DELAY BALANCING OF WAVE PIPELINED MULTIPLIER COUNTER TREES USING PASS TRANSISTOR MULTIPLEXERS

Hidechika Kishigami, Kevin J. Nowka, and Michael J. Flynn

Technical Report CSL-TR-96-692

January 1996

DELAY BALANCING OF WAVE PIPELINED MULTIPLIER COUNTER TREES USING PASS TRANSISTOR MULTIPLEXERS

by

Hidechika Kishigami, Kevin J. Nowka, and Michael J. Flynn

Technical Report CSL-TR-96-692

January 1996

Computer Systems Laboratory

Departments of Electrical Engineering and Computer Science

Stanford University

Stanford, California 94305-4055

Abstract

Wave pipelining is an attractive technique used in high-speed digital circuits to speed-up pipeline clock-rate by eliminating the synchronizing elements between pipeline stages.

Wave-pipelining has been successfully applied to the design of CMOS multipliers [1, 2, 11, 10, 12] which have demonstrated speed-ups of clock-rate 4 to 7 times over their non-pipelined design. In order to achieve high clock-rate by using wave-pipelining techniques, it is necessary to equalize (balance) all signal path delay of the circuit. In an earlier study a multiplier [1] was designed by using only 2-inputs NAND gates and inverters as primitives in order to reduce delay variations of the circuit.

Alternatively, there are several reports that use pass-transistor logic as primitives for multipliers to achieved very low latency [5] [6]. Pass-transistor logic seems attractive for reducing circuit delay variations.

In this report we describe a design of wave-pipelined counter tree, which is a central part of parallel multiplier, and detail a method to balance the delay of (4,2) counter using pass-transistor multiplexers (PTMs) as primitives to achieve both higher clock-rate and smaller latency. Simulations of the wave-pipelined counter tree demonstrated 0.8ns clock-rate and 2.33ns latency through the use of pass-transistor multiplexers (PTMs) for a 0.8μm CMOS process. This data suggests using pass-transistor multiplexers as primitives for wave-pipelined circuits is useful to achieve both higher clock-rate and lower latency.

Key Words and Phrases: Wave-pipeline, Pass-transistor multiplexer, Multiplier

Contents

List of Figures

List of Tables

1 Introduction

Wave-pipelining is a system design technique which allows digital synchronous system to be clocked at rates higher than can be achieved with conventional pipelining techniques.

Wave-pipelining has been successfully applied to the VLSI design of numerous arithmetic circuits including: a bipolar population counter[8], a CMOS adder[9], CMOS multipliers[1, 11, 2, 10, 12], and a VLSI vector unit [3]. These designs demonstrated speed-ups in clock-rate 2 to 7 times over the equivalent non-pipelined designs.

Figure 1 shows circuit models of non-pipelined (a), conventional pipelined (b), and wave-pipelined (c) system. In a wave-pipelined system, new data are applied to the inputs of the system before the previous outputs are available. In this way, coherent sets of data, *waves*, propagate through different stages of the system at the same time. The minimum clock period at which input data can be applied to the system is limited by the difference between maximum and minimum propagation delays through the system plus the clocking overhead. So to achieve maximum clock-rate, the propagation delays across difference paths of the system have to be equalized, or *balanced*. In the design of the parallel multiplier reported by [1], only 2-inputs NAND gates and inverters were used as primitives in order to minimize delay variations.

Alternatively, there are several reports that use pass-transistor logic as primitives for multipliers which have achieved very low latency [5] [6]. Ohkubo et. al. reported high performance (low latency) 54 x 54-b multiplier using pass-transistor multiplexers [5]. They used new (4,2) counters and a carry lookahead adder, both featuring pass-transistor multiplexers, which provide a speed advantage over conventional CMOS circuits because the number of critical-path gate stages is minimized due to the high logic functionality of pass-transistor multiplexers. Other researchers have developed wave pipelined designs using pass-transistor logic families: Using transmission gate logic, Zhang and Sridhar [12] developed an adder which had a delay variation of approximately 14.5%. With a CPL implementation of gates, Ghosh and Nandy [10] implemented a multipler.

Thus, it seems that wave-pipeline techniques promote high clock-rates, while pass-transistor multiplexers promote low latency of a multiplier.

In this report, we describes a design and simulation results of wave-pipelined counter tree, a main part of parallel multiplier, using pass-transistor multiplexers to achieve both higher clock-rate and lower latency.

