High-Performance Two-Phase Micropipeline Building Blocks: Double Edge-Triggered Latches and Burst-Mode Select and Toggle Circuits

Kenneth Y. Yun
Department of ECE
UC San Diego

Peter A. Beerel
EE-Systems Department
USC

Julio Arceo
Department of ECE
UC San Diego

Abstract

This paper presents new high-performance building blocks for two-phase micropipelines. We develop pseudo-static Svensson-style double edge-triggered D-flip-flops (DETDFF) for datapath storage in place of traditional capture-pass or transmission gate latches. We compare a DETDFF FIFO buffer implementation with the current state-of-the-art micropipeline implementation using four-phase controllers designed by Day and Woods for the AMULET-2 processor. We implemented both designs in the MOSIS 1.2μm CMOS process and simulated them under the worst-case process corner with a 4.6V power supply and at 100°C. Our SPICE simulations show that the DETDFF design has 70% higher throughput. This higher throughput is due to latching the data on both edges of the latch control, removing the need of a reset phase and simplifying the control structures. In addition, we present two commonly used micropipeline event-control structures, the select and toggle elements, implemented using the extended-burst-mode 3D synthesis system. Detailed simulations demonstrate that our implementations are up to 50% faster than traditional implementations. This speed advantage can be primarily attributed to careful applications of generalized C-elements rather than discrete basic gates.

1 Introduction

Micropipelines is a popular design style for building asynchronous circuits, introduced by Ivan Sutherland in his Turing Award Lecture in 1988 [5]. Micropipelines is a building block approach which makes designing deeply pipelined circuits relatively easy and facilitates the reuse of combinational functional units previously designed for use in synchronous circuits. Micropipelines is based on a two-phase, or transition-signaling, event-driven communication protocol in which an event is either a low-to-high or high-to-low transition on a control wire with no distinction being made between the two. Since traditional latches are level-sensitive, however, traditional designs required two-to-four and four-to-two phase converters, which hinder performance. The AMULET group recently designed a new set of control blocks using four-phase event-driven communication which eliminates the need for phase-converters thereby significantly improving performance.

This paper introduces a simple, but effective, idea of replacing traditional latches (*e.g.,* capture-pass and transparent) in two-phase micropipelines with pseudo-static double edge-triggered D-flip-flops (DETDFF). DETDFFs are fast and compact when designed with two Svensson-style latches placed in parallel [1] and can be easily made pseudo-static. They seamlessly integrate into the two-phase micropipeline methodology without the need of costly two-to-four phase converters. DET flip-flops, however, have higher input capacitance than level-sensitive latches and do not provide a "latch-done" signal which means that additional timing assumptions must be analyzed and accounted for to ensure correct operations. We designed both a DETDFF FIFO buffer and the leading (four-phase) FIFO buffer designed by Day and Woods [2] in the MOSIS 1.2μm CMOS process using the same transistor sizing. We simulated both designs under the worst-case process corner with a 4.6V power supply and at 100°C. Our design achieves a cycle time of 3.9ns while Day and Woods's design achieves a cycle time of 6.7ns — 2.8ns slower. Thus, our design is 70% percent faster than their design. The cost of this improvement is that the DETDFF design requires approximately twice the area compared with convential designs and consumes up to four times as much power.

To extend the application of micropipelines to general pipeline circuits, Sutherland also introduced a set of event-control building blocks [5]. A C-element acts as the AND of two events, synchronizing two independent pipeline streams. An xor gate acts to OR (or MERGE) two independent events. A select element routes an input event one of two ways depending on a select input, facilitating data-dependent choice. A toggle element alternates between routing an input event to one of two outputs. We present improved generalized-C-element-based implementations of the more complicated select and toggle elements obtained using the burst-mode 3D synthesis systems [8, 7, 9].

We designed both the traditional implementations of these elements [4] and our implementations in the MOSIS 1.2μm CMOS process and simulated them under the worst-case process corner with a 4.6V power supply and at 100°C. Our designs demonstrated significant improvements in latency and cycle time, as much as 1.8ns vs 2.5ns (40%) for latency and 2.2ns vs 3.3ns (50%) for cycle time for the select element and 1.3ns vs 1.9ns (50%) for latency and cycle time for the toggle element. The results demonstrate that although the synthesis paradigm is based on the fundamental-mode assumption [6, 7], the two synthesized designs have negligible settling time delay and thus realistic environments can easily adhere to the fundamental-mode assumption. Compared with traditional designs, our select element is approximately 7approximately 38designs is ...

The organization of the remainder of the paper is as follows. Section 2 reviews background of micropipelines. Section 3 presents the DETDFF FIFO and our comparison with traditional pipeline structures. Section 4 describes the synthesis of our burst-mode implementations of the select and toggle elements and compares them with traditional designs. Section 5 presents our conclusions.

2 Background: Micropipelines

In a one-dimensional asynchronous FIFO data is usually depicted moving from left to right, as illustrated in Figure 1. Once stage i receives data, it sends a request to stage $i+1$ to pass the data on. If stage $i+1$ is empty, it latches the new data and sends an acknowledgment to stage i. This tells stage i that the data transfer is complete and that stage i can accept new data.

Numerous implementations of this request/acknowledge communication has been suggested and analyzed. As depicted in figure 1, Sutherland's original paper [5] suggests an implementation in which neighboring stages communicate by two wires, one wire for request and one wire for acknowledgment. The request wire is labeled R_{out} at the output of stage i (the source) and R_{in} at the input to the stage $i+1$ (the sink). Similarly, the acknowledge wire is labeled A_{in} at the input to stage $i+1$ (the source) and A_{out} at the output of stage i (the sink). Sutherland original paper suggests a two-phase handshaking protocol, illustrated in Figure 2a), in which both the low-to-high transition and the high-to-low transitions, both referred to as *events*, of any control wire have the same meaning. For example, both R_{in} events indicates that new data is valid at the input to the stage.

PUT IN DESCRIPTION OF CAPTURE PASS LATCHES HERE

Because capture-pass latches are large and consumer significant power, the AMULET group proposed using transparent latches with with level-sensitive enables. Building fast implementations using these latches is challenging because they requires designing a two-to-four-phase interface around the level-sensitive enables. A simple implementation using an xor and a toggle element is illustrated in figure 3 [3, 4]. The cycle time of this two-phase design includes the delay overhead of both the xor and toggle elements, limiting its performance.

