# Double-Edge-Triggered Flip-Flops

STEPHEN H. UNGER

TABLE VI
IMPACT OF INDIVIDUAL DESIGN TECHNIQUES TO DECREASE MEMORY ACCESS TIME $t_2$ (SECONDS/BIT) ON PERFORMANCE IN THE PDP-11 FAMILY

| Implementation Technique | Percent Performance Improvement |
|---|---|
| Add Processor/Unibus overlap to PDP-11/04 | 4.7% |
| Redesign Unibus Arbitration Logic to eliminate processor wait on PDP-11/34 | 5.21% |
| Instruction Fetch Overlap on PDP-11/45 | 10.07% |
| Cache on PDP-11/60 | |
| Set Size One | 101.58% |
| Set Size Two | 116.69% |

of the midrange PDP-11's and thus needs fewer microcycles to complete an instruction. To obtain this increased functionality, however, a much more elaborate set of data paths is required in addition to a highly developed control unit to exercise them to maximum potential. Such a change is not an incremental one and involves rethinking the entire implementation.

2) *Structure the Microcode to Take Best Advantage of Instruction Features:* All processors except the 11/10 handle JUMP/SUBROUTINE instruction addressing modes as a special case in the microcode. Most models do the same for the destination modes of the MOVE instruction because of its high frequency of use. Varying degrees of sophistication in instruction dispatching at the end of every instruction fetch is evident in different models resulting in various performance improvements.

3) *Cut Effective Microcycle Count by Overlapping Processor and UNIBUS Operation:* The PDP-11/10 demonstrates that a large microcycle count can be effectively reduced by placing cycles in parallel with memory access operations whenever possible.

Increasing microcycle speed is perhaps more generally useful since it can often be applied without making substantial changes to an entire implementation. Several of the midrange PDP-11's achieve most of their performance improvement by increasing microcycle speed in the following ways.

1) *Make the Data Paths Faster:* The PDP-11/34 demonstrates the improvement in microcycle time that can result from the judicious use of Schottky TTL in such heavily traveled points as the ALU. Replacing the ALU and carry-lookahead logic alone with Schottky equivalents saves approximately 35 ns in propagation delay. With cycle times running 300 ns and less, this amounts to better than a 10 percent increase in speed.

2) *Make Each Microcycle Take Only as Long as Necessary:* The 11/34 and 11/40 both use selectable microcycle times to speed up cycles which do not entail long data path propagation delays.

Application to the IBM S/360-S/370 family indicates the general usefulness of the performance model to systems where the available data is less precise and less tightly controlled. Nevertheless, the model is sufficiently accurate to estimate the performance of a proposed implementation or to plan a family of implementations, given only the characteristics of the selected technology and a general estimate of data path and memory cycle utilization.

By focusing on the individual terms in the model, a particular implementation can be tested for balance between processor and memory subsystems ($K_1t_1/K_2t_2$). The model could also be used to estimate the impact on performance of individual design tradeoffs. The relative frequencies of each function (e.g., addressing modes, instructions, etc.), while required for an accurate prediction, may not be available. There are, however, alternative ways to estimate relative frequencies. Consider the three following situations.

1) *At Least One Implementation Exists:* An analysis of the implementation in typical usage (i.e., benchmark programs) can provide the relative frequencies.

2) *No Implementation Exists, But Similar Systems Exist:* The frequency data may be extrapolated from measurements made on a machine with a similar architecture.

3) *No Implementation Exists and There are No Prior Similar Systems:* From knowledge of the specifications, a set of most-used functions can be estimated (e.g., instruction fetch, register and relative addressing, move and add instructions). The design is then optimized for these functions. Of course, the relative frequency data should always be updated to take into account new data.

Our purpose in writing this correspondence has been two-fold: to provide data about design tradeoffs and to suggest design models based on this data. It is hoped that the design data will stimulate the study of other models, while the results of the design models presented here have demonstrated their usefulness to designers.

## ACKNOWLEDGMENT

The authors would like to thank M. Reich and M. Tsao for their assistance in formulating the IBM S/360-S/370 model. The engineering documentation was supplied by Digital Equipment Corporation.

