

PII: S0026-2692(96)00010-9

# Low power design for DSP: methodologies and techniques

T. Arslan, A.T. Erdogan and D.H. Horrocks

*Cardiff School of Engineering, Electronic Engineering Division, University of Wales,  
Cardiff CF2 1XH, UK. Tel: 01222-874000. Fax: 01222-874420*

Power dissipation is becoming a limiting factor in the realization of VLSI systems. The principal reasons for this are maximum operating temperature and, for portable applications, battery life. Because of the relatively greater complexity, the power dissipation in digital signal processing (DSP) applications is of special significance, and low power design techniques are now emerging. This paper provides an overview of the techniques and methodologies that have emerged in the past few years for DSP system design. These include techniques for minimizing power at architectural and algorithmic levels including DSP programming issues. In addition, the paper indicates some potential design directions. Copyright © 1996 Elsevier Science Ltd.

## 1. Introduction

The advance in VLSI technology, during the last decade, has led to the development of systems with high throughput capabilities. The impact of this has been most effective in the area of digital signal processing (DSP) where high throughput is most desirable. Today's DSP systems are capable of performing complex and computationally intensive signal processing tasks at speeds suitable for real-time operation. Example applications include speech recognition and video.

One of the fastest growing areas in the computing industry is the provision of high throughput DSP systems in a portable form [1]. A major limitation of these systems is the operating time provided by the battery, which is dependent on the characteristics of the battery and the power requirement of the system. Battery technology has peaked in recent years, whereas the number of applications requiring portability is increasing, placing an increasing demand on the power budget. A major concern in both portable and non-portable systems is heat dissipation. Overheating can reduce throughput and operation life. Cooling fans and heat sinks significantly add to the overall cost of a system [2].

Due to the reasons mentioned above, low power design techniques and methodologies are required to be adapted by VLSI system designers, especially those for portable DSP applications. This paper provides an overview of these techniques and aims to serve as a bibliography of the key papers relevant to low power DSP design. Emphasis is given to aspects of design at higher levels due to their importance in reducing the

overall design cost. In addition, the paper presents indications for potential design directions bearing in mind the architectural complexity and the speed requirements of today's systems.

CMOS logic is assumed since this is currently the most commonly used VLSI technology due to its high degree of integration, which is in turn allowed by its scaling properties and low power dissipation.

### 2. Power dissipation in digital CMOS circuits

The main sources of power dissipation, in a typical CMOS digital circuit (Fig. 1), are given by the following equation:

$$P_{ave} = P_s + P_{sc} + P_l \tag{1}$$

where  $P_{ave}$  is the average power dissipated by the circuit,  $P_s$  is the switching component of the power caused by charging/discharging of the circuit output load capacitance  $C_L$ , and  $P_{sc}$  and  $P_l$  reflect the power dissipated due to short-circuit and leakage currents, respectively ( $I_{sc}$  and  $I_l$ ). Equation (1) can be expanded in order to reveal the basic circuit parameters contributing to each of the above power components as follows:

$$P_{ave} = k \cdot C \cdot V_{dd}^2 \cdot f + I_{sc} \cdot V_{dd} + I_l \cdot V_{dd} \tag{2}$$

where  $V_{dd}$  is the supply voltage,  $f$  is the clock frequency,  $C$  is the physical capacitance of the

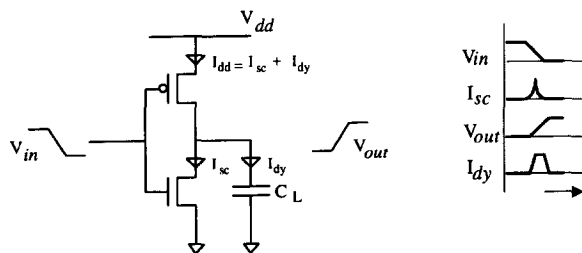


Fig. 1. Typical CMOS circuit power consumption parameter profiles.

circuit, and  $k$  is the transition activity factor which is defined as the average number of times the circuit makes a power consuming ( $0 \rightarrow 1$ ) transition in a single clock cycle. The parameters  $k$  and  $C$  are often combined in a single parameter,  $C^*$ , termed the effective capacitance of a design.

By employing appropriate design techniques both  $P_{sc}$  and  $P_l$  can be reduced to a negligible level, leaving  $P_s$  as the dominant factor of power consumption [1]. For this reason the rest of this paper will consider ways of reducing the effect of the constituent parameters of  $P_s$  (see eq. (2)) which can be exploited by the DSP engineer. Issues at the digital circuit level are not considered, such as the choice of static/dynamic logic or symmetric/non-symmetric organization of gates to overcome hazards, since these are covered in greater depth in papers which specifically target logic design (e.g. [1]).

### 3. Transformation techniques for reduced operation voltage

It can be seen from eq. (2) that the power consumed is a quadratic function of the operating voltage. This dependence on supply voltage has been verified for a number of CMOS logic circuits [2]. Hence reducing the operation voltage for such circuits could significantly reduce the consumed switching power. However, as verified in [2], reducing the operation voltage increases the delay of the different logic and memory elements in the circuit. This implies that operation under significantly reduced voltages is associated with an increase in delay and therefore reduced throughput. This reduction in throughput can be compensated by the application of a number of high level transformation techniques [3] which can be applied successively or individually to the DSP system under consideration for low power design [1, 4, 5]. The transformations can be applied to a DSP system at hardware level and/or a higher algorithmic level, represented using a data-flow

graph (DFG) [5]. Individual transformation techniques have specific advantages. However, they may have side-effect disadvantages, such as an increase in area, which could be overcome by applying another subsequent transformation technique [1, 4, 6]. The principal transformation techniques [3, 4] are as follows.

*Pipelining* [3,7,9]: a powerful transformation which can improve the performance of both general purpose and special purpose DSP systems. It involves the insertion of delay elements at specific points of a DFG [3] of an algorithm/structure, the aim being to increase the amount of concurrency. The application of pipelining to synchronous hardware architectures can allow operation with a faster system clock. However, pipelining increases both system latency and the number of delay elements in a system.

*Parallel processing* [1, 3, 7]: similar to pipelining in that it exploits parallelism in a system; however, here this is achieved by duplicating hardware sections in order to perform a number of similar tasks concurrently. The use of both pipelining and parallel processing in maintaining throughput under reduced supply voltage, thereby saving power, is demonstrated in [1] and [3].

