Low Power/ High Speed Design in VLSI with the application of Pipelining and Parallel processing

Champakamala B S
Dept. of M.Tech [VLSI Design & Embedded System],
Dr.A.I.T, Bangalore, India,
E-mail: bschampa@gmail.com

Shilpa K.C
Dr.A.I.T,Bangalore, India,
E-mail: shilpa.kc2@gmail.com

ABSTRACT

The main objectives of any VLSI design are Power, Delay and Area. Minimizing all the objectives is a challenge in present situation but all efforts to achieve one of these can lead to a better design. This paper proposes an EDA tool for low power/ high speed VLSI design, which solves any DFG to estimate the speed of operation and the percentage reduction in the power consumption using pipelining and parallel processing concepts.

Keywords

VLSI, power consumption, critical path, DFG, Unfolding.

1. INTRODUCTION

In an IC technology, as more and more complex functions are required in various data processing and telecommunications devices, the need to integrate these functions in a small system/package is also increasing. The levels of integration as measured by the number of logic gates in a monolithic chip has been steadily rising for almost three decades, mainly due to the rapid progress in processing technology and interconnect technology. The most important message here is that the logic complexity per chip has been (and still is) increasing exponentially. The monolithic integration of a large number of functions on a single chip usually provides Less area/volume and therefore, compactness, Less power consumption, Less testing requirements at system level, Higher reliability, mainly due to improved on-chip interconnects, Higher speed, due to significantly reduced interconnection length, Significant cost savings.

Most of the system level or circuit design is high performance and power optimization. For high performance system design, propagation delay minimization plays an important role. Basically size, cost, performance and power consumption are the crucial issues in low power portable battery operated system design.

The power dissipation and speed in a circuit present a trade-off, if we try to optimize on one, the other is affected. The choice between the two is determined by the way we chose the layout the circuit components. Layout can also affect the fabrication of VLSI chips, making it either easy or difficult to implement the components on the silicon.

To increase the speed or reducing power consumption pipeline and parallel processing are most appealing concept available which can easily apply on the chip level. Even if there is a difficulty to create the parallelism, unfolding provides transformation role to make it. [1, 2, 3].

2. PROBLEM FORMULATION

Design with low power consumption and also provide high speed is a challenge in the present situation. But all efforts to achieve one of these can lead to a better design.

3. VLSI DESIGN TECHNIQUES

Implementation of VLSI design algorithms includes high level architectural transformations. Pipelining , parallel processing, retiming, unfolding, folding and systolic array design methodologies plays an important role for optimized high performance design. There are numerous examples where we can deploy these concepts and some of them are Electrical Engineering, Microelectronics, Web search engines, Medical imaging and diagnosis. In this paper pipelining and parallel processing concepts [4] are used to build an EDA tool to compute low power consumption of any DFG (Data Flow Graph).

3.1 Pipelining

Pipelining transformation leads to a reduction in the critical path, which can be exploited to either increase the clock speed or sample speed or to reduce power consumption at same speed. Pipelining reduces the effective critical path by introducing pipelining latches along the data path. Pipelining has been used in the context of architecture design and compiler synthesis etc.

Consider the simple structure in Fig 1(a) where computation time of the critical path is $2T_A$. Fig 1(b) shows the 2-level pipelined structure, where 1 latch is placed between the 2 adders and hence the critical path is reduced by half.

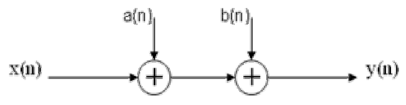

Fig 1 (a): A Data path

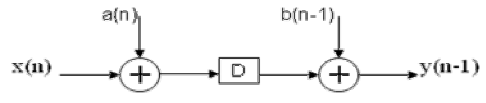

Fig 1 (b): The 2-level pipelined structure of (a)