## REFERENCES

[1] W. D. Connors, J. H. Florkowski, and S. K. Patton, "The IBM 3033: An inside look," *Datamation*, vol. 25, pp. 198-218, May 1979.
[2] C. G. Bell, R. Cady, H. McFarland, B. Delagi, J. O'Loughlin, R. Noonan, and W. Wulf, "A new architecture for minicomputer-The DEC PDP-11," in *Proc. AFIPS Conf.*, vol. 36, 1970, pp. 657-675.
[3] C. G. Bell, J. C. Mudge, and J. McNamara, *Computer Engineering: A DEC View of Hardware Systems Design*. Bedford, MA: Digital, 1978.
[4] E. A. Snow and D. P. Siewiorek, "Impact of implementation design tradeoffs on performance: The PDP-11, a case study," Dep. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, Res. Rep., 1978.
[5] ——, "Impact of implementation design tradeoffs on performance: The PDP-11, a case study," in *Computer Engineering: A DEC View of Hardware Systems Design*, 1978, pp. 327-364, ch. 14.
[6] J. C. Mudge, "Design decisions achieve price/performance balance in midrange minicomputers," *Comput. Design.*, vol. 16, pp. 87-95, Aug. 1977.
[7] W. D. Strecker, private communication, 1976.
[8] R. P. Case and A. Padegs, "Architecture of the IBM system/370," *Commun. Ass. Comput. Mach.*, vol. 21, pp. 73-96, Jan. 1978.
[9] D. P. Siewiorek, C. G. Bell, and A. Newell, *Computer Structures: Principles and Examples*. New York: McGraw-Hill, 1981.
[10] Data Pro Research Corporation, 70c-491-04a until 70c-491-05p, Delran, NJ, May 1978.
[11] Cobol Analysis System (CAS) Timing Test, U.S. Steel, May 1978.

# Double-Edge-Triggered Flip-Flops

STEPHEN H. UNGER

*Abstract*—A conventional positive-edge-triggered flip-flop (FF) senses and responds to the control input or inputs at the time the clock input is changing from 0 to 1. It does not respond at all to changes in the opposite direction. Negative-edge-triggered FF's behave in a complementary manner. Thus, these FF's can respond at most once per clock pulse cycle. It is proposed that double-edge-triggered (DET) FF's, responding to both edges of the clock pulse would have advantages with respect to speed and energy dissipation.

Manuscript received October 17, 1980; revised February 14, 1981.
The author is with the Department of Computer Science, Columbia University, New York, NY 10027.

**Several alternate designs are developed for both $D$ and $JK$ type DET-FF's. These vary in speed, reliability, and component complexity. The designs nicely illustrate several aspects of sequential circuit design, including races, hazards, decomposition techniques, and state assignments.**

*Index Terms*—**Asynchronous, clock pulses, decomposition, $D$-flip-flop, edge triggering, flip-flops, $JK$-flip-flops, sequential circuits.**

## I. Introduction

An edge-triggered $D$-flip flop ($D$-FF) [1] has a clock input ($C$) and a data input ($D$). Immediately after the $C$-signal changes from 0 to 1, the output $Q$ assumes the value of $D$ and holds that value until the next positive going $C$-signal. Such a FF is said to trigger on the leading, or positive, edge of the clock pulse. Some FF's are designed to trigger on the trailing, or negative, edges of the $C$-signals. There are also edge-triggered $JK$-FF's [1] that respond to the $J$ and $K$ signals following a $C$-transition (again they may be of either the positive or negative edge sensing type). The $JK$-FF output changes from 0 to 1 if $J = 1$ (independent of $K$) and changes from 1 to 0 if $K = 1$ (independent of $J$).

The advantages of edge triggering is that the control inputs ($D$ or $J$ and $K$) may be changed at any time not in the neighborhood of a triggering edge of the $C$-signal. It also reduces sensitivity to noise pulses.

If the minimum interval between consecutive changes in the state of an edge-triggered FF is $L$ in a synchronous system, then the clock pulse frequency must be at least $1/L$. During each clock pulse period, one of the two transitions of the $C$-signal accomplishes nothing, although it will produce changes in the outputs of some of the logic elements internal to the FF's. Such activity is undesirable, since it results in increased power dissipation for virtually every technology now in use for implementing logic circuits. (In the case of $C$-MOS logic [2], there is essentially *no* power dissipated except when a transition is occurring.) If FF's trigger on *both* edges of $C$-pulses, then the clock pulse generator operates at half the frequency for the same data rate. This in itself would reduce the cost and power dissipation of the clock pulse generator and of the clock pulse distribution system, and would also eliminate meaningless state changes at the outputs of various gates. One would also expect to be able to increase data rates to some extent.

Several designs are presented here for double-edge-triggered (DET) $D$-FF's and for DET $JK$-FF's. The simplest designs in terms of logic complexity require delay elements which reduce allowable operating speeds. With the other designs, roughly 50–100 percent more complex than the corresponding single-edge-triggered circuits, no delay elements are necessary so that maximum operating speeds are attainable. Only the basic operations are implemented; no set or clear operations are built in, and the complements of the outputs are not produced. These features would not be difficult to design in. Practical implementations would, in most cases, also utilize such elements as NAND and NOR gates or networks of pass transistors, rather than the AND-OR-INVERTOR logic shown here.

The design of DET FF's is a good application of the theory of asynchronous sequential switching circuits [3]; of particular interest perhaps is the use of decomposition techniques.

## II. Flow Table Descriptions of DET-$D$-FF's

Table I is a primitive flow table for a DET-$D$-FF. Note that simultaneous changes of $D$ and $C$ are treated as though one of these variables changed first, but that it does not matter *which* changed first. This is a realistic assumption, since we can never rely on *exact* simultaneity for any pair of events. The option of treating a simultaneous change in either of two ways is left open for exploitation later in the design process.

