An Asynchronous Pipelined 32×32-bit Iterative Multiplier Using Hybrid Handshaking Protocol

Yijun Liu
APT Group, the Department of Computer
the University of Manchester, Manchester M13 9PL, UK
yijun.liu@cs.man.ac.uk

Abstract

An asynchronous pipelined 32×32-bit iterative multiplier is presented in this paper. The multiplier supports 32×32-bit integer multiplication of both signed and unsigned operands. A 2-phase micropipeline latch controller is used which controls a 4-phase pipeline with standard transparent level sensitive latches. The design employs the modified Booth algorithm diminishing 8 bits at a time with an iterative structure. A sign extension algorithm is also employed in this work. Furthermore, the early termination scheme speeds up the multiplication operation. The multiplier consists of a total of 10700 CMOS elements and completes an 32×32-bit multiplication in 12 ns under the typical conditions. This work is also very low power and costs only 50% energy per operation of that of Amulet3i multiplier.

1 INTRODUCTION

Asynchronous logic gained a resurgence of interest among academic and industrial researchers due to its advantages in low power consumption, high operating speed, less emission of electromagnetic noise (EMI), better compensability and avoiding clock skew problems [1].

Synchronous design uses global clock to control dataflow in the datapath. A clocked system can be viewed as a finite-state machine (FSM) with registers (flip-flops) holding the current state. The clocked system changes from one state to the next state on the edges of the global clock. The state is held in a set of registers, and combinatorial logic is used to derive a new state and outputs. The new state is copied through the registers on every rising or falling edge of the global clock signal.

Asynchronous logic uses a different timing stratagem - handshaking protocol. There are two common handshaking protocols used by self-timed systems: 4-phase protocol [2] and 2-phase protocol [3]. Unlike 4-phase protocol, which uses level-sensitive control signals, 2-phase protocol employs events (rising or falling edges) to indicate the availability or absorption of data. The 4-phase protocol has better adaptability to most of VLSI designs but its superfluous return-to zero transitions cost unnecessary time and energy. In this work, we propose an asynchronous pipelined multiplier using a 2-phase pipeline control circuit to control a 4-phase datapath with level sensitive latches. Thus, we can keep the advantage of 4-phase protocol but gain a faster and lower power result.

The remainder of this paper is arranged as follows: Section 2 introduces hybrid handshaking protocol and asynchronous micropipeline. Section 3 provides the architecture of the multiplier. In section4, detailed circuit implementation is given and the performance is discussed. Section 5 presents the conclusions, and finally, section 6 give the acknowledgements.

2 HYBRID HANDSHAKING PROTOCOL AND MICROPIPELINE

Compared to 2-phase protocol, 4-phase protocol has better adaptability because most VLSI designs use level sensitive datapath. The asynchronous circuits using 4-phase handshaking protocol can easily use transparent latches in the pipeline without two- to four-phase signalling converters. While 2-phase handshaking protocol is faster and costs lower power because it eliminates the superfluous return-to-zero transitions. The motivation of employing the hybrid handshaking protocol is to combine the advantages of both 4-phase protocol and 2-phase protocol.

Figure 1 illustrates the block diagram of a hybrid latch controller. Rin is the request signal from the sender and Rout is the request signal to the receiver. While Ain is the acknowledge signal to the sender and Aout is the acknowledge signal from the receiver. Lt is the control line, which control the states of level sensitive latches. The control flow of the latch controller using hybrid handshaking protocol is shown as follows:

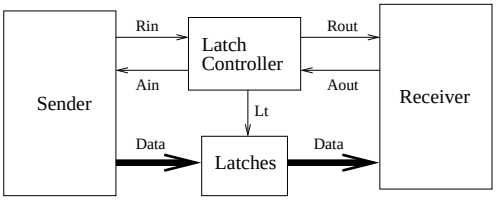

Figure 1: Block diagram of a hybrid latch controller

- At the beginning, Lt controls the state of latches as transparent (open).
- After the sender sets up data, it issues an event on Rin indicating the validity of data.

- The latch controller changes the state of latches to opaque (close) and let them absorb the data. Then the latch controller sends an event on Ain back to the sender to acknowledge the capture of data. At the same time it issues an event on Rout to indicate the availability of data.
- After some time, the latch controller detects an event on Aout from the receiver and it changes latches to transparent state again to prepare next cycle of data transmission.

By choosing different events (rising or falling edge), control circuits can be composed by different ways. The Signal Transition Graph (STG) [4] description of a latch controller is shown in Figure 2.

