International Journal of Applied Science and Engineering
2008. 6, 2: 181-197

Design of High Speed Asynchronous Pipelined FIR Filter Using Quasi Delay Insensitive Reduced Slack Pre-Charged Half Buffer

A.Senthilkumar[a], A.M.Natarajan[b*]

[a] *Professor, Department of ECE, Kongu Engineering College, Perundurai, Erode., TN, India.*
[b] *Professor, Department of ECE, Bannari Amman Institute of Technology, Sathy, Erode TN, India.*

Abstract: Asynchronous design is progressively becoming more attractive alternative to synchronous design because of its potential for high-speed and low-power. The pipelining technique is very effective for synchronous digital designs. This paper proposes the design of pipelined Finite Impulse Response (FIR) filter using asynchronous quasi-delay-insensitive (QDI) template based on Reduced Slack Pre-Charged Half Buffer (RSPCHB). Both synchronous and asynchronous pipelined FIR filter have been designed using TSMC 0.18-µm CMOS technology. HSPICE simulation shows that the speed of the asynchronous system has been improved 12 times with 2 times increased area over synchronous design.

Keywords: Asynchronous; FIR Digital Filter; Pipelining; QDI; Reduced Slack Pre-charged Half Buffer (RSPCHB)

1. Introduction

Most digital circuits designed and fabricated today are “synchronous”. In essence, they are based on two fundamental assumptions that all signals are binary and all components share a common and discrete notion of time, as defined by a clock signal distributed throughout the circuit. Asynchronous circuits are fundamentally different; they also assume binary signals, but there is no common and discrete time. Instead the circuits use handshaking between their components in order to perform the necessary synchronization, communication, and sequencing of operations. Expressed in ‘synchronous terms’ this results in a behavior that is similar to systematic fine-grain clock gating and local clocks that are not in phase and whose period is determined by actual circuit delays-registers are only clocked where and when needed. This difference gives asynchronous circuit’s inherent properties that can be exploited to advantage in the areas such as low power consumption, high operating speed, less emission of electro-magnetic noise and No clock distribution and clock skew problems [1].

Mainly the pipelining techniques are adopted in synchronous design of digital FIR filters to improve the performance such as speed and

[*] Corresponding author; e-mail: ask_rect@yahoo.com

Accepted for Publication: December 14, 2008

power consumption. The pipeline gating technique is used to reduce the power consumption [2], but it could not improve the speed appreciably, because the speed is limited by the flip flops and clock skew problem. This paper proposes a new design of pipelined FIR Filter using the asynchronous quasi delay insensitive template Reduced Slack Pre-Charged Half Buffer (RSPCHB) which achieves reasonable speed improvement**.**

Quasi delay insensitive (QDI) [3] design is a practical approximation to delay insensitive design. QDI circuit work correctly regardless of the delay of signal with in the circuit, except for some assumptions about certain delay. The QDI assumption has also been extended to include assumptions of isochronic propagation through a number of logic gates. Commonly used QDI templates are the Weak-Conditioned Half Buffer (WCHB), the Pre-charged Half Buffer (PCHB), and the Pre-charged Full Buffer (PCFB) templates. In this paper we have used the new QDI template called Reduced Slack Pre-Charged Half Buffer (RSPCHB).

The filter was designed with TSMC 0.18 μm CMOS technology at the nominal 2.5-V power supply, on average, it runs at 250 MHz. This paper also presents a flow and methodology for the design of QDI pipeline template-based FIR Filter. The remainder of this paper is organized as follows. Section II gives Literature Review on asynchronous channels and QDI pipeline templates, the FIR Filter, and the previously-presented synchronous Pipelined FIR Filter. Section III presents the proposed asynchronous architecture. Section IV and Section V presents the results and discussion, comparisons, and conclusions.

2. Materials and methods

This section first explains about asynchronous communication channels and RSPCHB template. Then, it describes the FIR filter and the previously designed synchronous pipelined design.

2.1. Asynchronous Communication Channel

In general, asynchronous designs are composed of blocks communicating using handshaking via asynchronous communication channels [4]. An asynchronous communication channel is a bundle of wires and a protocol to communicate data between blocks. The encoding scheme in which one wire per bit is used to transmit the data and an associated request line is sent to identify when data is valid is called single-rail encoding and is shown in Figure 1.The associated channel is called a bundled-data channel.

Alternatively, in dual-rail encoding the data is sent using two wires for each bit of information. Both single-rail and dual-rail encoding schemes are commonly used, and there are tradeoffs between each. Dual-rail allows for data validity to be indicated by the data itself and are often used in QDI designs. Single-rail, in contrast, requires the associated request line, driven by a matched delay line, to always be longer than the computation.

This latter approach requires careful timing analysis but allows the use of the same single-rail logic as is used in conventional synchronous design.

2.2. Reduced slack pre-charged half buffer (RSPCHB)

The key goal of the RSPCHB [5] compared to the precharged half buffer (PCHB) discussed by Recep O et.al is to eliminate the need of the enable signal en from the control of the function block. The need for this enable signal is only to support concurrency in the system that effectively does not improve performance.

More specifically, in the PCHB template the output of the left completion detector (LCD) and right completion detector (RCD) are combined using a C-element to generate the acknowledgment signal L_{ack}. This supports the integration of the handshaking protocol with the validity and neutrality of both input and output data, which removes the need for the function block to be weak-conditioned, but also requires the use of the enable signal. It is this replacement however that introduces more concurrency than is necessary.

In particular, in the case of a join, the non-weak-conditioned function block may generate an output as soon as one of the input channels provides data. In response, the RCD of the join will assert its output. Meanwhile, any subsequent stage can receive this data, evaluate, assert both its LCD and RCD outputs, and assert its acknowledgment signal. Although the join can receive this acknowledgment, it will not precharge until after en is asserted. The en signal delays the precharge of the circuit until after the acknowledgement to the input stages has been asserted. This delay is critical to prevent the precharge from triggering the RCD to de assert which would prevent the C-element from ever generating the acknowledgment. If only the generation of the acknowledgment signal from any stage subsequent to the join was delayed until all input data to the join has arrived and been acknowledged, then the en signal could be safely removed. In fact, such a delay of the acknowledgement would not generally impact performance because the join is the performance bottleneck for the subsequent stages. Therefore, this added concurrency is essentially unnecessary.

This pipeline template, which reduces this unnecessary concurrency to eliminate the internal en signal, thereby reducing the transistor stack sizes in the function block. This new QDI pipeline template, illustrated in Figure 2(a), as a Reduced Stack Precharged Half Buffer (RSPCHB). A specific form of this template for dual-rail data is shown in Figure 2(b). Notice that we optimized the RCD block by tapping its inputs before the output inverter and using a NAND gate instead of an OR gate.