*Retiming* [10–15]: the process of moving delays around a DFG of an algorithm/structure such that the overall computation is unaltered. It aims to move a computation in an attempt to reduce the critical path time, i.e. the path with the longest computation time without delays [15].

*Unfolding/folding* [3, 7, 13, 16–19]: used to unravel the hidden concurrency in a DFG. The iteration period of an  $N$  unfolded DFG is  $1/N$  times the critical path length of the unfolded DFG. By exploiting inter-iteration concurrency, unfolding can lead to a lower iteration period. Unfolding can also be used to improve processor utilization in time-constrained synthesis of DSP systems [3]. Folding is the reverse of unfolding,

but is more complex than unfolding since different transformations can result in different folded DFGs.

*Loop-unrolling (unwinding)* [1, 3, 4, 6, 20]: similar to the unfolding transformation [3]. It increases parallelism by utilizing more than one iteration of a loop [20]. Since it only duplicates the loop bodies, the iteration bound which is the loop with maximum ratio of computational operations to that of the number of delays cannot be changed by this transformation alone [6]. However, it enables other transformations which reveal large amount of concurrency [1, 4].

*Look-ahead* [3, 13, 21–23]: used to create additional concurrency by restructuring recursive DFGs [13]. This concurrency can then be exploited in the form of pipelining or parallel processing or both [3]. The look-ahead technique transforms any single-step recursions in the DFG to  $M$ -step recursions and therefore one iteration of the restructured recursion provides the results of  $M$  iterations of the original recurrence which will increase the number of operations in the DFG. These operations are performed using non-recursive DFGs [21]. Although look-ahead results in faster operations, it leads to increased circuit area which varies logarithmically with the speed-up factor [22]. Other types of look-ahead include: relaxed [3], scattered [3, 13, 22], and clustered look-ahead [13, 23] which differ in terms of implementation complexities and their application to different types of DFGs [13].

*Algebraic transformations* [3, 6, 24, 25]: based on simple algebraic properties. They are useful to move operations out of loops in DFGs in order to reduce the iteration bound. Their applicability is related to the number of operations in the DFG. The more operations in the DFG, the more likely it is that they can be applied [6]. The following are examples of algebraic transformations with their respective symbolic examples:

Distributivity:	$a.(b + c) \Rightarrow a.b + a.c$
Reverse distributivity:	$a.b + a.c \Rightarrow a.(b + c)$
Associativity:	$a.(b.c) \Rightarrow (a.b).c$
Commutativity:	$a.b \Rightarrow b.a$
Common sub-expression elimination (CSE):	$x = a + b, y = b + a \Rightarrow x = a + b, y = x$
Common sub-expression replication (CSR):	$x = a + b \Rightarrow x1 = a + b, x2 = a + b$

Another algebraic transformation is constant folding (also known as constant elimination) [6, 25] which involves the precomputation of constant values, e.g.  $[c = 1 + 2 \Rightarrow c = 3]$ . Among these algebraic transformations, associativity is the most important since it can directly move operations out of loops in a DFG in order to reduce the iteration bound [6]. The work in [6] classifies distributivity and CSR as enabling algebraic transformations and uses these to eliminate the inhibiting factors which disable associativity. For example, distributivity swaps addition and multiplication operations to put the same type of operations together, thus enabling associativity. Furthermore, since associativity cannot be applied to operations with multiple fanouts, CSR is applied first to remove this inhibiting factor by creating redundant operations.

In the case of non-recursive structures, such as FIR filters, one or more of the above transformations can be used for critical path time reduction. The order in which the transformations are applied is significant in achieving a reduced critical path time. However, in the case of recursive structures, such as IIR filters, either loop-unrolling or look-ahead can be used in

order to apply other transformations for critical path time reduction (as in the non-recursive case above) [1, 4]. The use of multiple transformations is demonstrated for a simple IIR filter in Fig. 2. The effect of transformations in power and circuit area has been investigated in [4, 5]. Figure 3 illustrates the power/area trade-offs obtained with different transformations for a number of signal processing circuits [5].

#### 4. Techniques for reducing the effective capacitance

In eq. (2), the parameters contributing to the effective capacitance of a DSP system are the activity factor  $k$  and the physical capacitance  $C$  of the system. This section considers a number of techniques which aid the DSP system designer in reducing  $k$  and  $C$  at architectural levels and circuit levels of the design.

##### 4.1 Reduced architectural complexity

Such techniques aim to reduce the number of complex area-consuming operations in a DSP system, hence reducing its overall capacitance. For example, the use of primitive operators [26] has proved effective in providing a significant reduction in computational complexity for digi-

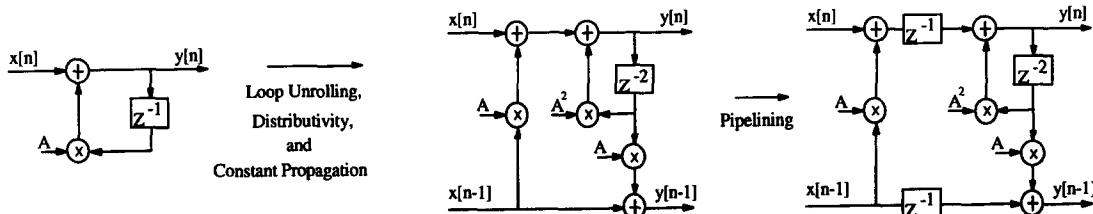


Fig. 2. The application of multiple transformations [4].