PAVER DEMONSTRATES THIS CAN BE ACHIEVED WITH a Xand a X

To improve the performance, the AMULET group suggested a number of optimizations [4, 2]. The fastest circuits they have developed abandon the two-phase paradigm and adopt a four-phase signaling depicted in figure 2b). In four-phase (or level-sensitive) signaling the rising and falling transitions of the interface signals are distinguished. The rising transitions are the active transitions that indicate data valid and data consumed and the falling transitions reset the control wires for the next communication.

To implement four-phase micropipeline control circuits careful consideration must be taken into the placement of the reset phase of the request and acknowledge control lines. The fastest four-phase micropipeline control circuit for FIFO applications that they developed is the semi-decoupled circuit depicted in figure 4.

3 Two-Phase Pipeline Circuit

We can build a simple 2-phase asynchronous pipeline circuit using *double edge-triggered D-flip-flops* as storage elements and a C-element to control each stage of the pipeline as shown in figure 5.

This pipeline circuit is similar to Sutherland's micropipeline, except that it uses double edge-triggered flip-flops in place of capture-pass latches and relies on a simple set of timing constraints for correct operations.

To make this circuit simpler to understand, assume negligible delays on latch control ("clock") buffers for DETDFFs for now. Consider stage i whose inputs are R_{in} and $\overline{A_{\text{out}}}$ and whose outputs are $\overline{A_{\text{in}}}$ and R_{out}. When its left neighbor toggles R_{in} signaling that the data D_{in} is valid, stage i's pipeline control toggles its output $\overline{A_{\text{in}}}$ (if the previous request to the right neighbor has been acknowledged). Toggling of the control C-element acknowledges the receipt of data to its left neighbor, latches the data in stage i's storage (DETDFFs), and enables R_{out} to toggle after a *bundling* delay. The bundling delay is necessary to provide sufficient data setup time for stage $i+1$, *i.e.*, D_{out} of stage i must be valid before stage $i+1$'s latch control toggles.

In practice, the latch control buffers incur significant delays (1.9ns delay to drive 32 DETDFFs in our simulation). Thus, the data can be delayed accordingly. In that case, toggling of the request

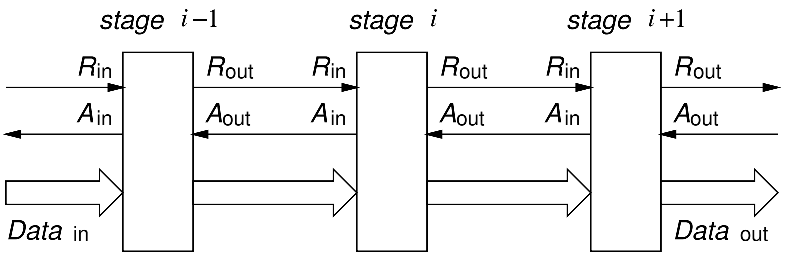


Figure 1: An asynchronous pipeline.

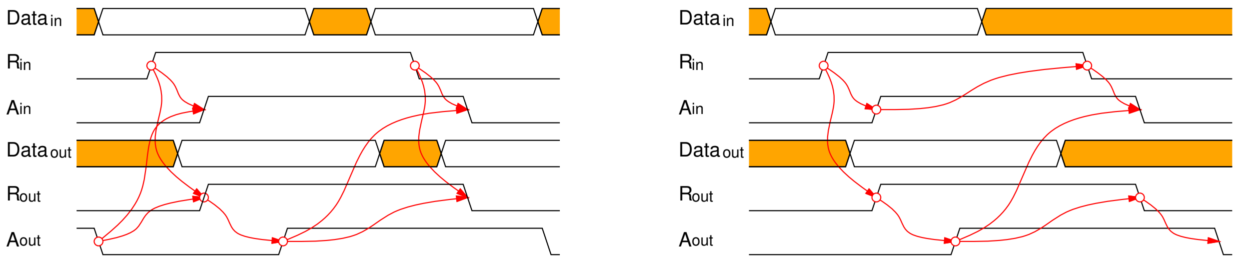


Figure 2: a) The two-phase signaling protocol. b) The four-phase signaling protocol.

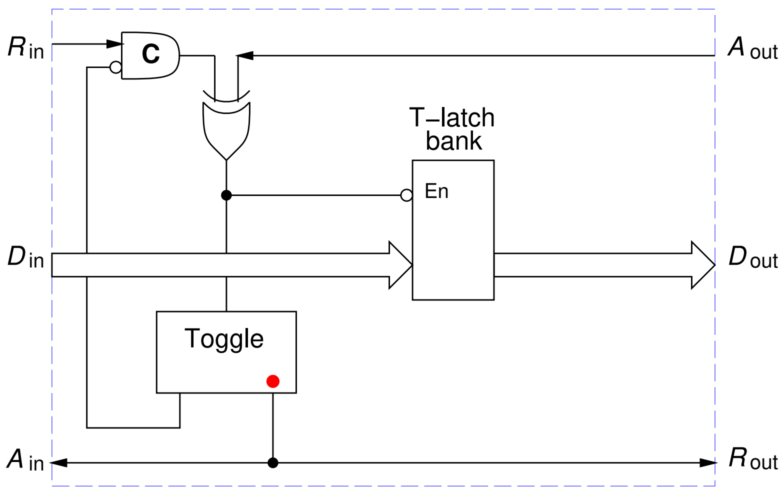


Figure 3: Simplified micropipeline stage based on transparent latches and a two-to-four-phase interface.