The unique feature of the RSPCHB is that it derives the request line from the output of the C-element instead of the RCD. (In particular, since the output of the C-element is active low and the request line is active high, the output of the C-element is sent through an inverter before driving R_{req}.) The impact of this change is that the assertion/de-assertion of R_{req} is delayed until after all L_{req}'s are asserted/de-asserted. As a consequence, the acknowledgment from a subsequent stage of the join may be delayed until well after its data inputs and outputs are valid. More specifically, the stage will delay the assertion of its acknowledgment signal until all L_{req}'s are asserted which can occur arbitrarily later than the associated data lines becoming valid. This extra delay, however, has no impact on steady-state system performance because the join stage is the bottleneck, waiting for all its inputs to arrive before generating its acknowledgement. In fact, this change yields a template with no less concurrency than WCHB. The advantage of this generation of the request line is that the function block does not need to be guarded by the enable signal [6]. In particular, it is now sufficient to guard the function block solely by the Pc signal because the Pc signal now properly identifies when inputs and outputs are valid. Namely, the function block is allowed to evaluate when Pc is deasserted which occurs only after all inputs and outputs data lines are reset. Similarly, it is allowed to precharge when Pc is asserted which occurs only after all input and output data lines are valid.

The RSPCHB is still QDI, however, the communications along the input channels to joins become QDI instead of delay-insensitive (other channels remain delay-insensitive). In particular, the assumption that must be satisfied is that the data should reset before the join stage enters a subsequent evaluation cycle. If we assert that the fork between the function block, the RCD, and the next stage is isochronic, this assumption is satisfied.

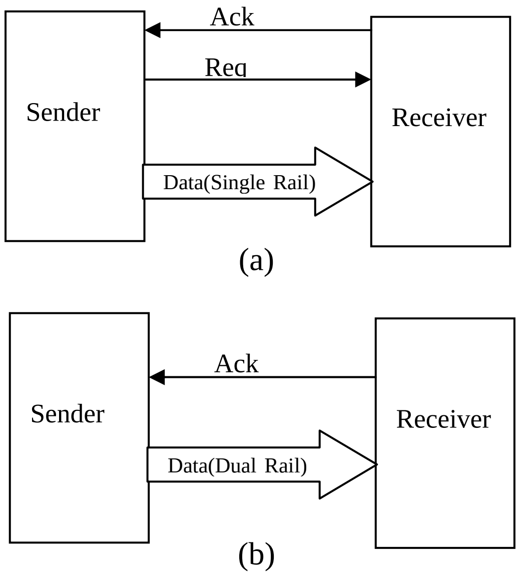
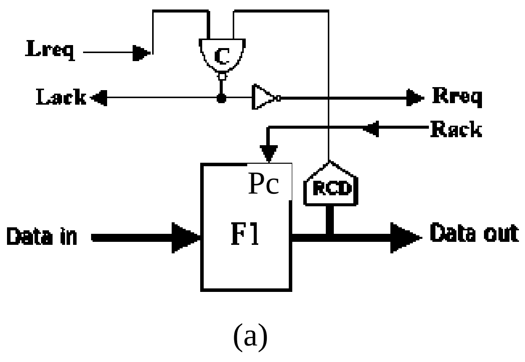


Figure 1. Pipeline channels (a) Bundled data channel.(b) Dual-rail channel.



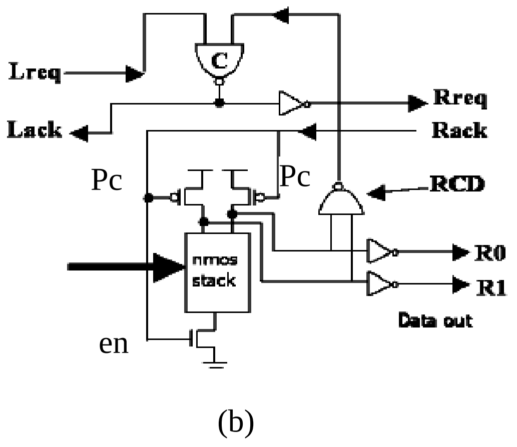


Figure 2. a)Abstract and b) detailed QDI RSPCHB pipeline template.

The analytical expression for the timing margin associated with this isochronic fork assumption can be derived from the abstract STG of the RSPCHB shown in Figure 3. Where F_1^e represents the Function F1 at evaluation state, F_1^P represents the Function F1 at pre-charge state, C_1^+ represents the completion detector C1 changes its state to 1, C_1^- represents the completion detector C1 changes its state to 0, RCD^+ represents the Right Completion Detector RCD changes its state to 1 and RCD^- represents the Right Completion Detector RCD changes its state to 0 of pipeline stage 1.

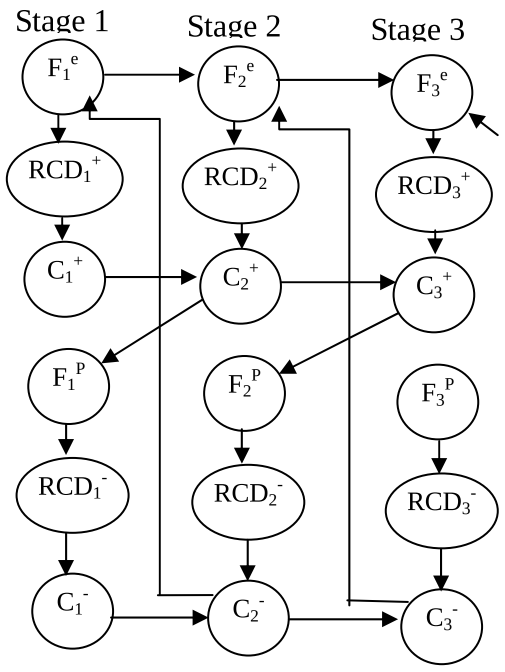


Figure 3. The STG of 3 stage popeline using RSPCHB

In particular, the delay difference between the resetting of the data and the associated request line should be less than T_{margin} .

$$T_{M\arg in} = 2t_{inv} + t_{cd} + 3t_c \qquad (1)$$

Where t_{inv} is inverter delay , t_{cd} is the completion detection delay and t_c is the C element delay.

This margin is between 6 and 8 gate delays depending on buffering. Notice that this timing assumption only applies to input channels of join stages because non-join stages must receive both valid data and a valid L_{req} before generating valid output data or valid R_{req}. Equation 2 shows the analytical cycle time of the RSPCHB which can be derived from the STG shown in Figure 3.

$$T_{RSPCHB} = Max(3.t_{Eval} + 2.t_{CD} + 2.t_C + t_{prech}, t_{Eval} + 2.t_{CD} + 4.t_C + t_{prech}) \tag{2}$$

Where t_{Eval} is the evaluation time, t_{CD} is the completion detection delay, t_c is the C element delay and t_{prech} is the time required for pre-charge.

With bubble shuffling, RSPCHB and PCHB have equal numbers of transitions per cycle. The advantage of RSPCHB is that the lack of an LCD and reduced stack size of the function block, which reduces capacitive load, and yields significantly faster overall performance. The cost of this increase in performance is that it requires one extra communicating wire between stages. A C-element may be used to implement a completion detection circuit for self-timed or delay-insensitive circuit. Figure 4 shows a two-input C-element with two input C-element, with two inputs a , b , output c and its timing diagram. Figure 5 shows a dual-rail self-timed component with an n-input C-element [7]. The self-timed component has completed an operation when DoneReset signal goes high and it has been reset when DoneReset signal goes down.

