A Pipelined Multiplier Using A Hybrid Wave-Pipelining Scheme

Suryanarayana B. Tatapudi and José G. Delgado-Frias
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752
Email: {statapud, jdelgado}@eecs.wsu.edu

Abstract— A hybrid wave-pipeline multiplier architecture is described in this paper. Mathematical analysis is provided to show the performance gains possible with hybrid wave-pipeline over conventional pipeline architectures. The clock period in conventional pipeline scheme is proportional to the maximum delay while in hybrid wave-pipelining it is proportional to the maximum delay difference. An 8×8-bit hybrid wave-pipeline multiplier using carry-save adder technique is fully described. The multiplier has been designed using TSMC 180nm (drawn length 200nm). Since in hybrid wave-pipelining clock period is proportional to delay difference, short clock periods can be obtained by minimizing the delay difference. The basic cells in multiplier are designed to have small propagation delay and delay variation. The pipelined multiplier is able to achieve 2.86 billion multiplications per second. The delay balancing necessary to reduce the delay variation is simpler in hybrid wave-pipeline architecture than in wave-pipeline architecture.

***Index Terms*—pipelined systems, high performance, hybrid wave-pipelining, multipliers.**

I. INTRODUCTION

Clocking is an essential component in sequential logic design in computer systems. Switching events must occur in well defined order and at precise moments for correct operation of the system. Such synchronous systems utilize a globally distributed clock signal for this purpose. Today's fast high frequency clocks have to be generated on chip by means of Phase-Lock Loops (PLL) using an external slow clock as reference. Apart from the performance improvements achieved from faster devices, very large scale integration, and faster clocks, newer architectures are needed to produce significant performance gains.

Pipelining has emerged as the design technique of choice that helps to achieve high throughput digital systems. Pipelining essentially breaks down a single complex computational block into discrete blocks separated by registers or clock storage elements (CSE like Flip-Flops, latches). Thought this does not reduce the actual time taken for computation (actually there is a minor increase in computation time due to the introduction of registers), the throughput and logic circuitry utilization increases. Figure 1 shows a block diagram of a simple pipeline scheme.

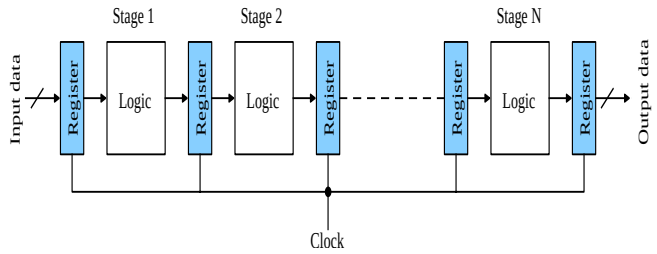

Figure 1. Simple pipeline scheme.

Such a pipelining scheme drastically increased the operating frequencies of the systems. In recent years the desire to scale clock frequencies and achieve higher performance has led to the implementation of super-pipelined systems resulting in additional overhead from the CSE. In a super-pipelined system, the latency from the logic blocks may be comparable to the latency of the CSE which envelope them, adversely effecting the performance of the systems. Also, with thousands of CSEs in a super-pipelined system distributing Giga-hertz clock on the entire chip is a complex task [1, 2].

As technology scaling continues, logic devices are becoming extremely fast. Interconnect delays are becoming the dominating delays. The clock distribution has become a major bottleneck for performance improvement. As the clock speeds increase with scaling, the clock uncertainties like uncontrolled transmission line effects, clock skew and clock jitter are not scaling

accordingly and increasing portion of clock time is being spent in countering these overheads. The clock distribution network is also increasing in size (longer wires with increased parasitic resistance, capacitance and inductance) and load. Once consequence of this is the increase in clock network's power consumption, which is around 50% of the total chip power consumption [3]. Also now the clock signal traverses different paths from its source (at the output of on-chip PLL or the input pin on the chip) to different parts of chip. Based on the path taken (physical dimensions and parametric values), buffers in the path, process variations and, load driven, the clock sinks will receive clock with uncertainties. This could jeopardize the systems performance or could result in its complete failure [1]. Significant research is being done to counter such uncertainties by design optimization and technology improvements [4, 5]. Considerable work has been put into studying various clock distribution schemes like the resistance-capacitance matched tree, the grid, the tree network, and the serpentine. But the gain from using these schemes is small. Also it is increasingly becoming difficult to study them due to parasitic (resistance, capacitance and inductance) modeling errors. With shrinking feature sizes, the interconnecting wires are becoming thin and long, significantly increasing the wire resistances. With high speed signals with fast rise and fall times running on thin long wires, the inductive component of wire parasitic is gaining significance [6]. This necessitates the use of more accurate delay models taking into account the wire inductances.