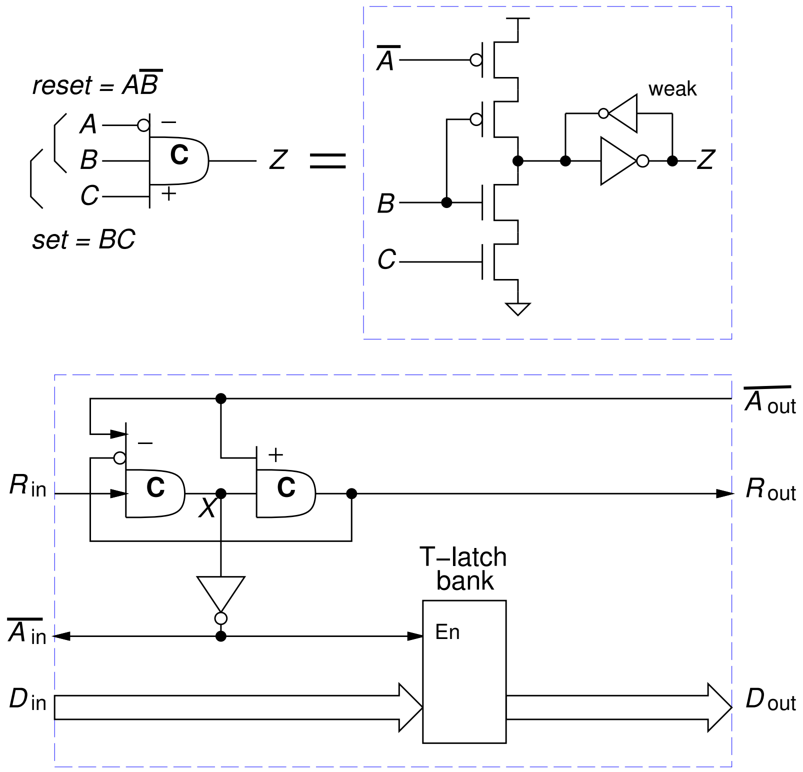


Figure 4: Generalized C-element notation (top) and a semi-decoupled four-phase micropipeline stage (bottom).

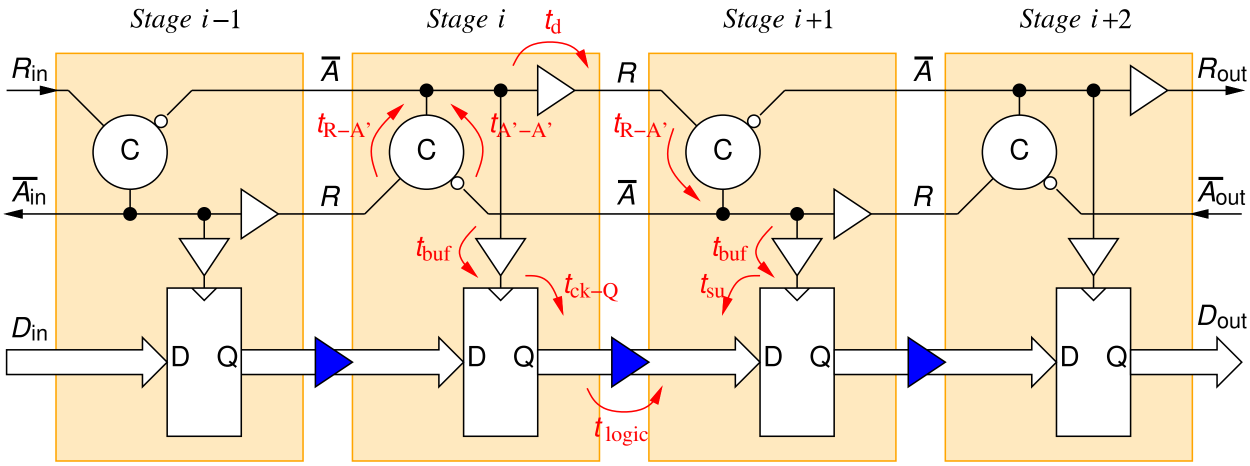


Figure 5: 2-phase pipeline circuits with double edge-triggered D-FFs.

line means that the data will be available at the next stage after some delay (roughly the same as the latch control buffer delay).

In order for this pipeline circuit to function correctly, two timing constraints must be met:

- *Data setup time*
  The bundling delay (t_d) must be long enough to meet the data setup time (t_{su}):

$$t_{\mathrm{d}}{}^{i} + t_{\mathrm{R}\to\mathrm{A}'}{}^{i+1} + t_{\mathrm{buf}}{}^{i+1} > t_{\mathrm{buf}}{}^{i} + t_{\mathsf{ck}\to\mathrm{Q}}{}^{i} + t_{\mathrm{logic}}{}^{i} + t_{\mathrm{su}}{}^{i+1}$$

  Thus

$$t_{\mathrm{d}}{}^{i} > t_{\mathsf{ck}\to\mathrm{Q}}{}^{i} + t_{\mathrm{logic}}{}^{i} + t_{\mathrm{su}}{}^{i+1} + (t_{\mathrm{buf}}{}^{i} - t_{\mathrm{buf}}{}^{i+1}) - t_{\mathrm{R}\to\mathrm{A}'}{}^{i+1}$$

  For carefully laid out data paths, latch control buffers would have nearly identical delays, so $t_{\mathrm{buf}}{}^{i} \cong t_{\mathrm{buf}}{}^{i+1}$. Then the minimum bundling constraint would be

$$t_{\mathsf{ck}\to\mathrm{Q}}{}^{i} + t_{\mathrm{logic}}{}^{i} + t_{\mathrm{su}}{}^{i+1} - t_{\mathrm{R}\to\mathrm{A}'}{}^{i+1}.$$

- *Data hold time*
  The C-element delay ($t_{A'\to A'}$) must be long enough to meet the data hold time (t_{h}):

$$t_{\mathrm{A}'\to\mathrm{A}'}{}^{i} + t_{\mathrm{buf}}{}^{i} + t_{\mathsf{ck}\to\mathrm{Q}}{}^{i} + t_{\mathrm{logic}}{}^{i} > t_{\mathrm{buf}}{}^{i+1} + t_{\mathrm{h}}{}^{i+1}$$

  This inequality is trivially satisfied because $t_{\mathrm{buf}}{}^{i} \cong t_{\mathrm{buf}}{}^{i+1}$ and $t_{\mathsf{ck}\to\mathrm{Q}}{}^{i} > t_{\mathrm{h}}{}^{i+1}$ in general.

The *minimum cycle time* of this pipeline is the minimum time interval between successive *toggling* of a request signal:

$$t_{\mathrm{d}}{}^{\mathrm{min}} + t_{\mathrm{R}\to\mathrm{A}'} + t_{\mathrm{A}'\to\mathrm{A}'}$$

which simplifies to

$$t_{\mathsf{ck}\to\mathrm{Q}} + t_{\mathrm{logic}} + t_{\mathrm{su}} + t_{\mathrm{A}'\to\mathrm{A}'}$$

Note that $t_{\mathsf{ck}\to\mathrm{Q}} + t_{\mathrm{logic}} + t_{\mathrm{su}}$ is the minimum clock period of the equivalent synchronous pipeline circuit.

