Design of low power reconfigurable DSP processor

Gurvinder Singh (03307915)
Supervisor: Prof. A. N. Chandorkar

Abstract: This paper addresses the problem of reducing power dissipation of digital signal processors basically used for implementing FIR and IIR filters and FFT algorithms. An approach is presented for minimizing power consumption for digital systems implemented in CMOS technology which involves optimization at all levels of the design. This optimization includes, at the highest level the algorithms that are being implemented, the architecture for implementing the circuit, the circuit topology and styles, and at the lowest level the technology used to implement the circuit. It describes a generic DSP architecture and identifies the main sources of power dissipation during FIR and IIR filtering and FFT algorithm implementations. It presents number of issues and transformations to reduce power dissipated in one or more of these sources. These transformations have been encapsulated in a framework that provide the comprehensive solution to low power realization of various FIR, IIR and FFT algorithms on programmable digital signal processors.

Index terms: Finite impulse response (FIR) & Infinite Impulse response (IIR) filters, Architecture driven voltage scaling, Bus invert coding, Gray coding, Transition zero coding, CMOS VLSI.

I. Introduction

Over the last two decades, technology scaling and systems development have primarily targeted increasing system throughput**;** reducing the minimum feature size has reduced gate delays and allowed high level of integration. However, as demand for portability increases, with speed, the need of low power design in systems has become increasingly important. To maximize battery life and minimize weight the systems and applications such as a portable communications terminals, which demand extreme computation capabilities requiring more then a billion operations/sec to support wide range of user applications including speech, graphics, and full motion video.

Many of these applications, built as a "systems on a chip", perform real time signal processing functions using a programmable DSP core embedded in them. Finite impulse response (FIR) filtering, infinite Impulse response filtering (IIR), FFT algorithms are most commonly used functions. FIR and IIR filtering is achieved by convolving the input data samples with the desired unit impulse response of the filters. The output Y[n] of the filter is given by

$$y[n] = \sum_{i=0}^{M-1} A[i].X[n-i] + \sum_{i=1}^{N-1} B[i].X[n-i]$$

The weights A[i] and B[i] in the expression are the filter coefficients. But B[i] = 0 for FIR filters. And IIR filters response depends upon both the previous and present input samples X[n-i] as well as upon the previous output samples Y[n-i].But for applications requiring minimum phase distortion FIR is the ideal choice because of symmetric coefficients (A[i]=A[N-1-i]) are possible. The typical DSP architecture[1],[10] to implement all the DSP applications is shown in Fig.1[1]. The techniques to reduce the power are targeted to this architecture.

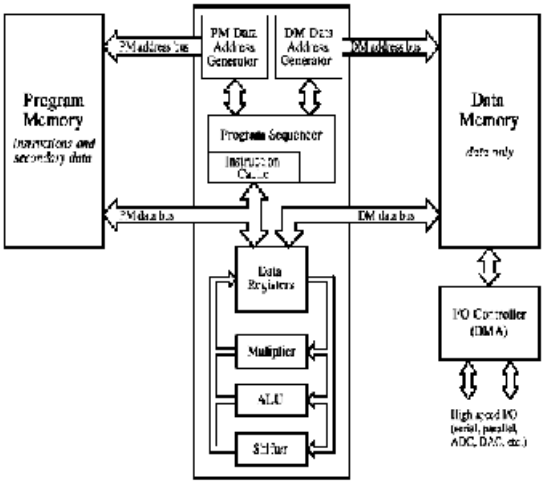

Figure 1 Typical DSP Architecture

The architecture has two separate memory spaces for program and data which can be addressed simultaneously. One of the memories can be used to store input data samples and the other to store the coefficients. The program is usually stored on the available on chip ROM which is usually electrically programmable but the coefficients as well as program can also be stored in same memory. Multiplier and accumulator unit (MAC) is composed of an adder, multiplier and the accumulator. Usually adders implemented in DSPs are Carry-Select or Carry-Save adders as speed is of utmost importance in a DSP[3],[10]. One implementation of the multiplier could be as a parallel array multiplier which computes partial products and adds them to the previous partial products. Other ways are multiplication based on LUTs. There are some Multiplier-less implementations too. Basically the multiplier will multiply the inputs and give the results to the adder, which will add the multiplier results to the previously accumulated results. This entire process is to be achieved in a single clock cycle. The FIR & IIR filtering can be performed using this architecture as a sequence of MAC's.

The sources of power dissipation in CMOS circuits can be classified as dynamic, short circuit and leakage power[12].

$P_{avg} = P_{switching} + P_{short\text{-}circuits} + P_{leakage}$

$= \alpha \, . \, C_L . V_{dd}^2 . f_{CLK} + I_{SC} . V_{dd} + I_{leakage} . V_{dd}$

The first term represents the switching components of power, where C_L is the load capacitance, f_{CLK} is the clock frequency and α is the switching factor or node transition

activity factor, which shows high transition level (V_{dd}) should be reduced to min. possible as it have quadratic effect on power. And $\alpha . C_L$ can be reduced by choice of logic function, logic style, circuit topology, data statistics and sequencing of operations. Second term represents short circuit component of power which is dependent upon rise and fall time of input signal. To reduce this component, the rise and fall time of input should be less then or equal to that of the output signal. If $V_{dd} < V_{tn} + | V_{tp} |$, I_{sc} can be completely eliminated. Architecture driven voltage scaling discussed in this paper allows reduction in supply voltage. Third term represent the leakage component of power, which can not be reduced much, as it depends upon the technology. The silicon on insulator technology have very less leakage and sub-threshold current.

The low power transformations described in this paper aim at reducing one or more of these factors while maintaining the throughput of the DSP system. The rest of the paper is organized as follows. In Section II, the main sources of power dissipation and measures for estimating the power are discussed. In Section III, optimizations at algorithm level and in Section IV, optimizations at architectural level are discussed. Similarly, section V, VI discussed the optimization at physical circuit and logic level, and at technology level respectively. Finally, conclusion have been made in Section VII with the description of a complete framework for applying the issues and transformation discussed in all the sections.

II. Power Dissipation- Sources And Measures

A. Components contributing to power dissipation

Each step in FIR, IIR, FFT filtering algorithms involves getting the appropriate coefficient and data values and performing the MAC operation over them. The address and data busses of both memories and multiplier-adder data path experience the highest signal activity during the filtering operations[8]. For a typical embedded processor, address and data buses are networks with a large capacitive loading. Hence signal switching in these networks has a significant impact on the power consumption. In addition to the net capacitance of each signal of the bus, inter-signal cross-coupling capacitance also contributes to the bus power dissipation. The power dissipation due to inter-signal capacitance varies depending on the adjacent signal values. The current required for signals to switch between “5’s” (0101b) and A’s (1010b) is about 25% more than the current required for the signals to switch between “0’s” (0000b) and F’s (1111b).

Another component is power dissipation in multiplier and adder. It is very difficult to find the transition density at each and every node of the internal circuits. Also most cause of run-time failure is the extent of circuit activity. As given in the literature [2], the algorithm of how transition density propagates from input, internal nodes of the circuit and then to the output nodes. This algorithm is implemented in simulator called DENSIM to measure the transition density of the circuit which accepts transition density and equilibrium probability at the primary inputs of the circuit. Where transition density of any signal x(t), t Є (-∞, +∞) is defined by

$$D(x)=\lim_{T\to\infty} N_x(t) / T$$

and N_x (t) is the number of transitions of x(t) in the time interval (-T/2, T/2). And equilibrium probability of x(t) is denoted by P(x) and given by

$$P(x)=\lim_{T\to\infty} (1/T) \int x(t)\, dt$$

As x(t) is the digital signal, P(x) is the fraction of time that x(t) is in ‘1’ state and D(x) is the average no of transition per unit time. Thus D(x) for 10 MHz clock is $20*10^6$. The various test on this simulator shows, the number of transitions at the circuit nodes is directly related to no. of transitions at the primary input of the circuits. Hence our main emphasis is to minimize the number of transitions at the input of the multiplier and adders.