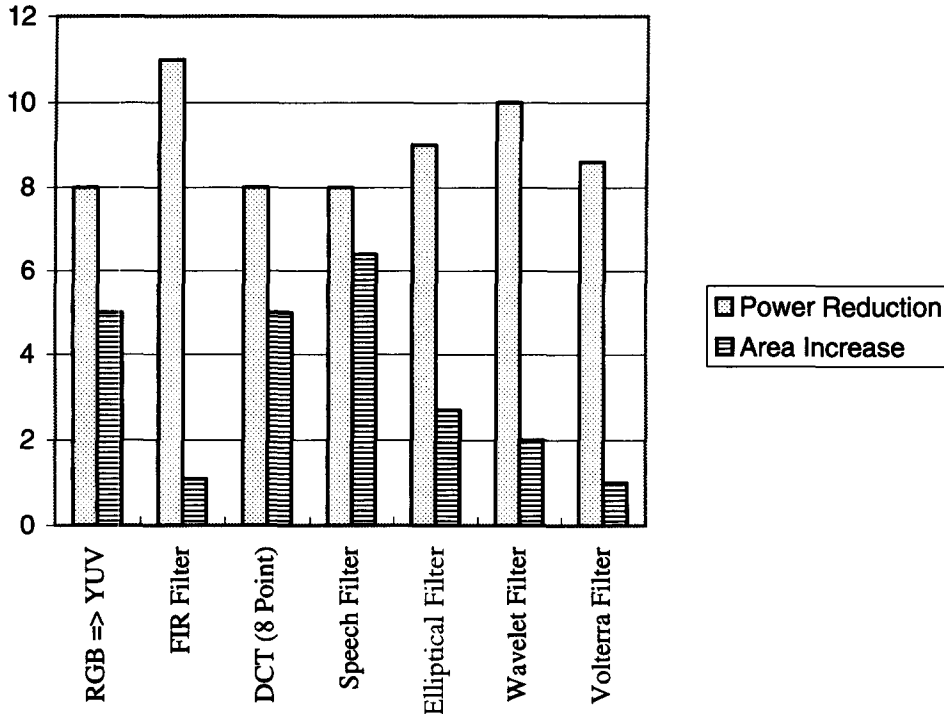


Fig. 3. Effect of transformations on example circuits [5].

tal filters. The above paper also provides references to other published work on reduced complexity techniques.

Transformation techniques such as linear transformation [27, 28] can be used for reducing the number of multipliers required for FIR filter design. This usually operates on the rows and columns of matrices in a state-space representation of the filter in order to reduce the numbers of additions and multiplications.

#### 4.2 Reduced switching activity

Another important factor influencing power consumption in signal processing circuits is the switching activity,  $k$ . The work in [29] provides an estimation of switching in the most commonly used adder circuits. Switching is estimated as the average number of transitions per addition observed for each adder in 50 000 randomly distributed input patterns with a

random distribution of carry inputs. This is illustrated graphically in Fig. 4.

The choice of data representation for the DSP system is one way of influencing the switching activity. For example, in two's complement representation the signal transition from positive to negative occurs frequently. This transition toggles all the bits and hence contributes significantly to the overall switching activity of the system. This is less so for the sign magnitude representation [1]. The choice of coding, wherever required in a DSP system, is important. The work in [30] demonstrates that Gray-coded instruction addressing results in a considerable reduction in switching activity. In [31], this type of addressing has been extended to microprocessors and the addressing is compared with cold scheduling, which is a software method which schedules instructions in a way that minimizes switching activity. With Gray-code

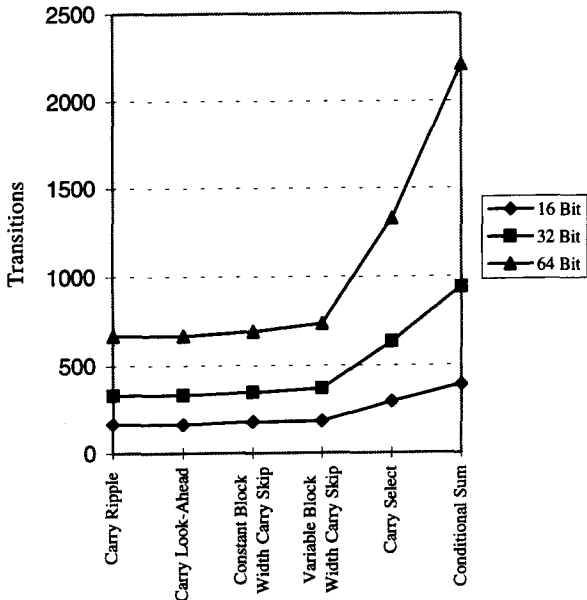


Fig. 4. Average number of transitions for main adder circuits [29].

addressing a 30–50% reduction in activity factor has been reported using a RISC microprocessor compared with normal binary coded addressing. This work also indicates that using cold scheduling with a VLSI-BAM, which is a RISC-like processor, a power reduction of 20–30% is achieved.

Another way of reducing the switching activity is by the choice of an appropriate ordering of the operations in a design. The work in [1] demonstrates that the switching activity, during an operation of ‘multiplication by a constant’ which is common in DSP, can be reduced by 30% by appropriate reordering. A number of transformations exist [32] that can affect the power consumption by reducing the average transition activity.

Reduced switching activity can also be achieved at a much higher level in the design hierarchy. An example is the choice of point-to-point data buses in favour of multiplexing [33].

#### 4.3 Dedicated DSP processor-based design

Programmability in general-purpose DSP processors can be the key to cost effectiveness but comes at a high energy cost relative to dedicated DSP chips. However, power saving techniques are emerging for both the DSP processor designer and its programmer. For example, work at AT&T laboratories [34] has shown that CPU power savings of up to 40% can be achieved by reordering the instruction sequence of the Intel 486DX2 processor. It was also shown that memory access instructions consume much more power than those for register access. Hence, reducing the number of memory operations can lead to power savings.

For the DSP processor designer, instruction-set optimization is one way of saving power. An example is to provide a special data path for often-executed instructions in order to reduce the capacitance switched for each execution of the instruction [35]. This approach has been utilized for modem and voice coder processors [36]. In [37], power consumption is analysed for search and sorting of software. An abstract machine with a simplified but realistic instruction set allowing reasonable implementation of several popular algorithms is defined. The energy requirement of each instruction is then estimated. The work concludes that sorting consumes approximately  $N$  times more energy than search, where  $N$  is the data size. For the same problem and the same data size, the energy requirement for different algorithms could also vary by orders of magnitude. Other work in the area of software-based power minimization includes [38–41].

#### 5. Power estimation

There is a need for accurate power estimates at all design levels. The use of effective power estimation models and techniques in modern design tools will provide the DSP system designer with additional flexibility since the designer will have access to accurate accounts of area, speed and

power. Power estimation tools developed so far can be classified according to either the design level at which estimation is made or the technique employed in power estimation.