Additional performance improvements can be gained by dealing at gate level and layout level issues to further minimize the clock uncertainties mentioned above. Architecture can be modified to eliminate the complex clock distribution mechanism to achieve significant gains in power consumption and performance. Alternate architectures like wave-pipelining, asynchronous pipelining and package wiring [4] have been proposed. While asynchronous pipelining may be appealing since it completely eliminates the need to distribute clock, it is complex compared to synchronous schemes and the performance improvement is higher in alternate synchronous schemes [4, 7].

In this paper we propose a novel multiplier architecture using a hybrid wave-pipelining, which achieves significant performance gains compared to the regular pipeline and wave-pipeline schemes. The organization of the paper is as follows. In Section II, the Hybrid wave-pipelining concept is introduced, with timing constrains and mathematical analysis to prove the performance gains. In Section III, we discuss the implementation of an 8-bit multiplier in the Hybrid wave-pipeline architecture. Performance analysis of the multiplier is presented in Section IV. Finally in Section V, some concluding remarks are presented.

II. The Proposed Pipeline Scheme

In this section, we introduce the proposed pipeline approach called Hybrid Wave-Pipelining. To compare with conventional pipeline we introduce the delays that affect the performance of a digital pipelined system.

A. Conventional Pipeline scheme

In a conventional pipeline scheme, the system is divided into small sub-systems (logic block) separated by registers/latches as shown in Figure 1. These intermediate registers synchronize the data from a stage with the reference clock edge (typically the leading edge) and propagate the data to the next stage. In this scheme a logic block operates on only one set of data vectors at any given time. New data is admitted into a stage only after the data in that stage has been cleared and latched by the intermediate register following it. This necessitates the clock to be free of all uncertainties, especially when the clock periods are shrinking in time. The pipeline stage with the longest computation time, dictates the clock-cycle time for the entire system. The intermediate registers are also an overhead on the clock times and clock power. Figure 2 shows a graphical representation of combined temporal and spatial variation for a pipeline scheme and Figure 3 is for a regular pipeline scheme with three stages.

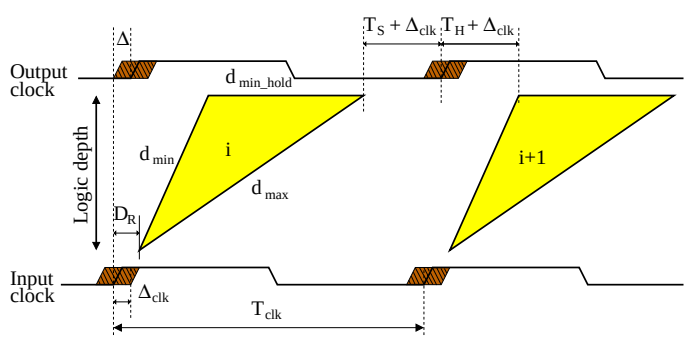

Figure 2. Temporal/Spatial diagram of a pipeline scheme.

The variables used in the Figure 2 are defined as follows.

T_{clk}	Clock period
Δ	Constructive clock skew
Δ_{clk}	Unconstructive clock skew or clock uncertainties
D_R	Clock-to-output delay of the register/latch

T_S, T_H	CSE setup and hold times
$d_{min(i)}$	Minimum propagation delay through the stage *i* of a multi-stage system
$d_{max(i)}$	Maximum propagation delay through the stage *i* of a multi-stage system
$d_{min\_hold(i)}$	Delay difference between $d_{max(i)}$ and $d_{min(i)}$
T_{hold}	Register overhead