Using well-known methods [3] it is not difficult to show that there are precisely two minimal-row covers of this table, as shown in Table II (A) and (B). The parenthesized sets of numbers to the right of each row of (A) and (B) indicate the rows of Table I that are covered by the rows of the reduced tables.

TABLE I
Primitive Flow Table for DET-$D$-FF

| | CD = 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 1 | (1),0 | 2,0 | 4,0/5,- | 3,0 |
| 2 | 1,0 | (2),0 | 5,- | 3,0/6,- |
| 3 | 1,0 | 2,0/7,- | 4,0 | (3),0 |
| 4 | 1,0/8,- | 7,1 | (4),0 | 3,0 |
| 5 | 1,-/8,1 | 7,1 | (5),1 | 6,1 |
| 6 | 1,0 | 2,-/7,1 | 5,1 | (6),1 |
| 7 | 8,1 | (7),1 | 5,1 | 3,-/6,1 |
| 8 | (8),1 | 7,1 | 4,-/5,1 | 3,- |

TABLE II
Minimal Row Covers of Table I

(A)

| | CD = 00 | 01 | 11 | 10 | |
|---|---|---|---|---|---|
| 1 | (1),0 | (1),0 | 4,- | 2,0 | (12) |
| 2 | 1,0 | 3,- | (2),0 | (2),0 | (34) |
| 3 | (3),1 | (3),1 | 4,1 | 2,- | (78) |
| 4 | 1,- | 3,1 | (4),1 | (4),1 | (56) |

(B)

| | CD = 00 | 01 | 11 | 10 | |
|---|---|---|---|---|---|
| 1 | (1),0 | 2,0 | 4,0 | (1),0 | (13) |
| 2 | 1,0 | (2),0 | 3,1 | (2),1 | (26) |
| 3 | 4,1 | (3),1 | (3),1 | 2,1 | (57) |
| 4 | (4),1 | 3,1 | (4),0 | 1,0 | (48) |

Table II (A) and (B) are equivalent to one another with respect to single-input-change (SIC) operation. But they describe different responses to multiple input changes, a difference without practical significance, but which leads to quite different implementations. Neither has any essential hazards, but (A) has $d$-transitions (e.g., from row 1 with $CD = 01$, when $C$ changes), while $B$ has none.

## III. A Simple Realization of the DET-$D$-FF

The theory of machine structure introduced by Hartmanis and Stearns [4] has been applied to asynchronous machines by Tan [5], [6], [3]. Here we shall use the term "closed partition" rather than "partition with SP," in conformity with current usage.

An examination of Table II(A) reveals that the partition (13, 24) is closed. This makes possible a serial decomposition with the front end machine described by Table III(A) corresponding to (13, 24). The second machine must distinguish state 1 from state 3 and state 2 from state 4, which can be done by realizing partition (12, 34) or (14, 23).

Choosing (12, 34) yields Table III(B) as the second machine. Note that its third input is the state of the first machine, corresponding to the value of $y_1$.

Machine III(A) is realizable very simply. We find that $Y_1 = C$. For machine III(B), the corresponding expression for $Y_2$ is rather more complex, namely

$$Y_2 = Dy_2 + \bar{C}\bar{y}_1y_2 + Cy_1y_2 + \bar{C}Dy_1 + CD\bar{y}_1.$$

Note that here, as in the sequel, the logic is designed to be hazard-free.

The $Y_2$ expression can be manipulated into the form

$$Y_2 = (C \oplus y_1)D + (\overline{C \oplus y_1})y_2 + Dy_2.$$

This can be recognized as a description of a latch [1], in which $C \oplus y_1$ is the $C_L$-input and $D$ is the $D_L$-input. The result is the circuit shown in Fig. 1.

This circuit is one that might easily have been produced in an intuitive manner, without benefit of any high powered theory. The circuit producing the $C_L$ input to the latch is a straightforward one for producing a pulse whenever $C$ changes in either direction. This

TABLE III
DECOMPOSITION OF TABLE II(A)

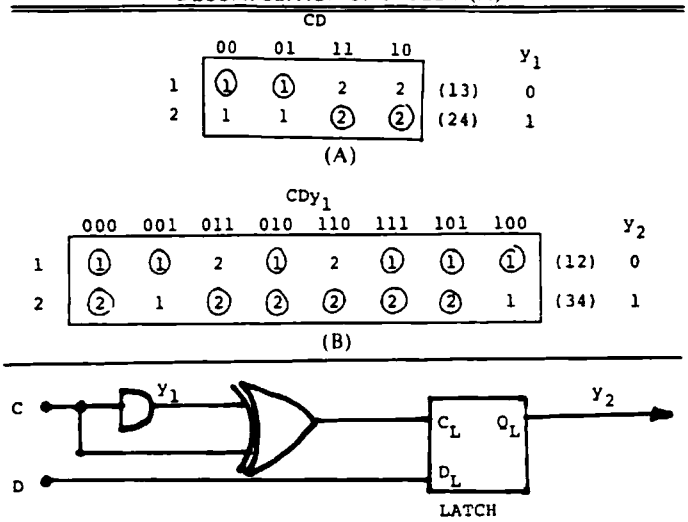



Fig. 1. Simple realization of a DET-*D*-FF.

pulse then causes the latch to copy the value of $D$ at the time of the $C$ transition. The latch holds this value at least until the next edge of the $C$-signal.