B. Measures of power dissipation

For the address and Data buses, the hamming distance between consecutive signal values and the number of adjacent signal toggling in opposite direction thus form the measure of power dissipation in the buses. For the multipliers, the power is directly proportional to the transition densities and the probabilities of the multiplier/adder inputs[2]. The transition densities of the multiplier inputs depend on the Hamming distance between successive input values. The input signal probabilities depend on the number of “1’s” in the input signal values of the multiplier[3]. These two thus form the measures of multiplier power dissipation.

Third measure of the power dissipation is the effect of circuit state on the power cost of an instruction stream[11], which is more marked in case of DSP processor. An instruction level power model based on the base cost and the overhead cost of an instruction where the base cost of a given instruction is defined as average current drawn by the processor during the repeated execution of instruction. The overhead cost is needed to account for the effect of circuit state change for an instruction sequence consist of different instructions. This suggests that changing the instruction order by appropriate scheduling of an instruction can lead to reduction in the power cost of the program. This is called cold scheduling.

III. Optimization at algorithm level

The main factor which decides the power consumption is the decision made for choice of algorithm. The ability for algorithm to be parallelized is critical and the basic complexity of computation must be highly optimized. This

section includes, how to minimize numbers of operations and switching activity at algorithm level.

A. Voltage reduction using algorithm transformations

Concept here is that due to quadratic effect of V_{dd} on power, reduction on supply voltage has a great effect, but it reduces the speed. This section discusses the transformations which allow the reduction of supply voltage for the same throughput.

1. Transformation for FIR filters--Signal flow graph of FIR filter as shown in figure 2[13]. require delayed versions of input samples multiplied by the stored coefficients. Due to which during every multiplication both the multiplier inputs are switched to different values leading to large switching losses in multiplier.

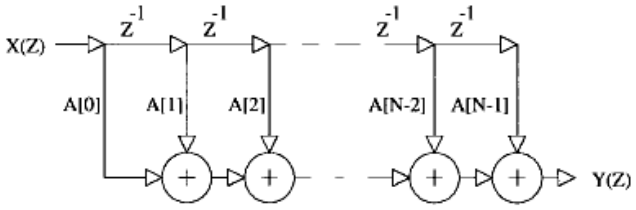

Figure 2 Signal flow graph of FIR filter

These losses can be minimized by transforming the signal flow graph using transposition theorem[13] into SFG shown in fig.3. This involves multiplying all the coefficients with same input data X[n].

Since during the filter computation, one of the multiplier inputs is fixed, it results in significant power saving in the multiplier. But It have one drawback also that the architecture is required to be change to perform the write operation after every

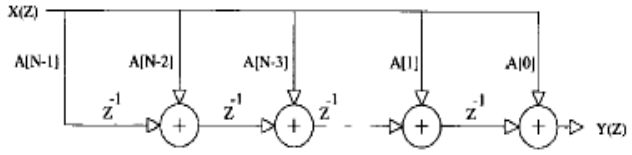

Figure 3 Signal flow graph of transposed FIR filter

cycle. Such architecture[4] is shown in Fig. 4 which supports N cycle implementation of N tap transposed filter.

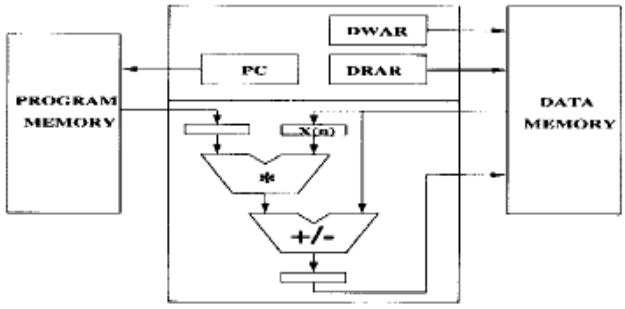

Figure 4 Architecture to support transposed FIR filter

But write operation after every cycle will result in extra switching on high capacitance data lines which results in increased power dissipation compared to the direct form implementation. Thus making such architecture overall less power efficient. Another Technique involves implementing FIR filter in terms of its decimated sub-filters using multi-rate architectures[4]. The signal flow graph of a multi-rate architecture that uses a decimation factor of two is shown in Fig. 5.

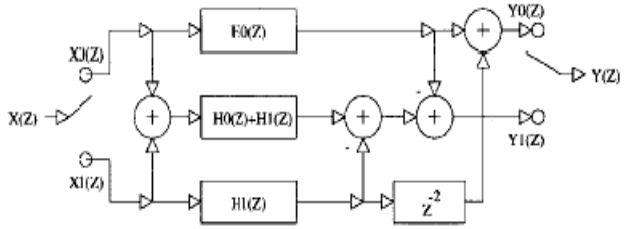

Figure 5 one level decimated multirate architecture

H0 and H1 are the decimated sub-filters formed by grouping even and odd coefficients of the original filter. The architecture processes two input samples simultaneously to produce the corresponding two outputs. From the signal flow graph shown in Fig. 2, an N-tap direct form FIR filter requires N multiplications and (N-1) additions per output. For the architecture shown in Fig. 5, assuming even number of taps, each of the sub-filters is of length N/2 and hence requires N/2 multiplications and (N/2-1) additions. There are four more additions required to compute the two outputs Y0 and Y1. This architecture hence requires 3N/4 multiplications per output which is less than the direct form architecture for all values of N and requires (3N+2)/ 4 additions per output which is less than the direct form architecture for N>6.

Because of the parallelism involved in this architecture number of cycles required to compute one output of filter are less than direct form architecture, since the multi-rate architecture requires fewer cycles, the frequency can be lowered by $f_{multi\text{-}rate}/ f_{direct.}$ and still achieve the same throughput. With the lowered frequency, the processor gets more time to execute the instruction. This time-slack can be used to appropriately lower the supply voltage using the following relationship:

$$\text{Delay } \alpha\ V_{dd} / (V_{dd} - V_{tn})^2$$

Based on above observation: $C_{total\_multirate} / C_{tatal\_direct} \approx 0.75$. This reduction in the switched capacitance translates into ≈ 0.75 saving in the power dissipation. Further reduction can be achieved by lowering the supply voltage as discussed above.

2. Transformation for IIR filters—As in IIR filters, there is a feedback, the computation can not be easily parallelized and algorithmic transformation are required to achieve recursive bottlenecks. Consider a first order IIR filter[5] as shown in fig. 6(a) with a critical path of 2 by assuming each operation takes one cycle. Due to recursive

bottleneck imposed by the filter structure, it is impossible to reduce the critical path using retiming and pipelining discuss in the section IV which we want, so that we are able to reduce the power for same throughput. Applying loop unrolling does not change the effective critical path as shown in fig.6(b) and therefore the supply voltage can not be reduced.