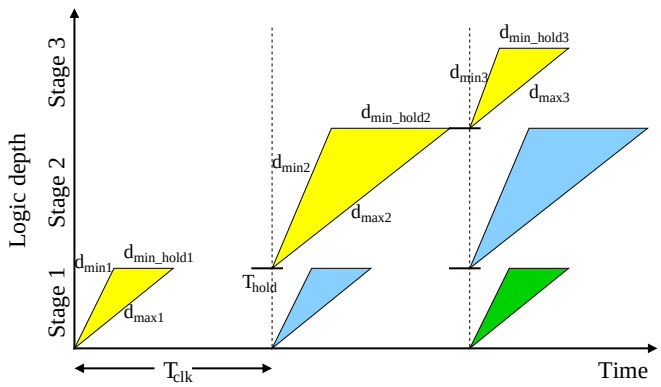

Figure 3. Temporal/Spatial diagram of a regular pipeline scheme.

Equation 1 defines the clock cycle time for a conventional pipeline scheme, where D_{max} is the computation time of stage with the longest delay, $D_{max} = max(d_{max(j)})$

$$T_{clk_{reg}} \geq D_{\max} + T_S + \Delta_{clk} \qquad (1)$$

B. Hybrid Wave-pipeline scheme

The proposed hybrid wave-pipeline scheme modifies the wave pipeline scheme [8, 9] to achieve improved power and performance gains. In this scheme the clock is also wave-pipelined as shown in Figure 4. The clock frequency is determined by the stage with the maximum delay difference. Contrary to this scheme's similarities to regular pipeline scheme, it allows multiple data waves to exist in any stage similar to wave-pipelining. Higher clock frequencies are possible and influence of clock uncertainties is mitigated. As can be seen, this scheme eliminates the need for complex clock distribution. Clock gating can be easily implemented to save power without affecting the pipeline's performance.

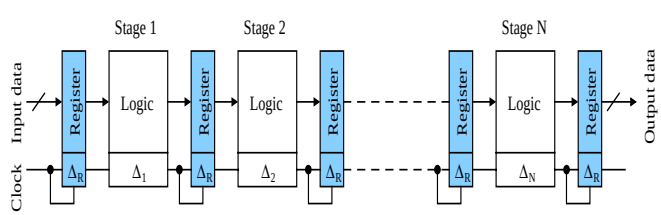

Figure 4. Hybrid wave-pipelining scheme.

The temporal and spatial variation of the proposed hybrid wave-pipeline architecture is shown in Figure 5. The timing constraint analysis of this system can be done using Figure 5. For this analysis a three-stage system was chosen and the computational cones of each stage have been made linear as shown in Figure 5, for a simpler analysis. In this scheme the clock is also wave-pipelined and travels along with the data.

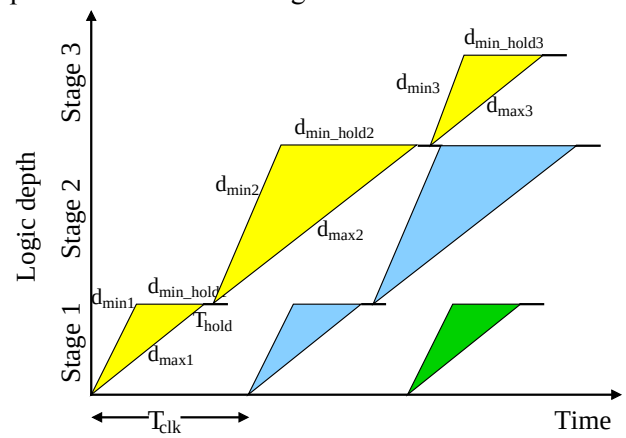

Figure 5. Temporal/Spatial diagram of proposed Hybrid wave-pipeline architecture.

The clock period T_{clk} of this system is determined by stage with the largest delay difference and safe time required before a new data vector/wave is admitted into this stage. The fundamental circuit limitations determine the safe time to separate any two adjacent data waves. The equation for T_{clk} can be derived as follows using Figure 5.

$$T_{clk_h} \geq \sum_{n=1}^{j} d_{\max(n)} - \left(\sum_{n=1}^{j} d_{\min(n)} + \sum_{n=1}^{j-1} d_{\min\_hold(n)} \right) + T_S + T_H + 2\Delta_{clk} \qquad (2)$$

Where *j* denotes the stage with the largest delay difference between the minimum and maximum propagation delays. In the example shown in Figure 5, stage 2 is the stage with maximum delay difference.