3.1 Double Edge-Triggered D-Flip-Flop Implementation

Our implementation of pseudo-static double edge-triggered D-FF is shown in figure 6. It is a staticized version of a true single-phase DETDFF described in [1]. The top half of the DETDFF is a *rising* edge-triggered DFF (RETDFF), and the bottom half a *falling* edge-triggered DFF (FETDFF). The outputs of these sections are *dotted* to form a DETDFF. When ϕ is high, the FETDFF section is cut off and Q' is driven by the RETDFF. Conversely, when ϕ is low, the RETDFF is cut off and Q' is driven by the FETDFF.

The RETDFF functions as follows. Input D must stabilize to 0 or 1 before ϕ rises, and the rising transition of ϕ latches the D value. If D stabilizes to 1 before ϕ rises, N_1 turns on and node x is pulled down, so the N-stack of the middle inverting stage is turned off. Because ϕ is low, node y is pulled up. When ϕ rises, the N-stack of the right inverting stage is turned on, which pulls down Q' and thus raises Q. If D stabilizes to 0 before ϕ rises, the P-stack of the left inverting stage is turned on, pulling up node x. When ϕ rises, P_3 turns off and the N-stack of the middle inverting stage turns on. This causes node y to be pulled down, which in turn turns on P_4 and turns off the N-stack of the right inverting stage. This causes Q' to rise and Q to fall. Once ϕ rises, switching D has no effect on the output.

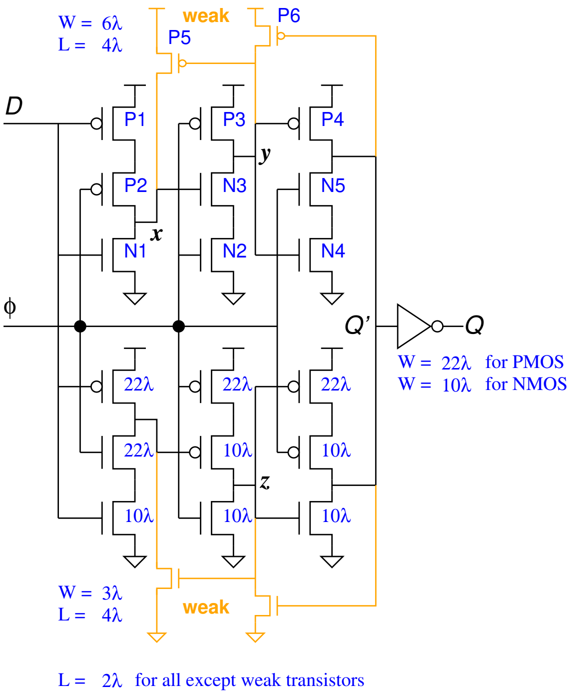


Figure 6: A true single-phase pseudo-static double edge-triggered D-FF.

In Afghahi and Yuan's original design, once ϕ rises, nodes x and y float, so it must rely on the capacitance on those nodes to hold the charge. However, DETDFFs used in asynchronous pipelines cannot depend on the capacitance to hold the logic level, because the next transition of ϕ may arrive at an arbitrary time. Our design uses two weak PMOS transistors to maintain the voltage level when ϕ is high. When Q' becomes low, y is pulled up by P_6, which keeps the N-stack of the right inverting stage turned on, keeping Q' low. On the other hand, to keep Q' high after it becomes high, y must remain low. Our circuit accomplishes this by turning on P_5 when y becomes low, which maintains the N-stack of the middle inverting stage turned on. The FETDFF functions similarly as ϕ switches from high to low.

Note that feeding back Q' to *both* RETDFF and FETDFF sections (which happens because the outputs are *dotted* to form the DETDFF) does not cause the *inactive* section to turn on inadvertently. For example, when ϕ is high, $Q' = 0$ has no effect. Neither does $Q' = 1$, because node z is already pulled down due to ϕ being high.

3.2 Simulation Results

We implemented 5 stages of our DET pipeline circuit and Day and Woods's semi-decoupled pipeline circuit in MOSIS 1.2μm CMOS process. We used consistent transistor sizing for both designs: W/L of the minimum size transistors for control circuits is $45\lambda/2\lambda$ for PMOS and $20\lambda/2\lambda$ for NMOS. In addition, for both designs, we adjusted the capacitive loading on the latch control signals to drive 32-bit latches/flip-flops. In Day and Woods's design, the capacitive loading of 32-bit latches corresponds to 1pF. Because the capacitive loading of the latch control signal in our DETDFF is 4 times as much as the n-type true single-phase latches used in Day and Woods's design (8 vs 2 transistors), we designed our latch control buffer to drive 4pF in each stage.

We simulated both designs with the worst-case process parameter using a 4.6V power supply at 100°C. Mentor Graphics Accusim analog simulator was used for simulation.

The simulation traces are shown in figure 7. A breakdown of the cycle times for both designs is given in table 1. [1] The table indicates that the maximum throughput simulated was 3.9ns cycle time for our DETDFF-based design and 6.7ns cycle time for Day and Woods's (see table 1). Thus our design is 70% faster than Day and Woods's. The worst-case $t_{\mathsf{ck}\rightarrow \mathrm{Q}}$ was 1.7ns for the double edge-triggered D-FF; $t_{\mathrm{R}\rightarrow \mathrm{A}'}$ and $t_{\mathrm{A}'\rightarrow \mathrm{A}'}$ were 1.0ns and 1.3ns respectively. The bundling delay needed was $t_{\mathrm{d}} = 1.6$ns.

There are two key factors that contributed to the difference in cycle times. The first obvious factor is that the DETDFF is capable of latching data on both edges of its enable signal. The second factor is that the acknowledgment of the receipt of data is made very early — as soon as the request from the left neighbor is detected (and, of course, the last request to the right neighbor has been acknowledged). We can afford to do that because the previous pipe stage can generate the next data no faster than the delay it takes for the acknowledgment signal to propagate through the latch enable buffer.