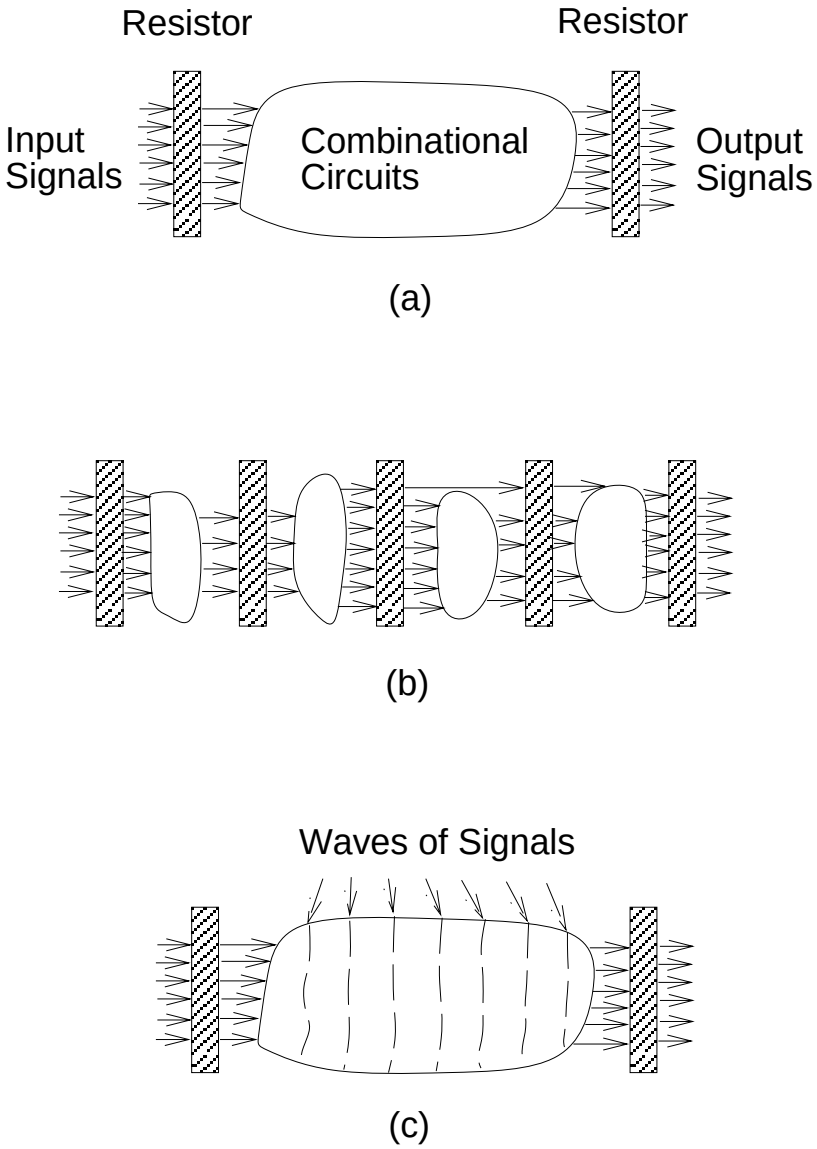

Figure 1: Circuit Models. (a) Non-pipelined System. (b) Conventional-pipelined System. (c) Wave-pipelined System.

2 16-bit Multiplier Design

2.1 Architecture for Multiplier

In order to evaluate the effectiveness of pass-transistor multiplexers (PTMs) as primitives for wave-pipelined system, we selected a counter tree which is a main part of parallel multiplier. Figure 2 is a block diagram of 16 x 16-bit integer parallel multiplier, which consists of partial products generator, counter tree, and 32-bit adder. The partial products generator generates a set of partial-products which are summed in the counter tree and the 32-bit adder. In a 16-bit integer parallel non-encoded multiplier, sixteen partial-products are generated by the partial products generator. The counter tree sums the partial-products and reduce the number of series addition. Figure 3 shows a block diagram of the counter tree that uses row of (4,2) counters which reduces four partial-products to two in one stage. The three stages of the rows of (4,2) counters reduces 16 partial-products to two, which are input to final the 32-bit adder.

The details of 16-bit integer parallel multiplier is described in [1].

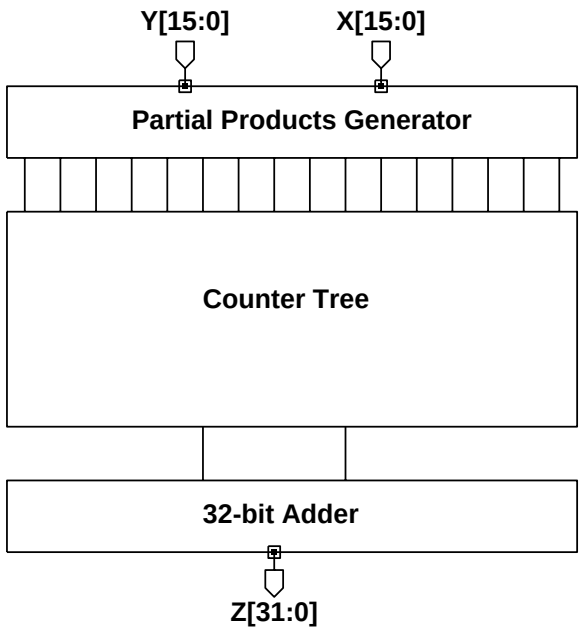

Figure 2: 16-bit multiplier block diagram.

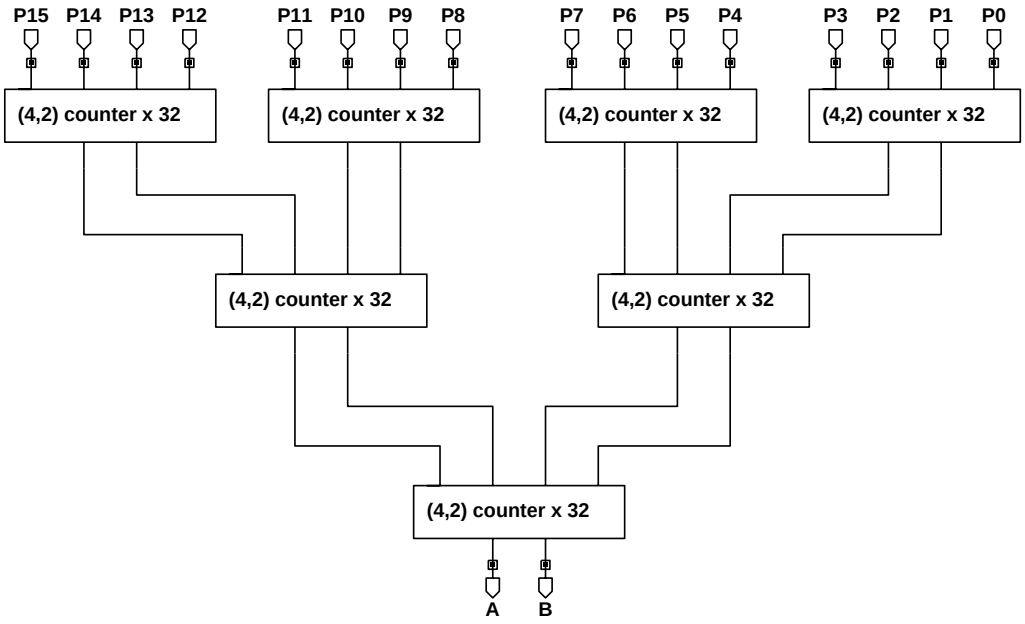

Figure 3: Counter tree block diagram.

