An 180 Mhz 16 bit Multiplier Using Asynchronous Logic Design Techniques

Richard G. Burford, Xingcha Fan and Neil W. Bergmann
CSIRO/Flinders Joint Research Centre in Information Technology
Flinders University, Adelaide, AUSTRALIA

Abstract

A CMOS digital logic design technique is described which exploits the advantages of fast precharged logic and efficient latch design commonly used in synchronous systems while maintaining the features of localized control inherent in asynchronous design. A pipelined sixteen bit multiplier is presented and its performance compared with several previously reported asynchronous and synchronous designs.

1 Introduction

Arithmetic operators are often the major building blocks and performance limiting factor for application specific Digital Signal Processing (DSP) and other numerical data processing hardware. It is generally believed that asynchronous arithmetic operators are slower and occupy a larger chip area than their synchronous counterparts and this is supported by several published studies [9]. We contend however, that with careful selection and design of control and data paths, asynchronous operators can be designed with performance equal to that of equivalent synchronous ones. When a *degradation factor* (typically 50 % or greater [2]) is applied to the clock speed to allow for temperature, supply voltage and process variations, asynchronous designs can exhibit a significant performance advantage. Here we present the design of a sixteen bit fixed point parallel multiplier which achieves a performance level similar to that of non-derated synchronous designs.

2 Asynchronous design techniques

Asynchronous digital logic design removes the global timing constraints of a clocked system. The flow of data is dictated by local timing considerations. This attribute is becoming increasingly important as feature size is reduced and chip complexity increases. Other potential advantages of asynchronous design include lower power consumption, simplified system level design, and greater product longevity. There are many approaches to the design of asynchronous systems, Hauck in [4] provides an excellent summary. Here we briefly review some methods used for reported multiplier designs which we will use for comparison.

An example of the *bounded delay technique* is the *micropipeline* [10]. Each bit of data is represented as a one bit variable carried over a single wire. Arbitrary data widths are *bundled* and the flow of data is controlled by the exchange of common request and acknowledge handshake signals. The computational delay in the data path is accounted for by introducing explicit delay elements in the control path. This technique of *delay modelling* must allow for the worst case delay of the data path. Two-cycle handshaking is used ie. a transition (either low-to-high or high-to-low) is used to signal an event. In order to use dynamic precharged logic, a pre-charge period between successive evaluation phases is required. Since two-cycle signalling does not have a return-to-zero phase which can be used for the pre-charge signal it is difficult to incorporate precharged logic into a micropipeline and so computation within the pipeline is usually implemented in static complementary logic (CMOS). This results in larger and slower logic cells than achievable using dynamic precharged logic because of the need for more complementary P type transistors.

The multiplier presented by Meng in [5] is an example of a *speed-independent* design utilizing precharged *Differential Cascode Voltage Switch Logic* (DCVSL) for computation. Data is *dual rail encoded* ie. two wires are used to encode each data bit. Inverters are necessary between succeeding computational stages to ensure that evaluation can commence only when the outputs of the preceding stage have settled. This introduces an inverter delay between each stage. The DCVSL provides complementary outputs which can be used for *completion detection*. Completion detection can produce a performance improvement when there is a data dependent variation in computation time. An example is a ripple carry adder where a completion signal can be generated when the carry propagation between stages is complete [3]. However the time and area overhead for completion detection may outweigh its advantages.

The *Latched Differential Pass Transistor Logic* (LDPL) design used by Salomon *et al* [8] is another dual rail technique which provides high speed with low power con-

IEEE 1994 CUSTOM INTEGRATED CIRCUITS CONFERENCE

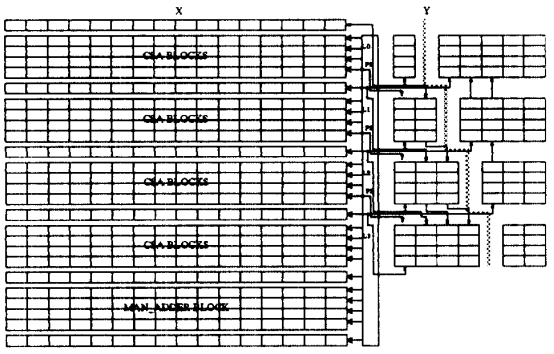

Figure 1: Multiplier floor plan

sumption. A feature of this method is the incorporation of data storage within the output buffer of the logic stage. This eliminates the need for data latching registers in pipelined designs.