If the latch enable buffer delay is removed from Day and Woods's semi-decoupled 4-phase micropipeline design, their design can also shave off somewhat less[2] than 1.1ns from the minimum cycle time as illustrated in figure 8. Even so, our design would still be 50% faster than Day and Woods's design.

	MOSIS 1.2μm	AMULET-2
$R_{\mathrm{in}} \uparrow \rightarrow R_{\mathrm{out}} \uparrow$	2.4ns	4.0ns
$R_{\mathrm{out}} \uparrow \rightarrow \overline{A_{\mathrm{out}}} \downarrow$	1.5ns	3.2ns
$\overline{A_{\mathrm{out}}} \downarrow \rightarrow \overline{A_{\mathrm{in}}} \uparrow$	2.3ns	3.8ns
$\overline{A_{\mathrm{in}}} \uparrow \rightarrow R_{\mathrm{in}} \uparrow$	0.5ns	1.7ns
Total	6.7ns	12.7ns

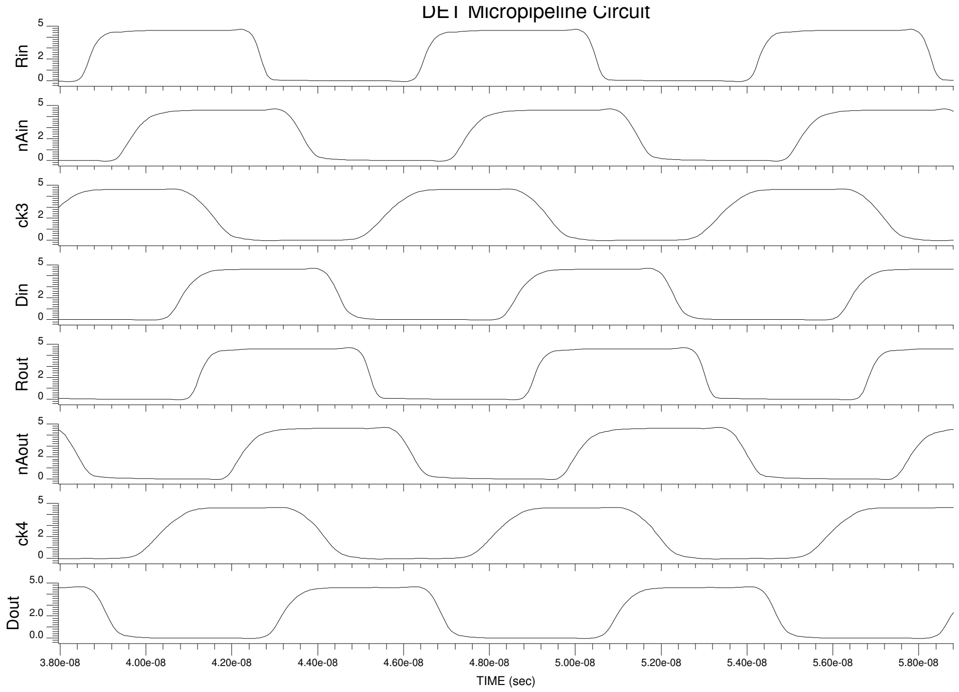
Table 1: Cycle time delay breakdown of Day and Woods's 4-phase micropipeline with the semi-decoupled latch control.

3.3 Area and power comparison

The main disadvantages of the pseudo-static DETDFFs are larger area and heavily loaded latch control lines. Each DETDFF is approximately twice the size of a standard edge-triggered flip-flop and the capacitance on the DETDFF control lines is approximately four times greater than standard level-sensitive latch control lines, yielding four-times more energy consumption in the control lines per transition. The overall effect on power consumption, however, is not clear because the DETDFFs are switched half as often as level-sensitive designs and they are naturally blocking [4], preventing glitches from propagating through multiple datapath stages.

[1] We note that the fraction of cycle time consumed by every component in the simulated implementation of Day and Woods's circuit is virtually the same as in their published delays, supporting the claim that the implementation of their design is reasonable.

[2] Still need to add a minimum size inverter to have the correct signal polarity.



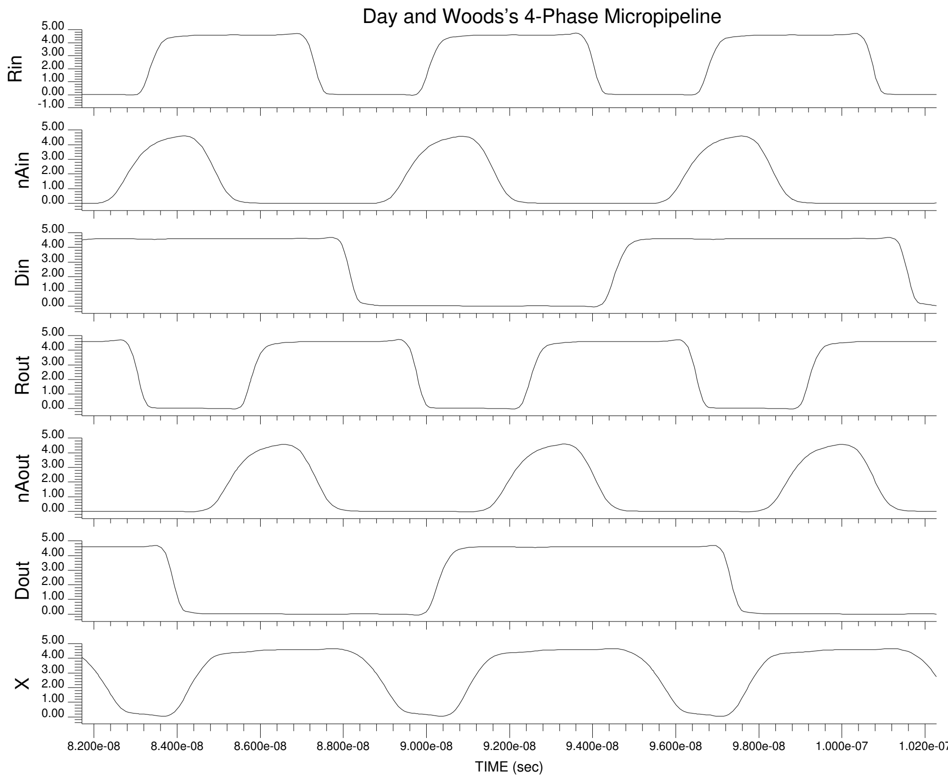


Figure 7: 2-phase DET (top) and Day and Woods's semi-decoupled 4-phase (bottom) micropipeline circuit simulations.

