International Journal of Electronics Engineering, 2(1), 2010, pp. 213-217

Analysis of Various Algorithms for Low Power Consumption in Embedded System using Different Architecture

Mandeep Kaur[1] & Vikas Sharma[2]

[1]*Lecturer in CIET, Rajpura*
[2]*Design Engineer, RF Silicon, Noida*

Abstract: Concurrent processors allow multiple processes to be executed simultaneously in other words, a concurrent program consists of several sequential programs to be executed in parallel. Concurrent processing is a computing model in which multiple processors execute instructions for better performance. Here the tasks are broken in to the subtasks that are assigned to separate processors to perform simultaneously. this will lead to increase in computation speed and increase in the complexity of the programming language and hardware and low in power consumption. The algorithms used in concurrent processing are generally represented in data flow graph model. the DFG is represented in terms of nodes and edges and then represented in intra-iteration and inter –iteration. retiming is a transformation technique used to change the location of delays elements in a circuit without affecting input/output characteristics of the circuit. retiming techniques include the cutest retiming and pipelining. pipelining is done through forward cutset method . unfolding is used to design parallel processing architecture from the serial processing architecture. Pipelining and unfolding techniques are used for the low power consumption and high speed. Look–ahead algorithm represents the pipelining and parallelism in recursive filters that leads to canceling the poles and zeros and hence results in making a stable filter.

1. INTRODUCTION

A sequential computer program consists of a series of instructions to be executed one after another. A concurrent program consists of several sequential programs to be executed in parallel. Each of the concurrently executing sequential programs is called a process. Concurrent process execution is simulated by randomly interleaving instructions of the sequential programs. All processes have access to a common pool of data. In contrast, multiprocessing systems have several processing units and one memory bank. Processes are executed in parallel on the separate processing units while sharing common data. Concurrent programming is also used when several computers are joined in a network. An airline reservation system is one example of concurrent processing on a distributed network of computers. A simple example of a task that can be performed more efficiently by concurrent processing is a program to calculate the sum of a large list of numbers [11]. Concurrent processors provide the computing power needed to solve many computations - intensive problem. A major challenging in these supercomputers lies in the partitioning the task of an algorithm in away that leads to better processor utilization (or reduced idle time). These algorithm or program posses more concurrency. Algorithm techniques lead to better processor utilization in the programmable supercomputer and dedicated processor. Algorithm transformation increases the level of concurrency of the implementation while preserving the input – output behavior of the original algorithm.

*Corresponding Author: ermandeep0@gmail.com[1], vikassharma.tu@gmail.com[2]

DATA FLOW GRAPH

Many DSP algorithm used in digital filters which contain feedback loops , which imposes inherent fundamental lower bound on the achievable iteration or sample period. This bound is referred to as the iteration period bound or iteration bound. The iteration bound is a characteristic of representation of an algorithm in the data flow graph (DFG). Data–flow representation of algorithm exhibits the available concurrency and forms a natural basis for a program specification in a multiprocessor environment. Consider the following program

Program 1	Program 2
for each $\{n = 1$ to $\infty\}$	for each $\{n = 1$ to $M\}$
$Z(n) = x(n) + y(n)$	$Z(n) = x(n) + y(n)$

Program 1 computes $z(1)$ through $z(M)$ and terminates after executing the loop M times. This is a terminating program. Program 2 operates on infinite time series $\{x(n)\}$ and $\{y(n)\}$ and computes the infinite time series $\{z(n)\}$. Program 2 runs forever and is a non-terminating program. The *for each* statement in program 1 and program 2 indicates sequential processing of the loop w.r.t loop index n. Each execution of the loop in the non-terminating program is referred to as iteration. The time required to perform each iteration is the iteration period or sample period to indicate the infinite time series $\{x(n)\}$ and $\{y(n)\}$ are sampled periodically at this time interval. Real–time

and signal -image processing programs are characterized by non –terminating program model. DSP programs are non-terminating programs that run from time index $n = 0$ to $n = \infty$. The input to this DSP program is the sequence $x(n)$ for n-0,1,2…and the initial condition $y(-1)$. The output is the sequence $y(n)$ for $n = 0, 1, 2\ldots$. A DSP program is often represented using a DFG, which is a directed graph (each has distinct direction) that describes the above program. For eg:

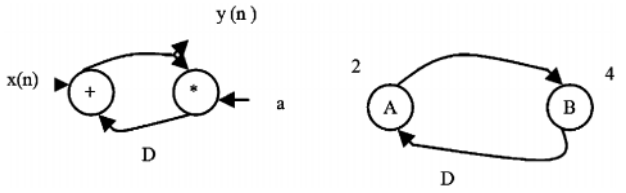


Fig (1a, 1b): The Graphical Representation of a DFG of the Program

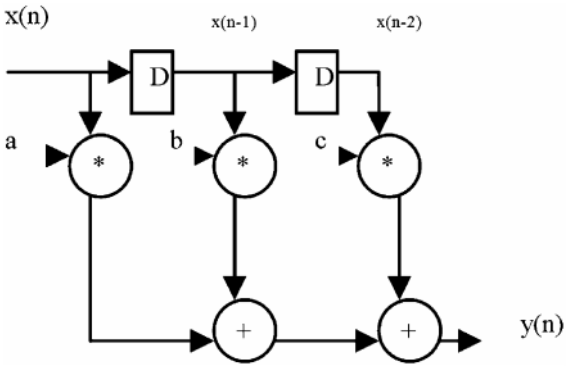
The structure of the DFG consists of a set of nodes and edges. The node represents the task or computations i.e. node A represents the addition and node B represents the multiplication and each node has the execution time with it. The edge represents the communication between the nodes and each node has non- negative number of delays associated with it. In this example, the edge $A \rightarrow B$ has zero delay with it and the edge $B \rightarrow A$ has one delay. An iteration of a node is the execution of the node exactly once, and the iteration of the DFG is the execution of each node in the DFG exactly once. [2]. Let X_k denotes the k-th iteration of the node X. It is important to distinguish between the intra-iteration and the inter-iteration precedence constraints, an intra-iteration precedence constraint is represented by a single arrow i.e. $A_k \rightarrow B_k$ and inter-iteration precedence constraint is represented by double arrow i.e. $B_k => A_{k+1}$. The critical path of a DFG is defined to be the path with the longest computation time among all the paths that contain zero delays. The critical path in above fig is 6 u.t. from $A \rightarrow B$. The critical path is the longest path for combinational rippling in the DFG, so the computation time of the critical path is the minimum computation time for one iteration of the DFG. A DFG is classified as recursive or non – recursive. A non- recursive DFG contains no loops, while the recursive DFG contains at least one loop.

PIPELINING AND PARALLEL PROCESSING

Consider the three tap FIR filter

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

The block diagram of above equation is



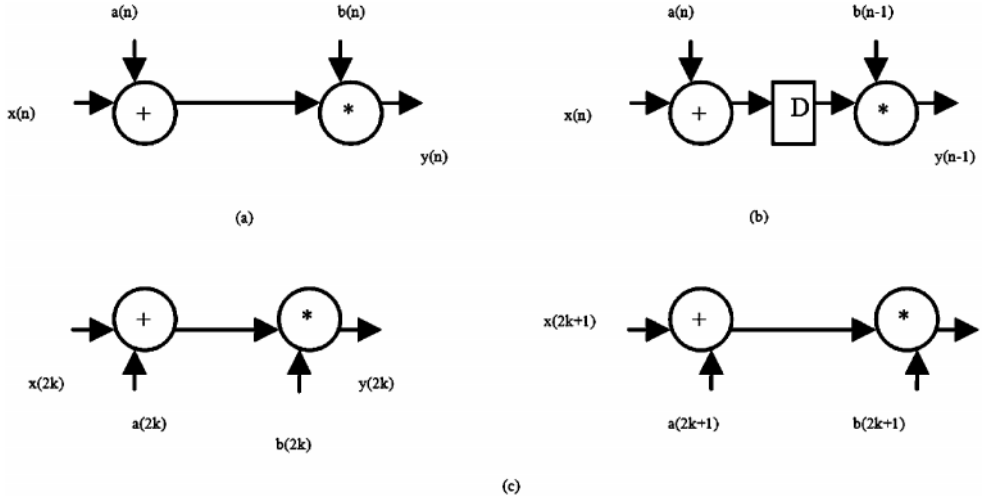
Here the critical path or the minimum time required for processing a new sample is limited by a 1 multiply and 2 add times i.e. if T_M is the time taken for multiplication and T_A is time needed for addition operation then the sample period is

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

Therefore, the sampling frequency is given as

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

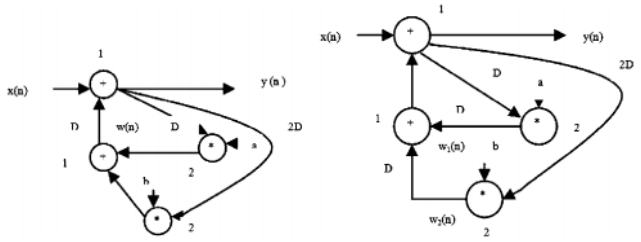
Note that the direct – form structure shown in fig is only be used only if it satisfies the equations. The effective critical path can be reduced by using either pipelining or parallel processing.Pipelining reduces the effective critical path by introducing the pipelining latches along the data path. Parallel processing increases the sampling rate by replicating the hardware so that several inputs can be processed in parallel and several outputs can be produced at the same time [26]. Consider the following example as shown in fig.