### 5.1 Power estimation levels

Power estimation could be made at either circuit, gate, architecture, or algorithm levels. The most accurate power estimation can be made at the circuit level where the transistor is the primitive element. At this level two models of transistor behaviour can be used: device-level or less detailed switch-level. The accuracy of circuit-level tools is determined by the use of either of the models above. For example, SPICE-like device-level circuit tools (e.g. PowerMill [42], TSIM [43]) account not only for the charging/discharging of power but also power consumption due to short-circuit, leakage, and non-linear device currents. However, the price to pay for this level of detail is the lengthy simulation time which could be impractical for even moderately sized circuits. To ease this problem switch-level modelling could be employed. This sacrifices some accuracy for an improvement in speed. Examples of switch-level circuit tools include IRSIM [44] and LDS [45].

As mentioned above, circuit-level tools are either relatively time consuming or require a large memory overhead for relatively large circuits. For this reason, tools for gate-level power estimation have recently emerged. Some of these tools (e.g. Hercules [46], CEST [47]) require a transistor-level circuit description as an input which is then partitioned into a gate-level representation. Power estimation is then performed by reducing the gates to equivalent inverters [48]. Other tools (e.g. [49–51]) take a Boolean logic description of a circuit as an input, for which gate-level probability propagation formulas can be easily defined. Usually this is followed by the employment of probabilistic techniques to calculate power consumption. Although gate-level tools have speed advantage

over circuit-level tools they lack in accuracy, especially in the presence of correlated signals [48].

With the increase in the complexity of current VLSI designs, specifically those targeted for DSP applications, the top-down design approach is achieving dominance. The majority of such designs are initiated at a very high level, using tools such as VHDL [52], in terms of highly complex functional models and are expanded later on through the design hierarchy into gate and transistor levels, usually through synthesis tools [53–55]. Most strategic decisions are made at higher levels throughout the design hierarchy and, as recognized by designers, such decisions are usually the most effective for optimizing design aspects such as speed, area and power. For this reason, high-level (architectural and algorithmic levels) power estimation tools are becoming very important. Although work has already started for the development of high-level power estimation tools, most of these have severe limitations. For example, for architectural level tools the limitations are lack of accurate power models which consider aspects such as signal activities which contribute significantly to power consumption. In the algorithmic level, where the aim is to predict the amount of power consumed as a result of executing a given algorithm, power estimation must consider the hardware platform targeted for execution. For this reason current tools are usually restricted to specific architectures and may involve using architectural techniques [48].

A number of researchers have already attempted to model power consumption for digital VLSI chips. For example, Liu and Svensson [56] describe a method of power modelling of CMOS VLSI chips, at the architectural level, in which the power consumption estimate is based on gate count, memory size, logic and layout styles. The estimation process is fast but less accurate than gate-level simulation. The work in [57] describes a technique for estimating

power dissipation of signal processing algorithms/architectures implemented using processor arrays or multiprocessor schemes. The estimation technique uses constant technology-dependent values for the different array elements which consist of elements such as multipliers, static memory and I/O. In [58], the use of the technique for one-, two- and three-dimensional algorithms implemented on linear, hexagonal and cubic processor arrays is investigated and appropriate models are developed. Powell and Chau [59] develop a power dissipation model for common DSP algorithms and architectures implemented as special purpose VLSI chips and chip-sets. The impact of design style and I/O is accounted for and power is estimated in terms of high level parameters, such as word length and number of available processing elements. Rabaey and Guerra [60] demonstrate that only a few fundamental algorithmic properties, which include aspects such as nodes, word lengths, the number of I/O operations, and ratio of sample rate to the critical path, can be used in classifying algorithms and predicting how responsive they are to various hardware platforms. The high level models developed in [61] are based on high level statistics such as mean, variance and autocorrelation. The technique used is abstracted from the gate level using modules such as adders, multipliers and registers. Mehra and Rabaey [62] assume that the power consumed by an ASIC is due to contributions from hardware resources such as data paths, interconnects, control and memory. Statistical power models are developed for each of these by analysing the control/data flow graph of the design. In [62], a power estimation technique is proposed with the aim of presenting a methodology for developing and validating an instruction-level power model for any given processor. The technique is based on estimating the average power through the use of measured current per instruction and information such as clock period of the processor, and number of clock cycles taken by each instruction. The technique has been applied to two commercial

microprocessors: Intel 486DX2 and Fujitsu SPARCHlite 934. A power reduction of up to 40%, obtained by rewriting code using the information provided by the instruction level power model, is reported.

### **5.2 Power estimation techniques**

On the other hand, power estimation techniques could be classified into the following categories: simulation-based, probabilistic and statistical techniques. The following subsections will describe these.

#### **5.2.1 Simulation-based techniques**

Here, a randomly generated logic vector is applied to the primary inputs of the circuit and the corresponding current waveform, drawn from the power supply rail, is determined. After a number of such simulation runs the average of the resulting current waveforms is computed and then, with the knowledge of the supply voltage, used to calculate the average power dissipation [64].

In addition to its accuracy, the technique is easy to use and can be applied to any circuit, regardless of design style, architecture, technology, etc. However, it suffers from memory and speed constraints and therefore is not suitable for large circuits [35]. Since simulation results are directly related to the specific input vector sequences and complete information about the input patterns is required, this technique is said to be strongly pattern dependent [64, 65].

#### **5.2.2 Probabilistic techniques**

Instead of simulating the circuit for a large number of input vector sequences and then calculating the average power dissipation, signal and transition probabilities could be used to compute the average power [64]. Signal probabilities can be propagated throughout the circuit to compute transition probabilities at all internal and output nodes. Once the transition probabilities have been computed, then the average power can be calculated as:

$$P_{av} = \frac{V_{dd}^2}{2T_c} \sum_{i=1}^n C_i \cdot P_t(x_i)$$

where  $T_c$  is the clock period,  $V_{dd}$  is the supply voltage,  $C_i$  is the total capacitance at node  $x_i$ , and  $n$  is the total number of circuit nodes. The transition probability  $P_t$  represents both low-to-high and high-to-low transitions.