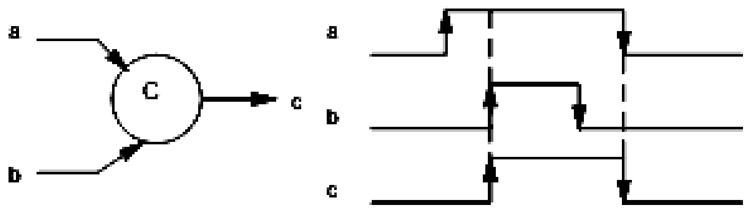


Figure 4. Two-input Muller C-element.

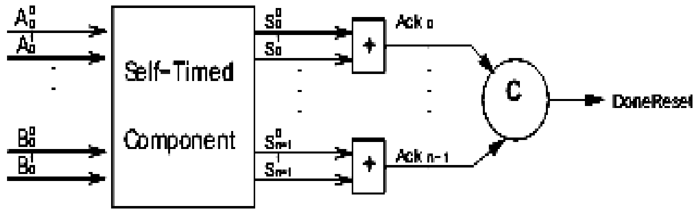


Figure 5. Self-timed component with completion detection circuit.

2.3. Synchronous popelined FIR filter

Figure 6 shows a pipelining scheme for FIR filter [3]. As has been discussed, an FIR filter must perform the following convolution equation:

$$Y(n) = h(k) * x(n) = \sum_{k=0}^{N-1} h(k).x(n-k) \tag{3}$$

Where h(k) is the filter coefficient array and x(n-k) is the input data array to the filter. The number N, in the equation, represents the number of taps of the filter and relates to the filter performance as has been discussed above.

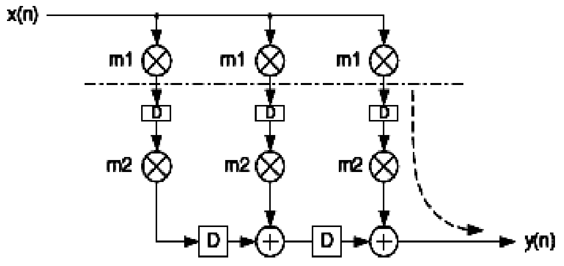


Figure 6. pipelining scheme for FIR filter.

There are different implementations of FIR filters. By pipelining multipliers as shown in Figure 6 can only achieve limited throughput improvement because the shortest critical path length limited by adder delays. In recent years, the word length of FIR filter has been growing up to 64-bit. Under long word length condition, addition also takes significant time [8]. To further improve throughput of FIR filters, the critical path in addition process needs to be shortened also. So adders, as well as multipliers, need to be pipelined [13]. Unlike pipelining multipliers, which doesn't change the relative timing sequence between adder inputs, pipelining adders just likes adding delay elements to the paths between adders. This will change the timing difference of the two inputs of the next adder from one clock cycle to more than one clock cycle, thus causes the incorrect output results

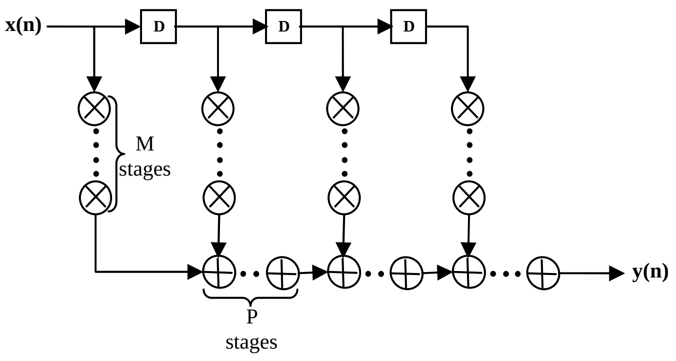


Figure 7. Pipelined FIR filter

To solve this problem, additional delay elements are added between next adder and its corresponding multiplier. The goal is to maintain the timing difference between the two inputs of the adder as one clock cycle. A 4-tap Data-Broadcast FIR filter with pipelining is shown in Figure 7. There are four multipliers and three adders. The general input-output relationship is shown in equation 1.

A. Synchronous Pipelined Adder

Synchronous pipelined implementation of 16-bit carry propagation adder is shown in Figure 8. Adder design is a part of the FIR filter design. 16-bit carry propagation adder is pipelined into 15 stages. Latency of the pipelined adder is 16 clock pulses. The throughput of the pipeline is determined by the slowest stage.

B. Synchronous Pipelined Multiplier

In pipelined multipliers, each pipeline stage contains a number of registers. Clock is connected to each register. In each clock cycle, a transition will occur on the clock input node of each register. This transition is independent of input data and will cause power dissipation even when the current input data of the register is the same as the current data output.

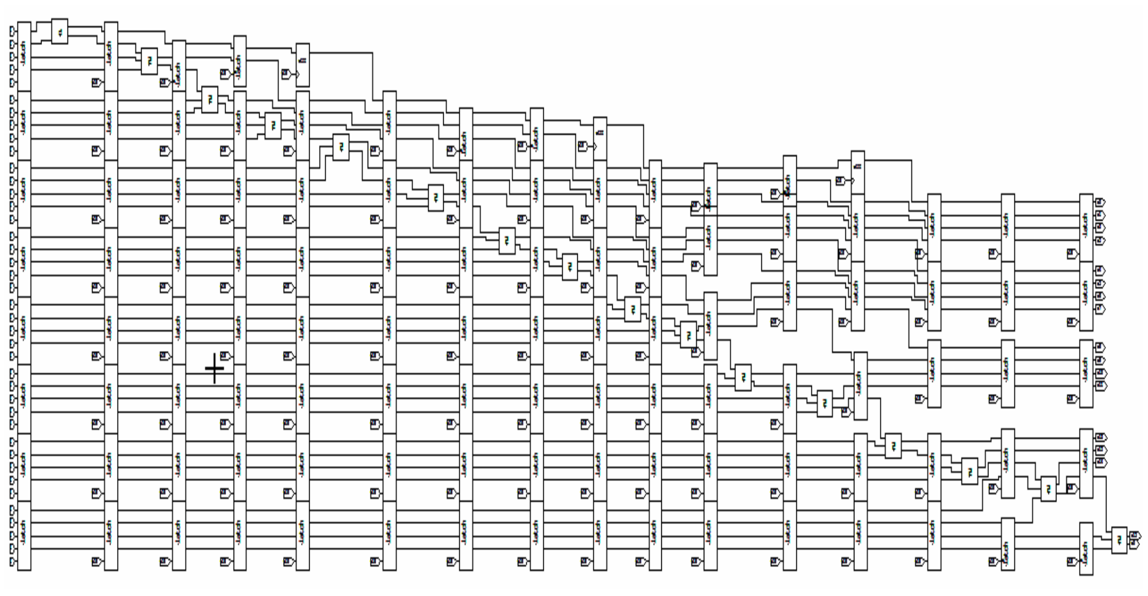


Figure 8. pipelined 16-bit carry propagation adder.

