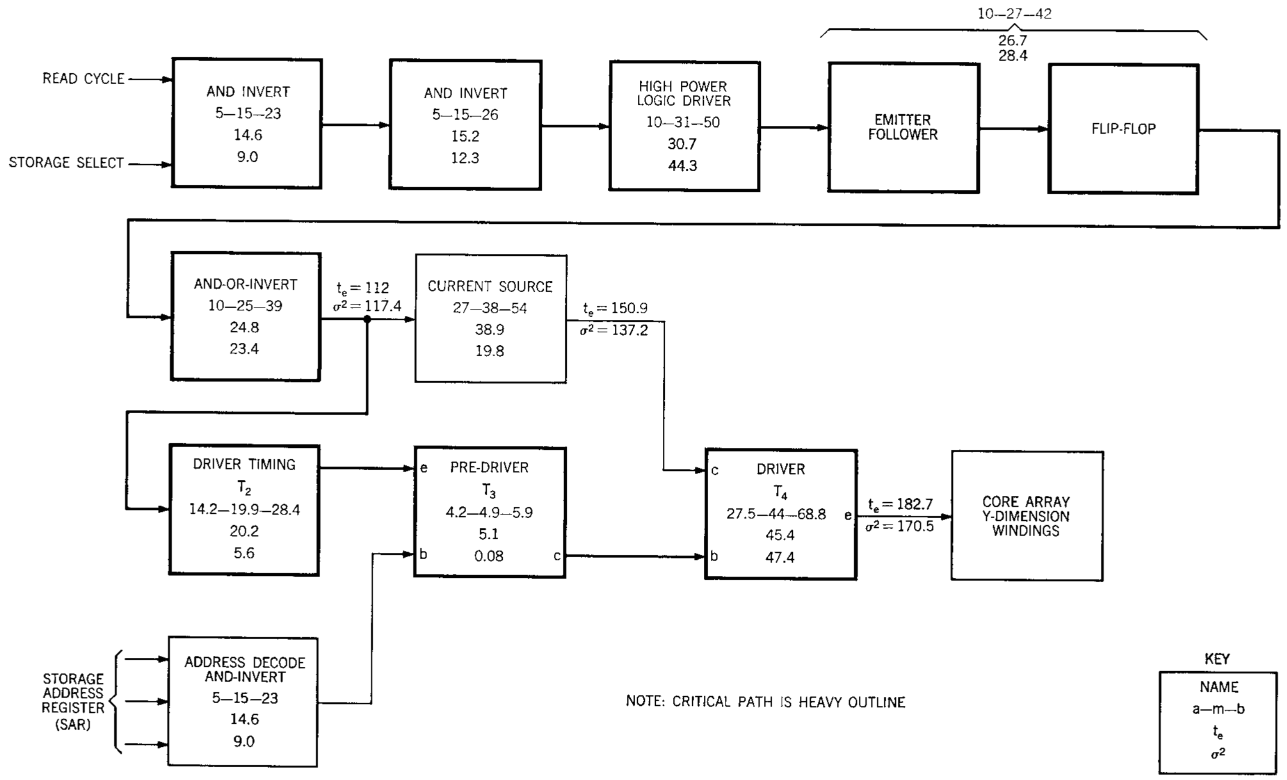


Figure 2 X-dimension timing logic.

Figure 3 Y-dimension timing logic.