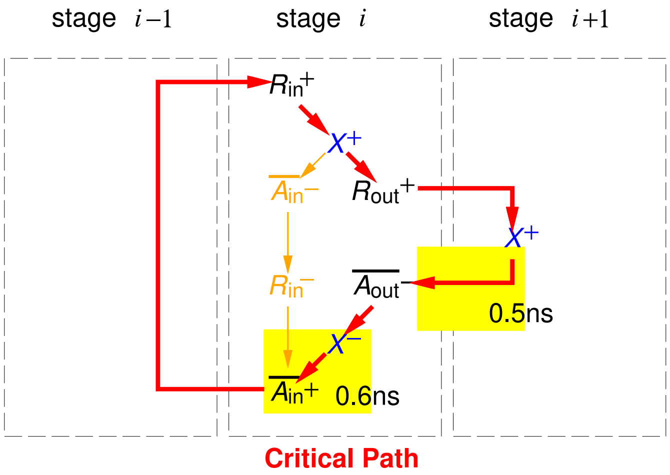


Figure 8: Semi-decoupled pipeline circuit: latch enable buffer in the critical path (see figure 4 for the circuit diagram).

4 Two-Phase Micropipeline Control Circuits

In this section, we present two commonly used micropipeline event-control structures: **select** and **toggle** elements. These elements were specified in extended burst-mode and synthesized using the 3D synthesis system. Detailed simulations demonstrate that our implementations are up to 50% faster than traditional designs.

4.1 Select

Figures 9 and 10 show the extended burst-mode **select** circuit and a conventional one used in AMULET-1.

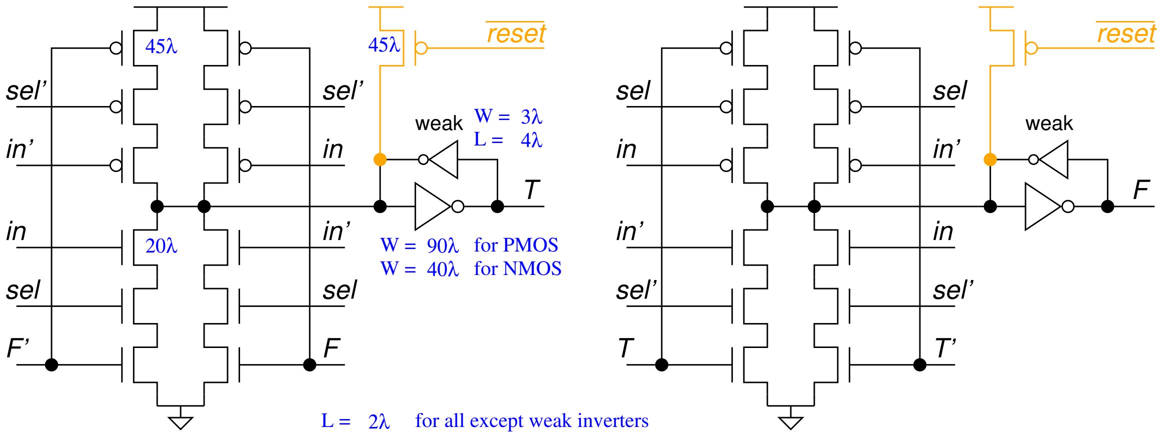


Figure 9: Extended burst-mode **select** circuit.

We implemented both circuits in MOSIS 1.2μm CMOS process and simulated them under the same condition as before. We simulated both circuits simultaneously using the same test inputs. T

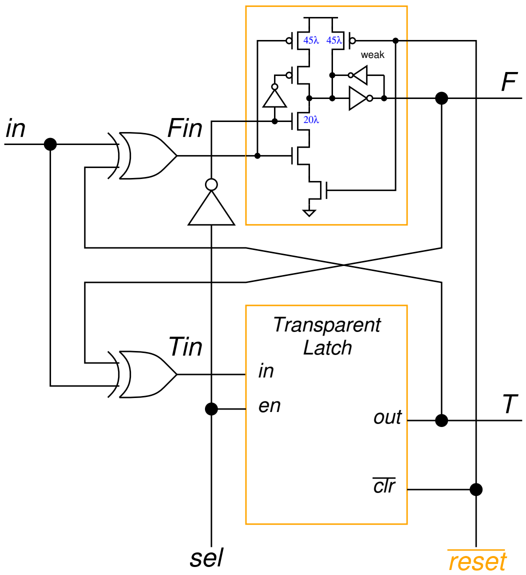


Figure 10: select circuit used in AMULET-1.

and F traces are from the outputs of our select circuit, and T.p and F.p are from the outputs of the AMULET-1 select circuit. Tin and Fin are outputs of xor gates (see figure 10). The AMULET-1 select circuit always propagates its input transition to both Tin and Fin; however, only one event is selected and propagated to the output of the associated transparent latch. For example, if sel is high, then the event on Tin propagates to T. When T changes, this change is fed back to the input of the xor whose output is Fin and cancels the event not selected. The environment of the circuit must wait for the feedback delay through xor gates before changing the sel input.

The *cycle time* of our select circuit corresponds to the delay from toggling of in to a change in T′ or F′. The cycle time of the AMULET-1 select circuit is the delay from toggling of in to the *cancellation* of Tin or Fin — not the first change in Tin or Fin after an input change but the change in Tin or Fin caused by the fed-back F or T transition. As shown in table 2, our select circuit has the latency of 1.8ns and the cycle time of 2.2ns. The circuit from AMULET-1 has the latency of 2.5ns and the cycle time of 3.3ns. Thus our circuit is faster than the AMULET-1 circuit by 40–50%.

Ignoring the small weak inverters, our select element design requires 21 p-transistors (W/L = $45\lambda/2\lambda$) and 19 (W/L = $20\lambda/2\lambda$) n-transistors, while the AMULET-1 design uses 19 p-transitors and 19 n-transitors. Thus our design is approximately 7

The number of transistors switched per select operation is XX

4.2 Toggle

The toggle is a circuit that routes input events to two outputs alternately. For example, if an input event causes the output dot to toggle, then the next event will cause the output blank to toggle, and the event after the next one will cause the output dot to toggle again.