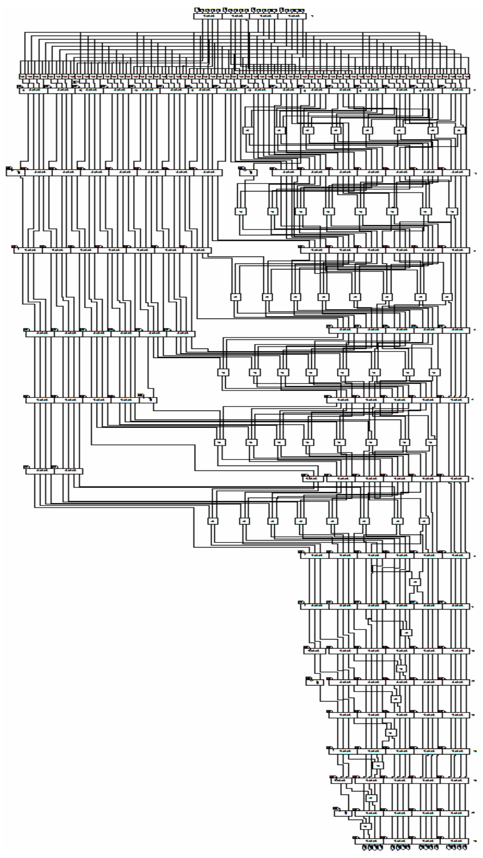


Figure 9. Pipelined 8X8 Array Multiplier

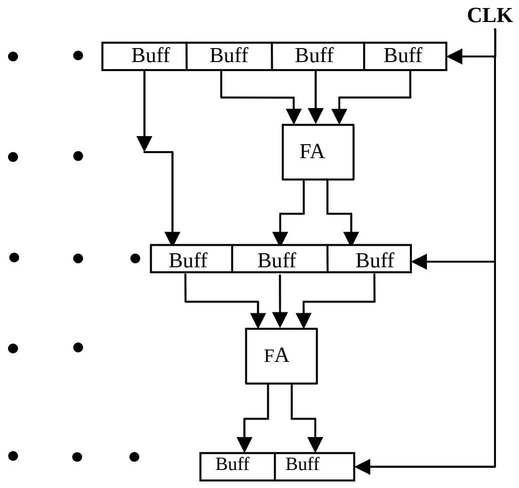


Figure 10. Detailed Diagram Of Pipelined Stage.

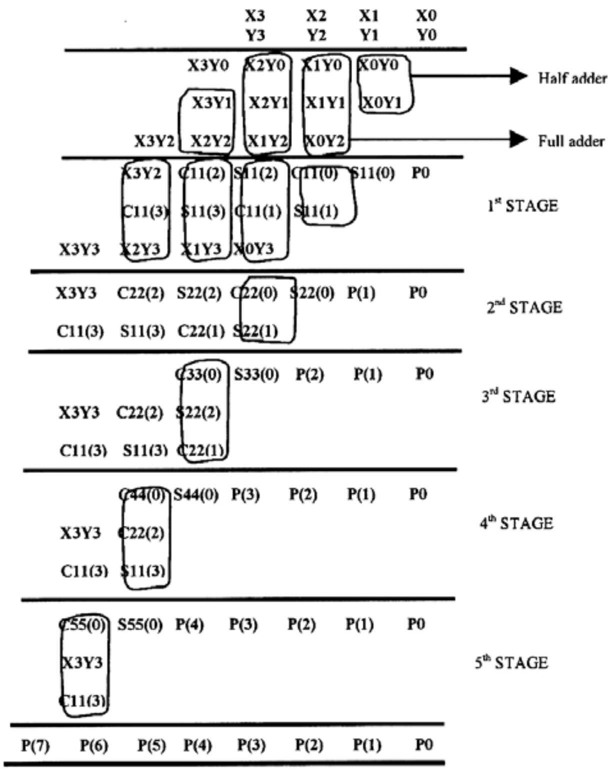


Figure 11. Multiplication process

The basic idea of synchronous pipelined implementation of a 4X4 array multiplier with 5 stages is shown in Figure 11

Figure 9 shows the pipelined 8X8 array multiplier, which is pipelined into 15 stages. The final carry propagation adder (CPA) is also pipelined to reduce the clock period. Last eight stages of the multiplier is pipelined CPA. Multiplier is designed in TSMC 0.18-µm CMOS technology. Multiplier has the latency of 16 clock cycles. After 16th clock pulse, each clock will produce output.

The last carry propagation adder in the multiplier is also pipelined. Figure 10 show the detailed diagram of pipelined multiplier stage.

3. Asynchronous fir fil ter design

The fact that the asynchronous circuit has no global clock, allows the asynchronous architecture to be naturally divided into two blocks, each operating at its ideal speed that communicate only when and where needed via the inter-block asynchronous channels [9]. Performing the same types of optimizations in the synchronous architecture may also be feasible but to obtain similar performance would either involve managing multiple clock domains or require all units, including the non critical units, to be highly pipelined, wasting energy, and increasing clock complexity.

3.1. Asynchronous pipelined adder

A. RSPCHB Adder Stage

Sum and Carry circuits are designed as per the RSPCHB template. The Figure 12 shows the circuit of Full adder with Dual rail logic, precharge and enable signal required to pipeline the different stages. This full adder is used to develop the pipelined 16 bit carry propagation adder as shown in Figure 15.

3.2. Completion detection

C_element [7, 10] is used to implement a completion detection circuit for self-timed or delay insensitive circuits. The Figure 13 shows the completion detection circuit used in this paper. If a = b = 1 then c = 1 and if a = b = 0 then c = 0, otherwise the value of c remains unchanged. This can be generalized to an n-input C-element. The output of an n-input C-element is 1 if all inputs are 1 and it is 0 if all inputs are 0; otherwise its value remains unchanged.

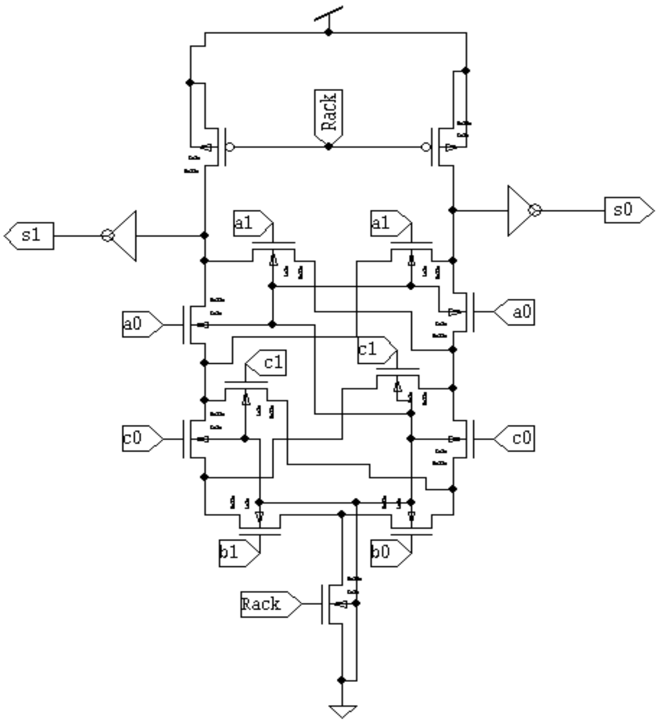


Figure 12. (a) Circuit for RSPCHB Sum.