Consider the three-tap finite impulse response (FIR) digital filter as shown in Fig 2(a)

$$y(n) = ax(n) + bx(n-1) + cx(n-2) \qquad (1)$$

Critical path or the minimum time required for processing a new sample is limited by 1 multiply and 2 add times, i.e., if T_M is the time taken for multiplication and T_A is the time needed for addition operation then the "sample period" (T_{sample}) is given by

$$T_{sample} \geq T_M + 2T_A$$

Therefore, the sampling frequency (f_{sample}) (also referred to as the throughput or the iteration rate) is given by

$$f_{sample} \leq \frac{1}{T_M + 2T_A}$$

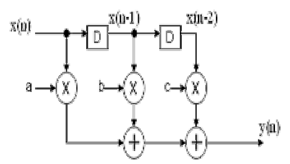

Fig 2 (a): A 3- tap FIR filter

Consider the pipelined implementation of the 3-tap FIR filter of Fig 2(a) obtained by introducing 2 additional latches is as shown in Fig 2(b).

The critical path is now reduced from $T_M + 2T_A$ to $T_M + T_A$. In this arrangement while the left adder initiates the computation of the current iteration the right adder is completing the computation of the previous iteration result. The schedule of events for this pipelined system is shown in the Table 1. In this system, at any time, 2 consecutive outputs are computed in an interleaved manner.

In M-level pipelined system, the number of delay elements in any path from input to output is $(M-1)$ greater than that in the same path in the original sequential circuit. While pipelining reduces the critical path, it leads to a penalty in terms of an increase in latency. Latency essentially is the difference in the availability of the first output data in the pipelined system and the sequential system. For example if latency is 1 clock cycle then k-th output is available in $(k+1)$-th clock cycle in a 1-stage pipelined system.

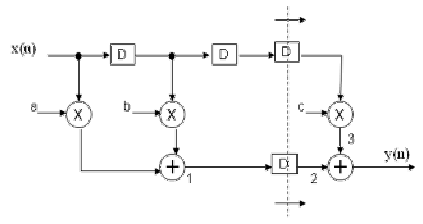

Fig 2 (b): Pipelined FIR filter. (The dashed vertical line represents a feed-forward cutset)

Table 1: Schedule of events in the Pipelined FIR filter

Clock	Input	Node 1	Node 2	Node 3	Output
0	$x(0)$	$ax(0)+bx(-1)$	–	–	–
1	$x(1)$	$ax(1)+bx(0)$	$ax(0)+bx(-1)$	$cx(-2)$	$y(0)$
2	$x(2)$	$ax(2)+bx(1)$	$ax(1)+bx(0)$	$cx(-1)$	$y(1)$
3	$x(3)$	$ax(3)+bx(2)$	$ax(2)+bx(1)$	$cx(0)$	$y(2)$

The following points may be noted:

The speed of architecture (or the clock period) is limited by the longest path between any 2 latches or between an input and a latch or between a latch and an output or between the input and output.

This longest path or the "critical path" can be reduced by suitably placing the pipelining latches in the architecture.

The pipelining latches can only be placed across any feed-forward cutest of the graph.

Cutset: A cutest is a set of edges of a graph such that if these edges are removed from the graph, the graph becomes disjoint.

Feed-forward Cutset: A cutset is called a feed-forward cutset if the data move in the forward direction on all the edges of the cutset. For example, the cutset used to pipeline the FIR filter in Fig 2(b) is a feed forward cutest.

We can arbitrarily place latches on a feed-forward cutest of any FIR filter structure without affecting the functionality of the algorithm.

3.2 Parallel Processing

It is of interest to note that parallel processing and pipelining techniques are duals of each other, and if a computation can be pipelined, it can also be processed in parallel. Both techniques exploit concurrency available in the computation in different ways. While independent sets of computations are computed in an interleaved manner in a pipelined system, they are computed using duplicate hardware in parallel processing mode.