2.2 Primitive Circuit

In the 16-bit integer parallel multiplier designed in [1], the implementation was constrained by the use of 2-input NAND gates and inverters exclusively for every logic function in order to minimize the delay variations. On the other hand, the use of pass-transistor multiplexers was proposed by [5] in order to design high performance (low latency) multiplier. The use of pass-transistor multiplexers as primitives provides a speed advantage over conventional CMOS circuits because the number of critical-path gate stages is minimized due to the high logic functionality of pass-transistor multiplexers. Figure 4 shows a (4,2) counter using pass-transistor multiplexers, which is similar to the circuit proposed by [5]. The only difference is that our multiplexer has output inverters to ease the balancing of the delay of the (4,2) counter.

3 Design of Wave-Pipelined Counter Tree

3.1 Pass-Transistor Multiplexer Delay Adjustment

To optimize the throughput of a wave pipelined design, the variation in the delay through the combinational logic along all paths must be minimized. To perform this optimization, gross adjustments to delay imbalance are made through the insertion of delay gates into the fast paths through the logic [8]. For variations less than a delay gate, transistor sizing methods are effective [4]. Along the fast paths through the logic, the transistor geometries are modified to decrease the speed of the gates; transistor lengths are increased and/or transistor widths are decreased to slow the propagation through the gate.

Several transistor sizing methods of fine-delay tuning are possible with pass-transistor multiplexers: (a) adjusting the width of the transistors in the output inverter, (b) adjusting the width of the pass-transistors in the multiplexers, (c) adjusting the length of the pass-transistors in the multiplexers, and (d) adjusting the widths and lengths of the pass-transistors concurrently so as to minimize the change in load on all inputs of the pass-transistor multiplexer. The first three options alter the effective capacitance of the input nodes of the pass-transistor multiplexer. This change makes delay balancing more difficult, since any delay adjustment to this gate affects the delay of predecessor gates. The final method allows relatively isolated delay adjustment, which eases the balancing task. This method is closely related to one used in [4].

The base transistor geometries for the delay adjustment methods detailed in Section 3.1 are given in Table 1.

The minimum delays through a pass-transistor multiplexer are 217ps for a single pass-transistor multiplexer load and 446ps for a four PTM load. The maximum delays are 441ps for a single PTM load and 864ps for a four PTM load. The ratios of maximum to minimum

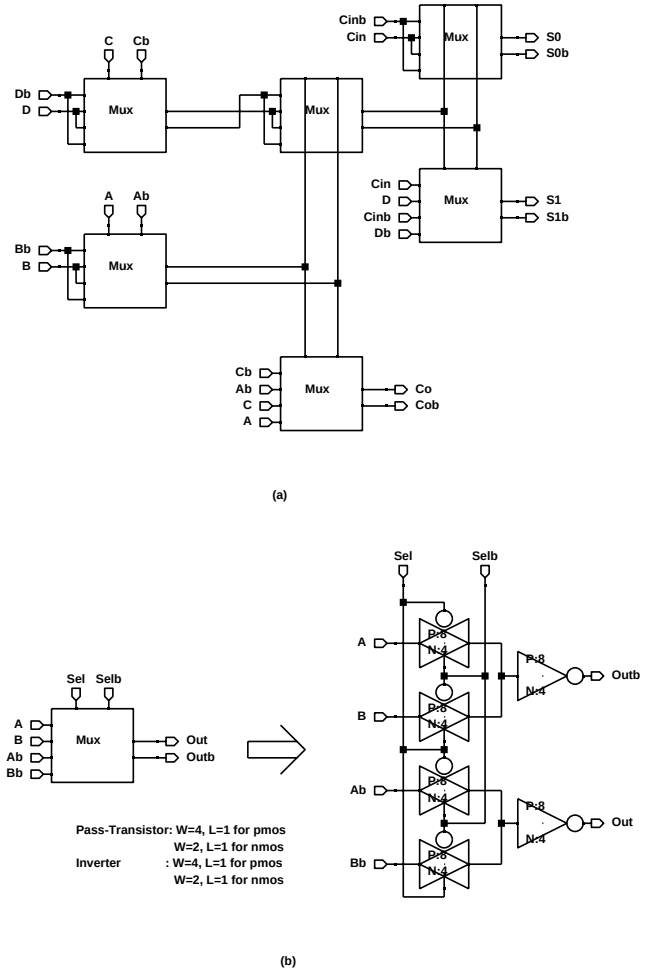

Figure 4: A (4,2) counter using pass-transistor multiplexers. (a) A (4,2) counter block diagram. (b) A multiplexer circuit.

Transistor	Width (micron)	Length (micron)
Pass Gate NMOS	8	1
Pass Gate PMOS	8	1
Buffer NMOS	8	1
Buffer PMOS	16	1

Table 1: Delay Adjustment Base Transistor Geometries

delay are thus 2.03 and 1.94, respectively. These wide variation in delays of individual gates must be managed when they are used in the design of wave pipelined circuits.

Effective fine balancing techniques should allow adjustment of PTM delay over a broad range with a fine degree of control while minimizing any additional induced data-dependent delay variation.

Figures 5 to 8 demonstrate the effects of the four transistor sizing techniques described above. They present HSPICE simulated delay of the fastest and slowest paths through the PTM as the transistors are sized. In each case, the fanout of the PTM output was four PTMs.

Adjusting the width of the transistors in the output inverter, as detailed in Figure 5, allows a broad range of adjustment of the delay of the critical path through the PTM. It does not, however allow sufficient adjustment of the fast-path delay. Thus, as this technique is used significant data dependent delay variation is introduced.

Adjusting the pass-transistor widths, as shown in Figure 6, does not significantly influence the delay of the PTM. It is, therefore, of little practical use in the optimization of wave pipelined PTM circuits.

Adjusting the length of the pass-transistors in the PTM, as shown in Figure 7, provides sufficient range of delay tuning with a reasonable degree of control. It results in a moderate increase in data-dependent delay variation.