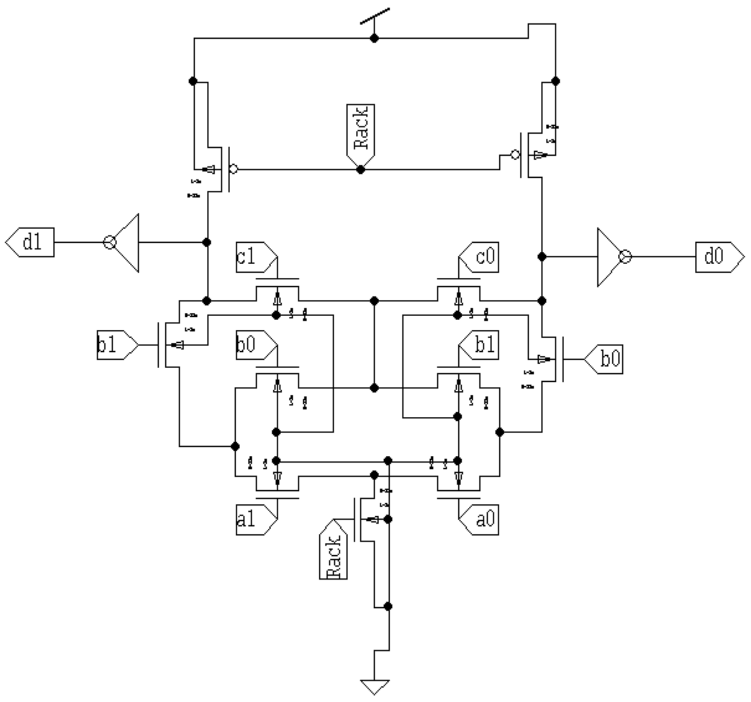


Figure 12. (b) Circuit for RSPCHB Carry

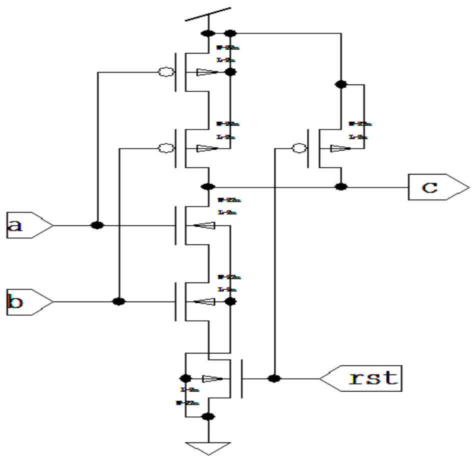


Figure 13. C_element.

3.3. Weak-Condition Half-Buffer Buffer (buffer WCHB)

Figure 14 show a WCHB template for a linear pipeline with a left (L) and right (R) channel and an optimized WCHB dual-rail buffer. L0 and L1, R0 and R1 identify the false and true dual rail inputs and outputs, respectively. Lack and Rack are active-low acknowledgment signals.

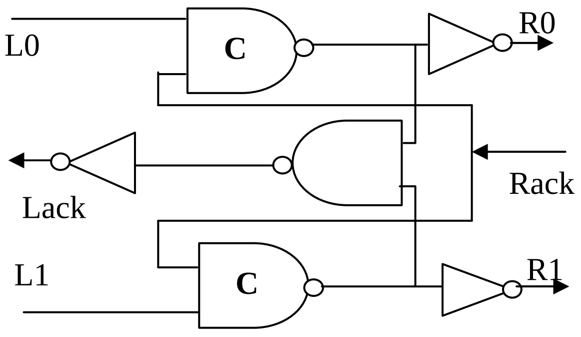


Figure 14. Weak-Condition Half-Buffer Buffer Circuit

The operation of the buffer is as follows. After the buffer has been reset, all data lines are low and acknowledgment lines, Lack and Rack, are high. When data arrives by one of the input rails going high, the corresponding C-element output will go low, lowering the left-side acknowledgment Lack. After the data is propagated to the outputs through one of the inverters, the right environment will assert Rack low, acknowledging that the data has been received. Once the input data resets, the template raises Lack and resets the output. This has been used in the adder and multiplier to adjust the slack time.

3.4. 16-Bit Pipelined Carry Propagation Adder

The carry signal has to travel through all the full adder cells, allocate tens of buffer WCHB cells around the full adders in order to match the slacks and optimize the throughput. After completion of the addition operation adder will generate the request signal to previous stage.