In terms of switching theory, the Fig. 1 circuit has a critical race between $y_1$ and $y_2$. Correct operation necessitates that $y_2$ respond faster than $y_1$ in all cases. This imposes the need for the delay element at one of the inputs to the EXCLUSIVE-OR-gate. This delay must exceed by a suitable amount the delay in the feedback path internal to the latch. In this case, "fixing" the race is feasible, since there is no need to make the latch delay exceed any other bound, i.e., it can be made as small as possible. However, the need for the $y_1$ delay element, which must be chosen so that its *minimum* value is guaranteed to exceed a value proportional to the *maximum* value of the latch delay, will clearly result in a lowered data rate.

## IV. TWO FASTER DET-*D*-FF'S

Referring to Table II(B), it can be shown that partitions (12, 34) and (14, 23) are both closed. Since their product is the 0-partition, this means that they correspond to a parallel decomposition. Table IV shows the submachines corresponding to the two partitions.

An examination of these tables reveals that both correspond to latches. The first latch(A) has as its $C_L$ and $D_L$ inputs, $C$ and $D$, while Table IV(B) latch also has $D$ as its $D_L$-input, but $\bar{C}$ as its $C_L$-input. Letting $y_1$ and $y_2$ be the outputs of the (A) and (B) latches, respectively, it can be seen from Table II(B) that for the overall circuit

$$Q = \bar{C}y_1 + Cy_2 + y_1y_2.$$

Manipulating this logic yields finally the circuit of Fig. 2. No delay elements are necessary, so that this is a relatively fast and reliable circuit.

Combining the logic internal to the latches with the output logic results in a slightly more economical realization specified by the expressions below:

$$Y_1 = \bar{C}y_1 + D(C + y_1)$$
$$Y_2 = Cy_2 + D(\bar{C} + y_2)$$
$$Q = [\bar{C}y_1] + [Cy_2] + y_1y_2.$$

(The bracketed terms in the $Q$ expression are shared with the $Y_1$ and $Y_2$ expressions, and the cost, measured in gate inputs, is 21.)

Table II(A) can also be realized by a parallel decomposition. In this case, the two closed partitions (1, 24, 3) and (13, 2, 4) each have three blocks, so that the component machines, shown in Table V, each have three states. (Note that they are equivalent except for an inversion of $C$.)

TABLE IV
DECOMPOSITION OF TABLE II(B)

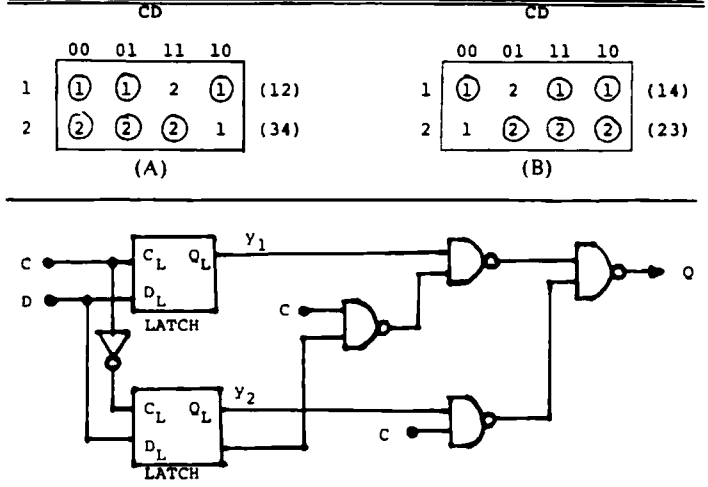
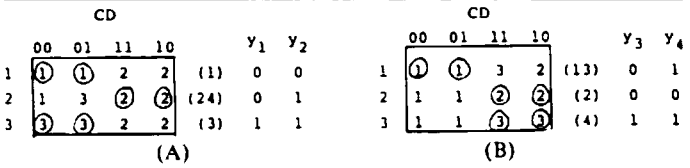



Fig. 2. DET-*D*-FF by parallel decomposition.

TABLE V
DECOMPOSITION OF TABLE II(A)




With the given state assignments, the $Q$-output, which is one for states 3 and 4 of Table II(A), is easily generated from $y_1$ and $y_3$. The logic expressions are

$$Y_1 = \bar{C}(y_1 + Dy_2)$$
$$Y_2 = C + y_1 + [Dy_2]$$
$$Y_3 = C(y_3 + DY_4)$$
$$Y_4 = \bar{C} + y_3 + [Dy_4]$$
$$Q = y_1 + y_3.$$

The cost measured in gate inputs is 20, slightly less than for the Table IV decomposition, even though four $y$-variables are used. With the given state assignments there is no problem with the $d$-transitions. In both cases $\bar{C}$ must be generated by a inverter if it is not already available.