$$T_{clk_h} \geq \sum_{n=1}^{j} d_{\max(n)} - \left(\sum_{n=1}^{j} d_{\min(n)} + \sum_{n=1}^{j-1} \left[d_{\max(n)} - d_{\min(n)} \right] \right) + T_S + T_H + 2\Delta_{clk} \qquad (3)$$

Eliminating the redundant terms, Equation 3 can be re-written as shown in Equation 4.

$$T_{clk_h} \geq d_{\max(j)} - d_{\min(j)} + T_S + T_H + 2\Delta_{clk} \qquad (4)$$

From Equation 4 it is clear that a smaller delay difference would result in a higher clock frequency. The delay difference can be minimized by balancing delay using buffers. Since each combinational logic stage in this scheme is wave-pipelined, the internal node

constrains must also be considered so that any two adjacent data waves do not collide. This can be easily satisfied by designing the logic stages such that a stage's delay difference is greater than the delay difference at any internal node in that stage or in other words the delay difference should monotonically increase from input to output of that stage [8].

The clock period for a wave-pipelined system has been derived in [8] and given as:

$$T_{clk_w} \geq (D_{MAX} - D_{MIN}) + T_S + T_H + 2\Delta_{clk}$$

Where D_{MAX} and D_{MIN} are the maximum and minimum propagation delays of the entire system.

It is not difficult to show that for any system the following expression is valid.

$D_{\max} \geq (D_{MAX} - D_{MIN}) \geq (d_{\max(j)} - d_{\min(j)})$, which implies that $T_{clk_h} \leq T_{clk_w} \leq T_{clk_{reg}}$. This in turn validates the claim that Hybrid wave-pipelining delivers an improved performance compared to regular and wave-pipelined schemes.

III. 8 × 8-BIT PIPELINED MULTIPLIER

Multiplier is a basic building block of Digital Signal Processing (DSP) processors. DSP processors performance is closely tied to the performance of multiplier, and since most signal processing algorithms can be implemented in pipelined architecture, a pipelined version of multiplier can be implemented to achieve high throughput. An 8-bit multiplier has been chose to be implemented in Hybrid wave-pipeline architecture as a proof of concept. The well-know Carry-Save Adder (CSA) technique has been used to implement the multiplier. In this technique, say for an N-bit multiplier, N stages reduce N-partial products to 2 partial products. Until this point the data flow is from one stage to next. In the last row of multiplier, the 2 partial products are reduced to the product, which involves the data flow within the stage. This can be implemented with any N-bit FA like carry-look-ahead adder, but this would make this stage the bottleneck. Instead of this, we chose to implement bit-by-bit addition using N-stages of Half Adders (HA) as a result each bit of final product is generated in each of these stages. This only affects the latency, but not the throughput. Effectively the architecture has 2N stages. Figure 6 shows the schematic of the 8×8-bit multiplier implemented in Hybrid wave-pipeline architecture. The placement of the flip-flops was determined based on the simulation of the basic cells in the multiplier and will be discussed in Section 4. In this implementation all the stages/logic enveloped between any two adjacent flip-flop stages become a single wave-pipelined stage.

From the mathematical analysis in Section II we understand that the secret for lower clock periods is to use logic blocks with small delay difference. The full adder used in the multiplier was implemented based on transmission gates so that the Sum and Carry signals are generated simultaneously. Also differential implementation (complimentary inputs are used and complimentary outputs are generated) was chosen to speed up the full adder and avoid glitches. Transistor level implementation of the FA is shown Figure 7.