Here the computation time of the critical path is 2 T_A. Fig b shows the 2-level pipelined structure, where 1 latch is placed between two adders and hence critical path is reduced by half. Fig c shows 2-level parallel processing where the same hardware is duplicated so that 2 inputs can be processed at the same time and 2 outputs are produced simultaneously. Therefore, sample rate is increased by 2.

RETIMING

Retiming is a transformation technique used to change the locations of delay elements in a circuit without affecting the input / output characteristics of the circuit. Retiming can be used to increase the clock rate of a circuit by reducing the computation time of the critical path. Critical path is defined as the path with longest computation time among all the paths that contain zero delays, and the computation time of the critical path is the lower bound on the clock period of the circuit The critical path of the filter as in first figure (a) passes through 1 multiplier and 1 adder and has a computation time of 3. u.t., so this filter cannot be clocked with a clock period of less than 3 u.t. The retimed filter in another fig of (b) has a critical path that passes through two adders and has a computation time of 2 u.t and hence this filter can be clocked with a period of 2 u.t. By retiming the filter as shown in fig (a) in to fig (b), the clock period has been reduced from 3 u. t. to 2 u. t.



Retiming can be used to decrease the number of registers in a circuit. Retiming can be used to reduce the power consumption of a circuit by reducing switching, which can lead to dynamic power dissipation in static cmos circuits. [6,8].Placing register s at the inputs of nodes with large capacitances can reduce the switching activities at these nodes, which can lead to low power solutions. Cutset retiming is a useful technique that is a special case of retiming. Cutset retiming only affect the weights of the edges in the cutset. If the 2 disconnected sub graphs are labeled as *G*1 and *G*2, then cutset retiming consists of adding k delays to each edge from G1 to G2 and removing k delays from each edge from G2 to G1.

For example, a cutset is shown with a dashed line in fig (a). The 3 edges in the cutset are $2 \rightarrow 1$, $3 \rightarrow 2$, $1 \rightarrow 4$. The 2 sub graphs *G*1 and *G*2 found by removing the 3 edges in the cutset are shown in fig (b), for k=1, the result of cutset retiming is shown in fig (c). The edges from G1 to G2 are $3 \rightarrow 2$ and $1 \rightarrow 4$, and one delay is added to each of these edges. The edges from G2 to G1 are $2 \rightarrow 1$, and one delay is subtracted from this edge. [7,13]. Cutset retiming is a special case of retiming where each node in the sub graph G1 has retiming value j and each node in the sub graph G2 has the retiming value $j + k$.

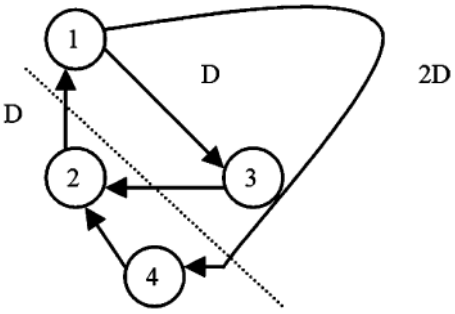
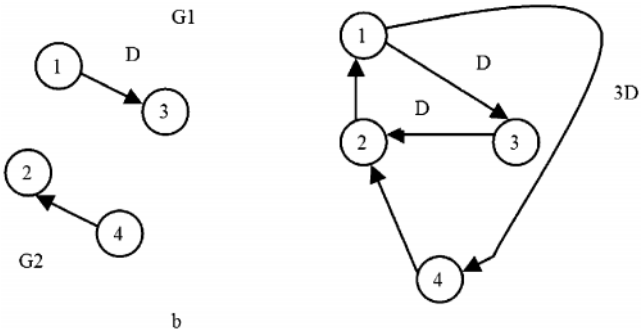
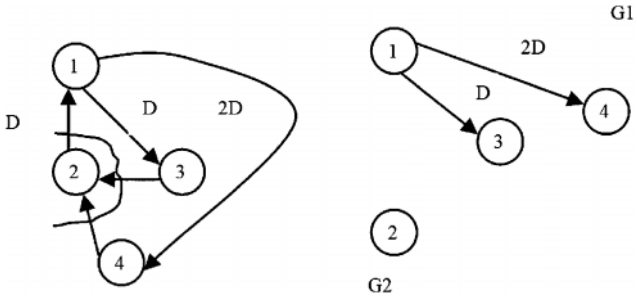


Figure a.



