Relating Data Characteristics to Transition Activity in High-Level Static CMOS Design

Russell Henning and Chaitali Chakrabarti *

Department of Electrical Engineering
Arizona State University, Tempe, AZ 85287-5706
rhenning@asu.edu, chaitali@asu.edu

Abstract

Significant power reduction can be obtained in the datapath of a CMOS VLSI circuit if data characteristics are carefully exploited. An improved approach that achieves such reduction by using a new model relating important data characteristics to the transition activity in static CMOS circuits is presented. Specifically, relationships between fixed-point, two's complement data and $0 \rightarrow 1$ transition activity in static CMOS circuits are identified. Models for computing transition activity in terms of a set of statistical parameters are developed, and their performance compared with the Dual Bit Type model. Then, the use of the relationships and models to analyze and significantly reduce $0 \rightarrow 1$ transition activity with little computational effort is illustrated by several, high-level synthesis examples.

1. Introduction

Though the study of power reduction techniques for VLSI implementations has received a great deal of attention in recent years, the majority of study has neglected the relationship between power dissipation and the data being processed. As a result, VLSI designers typically devote little attention to exploitation of data characteristics to reduce power. This paper addresses this issue by developing and demonstrating the importance of general relationships between fixed-point, two's complement data characteristics and power dissipation in static CMOS circuits.

Power reduction can be quite significant if data characteristics are exploited. The equation for average switching power consumed at a node in a conventional static CMOS circuit is

$$P_{switching} = \alpha_{0\rightarrow 1} C_L V_{dd}^2 f_{clk}.$$

Assuming that at most one transition takes place in a single clock cycle, data characteristics can be used to reduce the $0 \rightarrow 1$ transition activity, $\alpha_{0\rightarrow 1}$ (defined as the average number if power consuming transitions per clock cycle made by a node). Reducing load capacitance, C_L, supply voltage, V_{dd}, or clock frequency, f_{clk}, can also significantly reduce the switching power and have been the focus of much study. However, power reduction obtained by reducing $\alpha_{0\rightarrow 1}$ is desirable because it is fairly independent of integration technology, can be achieved without reducing processing speed, and more importantly, can be achieved on top of the other reduction techniques.

The conventional approach to finding a low power implementation in terms of activity is to effectively estimate $\alpha_{0\rightarrow 1}$ throughout the data path for all possible designs and choose the one with the lowest power [1]. The problem with this methodology is that for practical applications, there are too many design possibilities to consider. Therefore, other approaches have emerged that attempt to reduce this complexity by looking at a few choices, one control step at a time [2]. However, such methods potentially exclude good designs, because they narrow down the design space in an arbitrary manner. This problem can be avoided if data characteristics are used to efficiently narrow down the design space (ie., without throwing away good design possibilities.)

Relationships between data characteristics and $\alpha_{0\rightarrow 1}$ in the datapath were first illustrated for special cases [3], [4], [5]. Then, more general models for relating data characteristics and $\alpha_{0\rightarrow 1}$ began to appear [6], [7], [8]. However, these models are targeted for high-level power estimation accuracy (under certain assumptions), and do not analyze the causes of $\alpha_{0\rightarrow 1}$. More general relationships need to be defined and modeled specifically to aid in finding low $\alpha_{0\rightarrow 1}$ high-level designs.

The relationships presented in this paper are general and can be used to identify and exploit data characteristics to reduce $\alpha_{0\rightarrow 1}$ in the data path during high-level synthesis. Statistical models are developed for $\alpha_{0\rightarrow 1}$ in terms of a new set of parameters that are meant for analyzing the causes of $\alpha_{0\rightarrow 1}$. These models are shown to perform as well as the basic Dual Bit Type (DBT) activity model of [6] and better for some special cases. The special cases include nonstationary data such as speech samples, data that are slowly varying with high average magnitude, and data that contain truncated consecutive values. Although the shortcomings of the DBT model could possibly be handled, the real benefit of our model is its ability to easily analyze the causes of $\alpha_{0\rightarrow 1}$.

In this paper, the model and the manner in which the model can be used to analyze and significantly reduce $\alpha_{0\rightarrow 1}$ on highly capacitive buses and functional unit inputs is illustrated with some high-level synthesis examples. While

*This work was supported by the Center for Low Power Electronics.

our method does not specify the exact reduction in the power consumption, we operate under the assumption that reduction in $\alpha_{0\rightarrow1}$ on bus lines and functional unit inputs translates to reduction in power consumption. Glitching is assumed to be zero at these nodes.

This paper is organized as follows. In Section 2, the transition parameters that affect $\alpha_{0\rightarrow1}$, and the model that relates these parameters to $\alpha_{0\rightarrow1}$ are described. Examples illustrating the use of this knowledge to efficiently reduce $\alpha_{0\rightarrow1}$ during high-level synthesis is described in Section 3.