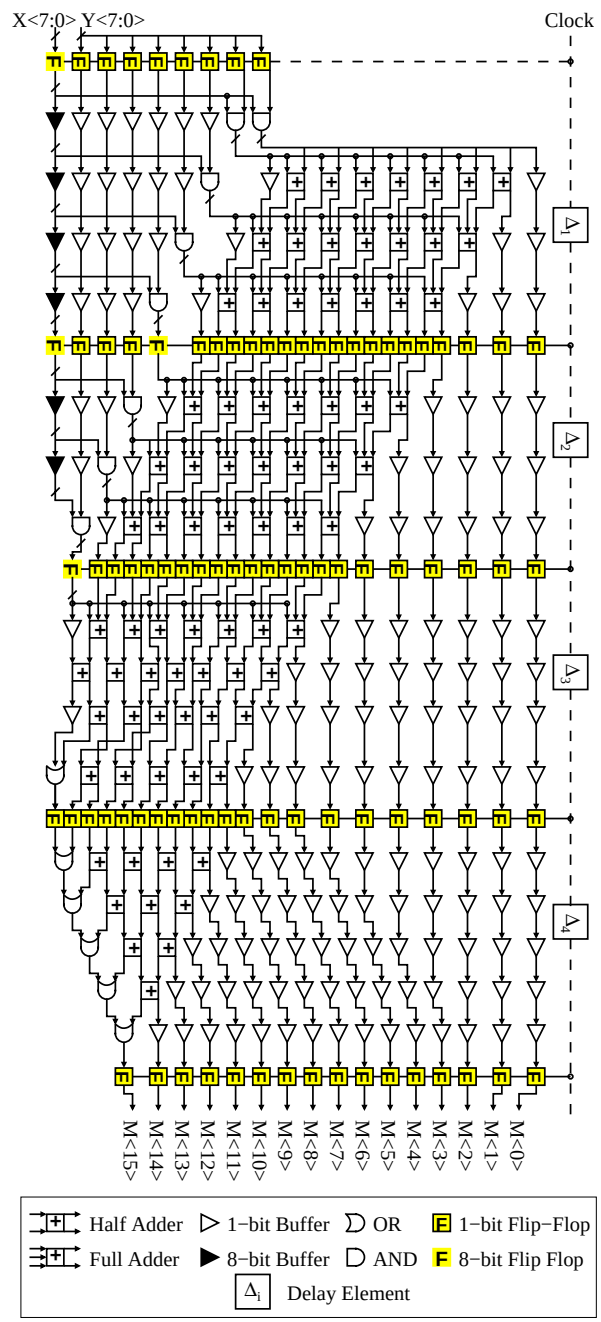

Figure 6. Hybrid wave-pipelined 8×8-bit multiplier.

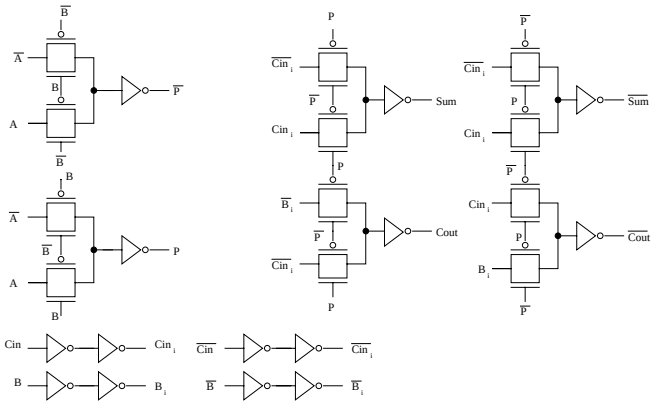

Figure 7. Transistor level implementation of Full Adder.

To retain the regular structure and delay variations of full adder, the half adder was implemented as full adder from Figure 7, with carry-in set to logic 0. Pseudo-nMOS implementation was chosen to realize the AND gates in partial product generation to obtain symmetric response for all possible input transitions. Buffers were added to the AND gates to equalize their delay to that of FA, this would avoid unnecessary transitions.

The registers/CSEs used in the multiplier are positive edge-triggered D flip-flops. The flip-flops sample their inputs at the clock rising edge, generate the outputs simultaneously for the next stage. The delay variations in the inputs to the flip-flops are eliminated when presented to the next stage. An improved version of Sense Amplifier based Flip-Flop (SAFF) with complementary push-pull [10] was the flip-flop implemented in the multiplier. The schematic of this flip-flop is shown in Figure 8. Since differential implementation was chosen for full adders, the SAFF is an ideal choice for this system as it is a differential implementation. The SAFF accepts true and complimentary inputs and generates true and complimentary outputs simultaneously. It uses single-phase clock and is a small load on clock distribution network. The first stage of the flip-flop is essentially a sense amplifier which assures accurate timing necessary in high speed applications [10]. This flip-flop also has short setup and hold times.