The result corresponding to Table IV could have been obtained by using a simple 2-variable state assignment for Table II(B) (a very simple example of a Tracey assignment) and the Table V result could have been obtained using a Liu assignment (requiring four variables) for Table II(A). A Tracey assignment for Table II(A), which would require only two $y$-variables, leads to a logic cost about double that for the solutions shown here.

Still another realization can be derived from a *serial* decomposition of Table II(A) based on the closed partition (13, 2, 4), which was used in the previous synthesis. The flow table for the corresponding front machine is repeated as Table VI(A). To complete the implementation, it is necessary that the second machine distinguish states 1 and 3. This is done here using a *cover* (124, 234) instead of a partition, since, in this case, it is then not necessary to have $C$ as an input to the second machine. Its flow table is Table VI(B), with inputs $S_i$, for the state of the front machine, and $D$. With the given state assignments, the $d$-transitions are all properly covered, so there need be no concern over stray delays.

The logic expressions are

$$Y_1 = C(y_1 + Dy_2)$$
$$Y_2 = \bar{C} + y_1 + [Dy_2]$$

TABLE VI
SERIAL DECOMPOSITION OF TABLE II(A)

CD

| | 00 | 01 | 11 | 10 | | $y_1$ | $y_2$ |
|---|---|---|---|---|---|---|---|
| 1 | (1) | (1) | 3 | 2 | (13) | 0 | 1 |
| 2 | 1 | 1 | (2) | (2) | (2) | 0 | 0 |
| 3 | 1 | 1 | (3) | (3) | (4) | 1 | 1 |

(A)

$DS_i$

| | 01 | 02 | 03 | 13 | 12 | 11 | | $y_3$ |
|---|---|---|---|---|---|---|---|---|
| 1 | (1) | (1) | (1) | 2 | 2 | (1) | (124) | 0 |
| 2 | (2) | 1 | 1 | (2) | (2) | (2) | (234) | 1 |

(B)

TABLE VII
FLOW TABLE FOR DET-*JK*-FF

CJK

| | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 1 | (1),0 | (1),0 | 2,0 | 2,0 | 4,0/7,- | 4,0/8,- | 3,0 | 3,0 |
| 2 | 1,0 | 1,0 | (2),0 | (2),0 | 7,- | 8,- | 3,0/8,- | 3,0/7,- |
| 3 | 1,0 | 1,0 | 2,0/6,- | 2,0/5,- | 4,0 | 4,0 | (3),0 | (3),0 |
| 4 | 1,0/5,- | 1,0/6,- | 6,- | 5,- | (4),0 | (4),0 | 3,0 | 3,0 |
| 5 | (5),1 | 6,1 | 6,1 | (5),1 | 7,1 | 4,-/8,1 | 3,-/8,1 | 7,1 |
| 6 | 5,1 | (6),1 | (6),1 | 5,1 | 4,-/7,1 | 4,- | 3,- | 3,-/7,1 |
| 7 | 5,1 | 1,-/6,1 | 2,-/6,1 | 5,1 | (7),1 | 8,1 | 8,1 | (7),1 |
| 8 | 1,-/5,1 | 1,- | 2,1 | 2,-/5,1 | 7,1 | (8),1 | (8),1 | 7,1 |

$$Y_3 = D(y_1 + \bar{y}_2 + y_3) + \bar{y}_1 y_2 y_3$$

$$Q = y_1 + [y_2 y_3].$$

The logic cost is 22 gate inputs and the complements of $y_1$ and $y_2$, as well as of $C$, are needed. The three lowest cost delay-free implementations, although quite different, are all very close together in cost.

It is interesting that the machine of Table VI(A) (with a different state assignment) is used as the front machine in a serial decomposition that corresponds to the most popular implementation of the conventional *single* edge-triggered *D*-FF [1].

## V. THE DET-*JK*-FF

Table VII is a minimal-row equivalent of a primitive flow table describing a DET-*JK*-FF. (On either edge of the *C*-input, the FF responds to the signals on the *J* and *K* terminals by going to the set state if only *J* is on, to the reset state if only *K* is on, and changing state if both are on.) Again, for multiple input changes the specification allows the option of choosing the response corresponding to serial changes in any order.

A quick and dirty solution analogous to that of Fig. 1 for the DET-*D*-FF is shown in Fig. 3. A delay and EXCLUSIVE-OR-gate are used to generate a pulse following each change in *C*. This pulse is then used as the *C*-input to a conventional edge-triggered *JK*-FF (it does not matter whether it is of the positive or of the negative edge triggered variety). As in the previous case, simplicity is purchased here at the cost of speed and possibly reliability. Two solutions of increasingly higher quality, but of greater cost, are developed below.

There are two different minimal row tables that cover Table VII, differing in their responses to multiple input changes. These are shown as Tables VIII (A) and (B) corresponding to grouping the states as [12, 78, 56, 34] or [13, 28, 57, 46], respectively.