Adjusting the length and width of the pass-transistors in the PTM concurrently so as to maintain constant capacitance of the pass-transistor control inputs allows sufficient tuning range and a reasonable degree of control with a moderate increase in data-dependent delay variation. Simulated results of this method are given in Figure 8. Adjustment of both width and length of transistors in the PTMs was used in this design.

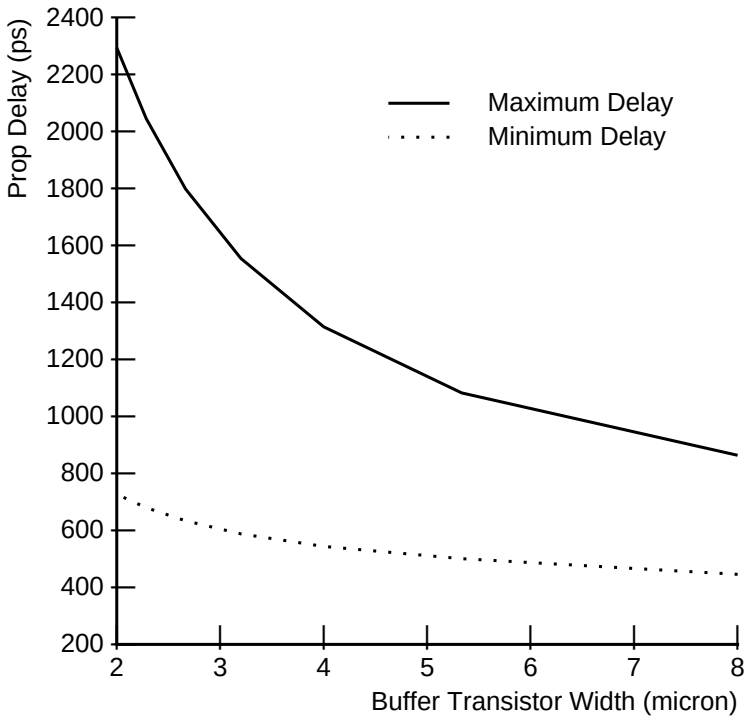

Figure 5: PTM Buffer Sizing Fine Tuning Results.

3.2 Balancing the Delay of the (4,2) Counter

In this design, the counter or compressor circuits used to form the multiplier were balanced individually. The delay of the (4,2) Counter is balanced manually by inserting delay elements and adjusting transistor sizes. This delay adjustment has been performed through extensive circuit level simulation using a SPICE-like simulator CAzM available from MCNC [7] and the chosen technology is CMOS26b, a 0.8μm triple-level-metal process developed at Hewlett-Packard. Delay data were obtained at 27C, 5.0V supply voltage, and 0.1ns input transition time signals were asserted to input inverters, which connect to each of the (4,2) counter inputs A, B, C, and D.

Figure 9 shows simulation results of (4,2) counter before and after delay adjustment and Table 2 summarizes the results.

The design process to balance delay of the (4,2) counter consists of the following steps:

1. No-delay adjustment

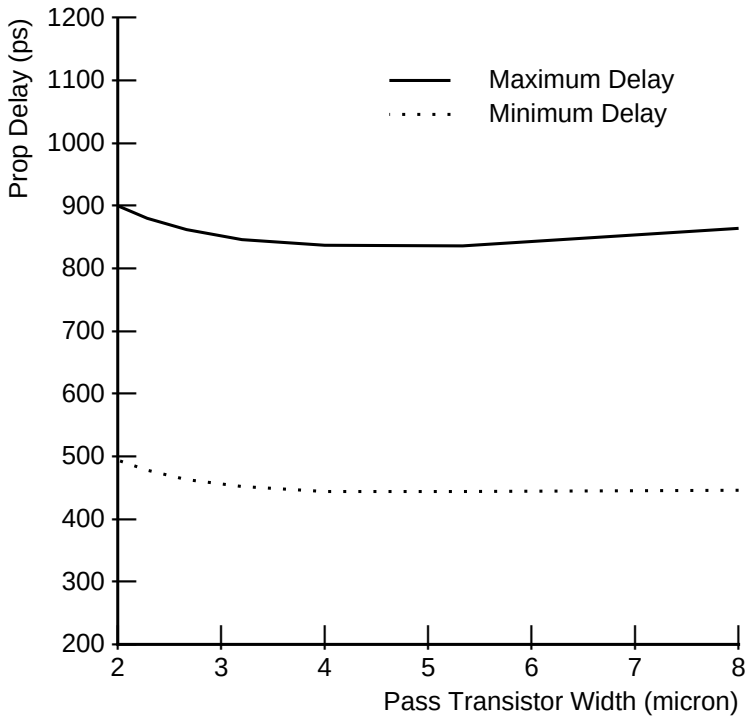

Figure 6: PTM Pass-Transistor Width Tuning Results.

The top figure of waves in Figure 9 shows simulation results of (4,2) counter without any delay adjustment. The simulation was performed with the 32 input vectors that change the (4,2) counter inputs (A, B, C, and D) from 0000 to **** and from **** to 0000 (* indicates either a 0 or 1). The outputs of all simulation results are superimposed in Figure 9. The (4,2) counter without any delay adjustment has a delay time varying from 0.23ns to 1.06ns, which means the delay variation is 0.83ns.

2. Insert delay elements

   The middle figure of waves in Figure 9 shows simulation results of (4,2) counter after inserting delay elements to equalize the number of gate-level stages. Because the longest path delay from input to output of the (4,2) counter is three pairs of a pass-transistor and an inverter, we inserted pair of a pass-transistor and an inverter as a delay element (See Figure 10(b)) to equalize the number of gate-level stages from all inputs to the outputs of the (4,2) counter as three. The (4,2) counter including delay elements has a delay time varying from 0.62ns to 1.02ns; thus the delay variation is 0.40ns.