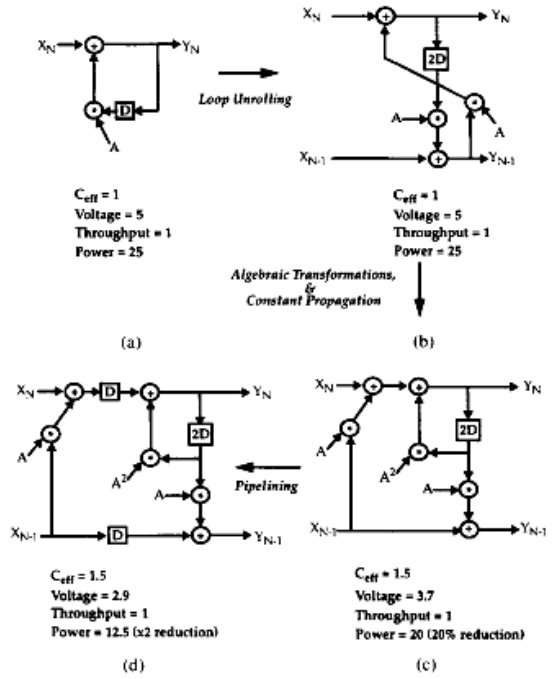

Figure 6 Transformation of IIR algorithm to reduce power

But after applying loop unrolling[5], distributivity and constant propagation transformation can be applied in a systemic way, the output of the samples can be represented as

$$Y_{N-1} = X_{N-1} + A * Y_{N-2}$$
$$Y_N = X_N + A*X_{N-1} + A^2 * Y_{N-2}$$

The transformed solution has a critical path of 3 (fig 6.(c)). However the pipelining can now be applied reducing the critical path further to 2 (fig 6(d)). Since we are processing the 2 samples in parallel with in 2 cycles. Hence the supply voltage can be reduced to half at which the delay increased by the factor by 2. But at the same time transformed SFG requires 3multiplications and 3 additions for processing 2 samples while the initial graph requires only one multiplication and one addition to process one sample, hence there is 50% increase in the capacitance. The reduction in the supply voltage, however, more than compensates for the increase in capacitance resulting in overall reduction of power by 2 due to quadratic effect of $V_{dd.}$ This means we can speed up the circuit by more loop unrolling combined with the other transformations and hence further reduction in voltage is possible for the same throughput. Unfortunately, the capacitance grows linearly[5] with unrolling factor and soon limits the gains from reducing the supply voltage. This result in an optimum unrolling factor for power is 3 beyond which the

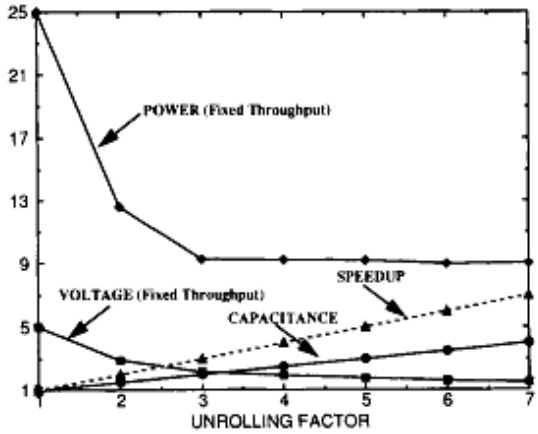

Figure 7 How power optimization is different from speed

power consumption starts to increase again as shown in fig 7.

B. Coefficient Scaling –

Scaling the output of a filter preserves its characteristics in terms of pass band ripple and stop band attenuation, but results in an overall magnitude gain equal to the scale factor[6]. For a scale factor K, from Eq (1) we get

$$K.Y[n] = K\sum_{i=0}^{M-1} A[i].X[n-i] + K\sum_{i=1}^{N-1} B[i].Y[n-i]$$

Thus the coefficients of the scaled filter are given by (K.A[i]) and (K.B[i]). Given the allowable range of scaling (e.g., 3 dB), an optimal scaling factor can be found such that the total Hamming distance between consecutive coefficient values is minimized. This can thus reduce not only the power dissipated in the coefficient memory data bus but also the power dissipated in the multiplier

C. Coefficient Optimization—

Given an N-tap FIR or IIR filter with coefficients (A[i], i=0.--.N-1) and (B[i], i=1,2--M-1) that satisfy the response in terms of pass band ripple, stop band attenuation, find a new set of coefficients (LA[i], i=0,1,---,N-1) such that the total Hamming distance between successive coefficients is minimized while still satisfying the desired filter characteristics[6].

This coefficient optimization problem can be formulated as a local search problem, where the optimum coefficient values are searched in their neighborhood. This is done via an iterative improvement process. During each iteration one of coefficients is suitably modified so as reduce the total Hamming distance while still satisfying the desired filter characteristics. The optimization process continues till no further reduction is possible.

D. Selective Coefficient Negation—

The number of 1s in A[i] and -A[i] (where negatives are stored in 2's complement format) can differ significantly. This fact can be taken advantage of. If rather than storing A[i]s we store –A[i] for all such numbers for which –A[i] has lesser number of ones and in the MAC instead of multiply/add we use multiply/subtract, the result will be same[6]. But significant power dissipation reduction is observed in multiplier and coefficient data bus power.

But to implement it some modification are required, If coefficients to be negated have random pattern then filtering can be performed using two loops- one for repeated multiply add and other for repeated multiply subtract, by grouping coefficients to be negated together.

E. Coefficient ordering —

The order of computing the products and then addition can be interchanged due to associative and distributive property of summation[6]. For a 3-tap IIR filter, the output can be computed as

$Y[n]= A_1X[n-1]+A_2X[n-2]+B_1Y[n-1]$ OR

$Y[n]= B_1Y[n-1]+A_1X[n-1]+A_2X[n-2]$

Since these coefficients determine the order of data to be appeared on the data buses for coefficient as well as input samples. Hence Hamming distance between two consecutive coefficients can be minimized by perfect ordering of above sum of products. For a N-tap filter there are such N! different coefficient orders are possible. To get the perfect order nearest neighbor algorithm should be used which provide the heuristic search to get the order corresponding to minimum switching activity on the data bus.

After the order is found, the data in the data memory is also required to reorder so that the desired sequence of coefficient-data product computations is achieved when the memory is accessed sequentially.

F. Minimizing the number of operations: Multiplication with constants

If the input is being multiplied with multiple coefficients, some of the shift-add terms can be shared and the number of operations can be reduced[5]. Fig.8 shows an example of exploiting multiple coefficients multiplied with same input.

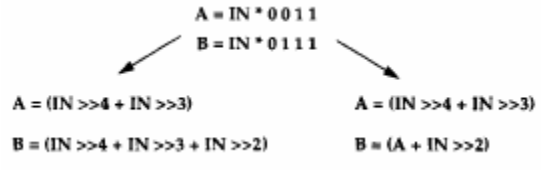

Figure 8. Example demonstrating sub-expression example

On the left is brute force implementation in which each multiply is computed separately. On the right side, approach exploits common terms in the coefficients and therefore some of the shift- add terms can be shared.

IV. Optimization at architecture level

An architecture driven voltage scaling strategy is presented in this section in which concurrent architectures are used to reduce supply voltage by retaining throughput.

A. Architecture driven voltage scaling—

Architecture can be modified to compensate for reduction in speed due to reduction in $V_{dd.}$ These modifications involve two major features called parallelism and pipelining as discussed below:

1. Parallelism—To illustrate how architectural techniques can be used to compensate for reduced speed a simple 8-b datapath consisting of adder and comparator is considered[5]. Fig.9 shows inputs A and B are added and the system is clocked with a clock period of T. Let's take this data path as a reference whose power dissipation is given by

$$P_{ref} = C_{ref}\ V_{ref}^2\ f_{ref}$$

Where C_{ref} is effective capacitance being switched per clock cycle. One way to maintain throughput for reduced V_{dd} is to utilize parallel

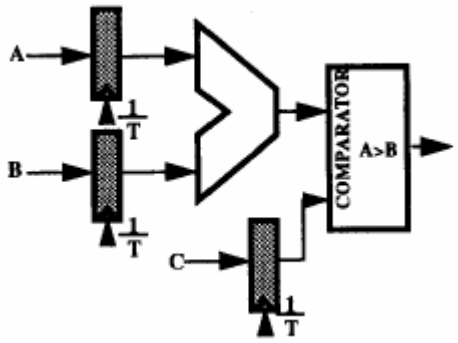

Figure 9 Simple data path for adder and comparator

architecture as shown in fig. 10, two identical data paths are used, allowing each unit to work at half the original rate while maintaining the original throughput. Since the speed requirements of adder and comparator are reduced to half, hence voltage can be reduced from 5V to 2.9V, the voltage at which the delay is doubled[5] as shown in fig.11.