For the state assignment shown, Table VIII (A) can be realized by the logic expressions

$$Y_1 = \bar{C} y_2 (J + y_1) + \bar{K} y_1 + C \bar{y}_2 [J + y_1]$$

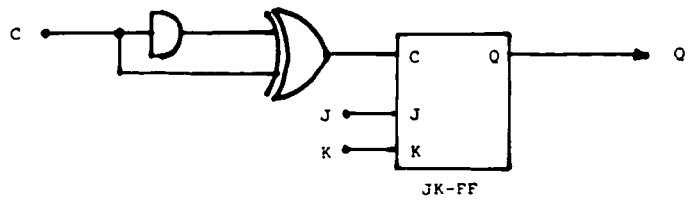



Fig. 3. A crude realization of a DET-*JK*-FF.

TABLE VIII
REDUCED VERSIONS OF TABLE VII

CJK

| | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 | $y_1$ | $y_2$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | (1),0 | (1),0 | (1),0 | (1),0 | 4,- | 4,- | 2,0 | 2,0 | 0 | 0 |
| 2 | 1,0 | 1,0 | 3,- | 3,- | (2),0 | (2),0 | (2),0 | (2),0 | 0 | 1 |
| 3 | (3),1 | (3),1 | (3),1 | (3),1 | 4,1 | 2,- | 2,- | 4,1 | 1 | 1 |
| 4 | 3,1 | 1,- | 1,- | 3,1 | (4),1 | (4),1 | (4),1 | (4),1 | 1 | 0 |

(A)

CJK

| | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 | $y_1$ | $y_2$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | (1),0 | (1),0 | 2,0 | 2,0 | 4,0 | 4,0 | (1),0 | (1),0 | 0 | 0 |
| 2 | 1,0 | 1,0 | (2),0 | (2),0 | 3,1 | (2),1 | (2)1 | 3,1 | 0 | 1 |
| 3 | (3),1 | 4,1 | 4,1 | (3)1 | (3),1 | 2,1 | 2,1 | (3),1 | 1 | 1 |
| 4 | 3,1 | (4),1 | (4),1 | 3,1 | (4),0 | (4),0 | 1,0 | 1,0 | 1 | 0 |

(B)

$$Y_2 = [\bar{C} y_2 (J + y_1)] + \bar{C}[\bar{K} y_1] + K y_2 ([J + y_1] + C) + \bar{y}_1 (J y_2 + C(\bar{J} + y_2))$$

$$Z = y_1.$$

Terms in square brackets are realized once and used several times via fan-out. The cost of this realization in gate inputs is 34. Inverters

are needed for $C$, $J$, and $K$, and the maximum logic depth (which is relevant to the circuit speed) is five.

There is, however, a problem with respect to sequential hazards. Consider initial state 1-010 (row −1, column 010). For a change in $C$, the resulting transition is a $d$-transition [3] which, with the given state assignments, could result in the final state being 2-110, instead of 4-110. This would occur if there were no sufficiently large delay element in the $y_1$ branch and if the stray delays in the logic and wiring are related to one another in an unfortunate manner. In particular, trouble results if the stray delays cause the following sequence of events after $C$ changes.

1) $Y_1$ sees the $C$-change and responds by changing $y_1$ from 0 to 1.

2) $Y_2$ sees the $y_1$-change (thus seeing the system in 4-010) causing $y_2$ to change from 0 to 1.

3) $Y_1$ sees the $y_2$ change (so that the system appears to be in 2-110) and responds by changing $y_1$ back to 0.

4) $Y_2$ sees the $y_2$ change, then the second $y_1$-change and then the $C$-change. Thus, it sees the system go from 4-010 to 3-010 to 2-010 and then to 2-110, holding $y_2$ at 1 throughout.

5) $Y_1$ sees no further changes, missing *both* of the $y_1$ changes due to a relatively large inertial stray delay from $y_1$ to $Y_1$.

The system then terminates in 2-110. (If the $y_1$ to $Y_1$ delay is pure, rather than inertial, there will be an oscillation between 2-110 and 4-110.)

The likelihood of the delays being so related is rather small, since this would require delays in certain paths to exceed the delays in paths containing several times as many gates. Thus, in practice the problem is not likely to be serious, although it should be examined. (There are three more similar transitions in the table, starting in states 2-110, 3-001, and 4-101, all involving changes in $C$.)

The $d$-transition problem can be eliminated altogether by choosing a different state assignment. In this case, however, it would mean using four additional $y$-variables and the resulting logic would be excessively costly. A better solution is to use a design based on Table VIII(B), which has no $d$-transitions. This costs only a little more than the given Table VIII(A) design and also has less logic depth.