3. Adjust transistor size (W and L)

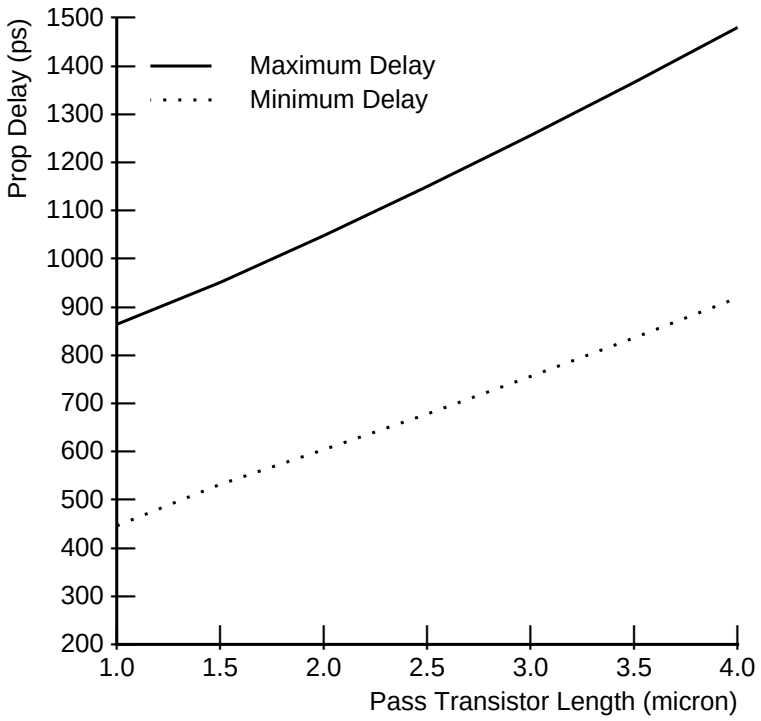

Figure 7: PTM Pass-Transistor Width Tuning Results.

Transistor size of all pass-transistors and inverters are W=2μm L=1μm for NMOS transistors and W=4μm L=1μm for PMOS transistor in the (4,2) counters described above. In order to decrease the delay variations, we adjusted the transistor sizes. As a first step, we changed the transistor width (W) of all inverters double (W=4μm L=1μm for NMOS and W=8μm L=1μm for PMOS) in order to equalize RC loads of all inputs (A, B, C, and D) and internal multiplexers of the (4,2) counter. Next we adjusted the transistor size (both width (W) and length (L)) of pass-transistor used in the delay elements inserted above. After varying the transistor sizes several times, we achieved a delay varying from 0.69ns to 0.85ns, which means delay variation is 0.16ns. The bottom figure of waves in Figure 9 shows the simulation results, and the (4,2) counter after adjusting transistor sizes is shown in Figure 10.

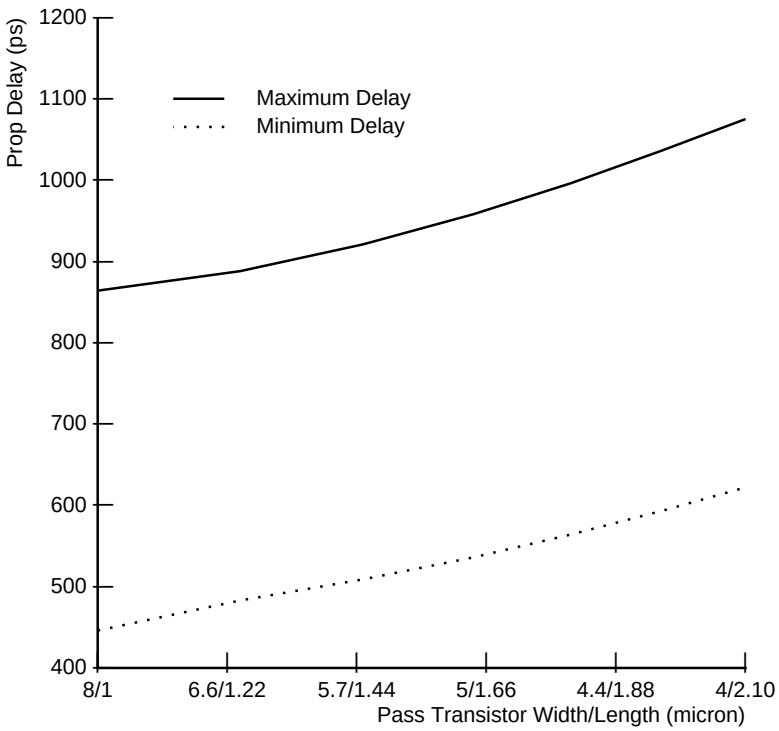

Figure 8: PTM W/L Fine Tuning Results.

3.3 Comparison of Balanced (4,2) Counter using Pass-Transistor Multiplexer and Static CMOS

Figure 11 shows simulation results of balanced delay (4,2) counter using pass-transistor multiplexers and using static CMOS (only 2-input NAND gates and inverters) reported by [1] to compare their delay and delay variations. Note that the simulations were exhaustive; they were performed by using the 256 possible different input patterns (from **** to ****) in order to conform worst case delay and delay variations. The average delay and the delay variations of the balanced (4,2) counter using pass-transistor multiplexers are 0.79ns and 0.21ns respectively, which are about 1/2 smaller than the equivalent counter implemented in two-input static CMOS. Table 3 summarizes the comparison.

Thus, using pass-transistor multiplexers as primitives for balanced (4,2) counters is effective in achieving both lower latency and higher clock-rate wave-pipelined circuits.