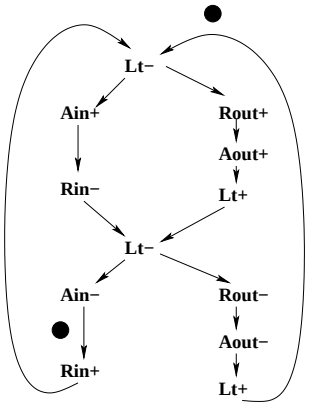

Figure 2: STG description of a latch controller

By a synthesis tool - Petrify [5], we can easily synthesize the latch controller from the STG description. The latch control circuit generated by Petrify is shown in Figure 3.

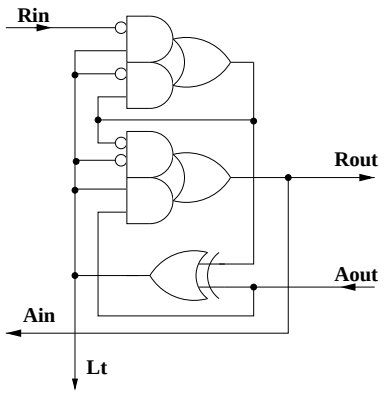

Figure 3: Latch controller implementation

But this circuit looks too big and not optimal. A better latch controller called “mousetrap” is proposed by Singh and Nowick in [6]. In this work, we use “mousetrap” as the latch controller to control the micropipeline.

3 MULTIPLIER ARCHITECTURE

Array multipliers [7] and tree multipliers [7] are rather fast but, on the other hand, are rather hardware hungry. Serial multipliers [7] need less area but have a very low throughput. Iterative multipliers are good choice considering the tradeoff between speed and silicon area. With pipeline technique, iterative multipliers can gain an equal throughput as that of parallel multipliers if we ignore the delay of registers or latches [8].

The multiplier of Amult3i [9] uses a synchronous pipeline structure clocked by an inverter oscillator. Compared to synchronous pipeline, self-timed pipeline has three main advantages:

- Higher speed: Asynchronous pipeline uses handshaking protocol and it operating speed is determined by actual combinational logic block latencies of each stage rather than the critical delay of all the stages.
- Less silicon area: Standard transparent latches are used in asynchronous pipeline whereas synchronous pipeline must employ edge-trigged registers to hold the stages. Edge-trigged registers occupy double silicon space compared to transparent latches. Furthermore, transparent latches are two times faster than edge-trigged registers, which furthers self-timed the pipeline’s advantage in high speed.
- More robust: Only after the availability and capture of data, will the sender and the receiver send the request and acknowledge signals. While a synchronous pipeline should satisfy the critical delay of all the stage circuits, which perhaps changes from time to time. For example, there comes a bug when the Amulet3i multiplier changes its state from normal calculation to early termination.

The multiplier in this paper uses modified booth algorithm and supports 32×32-bit integer multiplication of both signed and unsigned operands. The multiplier architecture is illustrated in Figure 4. This is a two-stage asynchronous pipeline structure using bundled-data protocol. The first stage includes a booth encoder, a pipeline latches row and a 4-2 compressors row. The second stage includes a pipeline latches row, a 4-2 compressors row and a shift registers row. The pipeline latch controllers are “mousetrap” introduced in the second section.

4 CIRCUIT IMPLEMENTATION AND PERFORMANCE

4.1 Booth multiplexer (Booth MUX)

The Booth MUX used in this multiplier is composed of 4 transmission gates. With elegant control signals, the Booth MUX can minimise the short circuit currents for low power reasons. The schematic of the Booth MUX is shown in Figure 5.