We have chosen to use a bundled data path with four-cycle handshaking and a delay model in the control path. The four-cycle signalling protocol provides a return-to-zero phase which can be used with precharged logic.

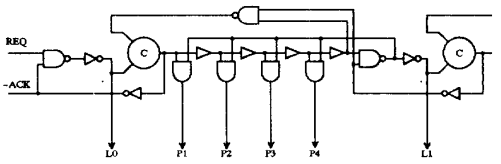

Figure 2: Control path for one pipeline stage

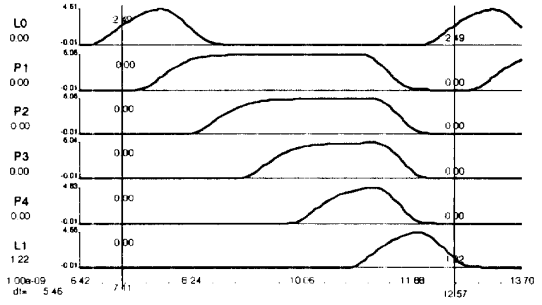

Figure 3: Latch and precharge signals for one pipeline stage

3 Overall structure

The multiplier uses sixteen stages of sixteen bit carry save adders (CSA) and a combination of Manchester carry adders and carry select adders to calculate the final thirty two bit result. It is implemented as a five stage pipeline. The five stage pipeline is the result of a trade off between a throughput, latency, and chip area. Increasing the number of stages of pipelineing increases the number of interstage registers. This results in greater throughput at the expense of a longer latency time and larger chip area. Five stages of pipelining maximize throughput while not exceeding the allowable active die area (excluding pads) of 3.24 mm^2 for the chosen prototype fabrication process (Orbit Semiconductor tiny chip) [7].

The floorplan of the multiplier is shown in Figure 1. The X operand is input at the top of the array, while Y is fed in from the right. The array of carry save adders and the final Manchester adder stages occupy most of the chip area. Pipeline register stages for Y input and lower order outputs are on the right, while control circuitry occupies a vertical strip between the input pipeline and the adder array. Precharged logic with a pull down evaluation tree of n-channel MOSFETS is used for all computational logic blocks.

4 Progressive evaluation

To ensure that the NMOS pull down tree evaluates correctly it is essential that evaluation of a given stage does not commence until all its inputs are correct and stable. One technique to ensure this is to insert register stages between each logic block, this however increases latency by adding register latching time. Another method is to interpose inverters between each evaluation stage which increases stage delay.

Our design is five stage pipeline using handshaking elements to generate local latching signals for the pipeline registers. Since the design is asynchronous, the precharge and evaluation phases of adjacent logic blocks are not constrained by a global clock system. Within each pipeline stage, adjacent evaluation stages are released from precharge after the output of the preceding stage has settled. The term we use for this technique is *Progressive Evaluation* (PE). This technique is similar to a multi-phase synchronously clocked precharged logic system. Timing for the precharge and evaluation phases is derived from taps in the delay model for the overall computational stage as shown in Figure 2. The string of buffers is a delay line which models the delay of the adder stages. Muller-C-elements [6] co-ordinate data transfer between pipeline stages. When an input request is generated, and the stage is empty, latch signal (L0) goes high, latching the data into the input register. The output of the C-element is then driven high starting the evaluation of the first CSA (signal P1 changes from pre-charge state to evaluate). The signal progresses through the inverter chain causing signals P2 - P4 to go high in succession. Finally a latching signal is generated for the next pipeline register and all evaluation stages are returned to the precharged state. A SPICE simulation for one pipeline stage of four carry save adders is shown in Figure 3.