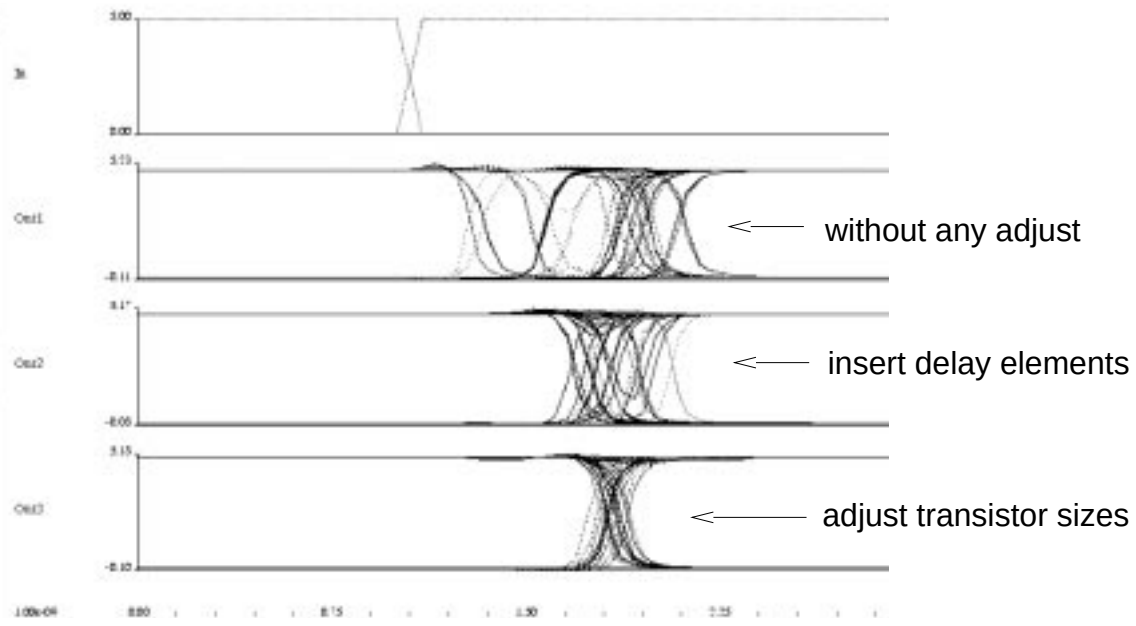

Figure 9: Simulation results of (4,2) counter before/after balancing delay.

	Circuit A	**Circuit B**	**Circuit C**
Delay *	0.23 - 1.06ns	0.62 - 1.02ns	0.69 - 0.85 ns
Delay Variation	0.83ns	0.40ns	0.16ns

Circuit A: no-delay adjustment
Circuit B: insert delay element
Circuit C: adjust transistor sizes (W and L)

Note: Delay shows a delay time of a input inverter + (4,2) counter.

Table 2: Delay variations of (4,2) counter before/after balancing delay.

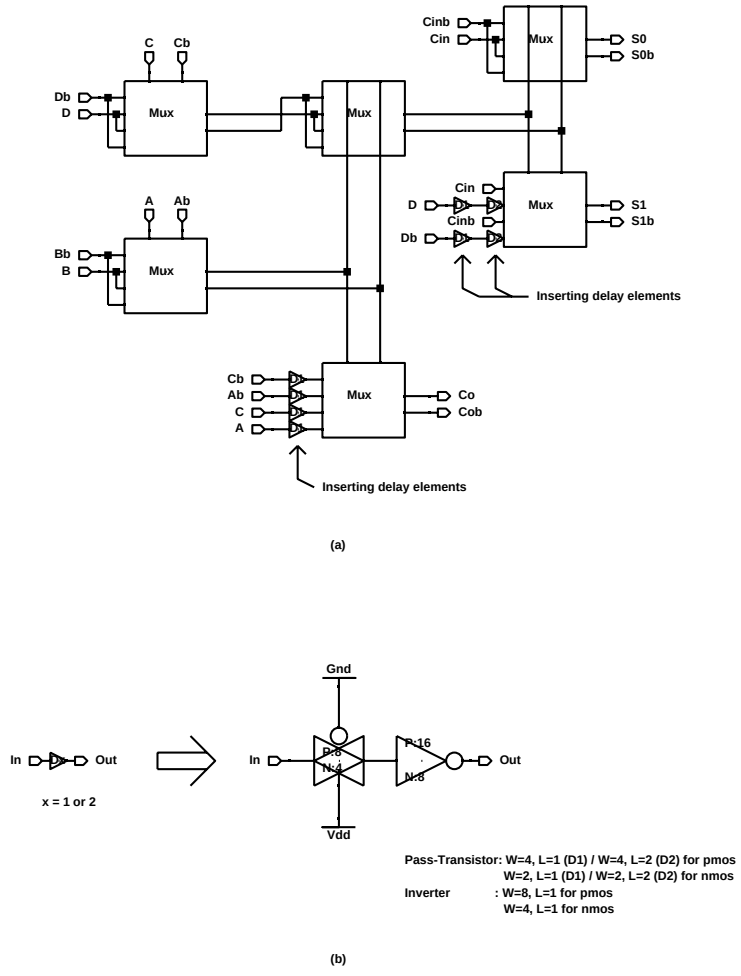

Figure 10: A (4,2) counter after balancing delay. (a) A (4,2) counter block diagram. (b) A delay element.

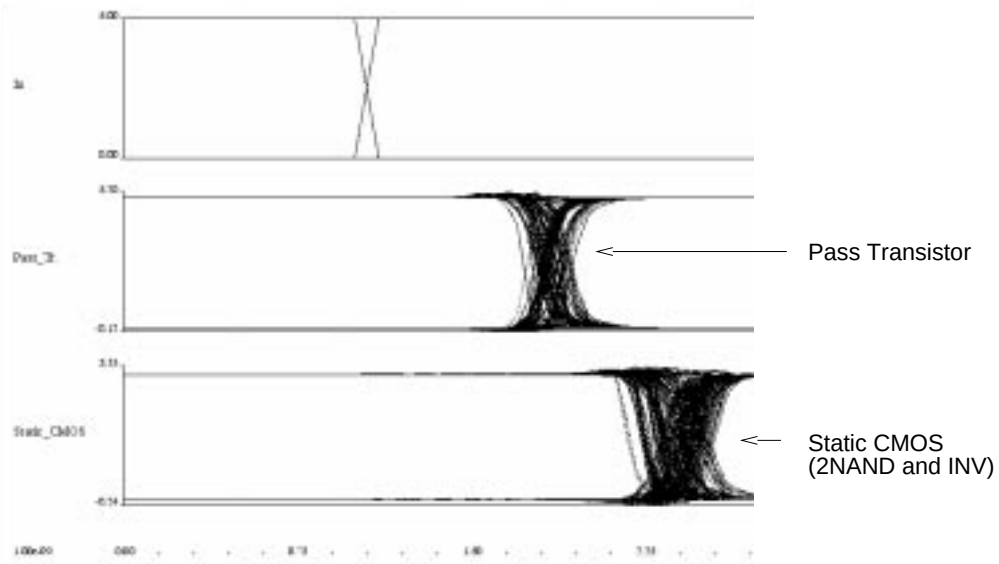

Figure 11: Simulation results of a (4,2) counter using pass-transistor multiplexers and static CMOS (2NAND and INV).