Only a single probabilistic analysis run is required and so probabilistic techniques allow fast power estimation. Figure 5 illustrates the main steps involved in probabilistic power estimation. Analysis results are still dependent on the supplied input probabilities but since complete and specific information is not required probabilistic techniques are said to be weakly pattern dependent [64]. However, they are not applicable directly to sequential circuits since it is very difficult to calculate the probability of the circuit being in each of its possible states [35, 66]. Nevertheless, a number of papers on sequential circuits have emerged [49, 66–68]. Probabilistic techniques usually use simplified delay models for the circuit components to ease the probabilistic analysis. Hence, their accuracy is limited by the quality of the delay models used [64]. For example, a zero-delay model does not account for power dissipation due to glitches. In order to achieve good accuracy, correlations among internal nodes must be taken into

account which comes at a high computational cost. In general, probabilistic techniques offer the advantage of speed but provide a less accurate estimation [65].

All reported probabilistic techniques utilize the knowledge of signal and transition probabilities. A signal probability  $P_s^1(x_i)$  at a node  $x_i$  is the probability that signal  $x$  has a logic value 1. On the other hand, a transition probability  $P_t^{01}(x_i)$  of a signal  $x_i$  at time  $t$  is the probability of a  $(0 \rightarrow 1)$  transition from  $t^-$  (just before  $t$ ) to  $t^+$  (just after  $t$ ). Other transitional probabilities  $(0 \rightarrow 0)$ ,  $(1 \rightarrow 1)$  and  $(1 \rightarrow 0)$  follow similarly. To calculate signal probabilities for the primary inputs of a circuit it is usually assumed that primary input signals are uncorrelated, and that each signal value is either 0 or 1, changing instantaneously at global clock edges [49]. Under the temporal independence assumption, that is, the values of the same signal in two consecutive clock cycles are independent, the transition probability at node  $x_i$  can be calculated from signal probabilities as:

$$P_t^{01}(x_i) = P_s^0(x_i) P_s^1(x_i) = P_s^1(x_i) [1 - P_s^1(x_i)]$$

A more accurate estimation of power dissipation can be obtained using probability waveforms. A probability waveform is a sequence of values indicating the probability of a signal being high for certain time intervals, and the probability that

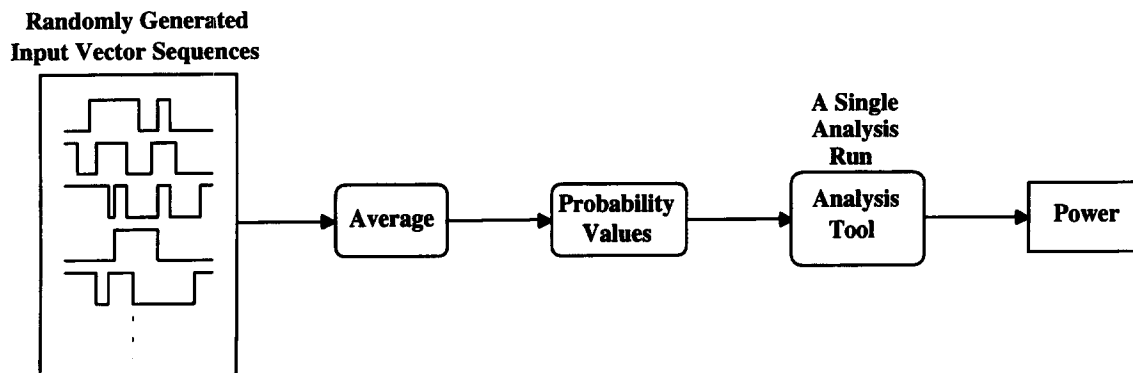


Fig. 5. Probabilistic simulation flow for power estimation.

it makes low-to-high transitions at specific points in time [50, 64, 69]. Given such waveforms at the primary inputs, they are propagated throughout the circuit in order to compute the corresponding probability waveforms at all the nodes.

Normally either random probabilities or signal probability waveforms are used to form input signal probabilities at the primary inputs. Propagating these probabilities is performed using specific formulas depending on the gate type [64, 70]. For example, for an AND gate this is the product of input probabilities. These formulas assume that signals are uncorrelated and no reconvergent fanout exists. Otherwise a number of other techniques are used. These include binary decision diagram (BDD) [71], disjoint cover [49, 72] and the cutting algorithm [70, 73].

It has been shown [51, 74] that exact signal probability calculation for a given function can be calculated by a linear traversal of a BDD representation of a logic function using the following expression:

$$P(\gamma) = P(x_1)P(f_{x_1}) + P(\bar{x}_1)P(f_{\bar{x}_1})$$

where  $f_{x_1}$  and  $f_{\bar{x}_1}$  are the cofactors of  $f$  with respect to  $x_1$  and  $\bar{x}_1$ , respectively. The probabilities of the cofactors of  $f_{x_1}$  and  $f_{\bar{x}_1}$  are expressed in the same way. For example, the Boolean function  $\gamma = x_1 \cdot x_2 + x_2 \cdot x_3$  can be represented by the BDD as shown in Fig. 6.

Each level in the BDD corresponds to a single variable and each node can branch in one of two directions, depending on the value of the relevant variable. For example, suppose that  $x_2 = 1$ ,  $x_1 = 0$  and  $x_3 = 1$ . To evaluate  $\gamma$ , start at the top node and branch to the right since  $x_2 = 1$ , then branch to the left since  $x_1 = 0$ , and finally branch to the right since  $x_3 = 1$  to reach the terminal node '1'. Thus the corresponding value of  $\gamma$  is 1. Similarly, if  $x_2 = 0$  then regardless of  $x_1$  and  $x_3$  the corresponding value of  $\gamma$  is '0'. Using

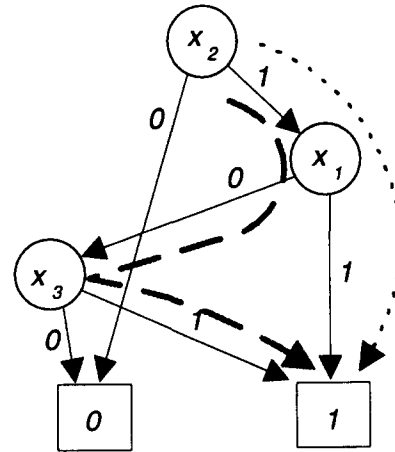


Fig. 6. BDD representation of  $\gamma$ .

the paths leading to the terminal node '1' in the BDD the signal probability of  $\gamma$  can be given as follows:

$$P(\gamma) = P(x_2)P(x_1) + P(x_2)(\bar{x}_1)P(x_3)$$

One disadvantage of this method is that it is computationally expensive. Since the BDD has to be built for the whole circuit, there may be cases where the BDD is too large to be processed in reasonable time. Hence, the use of the BDD approach is limited to moderately sized circuits [64].