5 Circuit elements

5.1 Carry save adder

The circuit used for the carry save adder stage is shown in Figure 4. Complementary inputs and outputs are used to eliminate the need for inverter stages thus minimizing computation time. Simulations using normal transistor process models show that both sum and carry outputs which evaluate to a low state reach 50 % of rail voltage in 0.6 ns and 25 % within 0.8 ns. Because no pull-up transistors are active during the evaluation phase, the evaluation of the following stage can be commenced once any low level inputs have settled below the threshold of the NMOS evaluation tree. Allowing 0.8 ns between successive evaluations gives a safety margin of 50 % with a typical half rail threshold.

5.2 Manchester carry adder

The final pipeline stage of the multiplier resolves the high order sixteen bits of the result. This achieved using a combination of Manchester carry adders, and carry select adders [11]. The result is evaluated and latched in approximately 5 ns, which matches the time taken for the previous pipeline stages.

5.3 Pipeline registers

The pipeline registers for X and Y inputs and output results, use nine transistor single-phase positive edge triggered D flip flops [1]. The circuit, shown in Figure 5(a) can be implemented with fewer transistors than micropipeline style transition registers [10] and exhibits near zero data hold time. An input inverter is used to give a non inverting latch, resulting in eleven transistors for each latch element. Simulations indicate, when implemented in the 1.2 micron CMOS process used for our design, the latch has a setup time of 0.8 ns and total delay of 1 ns under normal operating conditions.

Carry and sum outputs are latched using the circuit shown in Figure 5(b). This is a latched form of a Muller C-element. When the latch signal (L) is high and D is not equal to ~D, the output will equal D. When D equals ~D (ie when precharged) the output will be held. When L is low the output will also be held. When used with the latching signals shown in Figure 3, the latch is enabled while the latch signal is high and assumes the correct value when the precharged signals evaluate. Simulation results show a delay of 0.9ns from input to output.

6 Performance

SPICE simulations using normal transistor parameter models [7] at an operating temperature of 27 degrees Celsius indicate the multiplier is capable of accepting input operands every 5.46 ns or an effective throughput rate of 183 MHz. Latency is 25 ns when the pipeline is full, but because of the

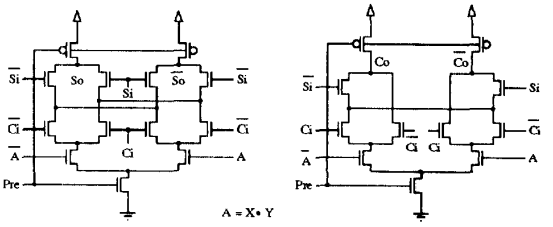

Figure 4: Carry save adder circuit

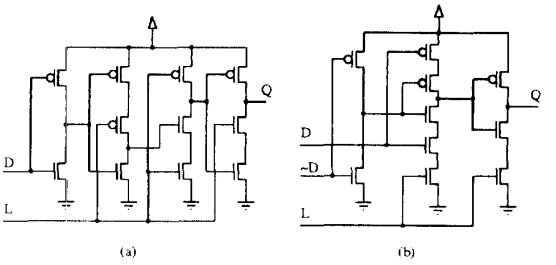

Figure 5: (a) Pipeline register latch (b) C-element latch

elastic nature of the pipeline this is reduced to 19.6 ns for an empty pipeline. Table 1 compares the performance of several multiplier designs, the DCVSL multiplier reported by Meng [5], LDPL, micropipeline and synchronous designs from Salomon *et al.* [8], with this design. The entries shown for Area(scaled) have been scaled to 1 micron feature size for comparison purposes. Note that the DCVSL design is not pipelined. A synchronous design using techniques similar to those presented here (including Progressive Evaluation) should be able to achieve performance equal or perhaps slightly better than our design (without allowing for a degradation factor). Such a design would not however have the elastic pipeline properties.