	Pass-Transistor Multiplexer	**Static CMOS**
Number of Transistors (Before Balanced) (After Balanced)	 72 104	 88 102
Delay Average Delay	0.68 - 0.89 ns 0.79 ns (0.59)	1.12 - 1.51ns 1.32 ns (1.0)
Delay Variation	0.21ns (0.54)	0.39ns (1.0)

Table 3: Comparison of balanced (4,2) counter using pass-transistor multiplexers and static CMOS (2NAND and INV).

3.4 Simulation Results of Wave-pipelined Counter Tree

By using balanced (4,2) counter described above, we designed wave-pipelined counter tree, which consists of three stages of 32-bit (4,2) counters shown in Figure 3. Figure 12 shows the simulation results of the counter tree. The simulation has performed at 27C, 5.0V supply voltage, and 0.1ns input transition time signals were asserted to the inputs of the counter tree. The input signals were changed their value randomly at a clock-rate of 1.0ns, 0.8ns, and 0.7ns. From these results, the counter tree can be operated up to 0.8ns clock-rate and the delay from inputs to the outputs is 2.33ns, which means that a speed-up of 2.9 (= 2.33ns / 0.8ns) is achieved by using wave-pipeline technique under typical conditions.

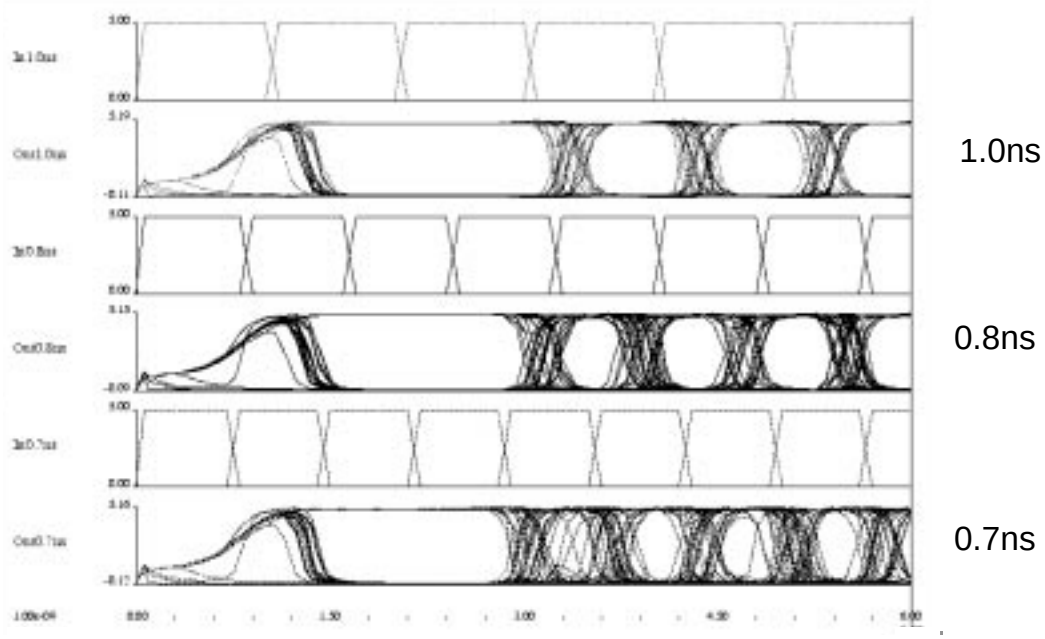

Figure 12: Simulation results of counter tree at typical condition.

Figure 13 also shows the simulation results of the counter tree at typical, fast, and slow conditions to consider the effects of temperature, supply voltage, and process variations on the wave-pipelined counter tree. Table 4 summarizes the results and Table 5 shows the difference of parameters at each conditions. From these results, the maximum clock-rate (cycle time), the delay from inputs to the outputs (latency), and the delay variations are almost proportional to each other and speed-ups of 2.7 - 2.9 can be achieved by using wave-pipeline technique under any conditions. These results suggest that the more advanced the

process technology used to design wave-pipelined circuits, the higher the clock-rate and the lower the latency achieved.

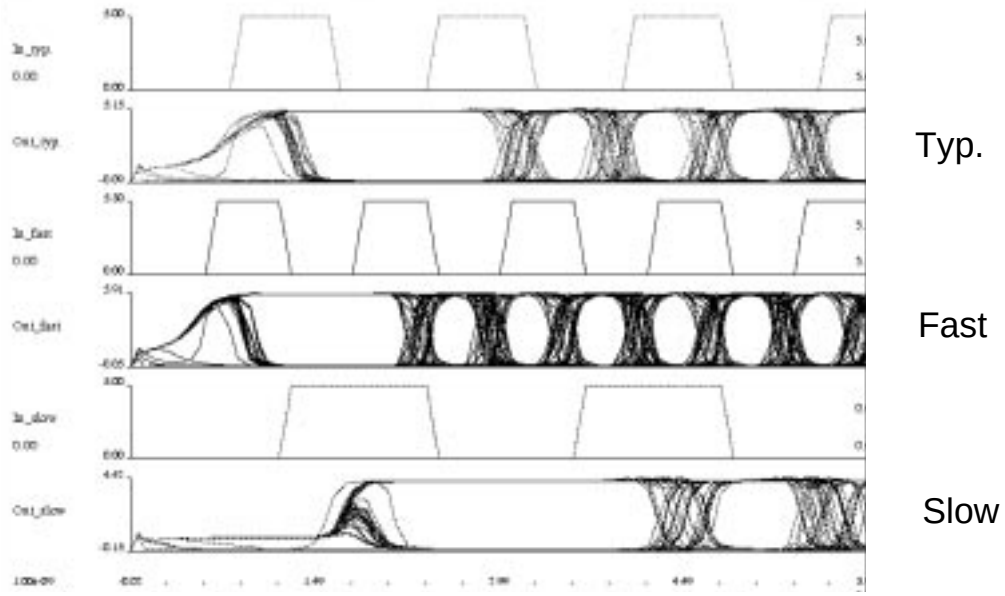

Figure 13: Simulation results of counter tree at typical, fast, and slow conditions.