The 8×8-bit multiplier was implemented in TSMC 180nm (drawn length 200nm) with 1.8V supply voltage and the layout is shown in Figure 9.

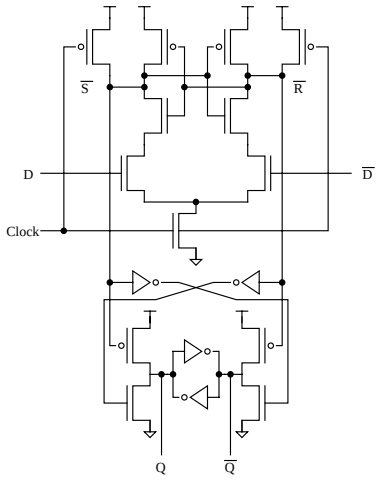

Figure 8. Sense Amplifier based Flip-Flop with complimentary push-pull S-R Latch [10].

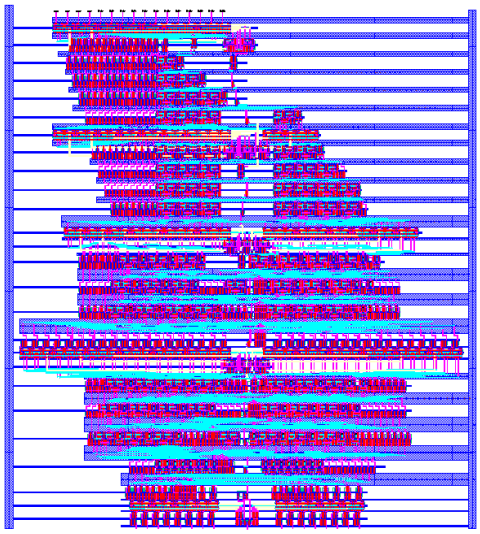

Figure 9. Multiplier Layout.

IV. Performance Analysis

Simulations of the proposed hybrid wave-pipelined multiplier have been performed in *SpectreS* under Cadence environment with a supply voltage of 1.8V. In the following sections results obtained for the basic cells and the entire system are presented.

A. Full Adder

Extensive simulations were performed on the Full adder to precisely characterize performance of this cell. Co-incident inputs (zero skew) were applied to the full adder cell and propagation delay was measured. There are a total of 56 transitions possible for the 3 inputs to a full adder. Of these 56 transitions 32 transitions only trigger a transition on the Sum and/or Carry output. For

these 32 transitions the propagation delay of the full adder was measured. Iterative process was used to optimize the transistor sizes to achieve minimum propagation delay and delay variation. The propagation delay values obtained for these 32 transitions are graphically represented in Figure 10. The propagation delay for the full adders varied from 210ps ($d_{\min}$) to 280ps ($d_{\max}$), resulting in a delay variation of 70ps. The internal node constraints dictate the rate at which new inputs can be applied to the full adder and from simulations it was observed that the fastest the inputs could be applied is at intervals of around 175ps.

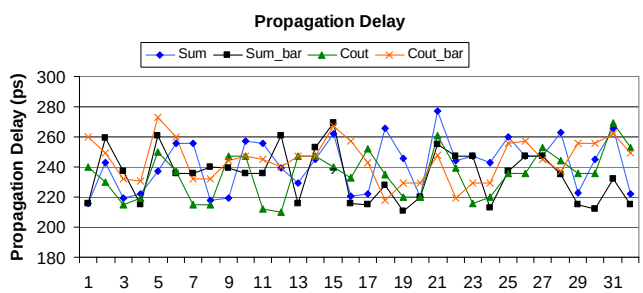

Figure 10. Propagation Delay of the Full Adder.

In the multiplier schematic shown in Figure 6, it can be observed that the other basic cells are AND gate, OR gate and buffers. These gates were designed to give a small propagation delay variation. For the AND and OR gates, buffers were added so that their propagation delay is close to $d_{\max}$ of the full adder.