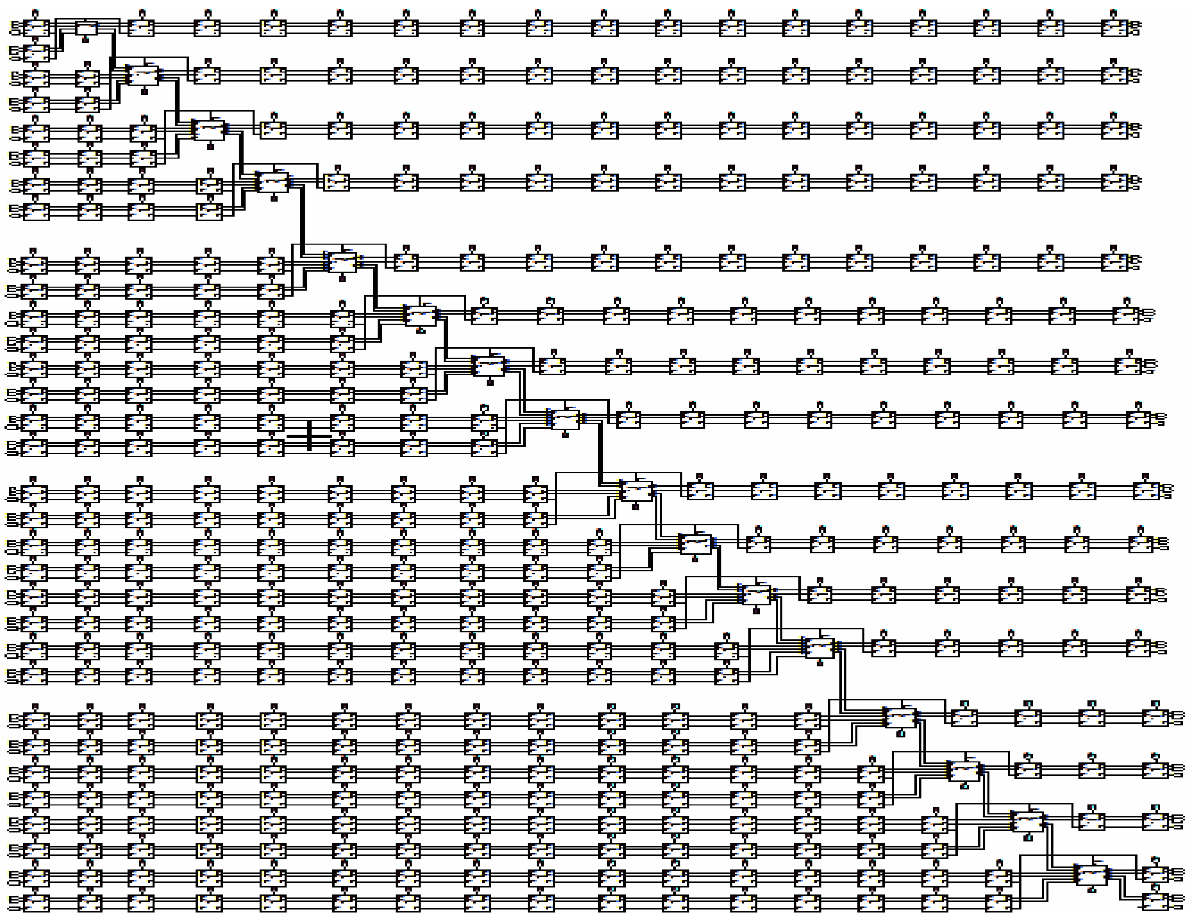


Figure 15. Asynchronous Pipelined 16-Bit Carry Propagation Adder.

Asynchronous pipelined 16-bit carry propagation adder is shown in Figure 15. Latency of the pipelined asynchronous adder is 72 ns.

3.5. Asynchronous pipelined multiplier

The design methodology in this paper is targets fine grain pipelining and small cells [11, 12], where the forward latency is two gate delays. Fine grain pipelining is achieved by dividing the processing blocks to even smaller cells where each cell has its own input and output completion detector. An 8 bit multiplier was implemented by using an 8 bit input completion detector at the inputs and a 16 bit output completion detector at the outputs. When the multiplier completes it processing and generates a 16 bit output, the output completion detector detects it and combined with the input completion detector generates the acknowledgement. The multiplier will accept a new input only after the first stage of the multiplier has finished processing. Therefore the throughput is limited to how fast the multiplier can multiply two numbers, generate the acknowledgement and then reset. As in the synchronous case the throughput of the multiplier can be increased by further pipelining the multiplier. In asynchronous design, this can be done by constructing the multiplier using small number of cells such as adders and other logic gates which have their own input and output completion detectors. The multiplier accept new input as soon as the first row of logic in the multiplier has evaluated and reset but also simplifies the 16 bit completion detectors into 1 bit input and output completion detectors. Buffer is used for completion detection in this design. The acknowledge signal from the buffer is used to indicate completion of a bit.

Asynchronous pipelined 8X8 array multiplier is shown in Figure 16. Inputs and outputs are individually detected by ack signal from buffer. Latency of the pipelined asynchronous adder is 162 ns. The throughput of the pipeline is determined by the slowest stage.

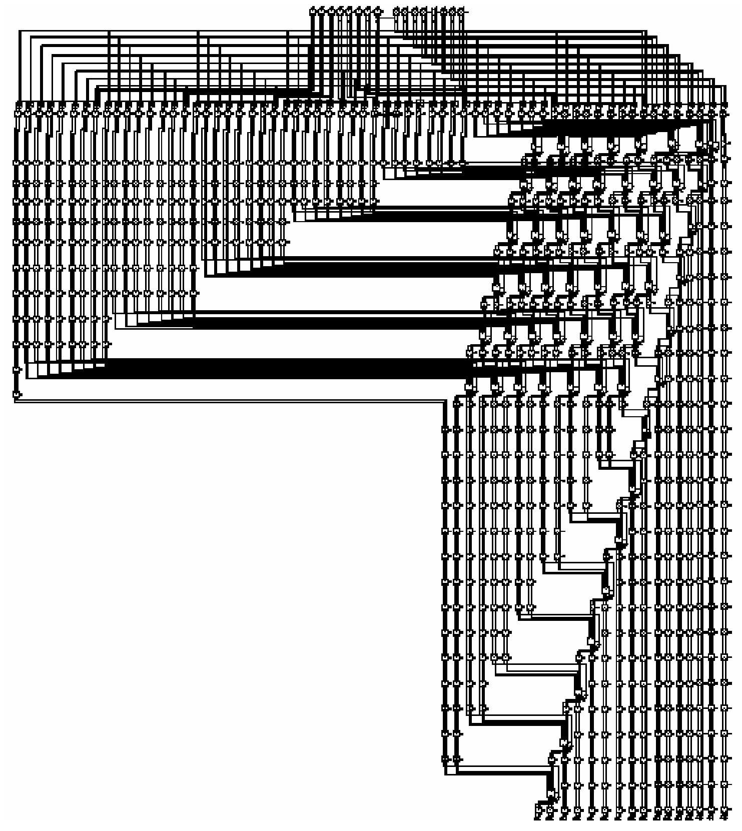


Figure 16. Asynchronous Pipelined 8X8 Array Multiplier

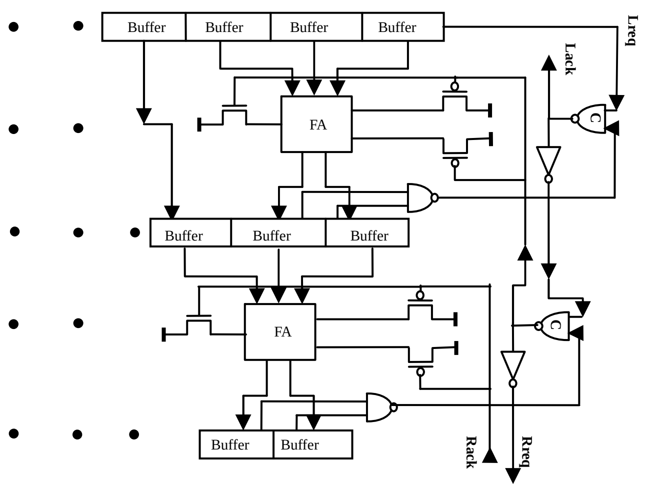
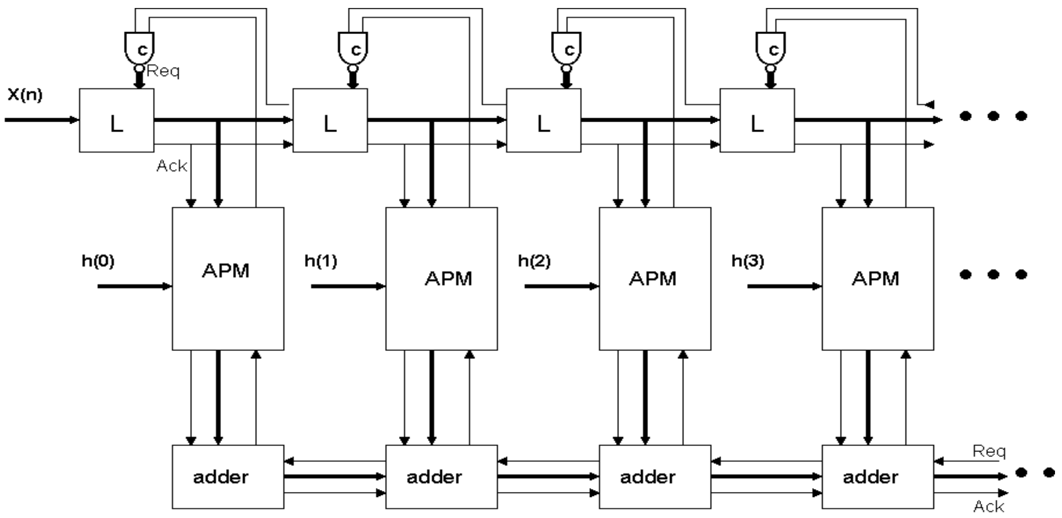