The disjoint cover method is based on the principle that two cubes are said to be disjoint if they have no input combinations in common, i.e. do not intersect. A disjoint cover for a function is simply a truth table, where each cube in the cover is disjoint from every other cube [72]. In general, given a disjoint cover for a function  $\gamma$  of uncorrelated inputs described by signal probabilities, it is easy to compute the signal probability of  $\gamma$  by simply summing the probabilities of each of the cubes in the cover [49].

The cutting algorithm computes signal probability bounds, rather than the exact figure. Its objective is to turn the combinational network into a tree, by cutting reconvergent fanout

branches, and inserting equivalent bounds at the cut points, which will guarantee that all the signal probability bounds computed on this tree will enclose the true values. The advantage of doing this is a reduction in computational complexity (speed and memory); the disadvantage is that the output is not an exact figure but just a bound [70, 73].

Another probabilistic approach is the use of transition density which is defined as the average switching rate at a circuit node. This is described in detail in [51, 75].

### 5.2.3 Statistical techniques

Statistically-based techniques combine the accuracy of simulation-based techniques with the weak pattern dependence of probabilistic techniques [65]. Randomly generated input vectors are applied to the circuit and the power being consumed is monitored using a timing or logic simulator. This is continued until the power converges to the true average power. In order to determine when the measured power has converged close enough to the true average power statistical mean estimation techniques are used and the decision when to stop is made based on a stopping criterion [64]. Examples of this can be found in [65, 76, 77].

## 6. Conclusions and future design directions

In this paper we have highlighted the main work which has been carried out to date in low power DSP system design. However, interest in this area is recent and significant future developments are both needed and can be expected. Some directions that these developments could take are as follows:

- (1) The full exploration of the transformation techniques for realistic DSP applications.
- (2) The use of the various low power design techniques is a combinatorial problem and

thus enables the use of AI-based techniques, such as genetic algorithms [78], and heuristics. Work has already started in this area [79].

- (3) The techniques of reordering general purpose CPU instructions can be applied to dedicated DSP processors.
- (4) More comprehensive low power investigations of coding and signal-representation techniques in DSP systems.
- (5) Extending the number of power estimation models, such as in [56], to cover the full range of DSP functional operators.
- (6) The full development of probabilistic and statistical based power estimation techniques and their inclusion in commercial VLSI design tools.

## References

- [1] A.P. Chandrakasan and R.W. Brodersen, Minimising power consumption in digital CMOS circuits, *Proc. IEEE*, 83(4) (Apr. 1995) 498–523.
- [2] A.P. Chandrakasan, S. Sheng and R.W. Brodersen, Low-power CMOS digital design, *IEEE J. Solid-State Circuits*, 27(4) (Apr. 1992) 473–484.
- [3] K.K. Parhi, High-level algorithm and architecture transformations for DSP synthesis, *J. VLSI Signal Processing*, 9 (1995) 121–143.
- [4] A.P. Chandrakasan, M. Potkonjak, R. Mehra, J.M. Rabaey and R.W. Brodersen, Optimizing power using transformations, *IEEE Trans. CAD*, 14(1) (Jan. 1995) 12–31.
- [5] A.P. Chandrakasan, M. Potkonjak, J.M. Rabaey and R.W. Brodersen, HYPER-LP: a system for power minimization using architectural transformations, *Proc. IEEE/ACM Int. Conf. CAD*, 1992, pp.300–303.
- [6] S.-H. Huang and J.M. Rabaey, Maximizing the throughput of high performance DSP applications using behavioral transformations, *Proc. EDAC-EUROASIC '94*, Paris, France, March 1994, pp. 25–30.
- [7] K. Hwang and F.F. Briggs, *Computer Architecture and Parallel Processing*, McGraw-Hill, Singapore, 1985.
- [8] P.J. Duncan, S. Swamy and R. Jain, Low-power DSP