B. Sense Amplifier based Flip-Flop (SAFF)

The transistor sizes in SAFF [10] were determined through an iterative process with knowledge of input signal driving strength and output drive needed. Simulations were performed to determine the setup time (T_S), hold time (T_H) and the sampling time. Setup time is define as the time for which data input must be stable before the arrival of clock active edge for the flip-flop to successfully store the data. Hold time is defined as the time for which the data must be held after the arrival of the clock active edge for the flip-flop to store the data. Simulations performed on the flip-flop revealed that the clock high time must be at least 160ps. In [1] it was reported that the SAFF has a negative setup time and simulation results revealed that at the clock frequencies of operation having a short positive setup time of 10ps gives a better response. The hold time was observed to be approximately 130ps. The *Clk-Q* delay was approximately 295ps.

Buffers were added to the flip-flop to increase the *Clk-Q* delay to this value to provide the driving capability and approximately equalize its delay to that of the full adder. Since the internal node constraints of the flip-flop are less than that of the Full Adder and their delays comparable, the flip-flop can be considered as another stage in the pipeline.

C. Multiplier

The simulation results reveal that the bottle neck in the pipeline is the flip-flop. The clock frequency is dependent on the timing limitations imposed by the flip-flop. The clock period has to be at least 320ps. Compensating for possible clock uncertainties ($2\Delta_{clk}$) a clock period of 350ps (≈2.86GHz) (T_{clk}) was chosen. According to Equation 4, $d_{max(j)} - d_{min(j)}$ can take a maximum value of 190ps. The placement of flip-flops as shown Figure 6 was based on this calculated limit on delay difference. The logic enclosed between any two adjacent flip-flop stages is wave pipelined and has a delay difference less than 190ps. Figure 11 shows the inputs to the first stage of the multiplier. Each data wave passes through the logic blocks shown in Figure 6, and as the wave propagates, each data path adds different delay. As a result the delay variation of the data waves increases. Figure 12 illustrates the delay variation of each data wave after the first stage. Since the delay variation at this point is close to the calculated limit, a flip-flop is used to synchronize the data waves. The rising edge of the clock is used by the flip-flop to sample its inputs. The synchronized data waves as stored by the second flip-flop stage are shown in Figure 13. The small variation observed in these signals is due to vertical clock skew and load variation.