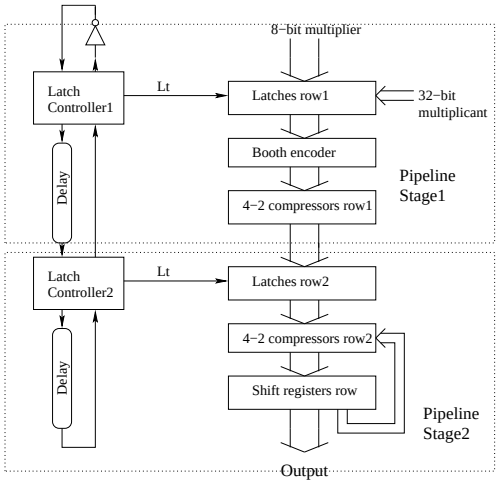

Figure 4: Multiplier micropipeline structure

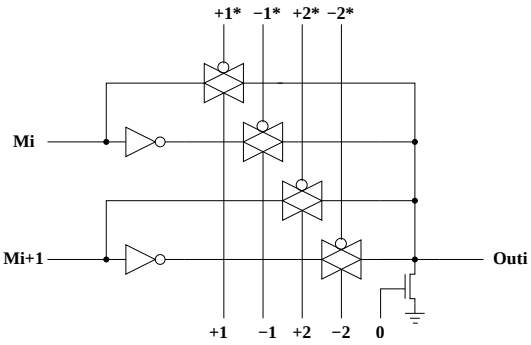

Figure 5: The Booth MUX

4.2 4-2 Compressor

With 4-2 compressors, we can build a multiplier with more regular structure compared to the one using 3-2 compressors. Moreover, 4-2 compressors row can diminish 4 partial products once compared to only 3 partial products once for 3-2 compressors row. So the multiplier with 4-2 compressors is faster than that using 3-2 compressors. Since the Cout signal is independent on Cin, there does not exist a propagation problem if several 4-2 compressors with same weight are abutted into the same row, which is the key idea behind 4-2 compressor. In this work, a new proposed 4-2 compressor using differential pass-transistor logic (DPTL) is employed and because of careful design, one XOR gate delay is saved compared to the 4-2 compressor constructed from two 3-2 compressors. Figure 6 shows the schematic of the 4-2 compressor. From Figure 6 we can see that the sum and carry are balanced for decreasing glitches. Balanced delay also results a lower worst-case latency.

4.3 Pipeline latch and shift register

Considering both speed and hardware consumption, we choose true single-phase clocking (TSPC) register to compose the shift register row. The pipeline latches are normal tranparent latches, which is very simple and fast. The schematic of TSPC register is illustrated in Figure 7.

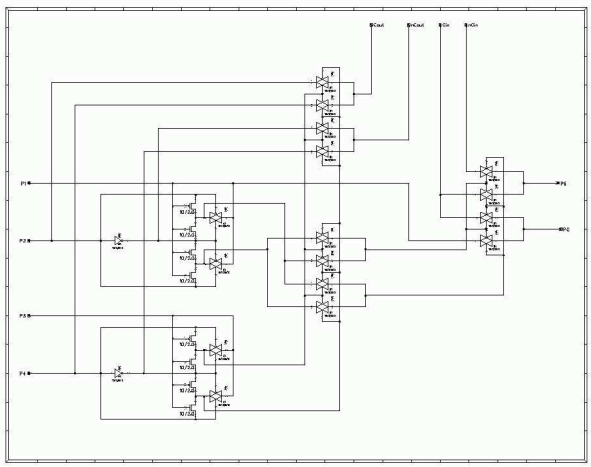

Figure 6: The schematic of the 4-2 compressor

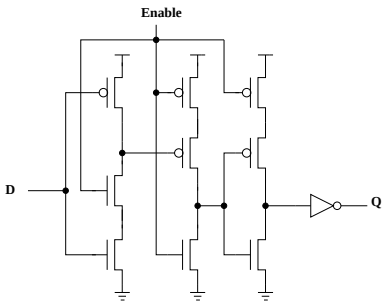

Figure 7: The schematic of TSPC register

4.4 Performance

The multiplier was analyzed using HSPIC on extracted layout under the conditions of 3.3 volt supply voltage and 100 degree temperature. The simulation results of delay (Typical process case and Worse process corner) are given in table 1.

Table 1: Simulated results of the Components Delay

Components	**Typical**	**Worse**
Booth MUX	0.61 ns	0.72 ns
4-2 Compressor	1.10 ns	1.40 ns
Transparent latch	0.18 ns	0.22 ns
TSPC register	0.66 ns	0.88 ns

The critical path of the first pipeline stage includes one Booth MUX, one 4-2 compressor and one pipeline latch and is totally equal to 2.34 ns (Under the worst-case conditions: Vdd = 3.3v, Vss = 0.1v, 100 degree temperature). The critical path of the second pipeline stage include one pipeline latch, one 4-2 compressor and one shift TSPC register and is totally equal to 2.5 ns. From the data given

above, we can know that it takes 12.34 ns ($2.34 + 2.50 \times 4$) to complement a 32×32-bit multiplication without using early termination. With 20% commercial timing margin, the multiplier completes a multiplication in 14.8 ns under worst-case conditions. The power of this work is about 80mw with full load in "peak time" and the average power is about 50mw.

Up to now, we have not designed the layout of the multiplier. So we cannot give the performance of silicon area.

5 CONCLUSIONS

A high performance, low hardware cost and low power asynchronous iterative multiplier has been developed in this work. The multiplier totally consists of 10700 CMOS elements and completes a 32×32-bit multiplication in 12 ns under the typical-case conditions. It suits for both signed and unsigned operands. The design uses the modified Booth algorithm. An early termination scheme is employed which efficiently speeds up the operation.