While the data path capacitance has increased by the factor of 2. and corresponding frequency decreased by the factor of 2. Unfortunately, there is a slight increase in the total capacitance due to extra routing resulting in an increased capacitance by the factor of 2.15. Thus the power for parallel datapath is given by

$$P_{par} = C_{par}\ V_{par}^2\ f_{par}$$
$$= (2.15C_{ref})\ (0.58\ V_{ref})^2\ (f_{ref}/2)$$
$$= 0.36\ P_{ref}$$

The amount of parallelism can be further increased to reduce the supply voltage. But up-to what limit, we can reduce the supply voltage, is

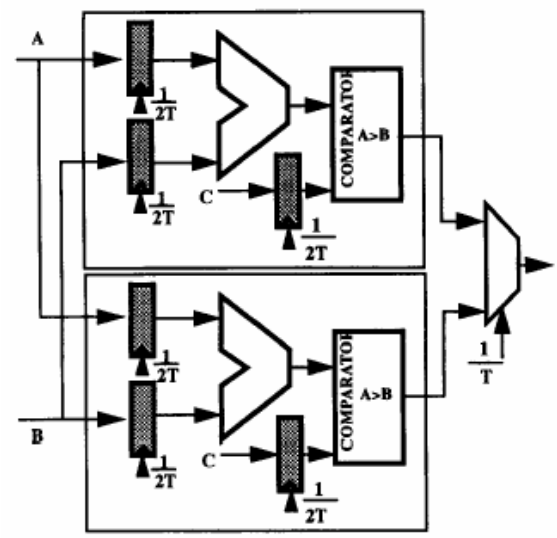

Figure 10 Parallel implementation of adder and comparator

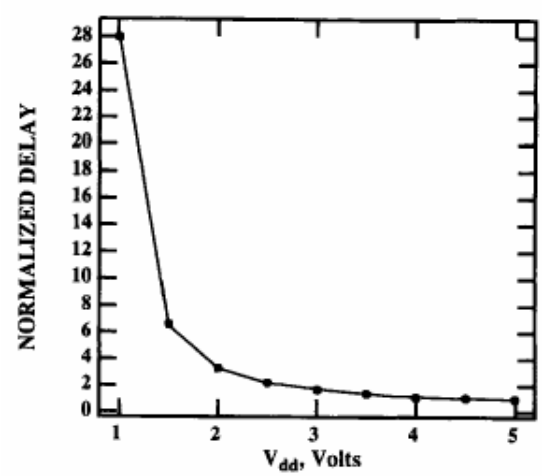

Figure 11 Normalized delay w.r.t supply voltage

decided by power consumption of overhead circuitry which dominates at some optimum voltage.

The same parallelism concept can be used for memory access as shown in fig.12 two alternate schemes for reading 8-b data from memory at throughput f. On the left hand side serial access scheme and second approach is to read several words from memory and clock the memory at lower rate for the same throughput. For example reading 8 bytes in parallel require memory to be clocked at 1/8 of the serial rate. Hence supply voltage can be reduced as time available to read the memory is 8 times the serial version. The parallel version can run at supply voltage of 1.1V.But this optimization is possible only when data access pattern is sequential in nature.

2.Pipelining-- Pipelining enables the more efficient use of on-chip silicon resources, allows multiple operations to

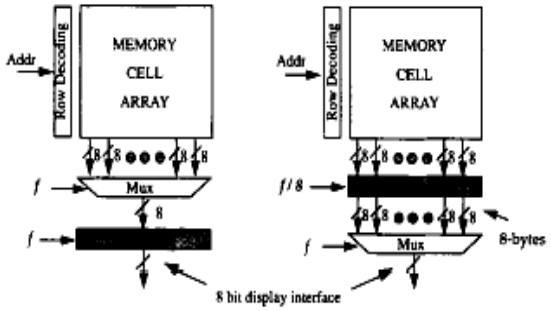

Figure 12 Parallel memory access for Vdd reduction

occur, and enables much faster cycle times[14]. What happens in pipelining is explained as follows: Throughput of any system consisting of a series of operations is limited by the single slowest operation in the complete series. So if this single slow operation is further broken down into a number of stages, the intermediate results will be available faster for the next stage. Each stage result(s) are stored in registers that follow that stage, and so the next stage can start operating on these results of the previous stage & simultaneously the previous stage can start operating on the next succeeding set of data. Thus simultaneously no stage of the

pipeline is free. DSPs can take advantage of pipelining as the data inputted in a DSP is always continuous.

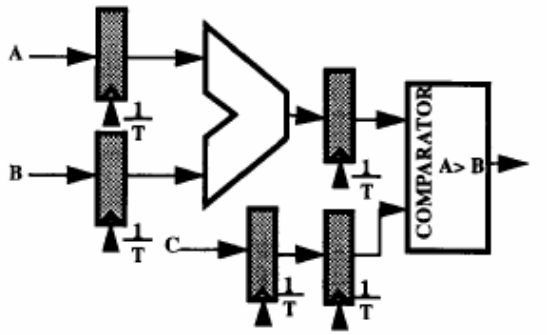

Figure 13 Pipelined implementation of simple data path

As shown in fig. 13, with the additional pipelined latch, the critical path becomes the max[$T_{adder,}T_{comaparator}$], allowing the adder and comparator to operate at slower rate[5]. For example the two delays are equal, allowing the supply voltage to again be reduced from 5V used in the reference data path to 2.9V. However, there is a much lower area overhead incurred by this technique, as we only need to add pipeline registers. Hence addition of extra latches, increasing the effective capacitance by approx. a factor of 1.15.The power consumed by pipelined data path is

$$P_{pipe} = C_{pipe}\ V_{pipe}^{2}\ f_{pipe}$$

$$=(1.15\ C_{ref})\ (0.58\ V_{ref})^2\ f_{ref} = 0.39\ P_{ref}$$

Hence power reduced by this technique is by the factor of 2.5, providing approx. the same power reduction as parallel architecture with the advantage of lesser area overhead.

Table 1 Architecture based voltage scaling results

Architecture	Voltage	Area (normalized)	Power (normalized)
Simple	5 V	1	1
Parallel	2.9 V	3.4	0.36
Pipelined	2.9 V	1.3	0.39
Pipelined-Parallel	2.0	3.7	0.2

Hence bigger improvement is possible if both parallelism and pipelining [5]are applied as shown in Table. I

B. Minimizing switching activity by choice of number presentation—

In most signal processing applications, 2's complement representation is used for arithmetic and logical operation. But fig.14 shows a short segment of a speech signal and the associated transition probabilities (Pr) assuming 2's complement for each bit[5]. LSB's are uncorrelated and hence Pr-0.5 and MSB's are highly correlated. If signal switches at higher rate form +ve to –ve value, hence uses small dynamic range then max. possible value determined by the bit width, it results in large switching activity.