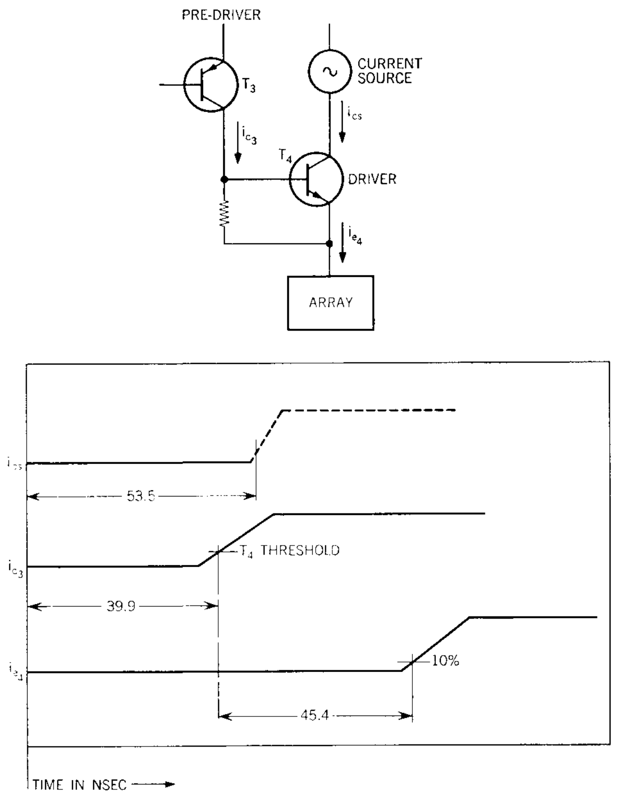


Figure 4 Read driver circuit and its delay.

the base-driven case, where the collector is conditioned well before the base input delay could allow collector current flow. Refer to Fig. 4, which shows the circuitry and timing in a little more detail. As is shown, the current source conditions the collector of T_4 before the base delay period is over. Therefore, the critical path is not through the current source. It should be noted that this is a special case not encountered in pure logic networks, since the base input delay is very much greater than the collector input delay. For pure logic networks, Rule 3 on page 145 would hold.

Computing over-all T_{ex} and Σ_x^2 along the critical path

$$T_{ex} = 14.6 + 20.2 + 5.1 + 45.4 = 85.3 \text{ ns}$$

$$\Sigma_x^2 = 9.0 + 5.6 + 0.1 + 47.4 = 62.1.$$

Figure 3 shows the Y-dimension timing logic. Again computing T_{ey} and Σ_y^2 along the critical path,

$$T_{ey} = 14.6 + 15.2 + 30.7 + 26.7 + 24.8 + 20.2 + 5.1 + 45.4 = 182.7 \text{ ns}$$

$$\Sigma_y^2 = 9.0 + 12.3 + 44.3 + 28.4 + 23.4 + 5.6 + 0.1 + 47.4 = 170.5.$$

Calculating the probability of achieving an output by 520 ns, we first must find the Z factor

$$Z = (T_{os} - T_{ey})/\sqrt{\Sigma_y^2}.$$

In this case, $T_{os} = 520 - 300 = 220$ ns, so

$$Z = \frac{220 - 182.7}{\sqrt{170.5}} = \frac{37.3}{13.1} = 2.85.$$

The probability (Pr) that the output will occur by 520 ns is then read off in Table 1:

$$\text{Pr} = 0.9978 \quad \text{for} \quad Z = 2.85$$

To interpret this result, the following points should be considered:

(1) The system is primarily composed of switching elements and therefore is not prone to random noise effects.

(2) The input data should be applicable to the actual operating conditions and should include any systematic effects caused by environment, as well as random tolerance variations.

Under these conditions, the result says that an average of 22 units per 10,000 manufactured will fail to meet the access-time requirement. It does not say that a given unit will fail 22 times per 10,000 accesses, since condition (1) above precludes this type of randomness. The only time a core memory exhibits random errors is when it is close to the limit of its operating region, where random noise can momentarily push the operating point outside the limit.

Next we must calculate the stagger between the X and Y pulses. We may treat the numbers T_{ex} and T_{ey} as random variables. The stagger then would be defined as T_{es}:

$$T_{es} = T_{ex} - T_{ey} = 182.7 - 85.3 = 97.4 \text{ ns}.$$

We need to know the probability that the stagger will be greater than 80 ns. To do this, we must consider the properties of a sum of random variables.

If the sum S_n is defined as

$$S_n = X_1 + X_2 + X_3 + \cdots X_n,$$

where $X_1 \cdots X_n$ are random variables, then the variance (σ^2) of S_n is given by Ref. 3:

$$\text{Var}(S_n) = \sum_{k=1}^{n} \sigma_k^2 + 2 \sum_{j,k} \text{Cov}(X_j, X_k).$$

In our case, the random variables are independent, therefore

$$\text{Cov}(X_j, X_k) = 0$$

and

$$\text{Var}(S_n) = \sum_{k=1}^{n} \sigma_k^2.$$

Using this result, we calculate for the variance of the stagger

$$\Sigma_s^2 = \Sigma_x^2 + \Sigma_y^2 = 62.1 + 170.5 = 232.6.$$

To find the probability that the stagger will be less than 80 ns, we define $T_{os} = 80$ and $T_e = T_{es}$ and compute

$$Z = \frac{T_{os} - T_{es}}{\Sigma_s} = \frac{80 - 97.4}{\sqrt{232.6}} = -1.15.$$