For example consider the 3-tap FIR filter described by (Fig 2(a)). This system is a single-input single-output (SISO) system and is described by

$$y(n) = ax(n) + bx(n-1) + cx(n-2)$$

To obtain a parallel processing structure, the SISO system must be converted into a MIMO (multiple-input multiple-output) system. For example, the following set of equations describe a parallel system with 3 inputs per clock cycle (i.e., level of parallel processing L=3)

$$y(3k) = ax(3k) + bx(3k-1) + cx(3k-2)$$

$$y(3k+1) = ax(3k+1) + bx(3k) + cx(3k-1)$$

$$y(3k+2) = ax(3k+2) + bx(3k+1) + cx(3k)$$

Here k denotes the clock cycle. As can be seen, at the k-th clock cycle the 3 inputs $x(3k), x(3k+1)$ and $x(3k+2)$ are processed and 3 samples are generated at the output. Parallel processing systems are also referred to as block processing systems and the number of inputs processed in a clock cycle is referred to as block size. Because of the MIMO structure, placing a latch at any line produces an effective delay of L clock cycles at the sample rate. Each delay element is referred to as a block delay (also referred to as L-slow). For example, delaying the signal $x(3k)$ by 1 clock cycle would result in $x(3k-3)$ instead of $x(3k-1)$ which has been input in another input line. The block architecture for a 3-parallel FIR filter is shown in Fig 3(a) and its details are shown in Fig 3(b).

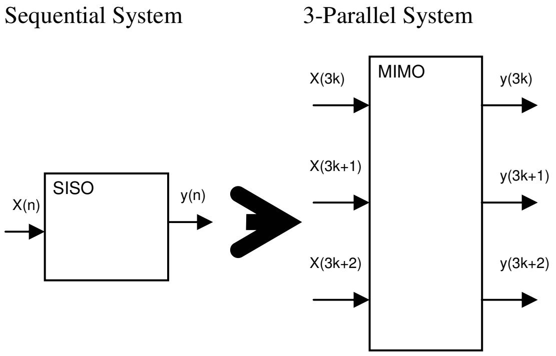

Fig 3 (a): A block processing example

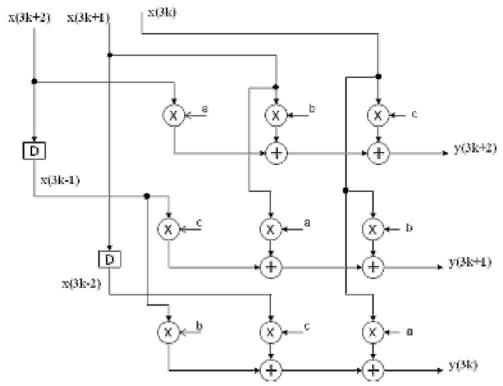

Fig 3 (b): Parallel processing architecture for a 3-tap FIR filter with block size 3

Note that the critical path of the block or parallel processing system has remained unchanged and the clock period (T_{clk}) must satisfy $T_{clk} \geq T_M + 2T_A$

But since 3 samples are processed in 1 clock cycle instead of 3, the iteration period is given by

$$T_{iter} = T_{sample} = \frac{1}{L} T_{clk} \geq \frac{1}{3}(T_M + 2T_A)$$

It is important to understand that in a parallel system $T_{clk} \neq T_{sample}$ whereas in a pipelined system $T_{clk} = T_{sample}$. Fig 3(c) shows the complete parallel processing system including serial-to-parallel and parallel-to-serial converters.

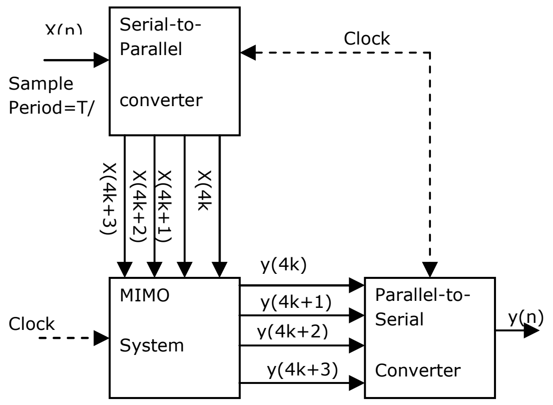

Fig 3 (c): Complete parallel processing system with block size 4

3.3 Unfolding

Unfolding is a transformation technique that can be applied to a DSP program to create a new program describing more than one iteration of the original program. More specifically, unfolding a DSP program by the unfolding factor J creates a new program that describes J consecutive iterations of the original program. Unfolding is also referred to as loop unrolling and has been used in compiler theory [5]. For example, consider the DSP program