The logic expressions are

$$Y_1 = \bar{C}y_1 + J\bar{y}_2(C + y_1) + \bar{K}y_2[C + y_1] + J\bar{K}y_1$$

$$Y_2 = Cy_2 + J\bar{y}_1(\bar{C} + y_2) + \bar{K}y_1[C + y_2] + J\bar{K}y_2$$

$$Q = [\bar{C}y_1] + [Cy_2] + [\bar{K}y_2(C + y_1)].$$

Again the terms in square brackets are generated once and used more than once. Note that it is more economical to fan out to two places from the $C + y_1$ OR-gate in the $Y_1$-circuit than to factor out the $(C + y_1)$ term. The third term in the $Q$-expression is needed to avoid combinational hazards in transitions between states 3-000 and 3-100 and between states 3-010 and 3-110. The term $\bar{K}y_1y_2$ would have sufficed, but the larger term used includes this and is available at no extra cost since it is needed to produce $Y_1$. Total cost is 37 gate inputs and the maximum logic depth is 3. Complements of $C$, $K$, $y_1$, and $y_2$, but not of $J$ are needed, and no *un*complemented $K$ is used. A simple transformation converts all of the gates to NAND-gates with the circuit (shown in Fig. 4) unchanged, except that $(C + y_1)$ becomes $(\bar{C} + \bar{y}_1)$ and $(\bar{C} + y_2)$ becomes $(C + \bar{y}_2)$. Adding clear or set controls is not difficult. There are no hazards of any kind to worry about and no delay elements are needed. This seems to be a clearly superior solution.

## VI. Conclusions

Double-edge-triggered flip-flops (DET-FF's) offer potential advantages with respect to speed and power supply requirements, since fewer redundant logic level changes occur. The price paid is in the form of a substantial increase (perhaps 50–100 percent) in the number of components required to build such devices.

Several designs have been developed for both $D$ and $JK$ type DET-FF's, and these illustrate a number of interesting techniques and principles of sequential circuit design. Tradeoffs are possible among speed, reliability, and circuit cost. Although the designs presented are based on the use of gate type logic, the same principles are applicable to designs using pass transistor logic.

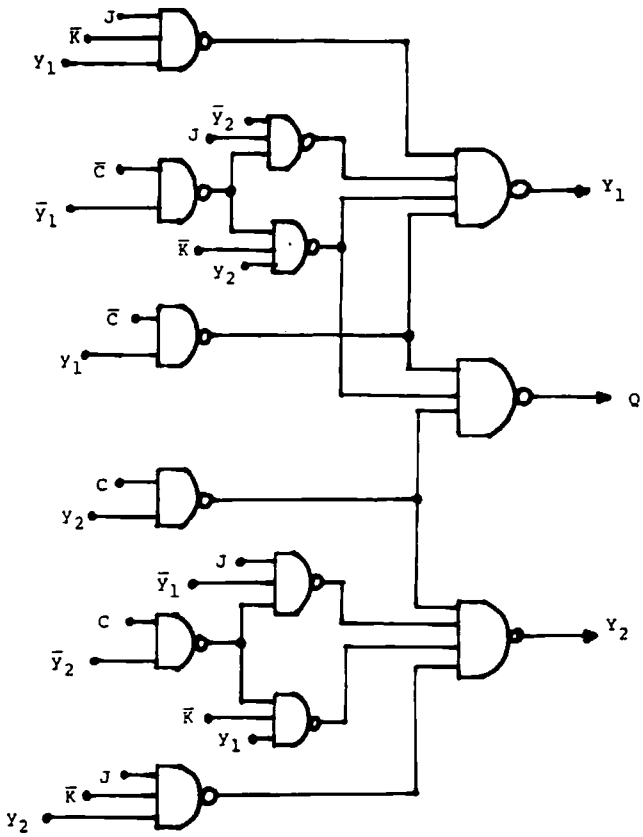



Fig. 4. High quality realization of DET-$JK$-FF.

In the case of single-edge-triggered $JK$-FF's, the functions involve both essential hazards and $d$-transitions. The former mandate control of delay paths. The usual IC-chip designs make use of delay control in very clever circuits [1] with relatively low gate-input counts. Strangely enough, the DET-$JK$-FF flow tables do not have essential hazards, and one version has no $d$-transitions either. Thus, delay control is not necessary. However, it remains to be seen whether delay control can be used to produce simpler, yet fast and reliable DET-FF's.

The circuits of Fig. 1, Fig. 2 (actually the more economical version described by the logic expressions), Fig. 3, and Fig. 4 have been built in the laboratory using small scale integrated circuit components (TTL) by N. Vi Pho, a Columbia University graduate student. They all worked properly.

## References

[1] *The TTL Data Book for Design Engineers*, 2nd ed., Texas Instruments Inc., 1976.

[2] "RCA COS/MOS integrated circuits," RCA Corp., 1978.

[3] S. H. Unger, *Asynchronous Sequential Switching Circuits*. New York: Wiley-Interscience, 1969.

[4] J. Hartmanis and R. E. Stearns, *Algebraic Structure Theory of Sequential Machines*. Englewood Cliffs, NJ: Prentice-Hall, 1966.

[5] C. J. Tan, "Synthesis of asynchronous sequential switching circuits," Ph.D. dissertation, Dep. Elec. Eng., Columbia Univ., New York, 1969.