We implemented the burst-mode toggle circuit (see figure 12) and the one used in AMULET-1 in MOSIS 1.2μm CMOS process and simulated them under the same condition as before. Again, we simulated both circuits simultaneously using the same test inputs. Simulation traces are shown in figure 13. dot and blank are from the outputs of our toggle circuit, dot.p and blank.p are from

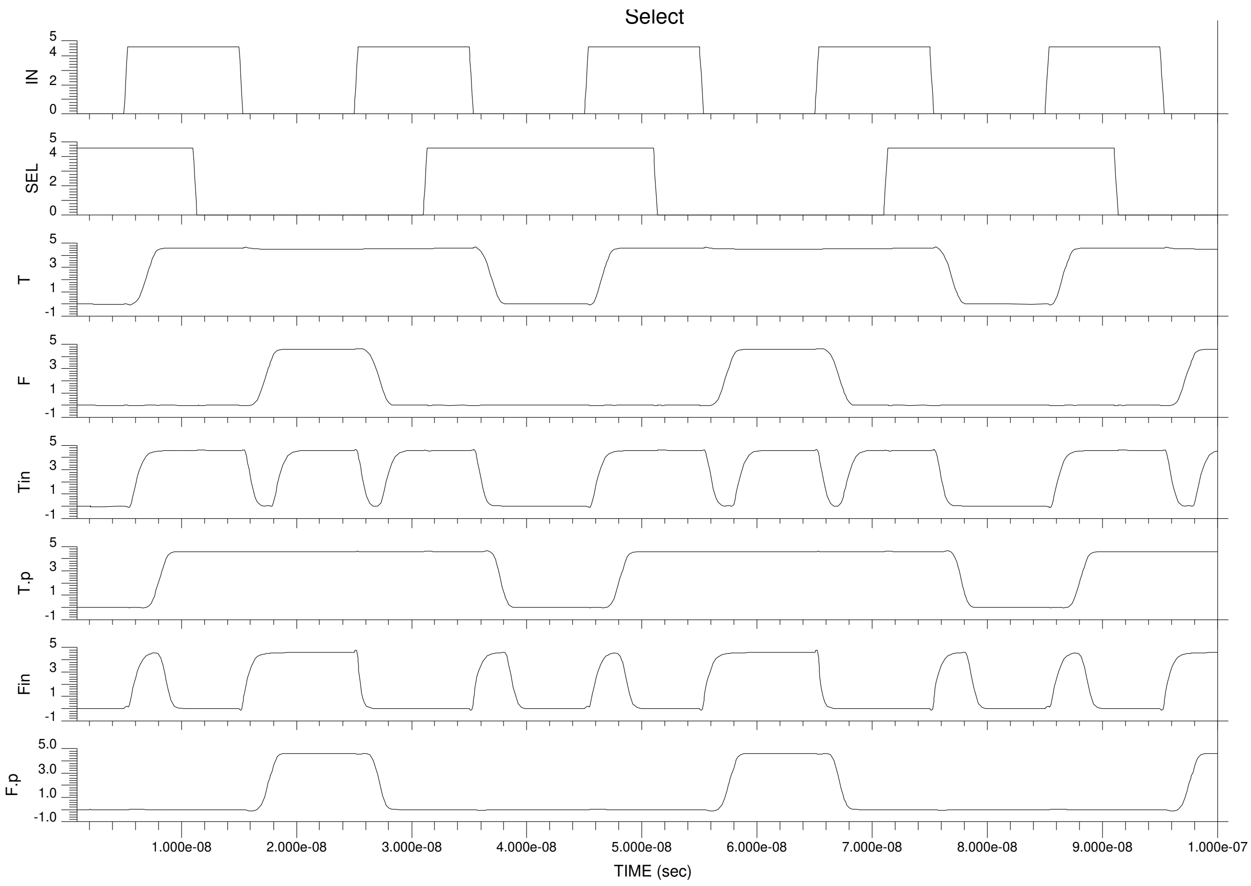


Figure 11: Simulation trace of select circuits

the outputs of the AMULET-1 toggle circuit.

As shown in table 3, in our circuit the delay from an event on in to a change in one of the outputs is 1.3ns, and in the AMULET-1 circuit it is 1.9ns. Thus our circuit is 50% faster than the AMULET-1 circuit. Note that both circuits are designed to be speed-independent. Our circuit was initially designed to be a fundamental-mode circuit but turned into a speed-independent one by feeding back $\overline{\text{blank}}$ from the internal node of the blank logic, instead of inverting the blank output and feeding the inverted signal back. This was done to make the comparison simpler.

PUT Area and power comparisions HERE.

Ignoring the small weak inverters, our toggle element design requires 11 p-transistors (W/L = $45\lambda/2\lambda$) and 7 (W/L = $20\lambda/2\lambda$) n-transistors, while the AMULET-1 design uses 15 p-transitors and 17 n-transitors. Thus, our design is approximately 38

5 Conclusions

This paper presents three improved two-phase micropipelines building blocks that facilitate significantly faster micropipeline designs compared with the current state-of-the-art two-phase and four-

		XBM		AMULET-1	
sel	transition	latency	cycle time	latency	cycle time
0	$\mathsf{in}^+ \rightarrow \mathsf{F}^+$	1.4ns	1.7ns	2.7ns	3.2ns
	$\mathsf{in}^- \rightarrow \mathsf{F}^-$	1.7ns	2.1ns	2.5ns	3.4ns
	$\mathsf{in}^+ \rightarrow \mathsf{F}^-$	2.0ns	2.4ns	2.0ns	2.7ns
	$\mathsf{in}^- \rightarrow \mathsf{F}^+$	2.1ns	2.4ns	2.4ns	3.3ns
1	$\mathsf{in}^+ \rightarrow \mathsf{T}^+$	1.4ns	1.7ns	2.9ns	3.5ns
	$\mathsf{in}^- \rightarrow \mathsf{T}^-$	1.7ns	2.1ns	2.5ns	3.5ns
	$\mathsf{in}^+ \rightarrow \mathsf{T}^-$	2.0ns	2.4ns	2.3ns	3.1ns
	$\mathsf{in}^- \rightarrow \mathsf{T}^+$	2.1ns	2.4ns	2.7ns	3.3ns
Average		1.8ns	2.2ns	2.5ns	3.3ns

Table 2: Comparison of select circuit performance.