$$y(n) = ay(n-9) + x(n) \qquad (1)$$

For n = 0 to ∞. Replacing the index n with 2k results in $y(2k) = ay(2k-9) + x(2k)$ for k = 0 to ∞. Similarly, replacing the index n with 2k + 1 results in $y(2k+1) = ay(2k-8) + x(2k+1)$ for k = 0 to ∞. Together, the 2 equations

$$y(2k) = ay(2k-9) + x(2k)$$

$$y(2k+1) = ay(2k-8) + x(2k+1) \qquad (2)$$

Describe the same program as (1). The program in (2) describes 2 consecutive iterations of the program in (1). For example, for k = 7, the equations in (2) are

$y(14) = ay(5) + x(14)$ and $y(15) = ay(6) + x(15)$. These equations are the same as those described in (1) for the 2 consecutive iterations n = 14 and n = 15. The program in (2) describes a 2-unfolded version of the program in (1).

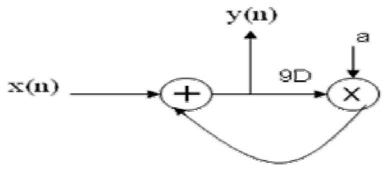

Fig 4 (a): The original DSP program describing $y(n) = ay(n-9) + x(n)$ **for n=0 to ∞**

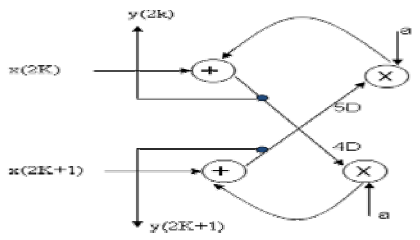

Fig 4 (b): The 2-unfolded DSP program describing

$y(2k) = ay(2k-9) + x(2k)$ **and** $y(2k+1) = ay(2k-8) + x(2k+1)$ **for k = 0 to ∞.**

In unfolded systems, each delay is J-slow. This means that if the input to a delay element

is the signal $x(kJ+m)$, the output of the delay element is

$x((k-1)J+m) = x(kJ+m-J)$.

The 2-unfolded program corresponding to the equations in (2) is shown in Fig 4 (b). Using the concept of J-slow delays, Fig 4 (b) describes the computations $y(2k) = ay(2(k-5)+1) + x(2k)$ and, $y(2k+1) = ay(2(k-4)+0) + x(2k+1)$, as desired.

The 2-unfolded program shown in Fig 4 (b) was created by substituting n = 2k and

n = 2k + 1 into (1). While this technique results in a correct 2-unfolded program, it can often be tedious to write the equations for the original and J- unfolded programs and then draw the corresponding unfolded data-flow graph (DFG), especially for larger values of J. This chapter describes a graph- based technique for directly unfolding the DFG to create the DFG of the J- unfolded program without explicitly writing the equations describing the original or unfolded program.

Unfolding has applications in designing high-speed and low-power VLSI architectures. One application is to unfold the program to reveal hidden concurrencies so that the program can be scheduled to a smaller iteration period, thus increasing the throughput of the implementation. Another application is to design parallel architectures at the word level and bit level. At the word level, these architectures can be used to design word-parallel architectures from word-serial architectures to increase the throughput or decrease the power consumption of the implementation. At the bit-level, unfolding can be used to design bit-parallel and digit-serial architectures from bit-serial architectures to achieve the same objectives as word-level unfolding.

4. ANALYSIS OF PIPELINING AND PARALLEL PROCESSING FOR POWER CONSUMPTION REDUCTION

It is of interest to note that parallel processing and pipelining techniques are duals of each other, and if a computation can be pipelined, it can also be processed in parallel. Both techniques exploit concurrency available in the computation in different ways. While independent sets of computations are computed in an interleaved manner in a pipelined system, they are computed using duplicate hardware in parallel processing mode. Unfolding has applications in designing high-speed and low-power VLSI architectures.