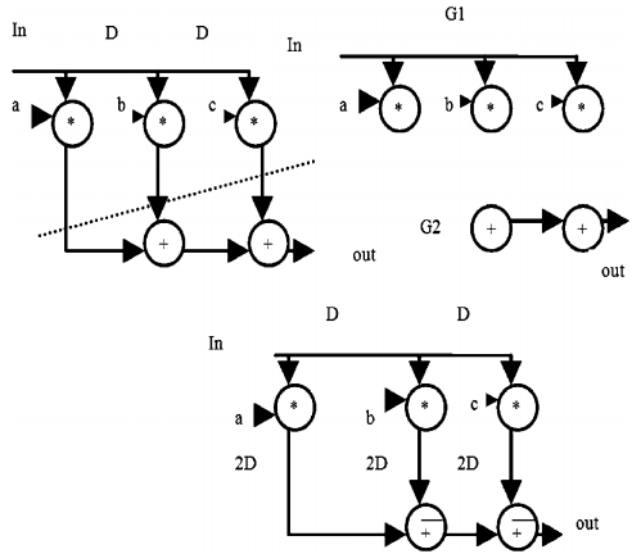
b

For e.g. : as shown in above fig, using the value s $j = 0$ and $k = 1$ result in $r(1) = 0$, $r(2) = 1$, $r(3) = 0$ and $r(4) = 1$ and this maps the DFG in fig (a) to the DFG in fig (c). Any value of j results in the same retimed graph. Another way of using cutset technique as follows:



Here cutset is so chosen so that the sub graph G2 is a single node and sub graph G1 is the rest of the graph minus the edges going into and out of the chosen node. The result of cutset retiming in this case is found by adding 1 delay to each edge incident into the node 2 and subtracting 1 delay from each delay from each edge outgoing from node 2 as shown in fig. Pipelining is a special case of cutset retiming where there are no edges in the cutset from the sub graph G2 to the sub graph G1, i.e. pipelining applies to graphs without loops. These cutset are referred to as feed-forward cutset. The feed – forward cutset shown in fig divides the graph in to 2 sub graph shown in fig given below. All 3 of

the edges in the cutset go from G1 to G2 and performing cutset retiming with k = 2 result in 2 additional delays on each edge in the cutset resulting in retimed graph.



A direct application of the general unfolding transformation is to design parallel processing architecture from the serial processing architecture. At the word level, it means that the word-parallel architecture can be designed from word-serial architecture. At the bit-level, it means it means that bit-parallel and digit-serial architecture can be designed from bit- serial architectures.

FOLDING

In synthesizing DSP architectures, it is important to minimize the silicon area of the integrated circuits, which is achieved by reducing the numbers of functional units (such as adders and multipliers), registers, multipliers and interconnected wires. The folding transformation is used to systematically determine the control circuits in DSP architectures where multiple algorithm operations (such as addition operations) are time-multiplexed to a single functional unit (such as pipelined adder). [9, 13].By executing multiple algorithm operations on a single functional unit, the number of functional units in the implementation is reduced, resulting in an integrated circuit with low silicon area. Folding technique can be used for synthesis of DSP architectures that can be operated using single or multiple clocks.

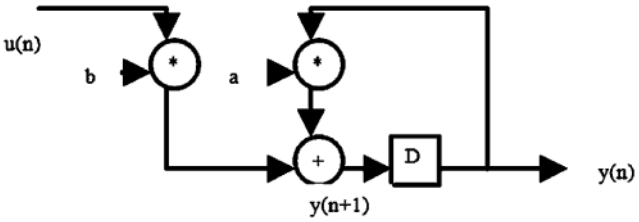
LOOK AHEAD ALGORITHM

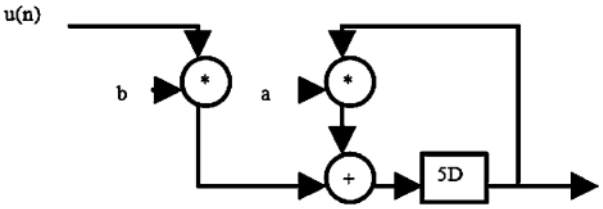
Inefficient single / multichannel interleaving

Consider a 1st order linear time – invariant recursion described by

$$y(n+1) = ay(n) + b\,u(n)$$

. . . . . T_M + T_A where $T_{M,\,TA}$ represent the word–level multiplication time and addition time.