7 Conclusions

A multiplier design which offers the advantages inherent in asynchronous design without sacrificing performance when compared to other approaches has been presented. The design is part of a toolkit of asynchronous logic building blocks for DSP being developed by the CSIRO/Flinders Joint Research Centre.

The multiplier has been fully designed and simulated and is about to be submitted for fabrication in a 1.2 micron

Technique	DCVSL	LDPL	Mpipe.	Sync.	PE
Feature size	1.6	1.0	1.0	1.0	1.2
Area mm^2	8.1	2.59	2.64	2.53	3.03
Area(scaled)	3.16	2.59	2.64	2.53	2.1
Speed (MHz)	28.6	156	104	172	180
Latency (ns)	35	64	-	-	25

Table 1: Performance comparison of several multipliers

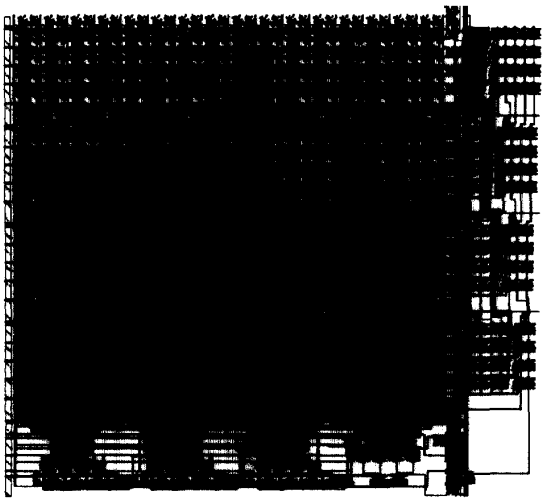

Figure 6: Mask layout of the multiplier

CMOS process. The layout is shown in Figure 6. Operational results should be available for presentation at the conference.

8 Acknowledgements

The support of the Australian Research Council and the Australian Telecommunications and Electronics Research Board is acknowledged.

References

[1] M. Afghahi and C. Svensson. A Unified Single-Phase Clocking Scheme for VLSI systems. *IEEE Journal of Solid-State Circuits*, 25(1):225–233, February 1990.

[2] M.E. Dean. Strip: A Self-Timed RISC Processor. Technical Report CSL-TR-92-543, Computer Systems Laboratory, Stanford University, July 1992.

[3] J.D. Garside. A CMOS VLSI Implementation of an Asynchronous ALU. In *IFIP Working Conference on Asynchronous Design Methodologies*, Manchester, England, April 1993.

[4] S. Hauck. Asynchronous Design Methodologies: An Overview. Technical Report 93-05-07, Department of Computer Science and Engineering, University of Washington, Seattle WA, 1993.

[5] T.H-Y. Meng. *Synchronization Design for Digital Systems*. Kluwer Academic Publishers, 1991. ISBN 0-7923-9128.

[6] D.E. Muller. *Asynchronous Logics and Application to Information Processing*. Switching Theory in Space Technology. Stanford University Press, 1963.

[7] Orbit Semiconductor Inc, Sunnyvale, CA. *Foresight Users Manual*, rev 1.4, July 1991.

[8] O. Salomon and H. Klar. Self-Timed Fully Pipelined Multipliers. In *IFIP Working Conference on Asynchronous Design Methodologies*, Manchester, England, April 1993.

[9] J. Sparso, C.D. Nielsen, L.S. Nielsen, and J. Staunstup. Design of Self-Timed Multiplers: A Comparison. In *IFIP Working Conference on Asynchronous Design Methodologies*, Manchester, England, April 1993.

[10] I.E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, June 1989. 1988 Turing Award Lecture.

[11] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design*. Addison Wesely, 1988.