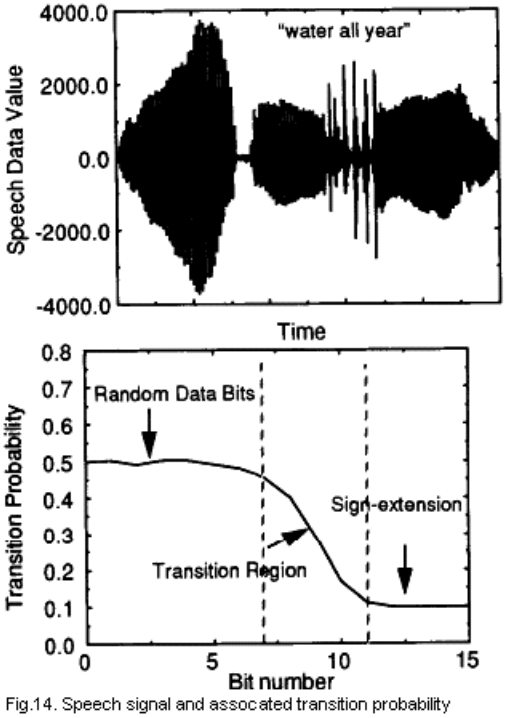

Fig.14. Speech signal and assocated transition probability

1. Reducing the number of transitions on buses

1.1 Data busses—

1.1.1 Using Sign magnitude—In sign magnitude only one bit is allocated for the sign and other for magnitude. In this case, if the dynamic range of a signal does not span the entire bit-width, only one bit will toggle when the signal switches sign, as opposed to the 2's complement representation

where number of bits switches due to sign change. To illustrate this, the transition activity on 16-b data bus with random gaussian data applied to it is shown in fig. 15 for 2's complement number representation and sign magnitude represent[5].

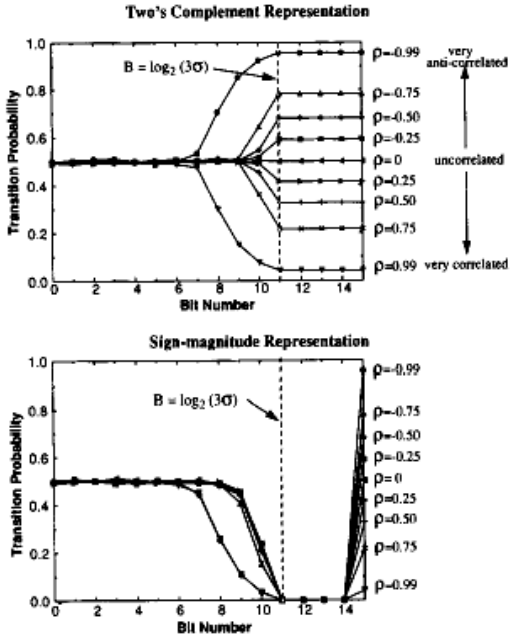

Figure 15 Transition activity for different number presentation

In which ρ is the correlation between two consecutive data. The above analysis suggest that sign magnitude has advantage in terms of number of transitions on busses. However addition and subtraction computation are difficult to implement in sign magnitude representation. So sign magnitude is helpful if large busses have to be driven (like external memory access), where overhead of converting into 2's complement is insignificant as compared to reduction in the capacitance switched in large busses.

1.1.2 Using bus invert coding—

Bus invert coding (BIC) also called starvation coding and limited weight coding is helpful in reducing the $P_{dis.}$ at I/O pads where due to huge dimensions of the devices, large currents are required to be drive the capacitance. In this coding we look for reducing I/O activity with assumption that each node has the same average load capacitance at I/O and internal nodes. Total power dissipated on the chip is given by

$$P_{chip}=C_{int}.N(\text{transitions})_{int} + C_{I/O}\ .\ N(\text{transitions})_{I/O}$$

Internal transitions are much larger then I/O transitions because of large number of internal nodes, but C_{int} is much smaller than $C_{I/O.}$ The Idea behind this coding is that: code the data in order to decrease the number of transitions on

the I/O node even at the expense of slightly increasing the number of transitions on the low capacitance node[7]. But total effect on power dissipation is reduction because of large I/O capacitance. The data on n-bit wide bus can have 2^n possible values with equal probability. The average number of transitions per clock cycle will be n/2. This code needs one extra control bit called *invert* bit. Let data'd' is to transmit and 'b' denotes the coded value on the bus.

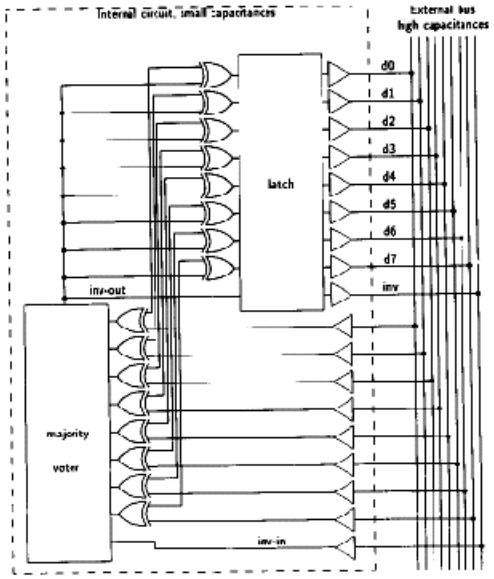

Fig.16. Encoder to implement bus invert coding on 8-bit bus

By convention when *invert=0,* b=d and when invert=1, bus value is the complement of data value. Power can be reduced by coding the I/O as follows:

i. Compute the hamming distance between the present bus value and next data value.
ii. If the hamming distance is larger than n/2, set *invert*=1, and make the next bus value equal to the inverted next data value.
iii. Otherwise let *invert*=0 and let the next bus value equal to the next data value.
iv. At the receiver side the contents of the bus must be conditionally inverted according to the invert line, unless the data is not stored encoded as it is.

With this coding average number of transition per time slot becomes 3.27(instead of 4). But maximum number of transitions is reduced by half. Hence peak power is reduced by 50% but average power is reduced only by 18.2%. If the numbers of data lines are large then average power is reduced by more amount. To implement the bus invert coding extra circuitry is needed which means extra area and sometimes lower speed. A possible circuit for 8-bit data bus at driver side[7] is shown in fig.16 that consists of 16 extra X-OR gates and a majority voter. This circuit would result in extra internal transitions and delay, but bus drivers and receivers and latches are also required in un-encoded case. But the delay and extra power consumption of majority voter should be considered carefully. At the receiver side because it only needs to invert the bus if invert line=1, it can be very easily implemented. If encoder and decoder have large delay then pipelining can be used to get same throughput.

1.2 Address bus— Bus invert coding is not good for address bus, as mostly the data on this bus is sequential and number of transitions on average are not more than n/2. Following techniques are more helpful to reduce power consumption on the address bus.

1.2.1 Gray coding— Address busses are used to fetch the data and coefficient (instruction also) from the external or internal data and program memory respectively. By providing the addressing by gray codes which changes by one bit only as sequenced from one number to next number[8]. Hence significant number of bit switches can be eliminated as shown in Table.2[8], the comparison of bit switches for binary and gray sequential addressing. It also shows that higher order bit on gray code sequence are not switched until all bit switch permutations for low order bits takes place. Hence gray coding is beneficial also for paging as binary coding. Instead of binary counter, the gray counter is required on chip to access the external or internal memory, to which we can optimize by good designing to consume low power. Following fig. 17 shows the comparison between

Table 2. Bit switches for consecutive numbers

Decimal	Binary code	Gray code
0	00000	00000
1	00001	00001
2	00010	00011
3	00011	00010
4	00100	00110
5	00101	00111
6	00110	00101
7	00111	00100
8	01000	01100
9	01001	01101
10	01010	01111
11	01011	01110
12	01100	01010
13	01101	01011
14	01110	01001
15	01111	01000
16	10000	11000
Total bit switches	31	16

binary and gray coding for various packages[8]. But it is unrealistic to assume that the address computation units, the data path, the memory decoder and even the compiler could be modified to generate gray code addressing.