- circuit design using bit-level pipelined maximally-parallel architectures, *Proc. 1st Symp. Integrated Systems*, March 1993, pp. 266–275.
- [9] J. Chung and K.K. Parhi, Pipelining of lattice IIR digital filters, *IEEE Trans. Signal Processing*, 42(4) (Apr. 1994) 751–761.
- [10] C.E. Leiserson, F. Rose and J. Saxe, Optimizing synchronous circuitry by retiming, *Proc. 3rd Caltech Conf. VLSI*, Pasadena, CA, March 1983, pp. 87–116.
- [11] B. Lockyear and C. Ebeling, The practical application of retiming to the design of high-performance systems, *Proc. IEEE/ACM Int. Conf. CAD*, Santa Clara, CA, Nov. 1993, pp. 288–295.
- [12] M. Potkonjak and J.M. Rabaey, Retiming for scheduling, in H.S. Moscovitz, K. Yao and R. Jain (eds), *VLSI Signal Processing IV*, IEEE Press, New York, 1991, pp. 23–32.
- [13] K.K. Parhi, Algorithm transformation techniques for concurrent processors, *Proc. IEEE*, 77(12) (Dec. 1989) 1879–1895.
- [14] J. Monteiro, S. Devedas and A. Ghosh, Retiming sequential circuits for low power, *Proc. IEEE/ACM Int. Conf. CAD*, Santa Clara, CA, Nov. 1993, pp. 398–402.
- [15] L.E. Lucke and K.K. Parhi, Data-flow transformations for critical path time reduction in high-level DSP synthesis, *IEEE Trans. CAD*, 12(7) (July 1993) 1063–1068.
- [16] K.K. Parhi, A systematic approach for design of digit-serial signal processing architectures, *IEEE Trans. Circuits Syst.*, 38(4) (Apr. 1991) 358–375.
- [17] K.K. Parhi, C. Wang and A.P. Brown, Synthesis of control circuits in folded pipelined DSP architectures, *IEEE J. Solid-State Circuits*, 27(1) (Jan. 1992) 29–43.
- [18] K.K. Parhi and D.G. Messerschmitt, Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding, *IEEE Trans. Computers*, 40(2) (Feb. 1991) 178–195.
- [19] L. Jeng and L. Chen, Rate-optimal DSP synthesis by pipeline and minimum unfolding, *IEEE Trans. VLSI Systems*, 2(1) (March 1994) 81–89.
- [20] C. Wang and K.K. Parhi, High-level DSP synthesis using concurrent transformations, scheduling, and allocation, *IEEE Trans. CAD*, 14(3) (March 1995) 274–295.
- [21] G.P. Fettweis and L. Thiele, Algebraic recurrence transformations for massive parallelism, in K. Yao, R. Jain, W. Przytula and J. Rabaey (eds), *VLSI Signal Processing V*, IEEE Press, New York, 1992, pp. 332–341.
- [22] K.K. Parhi, Impact of architecture choices on DSP circuits, *Proc. 1992 IEEE Region 10 Int. Conf. Computers, Communications and Automation Towards 21st Century (TENCON '92)*, Melbourne, Australia, Nov. 1992, pp. 784–788.
- [23] K.K. Parhi and D.G. Messerschmitt, Pipeline interleaving and parallelism in recursive digital filters — Part I: Pipelining using scattered look-ahead and decomposition, *IEEE Trans. Acoustic, Speech, Signal Processings*, 37(7) (July 1989) 1099–1117.
- [24] M. Potkonjak and J.M. Rabaey, Optimizing resource utilization using transformations, *IEEE Trans. CAD*, 13(3) (March 1994) 277–292.
- [25] J. Bhaskar and H.C. Lee, An optimizer for hardware synthesis, *IEEE Design Test Computers*, (Oct. 1990) 20–36.
- [26] D.R. Bull and D.H. Horrocks, Primitive operator digital filters, *IEE Proc.-G*, 12(7) (June 1991) 401–412.
- [27] M. Wintermantel and E. Lueder, Increasing the speed and saving multipliers in block parallel digital filters by a linear transformation, *Proc. IEEE ISCAS*, London, 1994, pp. 81–84.
- [28] E. Lueder, Generation of equivalent block parallel digital filters and algorithms by a linear transformation, *Proc. IEEE ISCAS*, Chicago, 1993, pp. 495–498.
- [29] T.K. Callaway and E.E. Swartzlander, Optimizing adders for WSI, *Proc. IEEE Int. Conf. Wafer Scale Integration*, 1992, pp. 251–260.
- [30] S. Wuytack, F. Catthoor, F. Franssen, L. Nachtergaele and H. DeMan, Global communication and memory optimizing transformations for low power systems, *1994 Int. Workshop on Low Power Design*, Napa Valley, CA, Apr. 1994.
- [31] C.L. Su, C.Y. Tsui and A.M. Despain, Low power architecture design and compilation techniques for high-performance processors, *IEEE COMPCON*, Feb. 1994, pp. 489–498.
- [32] R.W. Brodersen, A.P. Chandrakasan and S. Sheng, Low-power signal processing systems, in K. Yao, R. Jain, W. Przytula and J. Rabaey (eds), *VLSI Signal Processing*, IEEE Press, New York, 1992, pp. 3–13.
- [33] G.M. Blair, Designing low-power digital CMOS, *Electron. Commun. Eng. J.*, 6(5) (Oct. 1994) 229–236.
- [34] D. Bursky, Power-reduction schemes promise cool digital ICs, *Electronic Design J.*, 43(1) (Jan. 1995) 51–65.
- [35] D. Singh, J.M. Rabey, M. Pedram, F. Catthoor, S. Rajgopal, N. Sehgal and T.J. Mozdzen, Power conscious CAD tools and methodologies: a perspective, *Proc. IEEE*, 83(4) (Apr. 1995) 570–593.
- [36] Y. Be'ery, S. Berger and B. Ovidia, An application specific DSP for portable applications, *Proc. VLSI Signal Processing Workshop*, Veldhoven, The Netherlands, 1993, pp. 48–56.
- [37] P.W. Ong and R.H. Yan, Power-conscious software design: a framework for modelling software on hardware, *Proc. 1994 IEEE Symp. Low Power Electronics*, San Diego, CA, Oct. 1994, pp. 36–37.