Figure 17. Detailed Diagram Of Asynchronous Pipelined Stage.

Detailed description diagram of pipelined asynchronous multiplier stage is shown in Figure 17.

3.6. Asynchronous pipelined FIR filter

Asynchronous pipelined FIR filter general design is shown in Figure 18. In the implementation of FIR filter, acknowledgement signals from latch and Asynchronous Pipelined Multiplier(APM) is combined and given to the previous latch. This join has been done by the c_element. The basic building blocks of the FIR filter are designed in TSMC 0.18-μm CMOS technology.



*APM- Asynchronous Pipelined Multiplier, L-Asynchronous Delay Buffer

Figure 18. Asynchronous Pipelined FIR Filter general design.

It is a linear pipeline. When adder completes the operation it requests the data from multiplier. If the data is available multiplier gives acknowledge signal to adder.

4. Results and discussion

The FIR Filter was designed using a 0.18-µm TSMC process with a 2.5V power supply. It is simulated using HSPICE and the existing synchronous design and the proposed asynchronous designs are analyzed based on the speed , area and Power consumption.

4.1. Asynchronous pipelined adder

An asynchronous 16-bit carry propagation adder using Reduced Stack Pre-Charged Half Buffer pipeline template have designed and simulated. Figure 19 show the output waveform of a first stage of the asynchronous pipelined 16-bit adder. This adder is designed using dual rail logic.

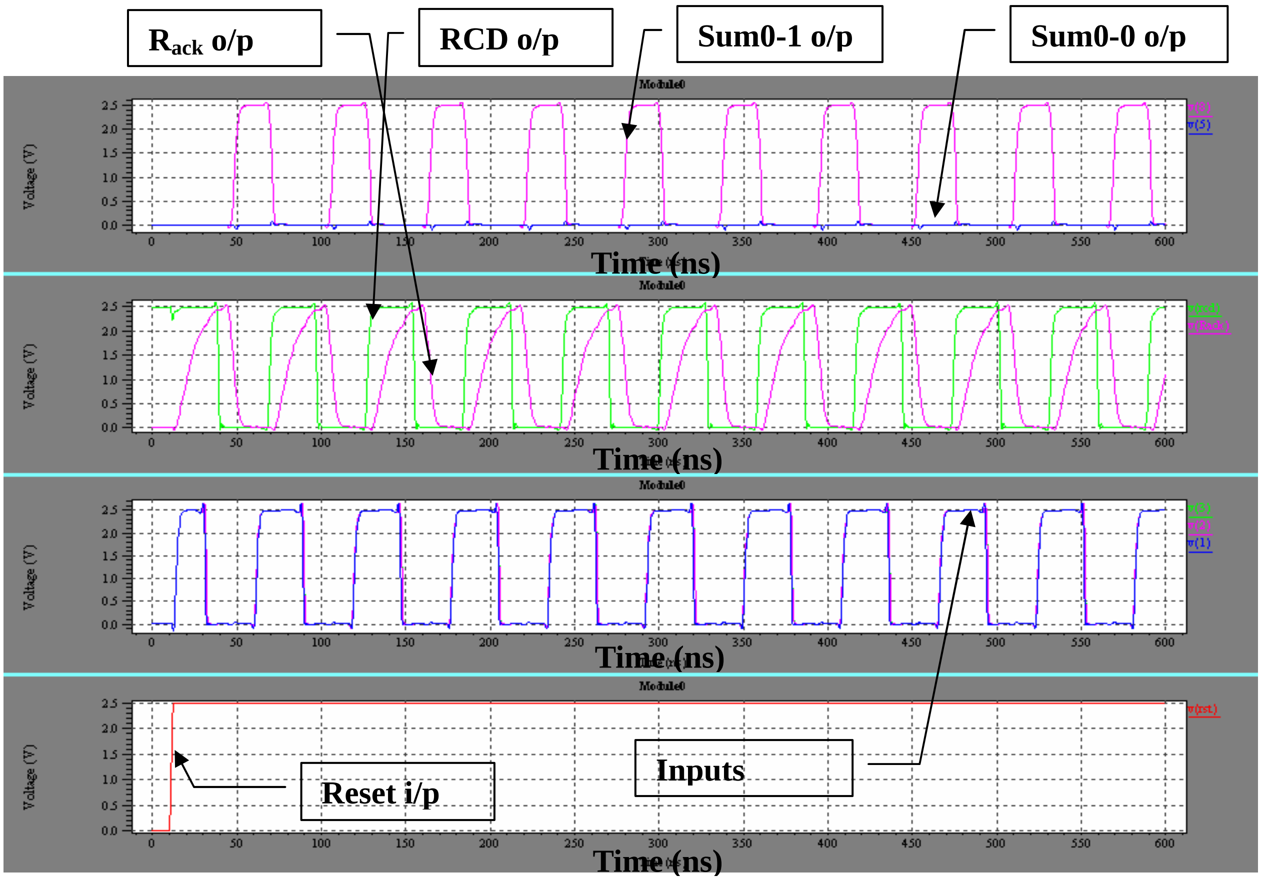


Figure 19. Asynchronous pipelined 16-bit Adder Stage 1 Output.

The Table 1 shows that the latency of the 15 stage pipelined asynchronous adder is only 72 ns compared to 800ns,the speed of proposed adder is 250 MHz compared to 20 MHz in synchronous design. Average power consumption of the asynchronous 16-bit carry propagation adder is 324 mW at 250 MHz. and 231 mW at 20 MHz. The speed has been increased without increase in power consumption

4.2. Asynchronous pipelined multiplier

An asynchronous 15 stage pipelined 8X8 array multiplier using Reduced Stack Pre-Charged Half Buffer pipeline template have designed using a 0.18- µm TSMC process with a 2.5V power supply and simulated. Figure 20 show the output waveform of asynchronous pipelined 8X8 multiplier.