This yields $\text{Pr}_s = 0.1251$. Then the probability that the stagger will *exceed* 80 ns is given by:

$$1 - \text{Pr}_s = 0.8749.$$

Next we wish to know how much slack we can allow between the storage-select pulse and the address lines. Referring to Fig. 2, we calculate the t_e and σ^2 at the emitter input to the predriver (labeled "e" on the block)

$$t_e = 14.6 + 20.2 = 34.8$$

$$\sigma^2 = 9.0 + 5.6 = 14.6.$$

Now proceeding as in Step 1 of the timing-slack procedure,

$$t_{eSAR} = 34.8 - 14.6 = 20.2 \text{ ns}$$

$$\sigma_{SAR}^2 = 14.6 - 9.0 = 5.6.$$

This means that as long as the address arrives on the average by 20.2 ns following storage select with variance not exceeding 5.6, the system performance will not be degraded.

Conclusion

A simple, formalized technique has been demonstrated for studying delays through logic networks. It yields quantitative estimates for:

(1) Probability that an output will occur by a given time
(2) Critical paths
(3) Timing slack allowable between various inputs.

The advantages over previous methods are:

(1) It is more realistic than the worst-case criterion.
(2) It effectively allows a nominal design to be used, and puts a confidence factor on that design.

The primary limitations are:

(1) Availability of the three necessary delay figures.
(2) Assumptions made about the validity of the beta and normal distributions.
(3) The bias resulting from the simplifying assumption made in the formulation of the PERT procedure (see Ref. 5, p. 654) which renders it a practical method to use.

The first of these can be alleviated by application of engineering judgment and experience with the particular class of logic circuits being used. The second and third merit further discussion.

With appropriate choice of constants, the beta distribution,[5]

$$f(t) = K(t - a)^{\alpha}(b - t)^{\gamma},$$

can closely approach the normal distribution.

In addition, a skew may be introduced by choosing $\alpha \neq \gamma$. The original PERT developers assumed that random delay processes could be adequately described by the beta distribution. In the equation $t_e = (a + 4m + b)/6$, choosing $(m - a) \neq (b - m)$ is equivalent to choosing $\alpha \neq \gamma$. The central limit theorem was used to justify use of the normal distribution to approximate the output of a network of beta distributions.

The third limitation listed above can be illustrated as follows: Consider the case of a two-input AND, shown in Fig. 5.

We wish to know t_{ec}, given t_{eA} and t_{eB}. The Cumulative Probability is the probability that the signal has arrived by time t. It is the integral of the density (beta distribution). Since *both* inputs must have occurred before the circuit will start to generate an output, we must multiply the two input distributions. Figure 5 shows two cases, one where $f_A(t) = f_B(t)$, and the other where $f_A(t) \neq f_B(t)$. It is apparent that t_{eeff} (the t_e of the effective input distribution) is later than either t_{eA} or t_{eB}. The worst case is where $f_A(t) = f_B(t)$.

Now consider the OR function shown in Fig. 6. Since $C = A + B$, the circuit will start to generate an output as soon as A or B or both have arrived. We may calculate the effective input distribution by observing the Boolean equality $A + B = A\bar{B} + \bar{A}B + AB$. There are three mutually exclusive ways in which the INCLUSIVE OR can be realized. Their probability distributions are

$$P(A\bar{B}) = f_A(t)[1 - f_B(t)]$$

$$P(\bar{A}B) = f_B(t)[1 - f_A(t)]$$

$$P(AB) = f_A(t)f_B(t).$$

Adding, we get

$$\begin{aligned} P(A + B) &= f_A(t) - f_A(t)f_B(t) + f_B(t) \\ &\qquad - f_A(t)f_B(t) + f_A(t)f_B(t) \\ &= f_A(t) + f_B(t) - f_A(t)f_B(t). \end{aligned}$$

It is seen that t_{eeff} is earlier than either t_{eA} or t_{eB}. Again, the amount of bias depends on the overlap of $f_A(t)$ and $f_B(t)$.