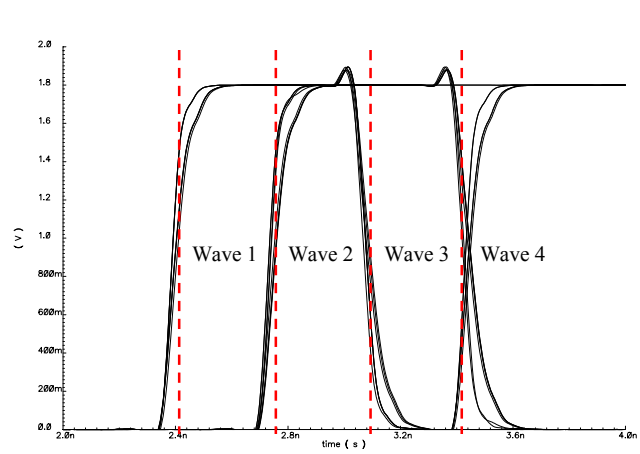

Figure 11. Inputs to the First Stage of the Multiplier

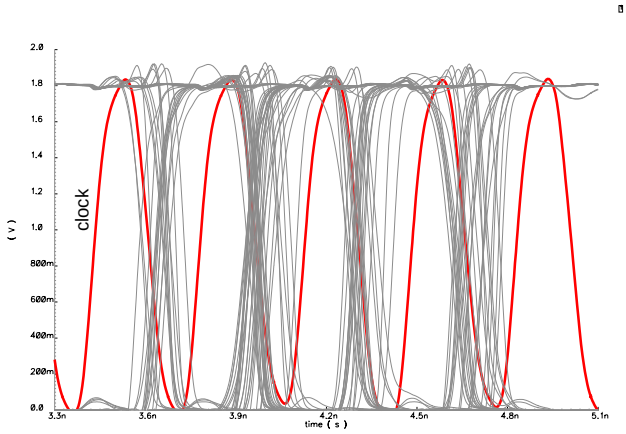

Figure 12. Outputs of the First Stage

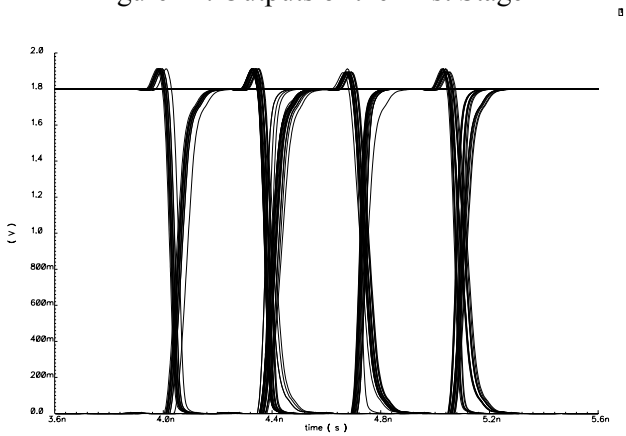

Figure 13. Outputs of Second Flip-Flop Stage

Simulations performed on the entire system revealed that the system successfully performed 8×8-bit multiplication every clock period i.e. 350ps. Using the same technology, a 3-inverter chain ring oscillator has been simulated; this circuit yields an oscillation period of approximately 260ps. A 3-inverter ring oscillator is considered to produce the highest clock frequency for a given technology. Comparing the multiplier and ring oscillator clock periods, it is remarkable the multiplier's clock period is just 35% longer than the shortest possible period.

V. Concluding Remarks

Hybrid wave-pipelining, a novel pipeline architecture, has been described in this paper. Mathematical analysis has been provided to show the performance gain possible with hybrid wave-pipeline over conventional pipeline architectures. The clock period in conventional pipeline scheme is proportional to the maximum delay while in hybrid wave-pipelining it is proportional to the maximum delay difference.

An 8×8-bit hybrid wave-pipeline multiplier using carry-save adder technique has been designed and simulated. The multiplier was implemented in TSMC 180nm (drawn length 200nm) and simulations performed proved the feasibility of the hybrid wave-pipeline architecture. Since in hybrid wave-pipelining clock period is proportional to delay difference, short clock periods can be obtained by minimizing the delay difference. The basic cells in multiplier have been designed to have small propagation delay and delay variation.

The pipelined multiplier was able to achieve 2.86 billion multiplications per second. The number of flip-flops need in this implementation is significantly less compared to a conventional pipeline. The delay balancing necessary to reduce the delay variation is simpler in hybrid wave-pipeline architecture than in wave-pipeline architecture.

References

[1] V. G. Oklobdzija *et. al.*, Digital System Clocking, Wiley-Interscience, 2002.

[2] F. Klass, *et. al.*, "*A New Family of Semidynamic and Dynamic Flip-Flops with Embedded Logic for High-Performance Processors,*" IEEE Journal of Solid State Circuits, vol. 34, no. 5, May 1999 pp. 712 – 716.

[3] D. E. Duarte, N. Vijaykrishnan, and M. J. Irwin, "*A Clock Power Models to Evaluate Impact of Architectural and Technology Optimizations,*" IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 10, no. 6, December 2002 pp. 844 – 855.

[4] P. J. Restle, and A. Deutsch, "*Designing the best clock distribution network,*" Symposium on VLSI circuits, 1998.

[5] S. Tam, R. D. Limaye, U. N. Desai, "*Clock Generation and Distribution for the 130-nm Itanium 2 Processor with 6-MB On-Die L3 Cache,*" IEEE Journal of Solid State Circuits, vol. 39, no. 4, April 2004 pp. 636 – 642.

[6] Y. I. Ismail, and E. G. Friedman, "*Effects of Inductance on Propagation Delay and Repeater Insertion in VLSI Circuits,*" IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 8, no. 2, April 2000 pp. 195 – 206.

[7] E. G. Friedman, "*Clock Distribution Networks in Synchronous Digital Integrated Circuits,*" Proceedings of IEEE, vol. 89, no. 5, May 2001 pp. 665 – 692.

[8] C. T. Gray, W. Liu, R. K. Cavin, "*Timing Constraints for Wave-pipelined Systems,*" IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, no. 8, August 1994 pp. 987 – 1004.

[9] W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, "*Wave-Pipelining: A Tutorial and Research Survey,*" IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 6, no. 3, September 1998 pp. 464 – 474.

[10] V. Stojanovic, V. G. Oklobdzija, FLIP-FLOP, US Patent No. 6,232,810, May 15, 2001.