4 Conclusions

In this report we described a design and simulation results of wave-pipelined counter tree for a parallel multiplier, using pass-transistor multiplexers as primitives to achieve both higher clock-rate and smaller latency. Our simulation results showed 0.8ns clock-rate and 2.33ns latency were achieved at typical condition for the HP 0.8μm CMOS26b process, 0.6ns(clock-rate) / 1.69ns(latency) at fast condition, and 1.2ns (clock-rate) and 3.27ns(latency) at slow conditions, which means speed-ups of 2.7 - 2.9 are achieved by using wave-pipelined technique.

Our simulation results suggest that using pass-transistor multiplexer as a primitive for wave-pipelined circuits is effective in achieving both higher clock-rate and lower latency, but a more complete study is necessary to clarify the advantages and disadvantages of

	Typical	**Fast**	**Slow**
Cycle Time	0.8ns(1.0)	0.6ns(0.6)	1.2ns(1.5)
Latency	2.33ns(1.0)	1.69ns(0.73)	3.27ns(1.40)
Delay Variation	0.38ns(1.0)	0.22ns(0.58)	0.60ns(1.58)
Speed-up(Latency/Cycle Time)	2.9	2.8	2.7

Table 4: Simulation results of counter tree on typical, fast, and slow conditions.

		Typical	**Fast**	**Slow**
Supply Voltage		5.0 V	5.8 V	4.2 V
Temperature		27 C	0 C	125 C
Process Parameters				
TOX	(NMOS)	1.78e-08	1.75e-08	1.85e-08
(Oxide Thickness)	(PMOS)	1.78e-08	1.69e-08	1.82e-08
VTO	(NMOS)	0.71	0.54	0.82
(Threshold Voltage)	(PMOS)	-0.90	-0.75	-1.50
LD	(NMOS)	1.44e-07	1.55e-07	0.80e-07
(Lateral Diffusion)	(PMOS)	1.03e-07	1.15e-07	0.95e-07
UO	(NMOS)	571.5	575.0	540.0
(Mobility)	(PMOS)	177.6	185.9	171.6

Table 5: Simulation parameters on typical, fast, and slow conditions.

using pass-transistor multiplexers for wave-pipelined circuits.

Some aspects of this future study are listed below.

1. *Whole multiplier design*: In this study, we designed and evaluated performance of a wave-pipelined counter tree, a part of parallel multiplier, but future study might be necessary to design whole multiplier and evaluate its performance including effect of wiring capacitance based on real layout information.

2. *Comparison of primitives*: In this study, we used pass-transistor multiplexers with output inverters as primitives, but other types of circuits as primitives should also be evaluated, and compared not only in speed, but in chip area and power.

5 Acknowledgements

Facilities used in the development of this work were provided by NASA under contract NAG2-842.

References

[1] F. Klass, M. J. Flynn, and A. J. van de Goor. "Fast Multiplication in VLSI using Wave Pipelining Techniques" *Journal of VLSI Signal Processing,* 7, 233-248 (1994)

[2] V. D. Nguyen, W. Liu, C. T. Gray, and R. K. Cavin. "A CMOS Signed Multiplier using Wave Pipelining" *IEEE 1993 Custom Integrated Circuits Conference,* 1993.

[3] K. Nowka and M. Flynn. System Design Using Wave-Pipelining: A CMOS VLSI Vector Unit. *Proceedings of the 1995 IEEE International Conference on Circuits and Systems*, pages 2301–2304, 1995

[4] K. Nowka. *High Performance CMOS VLSI System Design Using Wave Pipelining.* PhD thesis, Stanford University, Department of Electrical Engineering, 1995.

[5] Ohkubo et. al. "A 4.4ns CMOS 54 x 54-b Multiplier Using Pass-Transistor Multiplexer" *IEEE Journal of Solid-State Circuits,* Vol. 30, No 3 1995, pp. 251-257.

[6] Yano et. al. "A 3.8ns 16x16 Multiplier Using Complementary Pass Transistor Logic" *IEEE Journal of Solid-State Circuits,* Vol. 25, pp. 388-395, Apr. 1990.

[7] D. Rose, D. Erdman, and G. Nifong. "CAzM: Circuit analyzer with macromodeling user's guide," *Technical Report,* MCNC, June 1990.

[8] D. Wong. *Techniques for Designing High Performance Digital Circuits Using Wave Pipelining*. PhD thesis, Stanford University, Department of Electrical Engineering, 1991.

[9] D. Fan, C. Gray, W. Farlow, T. Hughes, W. Liu and R. Cavin. "A CMOS Parallel Adder Using Wave Pipelining" *MIT Advanced Research in VLSI and Parallel Systems,* March 1992, pp. 147-164.

[10] D. Ghosh and S. K. Nandy. " A 400 MHz Wave-Pipelined 8x8-bit Multiplier in CMOS Technology." *Proceedings of the International Conference on Computer Design*, pages 189–201, 1993.

[11] W. Lien and W. Burleson. "Wave-Domino Logic: Timing Analysis and Applications" *MIT Advanced Research in VLSI and Parallel Systems,* March 1992.

[12] X. Zhang and R. Sridhar. "CMOS Wave Pipelining using Transmission-Gate Logic." *Proceedings of Seventh Annual IEEE International ASIC Conference and Exhibit*, Rochester, NY, September 1994.