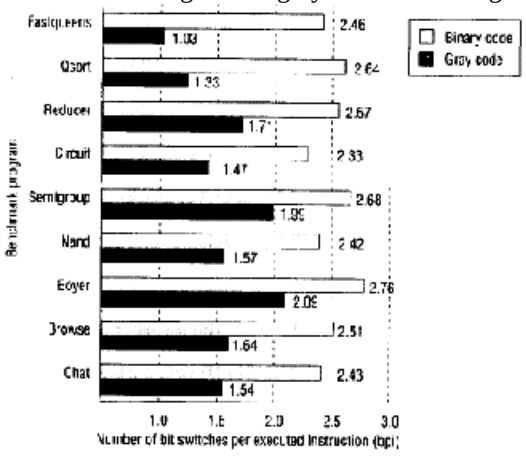

Fig 17 Bpi of selected benchmarks for binary and Gray code addressing

Hence it is not easy to implement it with simple gray counter. Another possibility is implementing the gray encoder and decoder on both sides.

1.2.2 Transition zero coding— This technique relies on the fact in a remarkable no. of cases, patterns traveling on the address bus are consecutive[9]. Hence devices located at receiving end of the bus can automatically calculate the address to be received at the next clock cycle. Hence transmission of new pattern can be avoided. But there are exceptions to this behavior, as control flow instructions cause interruption in the sequence of consecutive addresses on the instruction flow and data which not stored in arrays, are often addressed without any regular pattern. But usually sequential addressing dominates. This techniques also require one extra line called INC, If two addresses are consecutive then INC line=1. The address lines are frozen and new address is computed directly by the receiver. When two address are not consecutive then INC=0, bus lines operate naturally. As long as the addresses are consecutive, there is no any transition on the address bus and hence thus coding is transition zero coding (T0). This approach is helpful if consecutive addresses have very large lengths. To fully evaluate the effectiveness of this coding the cost of T0 encoding/decoding from binary addresses should be taken into account. At a given clock cycle t, the encoder as shown in fig.18[9] computed the incremented address of cycle t-1and compare it to the address generated at cycle t. If both are equal then INC line is raised and the old address is left

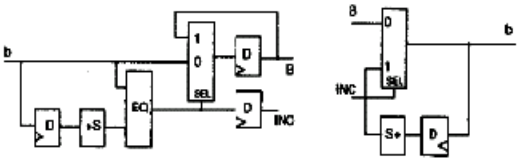

Fig.18. Block diagram of T0 encoder and decoder

on the bus otherwise INC line=0 and newer address is loaded onto the bus. The decoder is even simpler shown in fig.18(right side). At any given cycle the last cycle address is incremented. If INC line is high, the older incremented address is used otherwise the address coming from bus line is selected. If encoder/decoder are optimized for min. delay. The critical path at the encoder side is late arrival of data b, comparator and MUX. And on the decoder side is incrementer and MUX. Hence this architecture circuit can be make very fast. But gray encoder/decoder consume less power and area as compared to T0 encoder/decoder, but gray decoder is consist of chain of XOR's due to which its delay is more as compared to T0 encoder/decoder.

Hence for wider buses, if performance is main factor then T0 coding is the only alternative[9]. For power is main factor, there is trade-off between both as T0 encoder/decoder consume large power. For area is main factor, gray code is the best option, where power is the 2nd concern.

C. Adder input bit swapping— This technique is used to reduce the power dissipated in the adder, its input buses and the accumulator[6]. This not only reduces power dissipation in these buses, it also reduces the power dissipated in the adder and the accumulator that drives one of the buses. Bitwise commutativity implies that the result of an ADD operation is not affected even when one or more bits from one input are swapped with the corresponding bits in the second input.

The technique compares for every bit, the new value with the current value, and performs bit-swapping if the two values are different. Fig. 19 shows an implementation of this scheme[6]. As can be seen from Fig. 19, the reduction in the toggles in the adder inputs is achieved at the expense of additional logic, i.e., the MUXes and the XOR gates. The power dissipated in this logic offsets power savings in the adder and its input buses.

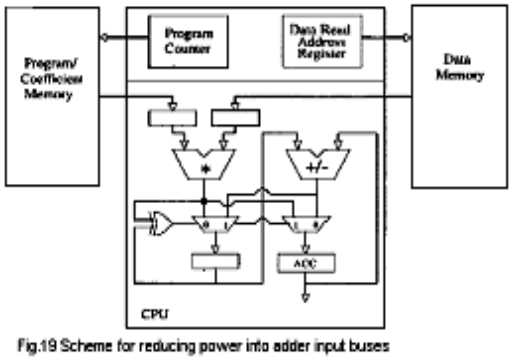

Fig.19 Scheme for reducing power into adder input buses

The final savings depend on the data values being accumulated and also on the relative capacitances of the adder input buses and the mux inputs.

D. Minimizing glitching activity – Due to finite propagation delays from one logic block to the next, the node can have multiple transitions in a single clock cycle before settling to correct logic level[5]. To minimize the extra transitions and power in a design, it is important to balance all signal paths and reduce the logic path.

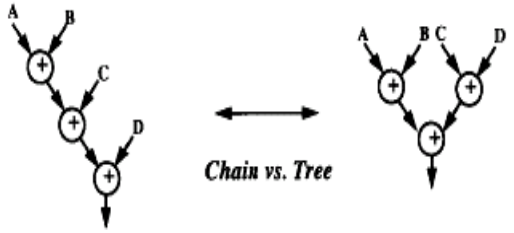

Fig.20 Reducing glitching activity by balancing signal paths

Fig.20 shows chain and tree implementations for adding four numbers if we assume that all the inputs arrive at same time. Then due to finite propagation delay of first adder for the chained case, the second adder is computing with the new C input and the previous output of A+B and so on . Hence there are more chances of glitches as compared to tree structure which is balanced. But decrease in the logic path as in case of tree structure will increase register power. Hence the decision to increase or decrease the logic path is based upon trade-off between the glitching capacitance and the register capacitance. But it is also notable that reducing the logic path allows reduction in supply voltage while keeping throughput same.

V. Physical circuit and logic level optimization

A. Bus bit ordering— At the layout level, this technique aims at reducing power dissipation due to cross-coupling capacitance. One approach to achieve this is to increase the spacing between the bus lines. This however results in increased area. In this approach the bus bits are reordered in such a way that the number of adjacent signals toggling in opposite direction are minimized[6].

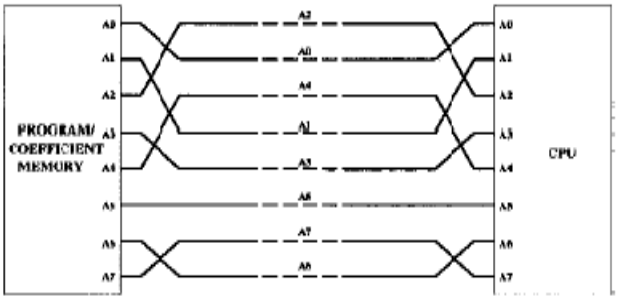

Figure 21 Bus bit reordering scheme

Fig. 21 illustrates[6] this approach and shows how the bus signals A0–A7 can be reordered in the sequence of A2-A0-A4-A1- A3-A5-A7-A6.

The numbers of algorithms are available to get optimum bus order. The input to these algorithms is coefficient values and there order in which they are accessed from memory for FIR or IIR computation. The most applicable algorithm is hill climbing search algorithm.

B. Place and route optimization— At the layout level, the place and route should be optimized such that signals that have high switching activity should be assigned short wires and signals with lower switching activities can be allowed progressively long wires.

C. Transistor sizing— For low power, as is true for high speed design, it is important to equalize all delay paths so that single critical path does not unnecessarily limit the performance of the entire circuit. As we known that to increase the speed we can increase the size of the transistor, hence it allows more reduction in supply voltage for same throughput. But optimizing for power is totally different[5].