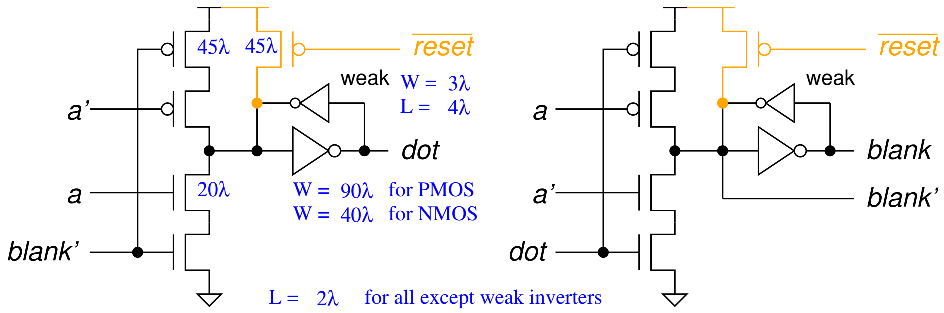


Figure 12: Burst-mode toggle circuit.

transition	XBM	AMULET-1
$\mathsf{in}^+ \rightarrow \mathsf{dot}^+$	1.1ns	1.8ns
$\mathsf{in}^- \rightarrow \mathsf{blank}^+$	1.4ns	1.9ns
$\mathsf{in}^+ \rightarrow \mathsf{dot}^-$	1.4ns	2.3ns
$\mathsf{in}^- \rightarrow \mathsf{blank}^-$	1.2ns	1.7ns
Average	1.3ns	1.9ns

Table 3: Comparison of toggle circuit performance.

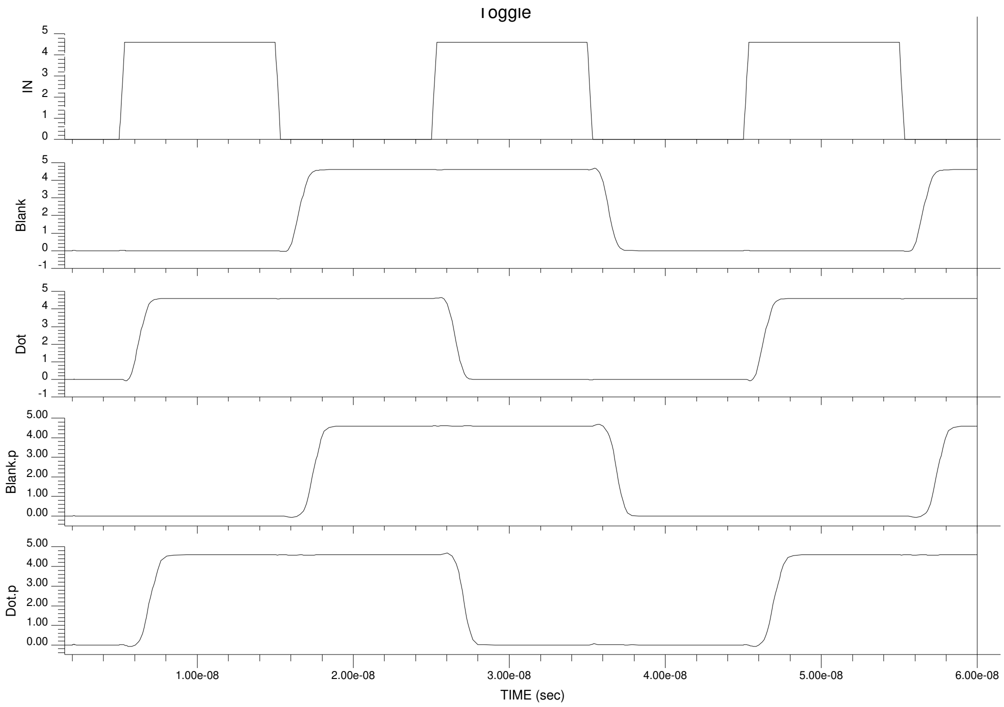


Figure 13: Simulation trace of **toggle** circuits

phase micropipelines. The first improved building block is a pseudo-static double edge-triggered D-flip-flops which updates data on both transitions of the latch control signal. As a result, no reset-phase is needed and efficient two-phase control structures can be used. The second and third are burst-mode select and toggle elements, which are significantly faster than previously hand-designed implementations. The source of the increased speed is the application of generalized C-elements which compactly implements the function of a collection of discrete gates. In spite of the fact that the synthesis method, 3D, is based on the fundamental-mode restriction, the synthesized circuits have looser environmental timing restrictions than the hand-designed circuits, demonstrating the versatility of the fundamental-mode assumption. The improved speed of these circuits also suggests that more complex two-phase control structures might benefit from being formally specified and automatically synthesized rather than decomposed by hand into basic event-control elements.

References

[1] M Afghahi and J. Yuan. Double edge-triggered d-flip-flops for high-speed cmos circuits. *IEEE Journal of Solid-State Circuits*, 26(8), August 1991.

[2] Paul Day and J. Viv Woods. Investigation into micropipeline latch design styles. *IEEE Transactions on VLSI Systems*, 3(2), June 1995.

[3] S. Furber. Computing without clocks: Micropipelining the ARM processor. In Graham Birtwistle and Al Davis, editors, *Asynchronous Digital Circuit Design*, Workshops in Computing, pages 211–262. Springer-Verlag, 1995.

[4] N. C. Paver. *The Design and Implementation of an Asynchronous Microprocessor*. PhD thesis, Department of Computer Science, University of Manchester, June 1994.

[5] Ivan E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, June 1989.

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

[7] K. Y. Yun. *Synthesis of Asynchronous Controllers for Heterogeneous Systems*. PhD thesis, Stanford University, August 1994.

[8] K. Y. Yun and D. L. Dill. Automatic synthesis of 3D asynchronous state machines. In *Proc. International Conf. Computer-Aided Design (ICCAD)*, pages 576–580. IEEE Computer Society Press, November 1992.

[9] Kenneth Y. Yun. Automatic synthesis of extended burst-mode circuits using generalized C-elements, 1996. Submitted to a conference.