Table 1. Performance comparison of the proposed adder.

parameter	Synchronous	Asynchronous
No. of transistor	5,978	9,563
speed	20 MHz	250 MHz
Latency	800 ns	72 ns
Avg. Power consumption	229 mW	324 mW

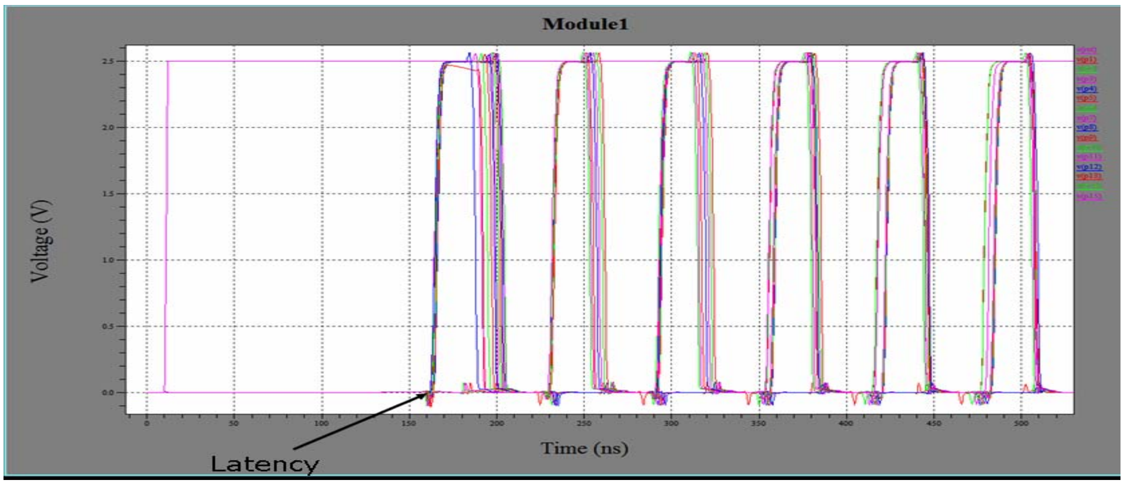


Figure 20. Asynchronous Pipelined Multiplier Output.

Table 2 gives the performance comparison between synchronous multiplier and proposed Asynchronous multiplier. It clearly shows that the latency of the pipelined asynchronous multiplier is 162 ns compared to 800 ns and speed is 250 MHz compared to 20 MHz than the existing synchronous counterpart. Average power consumption of the asynchronous 16-bit carry propagation adder is 517 mW at 250 MHz and when we compare the power consumption at the max operating speed of synchronous design, the proposed multiplier has 322mW and existing consumes 264 mW.

Table 2. Performance comparison Pipelined Multiplier

parameter	Synchronous	Asynchronous
No. of transistor	13,362	20,124
speed	20 MHz	250 MHz
Latency	800 ns	162 ns
Avg. Power consumption	264 mW	517 mW

4.3. Asynchronous pipelined FIR filter

4-tap FIR filter is designed using above asynchronous modules in 0.18-µm TSMC process with a 2.5V power supply . Latency of the pipelined asynchronous FIR filter is 372 ns shown in Figure 20. When it is operated at 20 MHz, average power consumption is 1.678 W.

Comparison of pipelined FIR filter is shown in Table 3. Asynchronous FIR filter shows improved performance up to 12 times and the latency decreases by 8.6 times with expense of is 2 times increase in area. The Power consumption is almost same at 20 MHz, which is the maximum operating speed of synchronous counterpart.

Table 3. Performance comparison of Pipelined FIR Filter.

parameter	Synchronous	Asynchronous(QDI)
No. of transistor	54,732	1,12,835
speed	20 MHz	250 MHz
Latency	3.2 us	372 ns
Avg. Power consumption	1.363 W	1.953 W (1.386 W @ 20 MHz)

5. CONCLUSION

New high-speed low-power architecture for the Digital FIR filter has been proposed. Asynchronous pipelined FIR filter is built using RSPCHB building blocks. Building blocks of asynchronous FIR filter like adder, multiplier and delay element are designed and compared with synchronous design. The advantage of proposed asynchronous pipelined FIR filer design over synchronous pipelined FIR design is speed and latency. The HSPICE simulation results demonstrate asynchronous design gives 12 times higher speed with almost same power consumption in the operating frequency range with the expense of 2 times increased area over synchronous design. Latency in the asynchronous design is greatly reduced by 8 times when compared to synchronous design.

References

[1] Jens Sparso. 2006. "*Asynchronous Circuit Design A Tutorial*", Technical University of Denmark.

[2] Di, J., Yuan, J. S. and DeMara, R. F. 2006. Improving power-awareness of pipelined array multipliers using 2-dimensional pipeline gating and its application to FIR design. *Integration the VLSI Journal*, 39(2):90-112.

[3] Recep, O. Ozdag., and Peter, A. Beerel. 2002. "*High speed QDI asynchronous pipelines,*" in Proc. ASYNC. 13-22.

[4] Fant. K., and Brandt. S. 1996. NULL Convention Logic a Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis, *Proc. IEEE International Conference Application Specific Systems, Architectures and Processors (ASAP 96),* 261-273.

[5] Recep, O. Ozdag, Peter, A. Beerel, 2006. An Asynchronous Low-PowerHigh-P erform ance Sequential Decoder Implemented With QDI Templates, *IEEE transactions on very large scale integration (VLSI) systems*, Vol.14, No.9.

[6] Recep, O. Ozdag. 2003. "*A template-ba sed standard-cell asynchronous design methodology,*" Ph.D. dissertation, Elect. Eng. Dept., Univ. Southern California, Los Angeles.

[7] Cheng, F. C. 1998. Practical Design and Performance Evaluation of Completion Detection Circuits, *Proceedings of International Conference on Computer Design: VLSI in Computers and Processors*, 354 -359.

[8] Singh. M., and Nowick. S. M. 2000. Fine-grain Pipelined Asynchronous Adders for High-Speed DSP Application, *IEEE Computer Society Annual Workshop VLSI*,111-118.

[9] Schuster. S., Reoher. M., Cook. P., Heidel. D., Immediato. M., and Jenkins, K. 2000. Asynchronous Integrated Pipelined CMOS Circuits Operating at 3.3-4.5 GHz, *Proc. ISSCC*, 292-293.

[10] Maitham Shams, Jo C. 1997. Ebergen, Mohamed I. Elmasry, Optimising CMOS Implimentation of the C-element, *IEEE International Conference on Computer Design (ICCD'97)*, 700-705.

[11] Singh. M., and Nowick. S. M. 2001. MOUSETRAP: Ultra-High-Speed Transition-Signaling Asynchronous pipelines, *Proc. ICCD 2001*, 9-17.

[12] Shojaee. K., Gholipour. M., Afzali-Kusha. Z., and Nourani.M. 2006. Comparative Study of Asynchronous Pipelined Design Methods, *EICE Electronics Express*, Vol.3, No.8, 163-171.

[13] Singh. M., and Nowick.S. M. 2000. High-Throughput Asynchronous Pipelines for Fine-Grain Dynamic Datapaths, *Proc. Int'l symp, Advanced Research in Asynchronous Circuits and Systems (ASYNC 2000)*, 198-209.