Consider an M – stage pipelined version of this implementation obtained by inserting (M-1) additional latches inside in the loop. The clock period of this Implementation be reduced by M times, but the latency associated with the loop computation and the sample period of this implementation will increase toM clock Periods. As an example, M = 5, if we begin with a state $y^1(0)$ in clock period 0 , the next state $y1$ (1) wil lbe available in clock period 5. For the case of a single time this series, array will be useful for only 20 % of the time. Hence the sample rate of this implementation is five times lower than the clock rate, and is no higher than that of the unpipelined version. Thus, pipelined interleaving approach is well suited for the applications requiring nominal concurrency. To conclude, if a recursive loop with a single delay element is pipelined by M stages by inserting $(M-1)$ additional delay elements , then the input data must be M–way interleaved. This implementation is M–slow circuit. The slow hardware in this interleaved implementation is inefficiently utilized if M independent computations are not available to be interleaved.

CONCLUSION

These algorithm used for different architecture is to reduce the critical path and hence , increase the sampling rate. In parallel processing, multiple outputs are computed in a clock period. The effective sampling speed is increased by the level of parallelism. Parallel processing and pipelining are used for reducing the power consumption. Parallel processing increases the sampling rate by replicating the hardware so that several inputs can be processed in parallel and several outputs can be produced at the same time. Pipelining introduces latches in between path and reduces the critical path. Retiming include pipelining, feed forward cut-set which results in reducing the clock period of the circuit, reducing power consumption of the circuit. Folding is used to reduce the silicon area which results in reducing the numbers of functional units. Look –ahead algorithm includes the

additional concurrency. It will include pipelining interleaving which will result in canceling poles and zeros, and make the filter stable. As a result of this, numbers of operations are performed simultaneously in a concurrent processor. This leads to high speed and result in increase the performance of the processor.

REFERENCES

[1] Jerzy Tokarzewski "Output–nulling Subspaces, Zero Dynamic and Zeros in MIMO LTI Proper System", *Proceeding 18 th International Conference*, IEEE 2005.

[2] Yin Wang Zhiming Wu "Avoiding Unsafe States in Manufacturing System based on Polynomial Digraph Algorithm", *Proceeding International Conferences*, *IEEE* 2003.

[3] Ran Yang, Shiyang Qui and Guoli Wang "Periodic Look Ahead Filter Design for Pipelining 2D IIR Digital Filter" *IEEE* 2007.

[4] Dennis K. Y Tong, Evangeline, *f, y*. Young, Chris Chu, Sampath Dechu "Wire Retiming Problem with Net Topology Optimization" *IEEE,* **26**, SEPT 2007.

[5] Shih – hsu – huang , Yow – tyng – neih , Feng – pin –lu and Wei – chieh yu " Race – condition –aware –retiming", *IEEE* 2005.

[6] Jia Wang and Hai Zhou, "An Efficient Incremental Algorithm for Minimum Area Retiming", *IEEE*, July, 2008.

[7] Ingmar Neuumann, Wolfgang Kunz, "Placement Driven Retiming with Coupled Edge Timing Model", *IEEE* 2001.

[8] Jie–hong R. Jiang and Wie–lun Hung, "Inductive Equivalence Checking under Retiming and Resynthesis" *IEEE* 2007.

[9] Vijay Sundararajan, Keshab K. Parhi, "Synthesis of Low–power Folded Programmable Coefficient FIR Digital Filter" *IEEE* 2000.

[10] Marvi Teixeria "An Operator–based Approach to Unfolding Transformation", *IEEE*, 2008.

[11] Ishwar Parulkar, Thomas Ziaja, Rajesh Pendurkar, Anand D'souza, Amitava Majumdar, "A Scalable, Low Cost Design – for – test Architecture for Ultra Sparc Chip Multiprocessor" *IEEE* 2002.

[12] Tillie Tang "Parallel Sorting on the Hypercube Concurrent Processor", *IEEE* 1990.

[13] Lori E. Lucke, Andrew P. Brown, Keshab K. Parhi "Unfolding and Retiming for High–level Synthesis", *IEEE* 2000.