4.1 Pipelining for Low Power

Two main advantages of using Pipelining are

- Higher Speed
- Lower Power

Pipelining can increase the sample speed. This technique can be used for lowering the power consumption where sample speed does not need to be increased. The propagation delay T_{pd} is associated with charging and discharging of the various gate and stray capacitance in the critical path [6]. For CMOS circuits, the propagation delay can be written as

$$T_{pd} = \frac{C_{charge}V_0}{k(V_0 - V_t)^2} \qquad (1)$$

Where C_{charge} denotes the capacitance to be charged/discharged in a single clock cycle, i.e., the capacitance along the critical path, V_0 is the supply voltage and V_t is the threshold voltage. Parameter k is a function of technology parameter μ, $\frac{W}{L}$ and C_{ox}. The power consumption of a CMOS circuit can be estimated using the following equation,

C_{total}

$$P = C_{total}\ V_0^2 f \qquad (2)$$

Where denotes the total capacitance of the circuit, V_0 is the supply voltage, and f is the clock frequency of the circuit. The above two equations are based on simple approximations and are appropriate only for a 1st-order analysis. Let

$$P_{seq} = C_{total}\ V_0^2 f \qquad (3)$$

Represent the power consumed in the original filter. It should be noted that $f = \frac{1}{T_{seq}}$ where T_{seq} is the clock period of the original sequential filter. Now consider an M-level pipelined system, where the critical path is reduced to $\frac{1}{M}$ of its original

length and the capacitance to be charged/discharged in a single clock cycle is reduced to $\frac{C_{charge}}{M}$. Notice that the total capacitance does not change. If the same clock speed is

maintained, i.e., the clock frequency f is maintained, only a fraction of the original capacitance, $\frac{Ccharge}{M}$, is being charged/discharged in the same amount of time that was previously needed to charge/discharge the capacitance, C_{charge} (see Fig 5(a)). This implies, then, that the supply voltage can be reduced to βV_0, where β is a positive constant less than 1. Hence the power consumption of the pipelined filter will be

$$P_{pip} = C_{total}\beta^2 V_0^2 f = \beta^2 P_{seq} \quad (4)$$

Therefore, the power consumption of the pipelined system has been reduced by a factor of β^2 as compared with the original system.

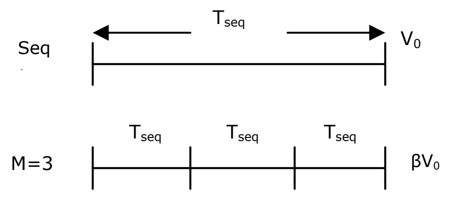

Fig 5 (a) Critical path lengths for original and 3-level pipelined systems

The power consumption reduction factor, β can be determined by examining the relationship between the propagation delay of the original filter and the pipelined filter. The propagation delay of the original filter is given by

$$T_{seq} = \frac{C_{charge}V_0}{k(V_0 - V_t)^2} \quad (5)$$

The propagation delay of the pipelined filter is given by

$$T_{pip} = \frac{\frac{C_{charge}}{M}\beta V_0}{k(\beta V_0 - V_t)^2} \quad (6)$$

It should be noted that clock period T_{clk} is usually set equal to the maximum propagation delay T_{pd} in a circuit. Since the same clock speed is maintained for both filters, from the equations 3 and 4 the following quadratic equation can be obtained to solve for β. $M(\beta V_0 - V_t)^2 = \beta(V_0 - V_t)^2$

Once β is obtained, the reduced power consumption of the pipelined filter can be computed using equation 4.

4.2 Parallel Processing for low power

Parallel processing like pipelining can reduce the power consumption of a system by allowing the supply voltage to be reduced. In an L-parallel system, the charging capacitance does not usually change while the total capacitance is increased by L times. In order to maintain the same sample rate, the clock period of the L-parallel circuit must be increased to LT_{seq}, here T_{seq} is the propagation delay of the sequential circuit given by (equation 3). This means that C_{charge} is charged in time LT_{seq} rather than in time T_{seq}. In other words, there is more time to charge the same capacitance (see Fig 5(b)). This means that supply voltage can be reduced to βV_0.