In the original PERT application, the assumption was made that the input distribution with the latest t_e predominated at each multiple-input node. Since all the activities in the PERT chart involved AND functions, the

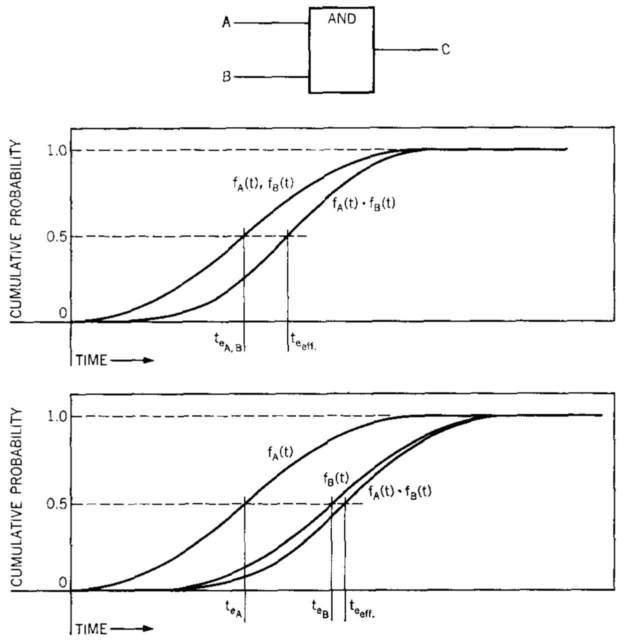


Figure 5 Two cases for Cumulative Probability, two-input AND circuit.

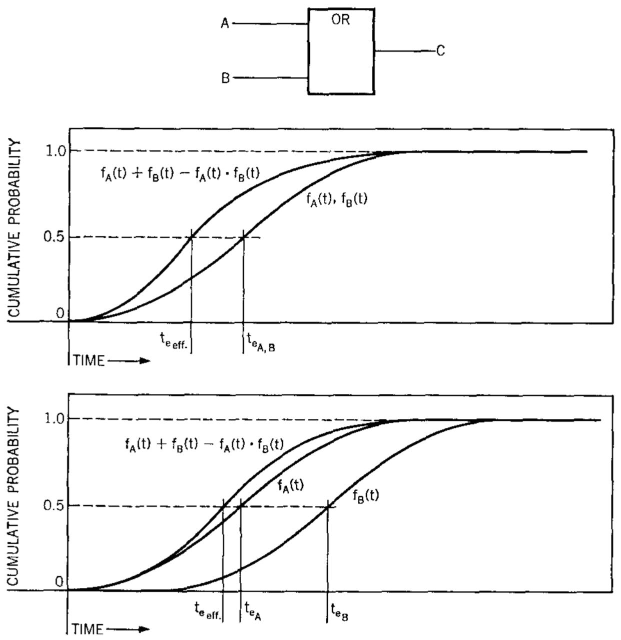


Figure 6 Two cases for Cumulative Probability, EXCLUSIVE OR circuit.

answers obtained were systematically biased early (Ref. 5, p. 654). Interestingly enough, the great bulk of articles, papers, and textbooks which have since been written about PERT appear to ignore this point entirely.

In applying the PERT procedure to logic networks, it would appear the assumptions concerning the shape of the delay distributions would be as valid as in the original application. The systematic t_e bias should actually be less, since usually there are about as many OR as there are AND functions.

• *Further development*

Thus far, only limited use has been made of this technique. In the future, application to large computing systems might yield considerable insight into system performance, critical timing situations, and areas in which requirements could be relaxed without performance degradation. All this is achieved without imposing any cumbersome procedures on the designer. In fact, it could reduce the work required of him if an automated PERT system is available.

References

1. L. P. Hartung and J. E. Morgan, "PERT/PEP—A Dynamic Project Control Method," IBM FSD Space Guidance Center, Report 61-816-2005, Owego, New York, 1961.
2. N. C. Loeber, "PERT for Small Projects," *Machine Design* 134–139 (Oct. 25, 1962).
3. William Feller, *An Introduction to Probability Theory and Its Applications*, Vol. I, Wiley, New York, 1957.
4. Jordon Kadet and Bruce H. Frank, "PERT for the Engineer," *IEEE Spectrum*, 131–137 (Nov. 1964).
5. D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fazar, "Application of a Technique for Research and Development Program Evaluation," *J. Operations Research Soc. Amer.* 7, 646–669 (1959).

Received July 26, 1965.