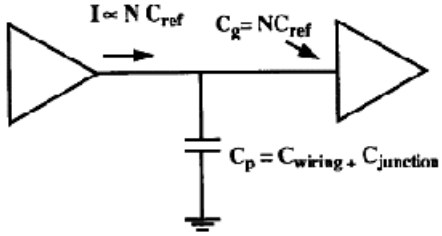

Fig.22 Ckt. model for analyzing the effect of transistor sizing

Fig.22 shows the two-gate circuit[5], with first stage having input gate capacitance C_{ref} driving the next stage with the parasitic capacitance due to substrate coupling and interconnect C_p between them. Then the delay of first stage at supply voltage V_{ref} is given by

$$T_N = K(C_p + C_{ref})V_{ref} / N\, C_{ref}(V_{ref} - V_t)^2$$
$$= K(1 + \alpha/N)\, V_{ref} / V_{ref}(V_{ref} - V_t)^2$$

where $\alpha = C_p / C_{ref}$, and K is constant. Speed can be increased by factor of $(1+ \alpha/N)/1+ \alpha$ if the size is increased by N w.r.t reference circuit having N=1.

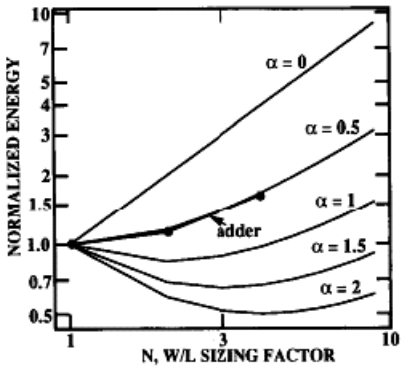

Fig.23 Energy Vs. transistor sizing for various parasitics contributions.

But if we want the supply reduction to decrease the power then speed to the V_{dd} can be scaled down to get same voltage where delay of the scaled design and the reference design are same which is given by

$$V_N = (1+ \alpha/N)V_{ref}/(1+ \alpha)$$

Then energy consumed by first stage is given by

$$\text{Energy}(N) = (C_p + N C_{ref}) V_N^2$$
$$= N C_{ref} (1 + \alpha/N)^3 V_{ref}^2 / (1 + \alpha)^2$$

For $\alpha=0$, energy increases linearly w.r.t N. For $\alpha>0$, Initially energy decreases with increases in N, due to reduction in delay compensates the increase in capacitance due to increasing N and after some point energy increases with increase in N as the capacitance factor NC_{ref} dominates and also parasitic capacitance increases as shown in fig.23 (which shows the simulation result of 8-bit adder) for various values of N[5]. Which shows that if C_p due to interconnects is very large as in case to drive the address and data buses, then increase in the size is helpful in reduction of power by using large size buffer at the driving end. But if C_L is not dominated by interconnects then minimized sized device should be should.

D. Reduced swing logic— Another approach to reduce the supply voltage is to reduce the swing at the output node[5]. For example, using an NMOS device to pull up the output will limit the swing to $V_{dd} - V_{tn.}$ The power consumed will be reduced to $C_L V_{dd}(V_{dd} - V_{tn})$. But this scheme has two negative effects: First, noise margin is reduced at the output by V_{tn}. Second, since the output does not rise to the maximum value, the PMOS of the next stage will be on and hence the next static stage will have static power loss, increasing the effective energy per transition. Hence special gates are required to increase the noise margin at signal level and eliminate the short circuit currents as shown in fig. 24[5], which requires extra transistors and hence have more parasitic capacitance. This circuit uses a precharged scheme and the device M3 is used to clip the voltage of bit line to $V_{dd} - V_{tn.}$ During $\varphi=1$, the evaluation period, if V_{in} is high, the bit

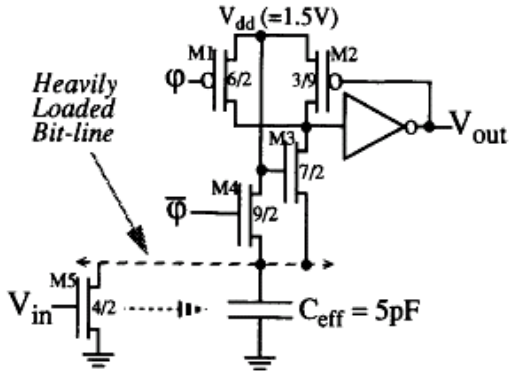

Fig. 24 Signal amplification/swing reduction in memory circuits.

line will begin to drop as shown in fig.25.As the ratio of the capacitance of the bit line to the internal node is very high, once the bit line is dropped to 0.2V to sufficiently turn on M3, the internal node drops to the potential to the potential of bit line, providing signal amplification. Thus the circuit greatly reduces the voltage swing on the high capacitance line, which reduces the energy, and provides the amplification, which reduces the delay as well. This technique is useful for high capacitance bit line or nodes.

E. Low voltage support circuitry— One important circuitry to operate the circuit at low voltages is the level shifter[5]. That can convert the low voltage swing form the core of

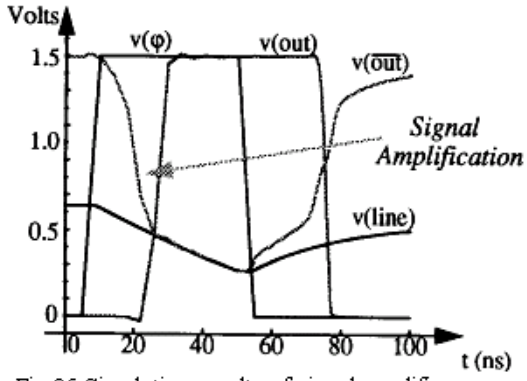

Fig. 25 Simulation results of signal amplifier.

the chip to the high voltage swing at the I/O pin or vice-versa. Also different parts of the system could operate at their own optimum supply voltage and communicate with each other using level conversion circuitry, in which the design of high efficiency low voltage in which voltage is programmable, must be considered.

F. Logic level power down and gated clocks— Power down is not only useful at chip and module level but also at the logic level by reducing the switching the switching activity at the expense of some additional control circuitry[5].

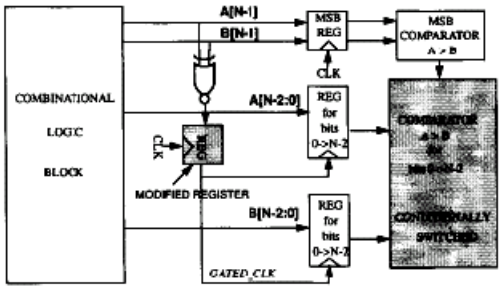

Fig. 26 Using gated clocks to reduce power

Taking example of comparing the two numbers at the output of combinational circuit as shown in fig26.If the most significant bits, A[N-1] and B[N-1], are different then the computation of A>B can be strictly performed from MSB's and therefore the comparator logic for bits A[N-1:0] and B[N-1:0] is not required and hence the logic can be power down. One approach to accomplish the power down is shown in fig26, is to gate the clock. The XNOR output of the A[N-1] and B[N-1] is latched by the special register to generate the gated clock. This gated clock is then used to clock the low order registers.

VI. Optimization at technology level—

We know that the supply voltage have quadratic effect on power reduction at the expense of delay. As shown in fig.11, the delay increases drastically as supply voltage approaches threshold value[5]. Since the objective is to reduce power consumption while keeping the throughput of overall system fixed, compensation for these increased delays is required. As discussed in Section II architecture driven voltage scaling strategy uses the concept of parallelism and pipelining to compensate for increased delays. Another approach is to reduce the threshold voltage of the device. Low threshold devices should be used. Reducing the threshold voltage allows the reduction in supply voltage without loss in speed. For example the circuit running at the supply voltage of 1.5V with V_t=1V have same performance as the circuit running at supply voltage of 0.9V and V_t=0.5V according to the following equation

$$T_d = C_L V_{dd} / K_n (W / L) (V_{dd} - V_t)^2$$

Fig.27 shows the plot of normalized delay Vs. threshold voltage for various supply voltages[5].