- [38] J.H. Snyder, J.B. McKie and B.N. Locanthi, Low-power software for low-power people, *Proc. 1994 IEEE Symp. Low Power Electronics*, San Diego, CA, Oct. 1994, pp. 32–35.
- [39] V. Tiwari, S. Malik and A. Wolfe, Compilation techniques for low energy: an overview, *Proc. 1994 IEEE Symp. Low Power Electronics*, San Diego, CA, Oct. 1994, pp. 38–39.
- [40] A. Alomary, T. Nakata and Y. Honma, An ASIP instruction set optimization algorithm with functional module sharing constraint, *Proc. IEEE/ACM Int. Conf. CAD*, Santa Clara, CA, Nov. 1993, pp. 526–532.
- [41] M. Corazao, M. Khalaf, L. Guerra, M. Potkonjak and J.M. Rabaey, Instruction set mapping for performance optimization, *Proc. IEEE/ACM Int. Conf. CAD*, Santa Clara, CA, Nov. 1993, pp. 518–521.
- [42] A. Deng, Power analysis for CMOS/BiCMOS Circuits, *1994 Int. Workshop on Low Power Design*, Napa Valley, CA, Apr. 1994, pp. 3–8.
- [43] A. Deng, Y. Shiao and K. Loh, Time domain current waveform simulation of CMOS circuits, *Proc. IEEE/ACM Int. Conf. CAD*, 1988, pp. 208–211.
- [44] A. Salz and M. Horowitz, IRSIM: an incremental MOS switch-level simulator, *Proc. 26th ACM/IEEE Design Automation Conf.*, 1989, pp. 173–178.
- [45] R. Tjarnstrom, Power dissipation estimate by switch level simulation, *Proc. IEEE Int. Symp. Circuits and Systems*, May 1989, pp. 881–884.
- [46] A. Tyagi, Hercules: a power analyzer for MOS VLSI circuits, *Proc. IEEE/ACM Int. Conf. CAD*, Nov. 1987, pp. 530–533.
- [47] S. Chowdhury and J. Barkatullah, Estimation of maximum currents in MOS IC logic circuits, *IEEE Trans. CAD*, 9(6) (June 1990) 642–654.
- [48] P.E. Landman, Low-power architectural design methodologies, *Memorandum No. UCB/ERL M94/62*, College of Engineering, University of California, Berkeley, CA.
- [49] A. Ghosh, S. Devedas, K. Keutzer and J. White, Estimation of average switching activity in combinational and sequential circuits, *Proc. 29th ACM/IEEE Design Automation Conf.*, Anaheim, CA, June 1992, pp. 253–259.
- [50] C.-Y. Tsui, M. Pedram and A. Despain, Efficient estimation of dynamic power consumption under a real delay model, *Proc. IEEE/ACM Int. Conf. CAD*, Santa Clara, CA, Nov. 1993, pp. 224–228.
- [51] F.N. Najm, Transition density, a stochastic measure of activity in digital circuits, *Proc. 28th ACM/IEEE Design automation Conf.*, 1991, pp. 38.1.644–649.
- [52] R. Camposano, L.F. Saunders and R.M. Tabet, VHDL as input for high-level synthesis, *IEEE Design Test Computers*, 8 (March 1991) 43–49.
- [53] R. Camposano, From behavior to structure: high-level synthesis, *IEEE Design Test Computers*, 7 (Oct. 1990) 8–19.
- [54] J.M. Rabaey, C. Chu, P. Hoang and M. Potkonjak, Fast prototyping of datapath-intensive architectures, *IEEE Design Test Computers*, 8 (June 1991) 40–51.
- [55] N. Kumar, S. Katkooi, L. Rader and R. Vemuri, Profile-driven behavioral synthesis for low-power VLSI systems, *IEEE Design Test Computers*, 12(3) (Fall 1995) 70–84.
- [56] D. Liu and C. Svensson, Power consumption estimation in CMOS VLSI chips, *IEEE J. Solid-State Circuits*, 29(6) (June 1994) 663–670.
- [57] S.R. Powell and P.M. Chau, Estimating power dissipation of VLSI signal processing chips: the PFA technique, in H.S. Moscovitz, K. Yao and R. Jain (eds), *VLSI Signal Processing IV*, IEEE Press, New York, 1991, pp. 250–259.
- [58] P.M. Chau and S.C. Powel, Power dissipation of VLSI array processing systems, *J. VLSI Signal Processing*, 4 (1992) 199–212.
- [59] S.R. Powell and P.M. Chau, A model for estimating power dissipation in a class of DSP VLSI chips, *IEEE Trans. Circuits Syst.*, 38(6) (June 1991) 646–650.
- [60] J.M. Rabaey and L.M. Guerra, Exploring the architecture and algorithmic space for signal processing applications, *Proc. Int. Conf. on VLSI and CAD*, Taejon, Korea, Nov. 1993, pp. 315–319.
- [61] P.E. Landman and J.M. Rabaey, Power estimation for high level synthesis, *Proc. EDAC-EUROASIC '93*, Paris, France, Feb. 1993, pp. 361–366.
- [62] R. Mehra and J.M. Rabaey, Behavioral level power estimation and exploration, *1994 Int. Workshop on Low Power Design*, Napa Valley, CA, Apr. 1994.
- [63] V. Tiwari, S. Malika and A. Wolfe, Power analysis of embedded software: a first step towards software power minimization, *IEEE Trans. VLSI Syst.*, 2(4) (Dec. 1994) 437–445.
- [64] F.N. Najm, A survey of power estimation techniques in VLSI circuits, *IEEE Trans. VLSI Syst.*, 2(4) (Dec. 1994) 446–455.
- [65] R. Burch, F.N. Najm, P. Yang and T.N. Trick, A Monte Carlo approach for power estimation, *IEEE Trans. VLSI Syst.*, 1(1) (March 1993) 63–71.
- [66] C.Y. Tsui, J. Monteiro, M. Pedram, S. Devedas and A.M. Despain, Power estimation methods for sequential logic circuits, *IEEE Trans. VLSI Syst.*, 3(3) (Sept. 1995) 404–415.
- [67] L. Benini and G.D. Micheli, State assignment for low power dissipation, *IEEE J. Solid-State Circuits*, 30(3) (March 1995) 258–267.
- [68] K. Roy and S.C. Prasad, Circuit activity based logic synthesis for low power reliable operations, *IEEE Trans. VLSI Syst.*, 1(4) (Dec. 1993) 503–513.

- [69] F.N. Najm, R. Burch, P. Yang and I.N. Hajj, Probabilistic simulation for reliability analysis of CMOS VLSI circuits, *IEEE Trans. CAD*, 9(4) (Apr. 1990) 439–450.
- [70] B. Krishnamurthy and I.G. Tollis, Improved techniques for estimating signal probabilities, *IEEE Trans. Computers*, 38(7) (July 1989) 1041–1045.
- [71] R.E. Bryant, Graph-based algorithms for Boolean function manipulation, *IEEE Trans. Computers*, C-35(8) (Aug. 1986) 677–691.
- [72] S. Devadas, K. Keutzer and J. White, Estimation of power dissipation in CMOS combinational circuits using Boolean function manipulation, *IEEE Trans. CAD*, 11(3) (March 1992) 373–383.
- [73] J. Savir, G.S. Ditlow and P.H. Bardell, Random pattern testability, *IEEE Trans. Computers*, C-33(1) (Jan. 1984) 79–89.
- [74] S. Chakravarty, On the complexity of using BDDs for the synthesis and analysis of Boolean circuits, *Proc. 27th Annual Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Sept. 1989, pp. 730–739.
- [75] F.N. Najm, Transition density: a new measure of activity in digital circuits, *IEEE Trans. CAD*, 12(2) (Feb. 1993) 310–323.
- [76] P. Vanoostende, P. Six, J. Vandewalle and H.J. De Man, Estimation of typical power of synchronous CMOS circuits using a hierarchy of simulators, *IEEE J. Solid-State Circuits*, 28(1) (Jan. 1993) 26–39.
- [77] C. Huizer, Power dissipation analysis of CMOS VLSI circuits by means of switch-level simulation, *Proc. IEEE European Solid-State Circuits Conf. '90*, Grenoble, France, 1990, pp. 61–64.
- [78] D.E. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison Wesley, 1989.
- [79] T. Arslan, E. Ozdemir, M.S. Bright and D.H. Horrocks, Genetic synthesis techniques for low-power digital signal processing circuits, *IEE Colloquium on Digital System Design Using Synthesis Techniques*, London, 1996, pp. 7/1–7/5.