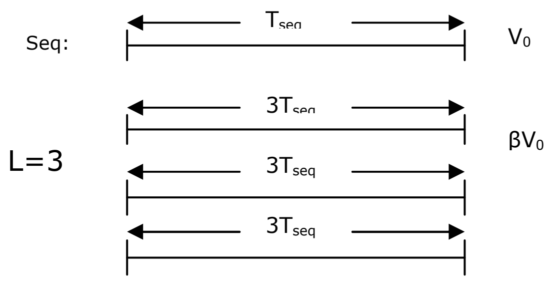

Fig 5 (b): Critical path lengths for sequential and 3-parallel systems

The propagation delay considerations can again be used to compute the supply voltage of the L-parallel system. The propagation delay of the original system is given by (5), while the propagation delay of the L-parallel system is given by

$$LT_{seq} = \frac{C_{charge}\,\beta V_0}{k(\beta V_0 - V_t)^2} \quad (7)$$

From equations 5 and 7, the following quadratic equation can be used to compute β

$$L(\beta V_0 - V_T)^2 = \beta(V_0 - V_T)^2 \; \beta\,(\beta V_0 - V_T)^2 = \beta(V_0 - V_T)^2$$

Once β is computed, the power consumption of the L-parallel system can be calculated as

$$P_{par} = (LC_{charge})(\beta V_0)^2 \frac{f}{L}$$

$$= \beta^2 C_{charge} V_0^2 f$$

$$= \beta^2 P_{seq}$$

Where P_{seq} is power consumption of the sequential system given by (equation 3). Therefore as in the pipelined system, the power consumption of the L-parallel system has been reduced by a factor of β^2 as compared with the original sequential system.

4.3 Combining Pipelining and Parallel Processing

The techniques of pipelining and parallel processing can be combined for lower power consumption. The principles remain the same, i.e., pipelining reduces the capacitance to be charged/ discharged in 1 clock period and parallel processing increases the clock period for charging/discharging the original capacitance (see Fig 5(c)).

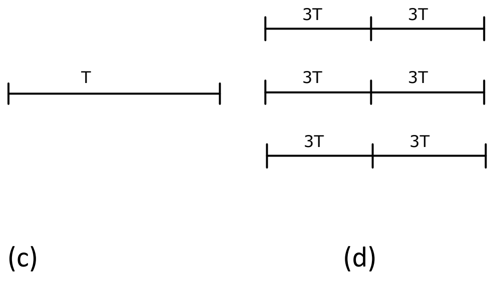

Fig 5 (c) Charging/discharging of entire capacitance in clock period T. 5 (b) Charging/discharging of capacitance in clock period 3T using a 3-parallel filter and 2 stages of pipelining

The propagation delay of the parallel-pipelined filter is obtained as follows

$$LT_{pd} = \frac{\left(C_{charge}/_{M}\right)\beta V_0}{k(\beta V_0 - V_t)^2} = \frac{LC_{charge}V_0}{k(V_0 - V_t)^2}$$

Using this equation, the following quadratic equation is obtained

$$ML(\beta V_0 - V_t)^2 = \beta(V_0 - V_t)^2$$

5. EXPERIMENTAL SETUP

This VLSI design is developed on MATLAB platform. MATLAB is an efficient platform that has lots of library functions, graphical facilities, GUI, platform independent, factorization, high speed of execution and many more.

For a given DFG, it computes the critical path with and without applying pipelining technique. By comparing the critical path of original circuit with pipelined circuit, it is reduced by half after pipelining and hence higher speed is achieved. Required power consumption in pipelined system compared to non pipelined system can be achieved by taking the finegrain pipelined version of the circuit with related equations. Similarly required power consumption in parallel processing system compared to non parallel processing system is computed with the help of certain assumption and equations. Required power consumption in combined pipeline and parallel system is computed with certain assumptions and equations.