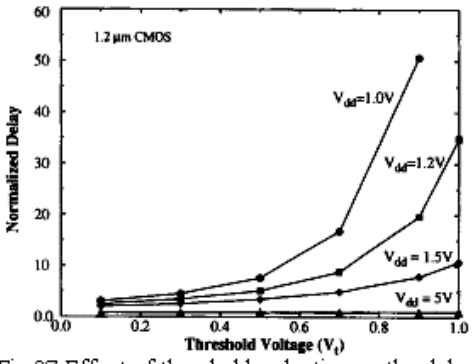

Fig.27 Effect of threshold reduction on the delay for various supply voltages

But how low the threshold can be reduced. This limit is set by the adequate noise margin and the increase in the sub-threshold currents. Noise Margins will be relaxed in low power designs because of the reduced current being switched, but sub-threshold current will result in significant static power dissipation.

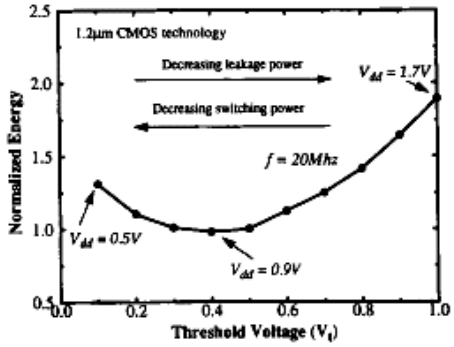

Fig.28 Compromise between dynamic and leakage power dissipation through threshold variations.

Fig.28 shows a plot of energy Vs. threshold voltages for a fixed throughput for a 16-b data path ripple carry adder. Which shows the optimum threshold voltage must compromise between improvement of current drive at low supply voltage operation and control of threshold leakage.

If feature size shrinks below 1.0µm, the delay characteristics as a function of supply voltage does not have quadratic relationship. As a result of velocity saturation, the current is no longer a quadratic function of voltage but linear; hence current drive is significantly is reduced to

$$I = W C_{ox} (V_{dd} - V_t) v_{sat.}$$

And hence the delay of the circuit is given by CV_{dd} / I, by comparing these two equations delay for submicron technology is relatively independent of supply voltages at high electric fields. Hence V_{dd} can be reduced to some extent for velocity saturated device with little penality in speed performance.

VII. Conclusions—

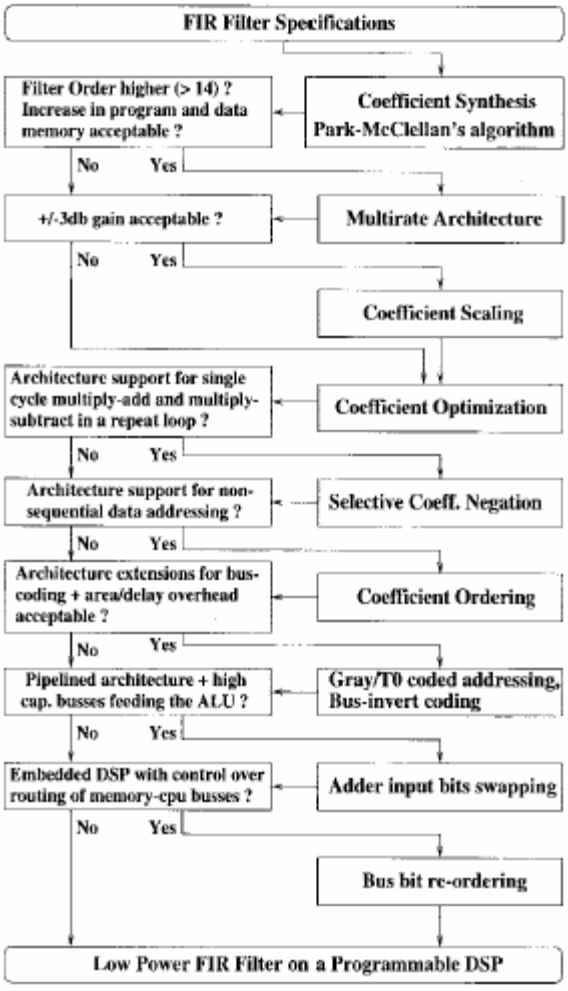

Figure 29 Frame work for designing low power digital signal processor

This paper discusses the identification of main sources of power dissipation in DSP and their measures. And provide the techniques to reduce them at each level of designing the digital signal processors. First of all, the specifications are given to us, and we need to design the algorithm using appropriate method of designing FIR, IIR filters and modify the algorithm to reduce the computations, this can be performed by various transformations as discussed above. Then coefficient modifications (like scaling, ordering, optimization, selective coefficient negation) are discussed at algorithm level to reduce the power further. Then at architecture level, the architecture driven voltage scaling which involve parallelism, pipelining is discussed. And then different coding techniques to reduce the power consumption on address bus and data bus are discussed. At physical layout level and logic level, this paper include activity driven place and route, the use of minimize sized devices, reduced swing logic, and logic level optimization and power down. At the technology level, this paper discusses the reduction in the threshold voltage V_t and its trade-off with leakage and sub-threshold currents. We can reduce the V_t to 0.3-0.4V which is 0.7V current day technology. All the above techniques are encapsulated in the framework to design the low power programmable processor as shown in fig.29[6].

Acknowledgment

The author would like to thank Prof. A. N .Chandorkar for his great co-operation and guidance. And the author would also like to thank the reviewers for their constructive criticism and comments.

References:

[1] S. W. Smith, The Scientist and Engineer's Guide to Digital Signal Processing, California Technical Publishing, 1997, pp. 503-534.
[2] F. Najm, “Transition density: A new measure of activity in digital circuits,” *IEEE Trans. Computer-Aided Design*, pp. 310–323, Feb. 1993.
[3] G.-K. Ma and F. J. Taylor, “Multiplier policies for digital signal processing,” *IEEE ASSP Mag.*, Jan. 1990, pp. 6–19.
[4] M. Mehendale, S. D. Sherlekar, and G. Venkatesh, “Extensions to programmable DSP architectures for reduced power dissipation,” in *Proc. 11th Int. Conf. VLSI Design*, Jan. 1998.
[5] A. P. Chandrakasan and R. W. Brodersen, “Minimizing power consumption in digital CMOS circuits,” *Proc. IEEE*, Apr. 1995, pp. 498–523.
[6] M. Mehendale, S. D. Sherlekar, and G. Venkatesh, “Low-Power Realization of FIR Filters on Programmable DSP’s”, IEEE Trans. Signal Processing, Dec. 1998, pp. 546-553.
[7] M. R. Stan and W. P. Burleson, “Bus invert coding for low power I/O,” *IEEE Trans. VLSI Syst.*, Mar. 1995.
[8] C.-L. Su, C.-Y. Tsui, and A. M. Despain, “Saving power in the control path of embedded processors,” *IEEE Design Test Comput.*, pp. 24–30, Winter 1994.
[9] L. Benini, G. D. Micheli, E. Macii, D. Sciuto, and C. Silvano, “Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems,” in *Proc. GLS-VLSI'97, 7th Great Lakes Symp. VLSI*, Mar. 1997.
[10] E. A. Lee, “Programmable DSP architectures: Part I,” *IEEE ASSP Mag.*, Oct. 1988, pp. 4–19.
[11] M. T.-C. Lee, V. Tiwari, S. Malik, and M. Fujita, “Power analysis and minimization techniques for embedded DSP software,” *IEEE Trans.VLSI Syst.*, pp. 123–135, Mar. 1997.
[12] Eshraghian and Weste, The principles of CMOS VLSI design, Pearson Education Asia, 1993, pp-231-237.
[13] John G. Proakis and dimitris G.Monoloakis, Digital signal processing, Prentice Hall India, Dec.2000, pp.-500-590.
[14] Vijay K. Madisetti, VLSI digital signal processors, IEEE press, 1995, pp.-121-190.