[6] C. J. Tan, P. R. Menon, and A. D. Friedman, "Simplification and decomposition of asynchronous sequential circuits," *IEEE Trans. Comput.*, vol. C-18, pp. 830–838, Sept. 1969.

## Comments on "Very Fast Fourier Transform Algorithms Hardware for Implementation"

S. PRAKASH AND V. V. RAO

The object of this correspondence is to point out and correct some errors which have occurred in the above paper.[1]

1) The definition of DFT as given by (1) is

$$A_r = \sum_{K=0}^{N-1} B_K W_N^{rK}, \qquad r = 0, 1, 2, \cdots, N-1$$

where

$$W_N = \exp(-j2\pi/N), \qquad j = \sqrt{-1}.$$

But the 16-point DFT as calculated by the signal flow graph of Fig. 2 follows the equation

$$A_r = \sum_{K=0}^{N-1} B_K W_N^{-rK}.$$

This fact should have been clearly mentioned in the paper, as it often leads to confusion. The standard terminology for digital signal processing has been defined by Rabiner *et al.* [1].

2) In Fig. 2 the "Twiddle factor" appearing in the line corresponding to $B_{15}$ should be $-w_3^- w_4^+$ instead of $-jw_3^- w_4^+$, as given.

3) Some errors have occurred in Fig. 9 which gives the radix-16 FFT algorithm:

a) in view of the previous correction the last but one step in the subroutine $W16\ (N, K)$ should be changed to

if $(i = 6, 7, 14)$ then $B_1 \leftarrow jB_1$; instead of

if $((i/2) \bmod 4 = 3)$ then $B_1 \leftarrow jB_1$;

b) in the subroutine $W4(N, K)$ the last but one step should be changed to

for $m := (i + 3K)$ to $(i + 4K - 1)$ do instead of

for $m := (i - K + 1)$ to $i$ do;

c) some typographical errors have occurred in the last two steps of the subroutine $\theta_3\ (OP)$. They should be read as

$$\{B_1(\text{real}) \leftarrow \text{temp(real)}2^{-1}\ OP(-B_1(\text{imag})2^{-2})$$
$$B_1(\text{imag}) \leftarrow \text{temp(imag)}2^{-1}\ OP(+B_1(\text{real})2^{-2})\}.$$

4) Some more typographical errors have occurred in the equations appearing on page 337. Starting from the equation in the last line of the first column, the corrected equations are

$$(f_2/f_2) = (128/309)_{\text{decimal}} = \left(\frac{0.010000000}{0.100110101}\right)_{\text{binary}}$$

$$x' \leftarrow 0.100110101x \mp 0.01000000y$$

$$y' \leftarrow 0.100110101y \pm 0.01000000x$$

$$x' \leftarrow ((x + x2^{-2}) - (x + x2^{-2})2^{-5} - x2^{-8})2^{-1} \mp y2^{-2}$$

$$y' \leftarrow ((y + y2^{-2}) - (y + y2^{-2})2^{-5} - y2^{-8})2^{-1} \pm x2^{-2}.$$

Manuscript received March 5, 1981.

The authors are with the Department of Electrical Engineering, Indian Institute of Technology, Madras, India.

[1] A. M. Despain, *IEEE Trans. Comput.*, vol. C-28, pp. 333-341, May 1979.

### REFERENCES

[1] L. R. Rabiner *et al.*, "Terminology in digital signal processing," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 322-337, Dec. 1972.

*Author's Reply*[2]

ALVIN M. DESPAIN

The author would like to acknowledge Prakash and Rao for their corrections. Unfortunately, there are further errors in need of correction.

1) On p. 333, first column, the last equations should read

$$W_{16T}^{16mtT} = 1$$

and

$$W_{16T}^{Ttl} = W_{16}^{lt}$$

$$W_{16T}^{16sm} = W_T^{sm}.$$

2) On p. 334, first column, the fifth equation should be

$$W_{N_1} = W_N^{sj}.$$

3) On p. 334, equation (6) should be

$$A_t = \sum_{l=0}^{15} B_l W_{16}^{lt}. \tag{6}$$

4) On p. 334, equation (7) should be

$$A_{u+4v} = \sum_{e=0}^{3} \sum_{f=0}^{3} B_{e+4f} W_{16}^{(u+4v)(e+4f)}. \tag{7}$$

5) In Fig. 5, the control line for adder $A_1(r)$ should be extended to the left and attached to the control line that leads to $A_1(i)$.

6) In Fig. 7, the "+" input lines to both the ADD/SUB units should be weighted by $2^{-1}$ to correspond to the equations at the bottom of p. 337.

7) The last two control lines at the bottom right of Fig. 11 should be labeled "$J10$" and "$J20$." The logic for "$J10$" should be modified to correspond to the revised routine "$W16$" of Fig. 9.

[2] Manuscript received November 11, 1980.

The author is with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720.

0018-9340/81/0600-0452$00.75 