Pipelining reduce the length of critical path and hence the reduction in capacitance over the critical path which in turn reduce the power consumption. Parallelism increase the number of samples within the same time period and hence more time available to charge or discharge the capacitor. To reduce the power consumption the voltage is reduced, in result with extended value of time for charging or discharging. If voltage is reduced by a k factor in effect power consumption is reduced by the square factor [8].

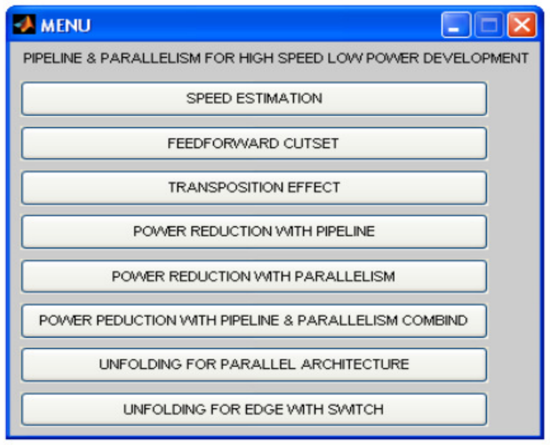
MENU
PIPELINE & PARALLELISM FOR HIGH SPEED LOW POWER DEVELOPMENT
SPEED ESTIMATION
FEEDFORWARD CUTSET
TRANSPOSITION EFFECT
POWER REDUCTION WITH PIPELINE
POWER REDUCTION WITH PARALLELISM
POWER PEDUCTION WITH PIPELINE & PARALLELISM COMBIND
UNFOLDING FOR PARALLEL ARCHITECTURE
UNFOLDING FOR EDGE WITH SWITCH

Figure: 6. Implementation order snapshot

6. CONCLUSION

The current challenge of VLSI in terms of the speed and power has taken the research of the presented work. The two effective means namely pipelining and parallel processing has taken to meet these challenges.

Pipelining reduce the length of the critical path while placing the delay element in between. Feed forward cutest, transposition, fine-grain division of modules are the fundamental method under the pipelining. Parallelism takes the more number of samples in the same cycle to increase the speed, power consumption has been reduced with the concept of reducing the effective capacitance in the critical path or providing more time for charging or discharging.

7. ACKNOWLEDGMENTS

The author would like to thank Mrs. Shilpa K C for her assistance, implementations, insight and valuable discussion over the course of this project.
Parts of this work have been supported by Dr.Ambedkar Institute of Technology, Bangalore and also our beloved friends who lent a helping hand in the success of this project.

8. REFERENCES

[1] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou. " Precomputation- based sequential logic optimization for low power. " In Proceedings of the 1994International Workshop on Low Power Design, pages 57-62, April 1994.

[2] W.C.Athas, L. J. Svensson, J.G.Koller, N.Thartzanis and E. Chou. " Low-Power Digital Systems Based on Adiabatic-Switching Principles. " IEEE Transactions on VLSI Systems, 2(4)398-407:, December 1994

[3] H. Bakoglu, Circuits, Interconnections, and Packaging for VLSI, Addison-Wesley, Menlo Park, CA, 1990.

[4] K.K.Parhi, “VLSI Digital Signal Processing systems-Design and Implementation”,Wiley Student Edition.

[5] Miodrag Potkonjak ,Jan Rabaey , “Maximally Fast and Arbitrarily Fast Implementation of Linear Computations”, 1992 IEEE proceedings.

[6] K. Martin, “A high-speed, high-accuracy pipeline A/D converter, “ Proc. Asilomar Conf. Circuits Syst., Computers,” pp. 489-492, 1981.

[7] R. I. Hartley and K. K. Parhi, Digit-Serial Computation. Boston, MA: Kluwer, 1995.

[8] B. Davari, R. H. Dennard and G. G. Shahidi. " CMOS scaling for high performance and low power. " Proceedings of IEEE, 83(4):408-425, April 1995.