Table 2 shows the comparision between this multiplier and Amulet3i multiplier (* The numbers mean the multiplication cycles). Compared to Amulet3i multiplier, this multiplier is smaller. It contains 10700 CMOS elements, while Amulet3i multiplier contains 13600 CMOS elements. This dues to the simplify of the control circuit and the employment of transparent latches. And this multiplier is also 10% faster than Amulet3i multiplier because the delay of transparent latches is shorter than that of edge-triggled registers. Forthermore, this multiplier is more robust because the handshaking protocol can match the exact latency of each pipeline stage. Under the same conditions and with the same operands, this multiplier costs only 50% energy per operation of that of Amulet3i multiplier.

Table 2: The comparision between two multipliers

Multiplier	**This work**	**Amulet3's**	**Ratio**
CMOS	10700	13600	1/1.27
Latency(4*)	12.3ns	14.0ns	1/1.14
Latency(3)	11.4ns	13.4ns	1/1.18
Latency(2)	9.9ns	11.7ns	1/1.18
Latency(1)	8.0ns	10.7ns	1/1.34
Power(4)	80.0mw	86.2mw	1/1.08
Power(3)	67.4mw	82.1mw	1/1.22
Power(2)	50.0mw	82.7mw	1/1.66
Power(1)	25.1mw	66.0mw	1/2.63

$Power \times Latency$ is often used as a metric for the power consumption of a CMOS system. Given two designs A and B, if the $Power \times Latency$ of A is smaller than that of B, then A consumes less power then B when they operate the same number instructions. From Figure 8, we can see the average $Power \times Latency$ of this work is only $1/2$ of that of Amulet3 multiplier. So this work is much more power efficient than Amulet3 multiplier. The reason is eliminating the propagation of the glitches through the whole datapath. The unnecessary switches waste quite a lot power especially when they are propagated through the whole datapath. The immediately closing of the latches provents glitches from propagating to the next stages, thus saves power. From this example, we can see that the hybrid handshaking protocol is a good choice for low power circuits. Another reason is that during early termination period, only shift registers row consumes power, while compressors rows are "free". But for synchronous system, it is very difficult to stop some parts of the datapath.

Moreover, this work introduces a STG description for the hybrid pipeline latch controllers which uses 2-phase protocol to control level sensitive latches and gives an example of the latch control circuit synthesized by Petrify. The hybrid pipeline latch controllers make datapaths run faster comparing to those using traditional 4-phase latch controlleres because they avoid the superfluous return-to-zero transitions.

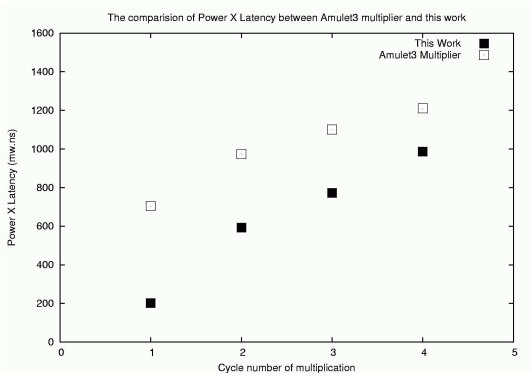

Figure 8: The comparision of $Power \times Delay$

6 ACKNOWLEDGEMENTS

The author would like to thank Dr. Montek Singh for the discussion on "mousetrap".

Yijun Liu is funded jointly by a research studentship from the Department of Computer Science at the University of Manchester and an ORS scholarship awarded by Universities UK. The author would also like to acknowledge with gratitude the grants from the Department of Computer Science at the University of Manchester and Universities UK.

7 REFERENCES

[1] J. Sparsø, S. Furber. "Principles of Asynchronous Circuit Design: A systems Perspective". Kluwer Academic Publishers, 2001

[2] Furber, S.B. and Day, P., "Four-Phase Micropipeline Latch Control Circuits", IEEE Trans. on VLSI, 4 (2), June 1996, pp. 247-253

[3] Sutherland, I.E., "Micropipelines", Communications of the ACM, 32 (6), June 1989, pp 720-738

[4] T.-A. Chu, "Synthesis of self-timed VLSI circuits from graph-theoretic specifications, PhD thesis", MIT, 1987.

[5] J. Cortadella et al, "Petrify: a Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers", IEICE Transactions on Information and Systems, E80-D(3): 315-325, 1997

[6] Montek Singh and Steven M. Nowick, "MOUSETRAP: Ultra-High-Speed Transition-Signaling Asynchronous Pipelines", ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU-2000), Austin, TX, December 2000.

[7] Amos R. Omondi, "Computer Arithmetic Systems: Algorithms, Architecture and Implementations", Prentice Hall, 1994.

[8] Santoro, M., "SPIM: a pipelined 64×64-bit iterative multiplier", IEEE Journal of Solid-State Circuits, vol. 24, April 1989, pp. 487-493.

[9] J. Liu, "Arithmetic and Control Components for an Asynchronous System, PhD thesis", The University of Manchester, 1997.