2. Transition Activity Model

In our previous work in [9], we identified three main types of two's-complement, fixed point data that can significantly affect $0 \rightarrow 1$ transition activity of a bus or functional unit input. The first type is characterized by a significant amount of sign extension evident in consecutive words appearing at a multi-bit node. The second type tends to result when consecutive words differ by a very small magnitude. The third type has consecutive words that are significantly truncated. To understand the effect these types of data have on power dissipation, we propose the use of four basic parameters. These parameters are very intuitive, because they are derived from the simple case of a single word transition at a multi-bit node. These single transition parameters are described next.

Intersecting sign bits: Sign bits are consecutive bits following the most significant bit of a fixed-point, two's complement value that are the same as the most significant bit. The number of intersecting sign (IS) bits between the two values involved in the transition is the number of consecutive bits of a multi-bit node, beginning with the most significant bit, that make the same transition as the most significant bit. Two examples of single transitions are shown in Figure 1. Each transition shows consecutive binary values occurring at a 16-bit node driven by glitch-free static CMOS circuitry. The binary numbers are fixed-point, two's-complement representations of decimal values between -1 and 1 (also shown in Figure 1). For the transition in Figure 1a, the number of IS bits is five, while in Figure 1b, it is four.

Sign transitions: To characterize the effect of significant sign bit-related activity in a single transition, identification of the sign transition is the next step. As can be seen in Figure 1b, if the sign transition is $+ \rightarrow -$, all the IS bits will consume power in that transition. If the single transition is anything else, they will not consume power. So, the second single-transition parameter is just a flag that is 1 for a $+ \rightarrow -$ transition and 0 for any other transition.

Intersecting truncation bits: Truncation bits are defined as the consecutive bits beginning with the significant bit (LSB) that are zero. For instance, the number of truncation bits in Figure 1a is four in the previous value and is three in the current value. Similar to sign bits, the single-transition parameter associated with truncation bits is the number of intersecting truncation (IT) bits between consecutive node values. This parameter is defined as the min-

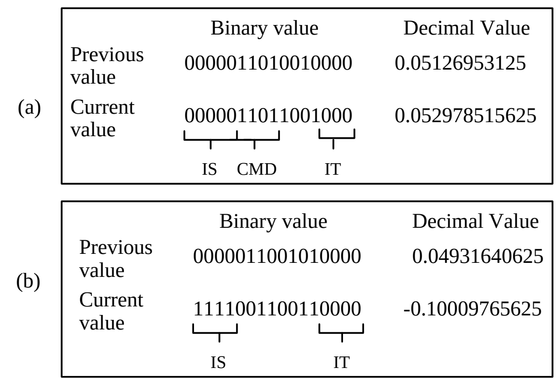


Figure 1. Transition between (a) two positive 16-bit values and (b) a positive and a negative 16-bit value.

imum of the number of truncation bits in the two values involved in the transition. In Figure 1a, the number of IT bits is three. In Figure 1b, it is four. The property of IT bits that relates them to significant power reduction is they all make $0 \rightarrow 0$ transitions (they never make power consuming $0 \rightarrow 1$ transitions).

Consecutive Matching Data Bits: Data bits include all other bits of a value that are not sign or truncation bits. In Figure 1a, the previous value has seven data bits and the current value has eight data bits. The single-transition parameter associated with data bits is the number of consecutive matching data (CMD) bits between consecutive node values. This parameter is defined as the number of consecutive data bits that match in value and position between the two values involved in a transition beginning with the most significant data bit of both values. In Figure 1a, the number of CMD bits is four, whereas in Figure 1b, it is zero. The property of CMD bits that relates them to significant power reduction is they all make $0 \rightarrow 0$ or $1 \rightarrow 1$ transitions. They never make power consuming $0 \rightarrow 1$ transitions.

Multiple transition parameters: With all the single transition parameters defined, multiple transition parameters are found by simply averaging each of the single-transition parameters over multiple transitions. The resulting parameters are denoted α_{OS}, n_{IS}, n_{CMD}, and n_{IT}. α_{OS} is the fraction of transitions that are opposite sign ($+ \rightarrow -$ or $- \rightarrow +$) transitions. Thus α_{OS} is approximately two times the single transition parameter averaged over all transitions. n_{IS} is the average number of IS bits per transition. n_{CMD} is the average number of CMD bits per transition. n_{IT} is the average number of IT bits per transition. These parameters can be used to estimate $\alpha_{0\rightarrow1}$, the average $0 \rightarrow 1$ transition activity per bit of an N bit node, through the model

$$\begin{aligned}\hat{\alpha}_{0\rightarrow1} &= \frac{1}{4N}\{N + \frac{1}{3} - n_{IT} + \alpha_{OS}(n_{IS} - \frac{1}{3}) \\ &\quad +(1-\alpha_{OS})(\frac{3}{2} - n_{CMD} - n_{IS})\} \qquad (1)\end{aligned}$$

This model was found through experimentation with a wide variety of synthetic data streams containing the char-

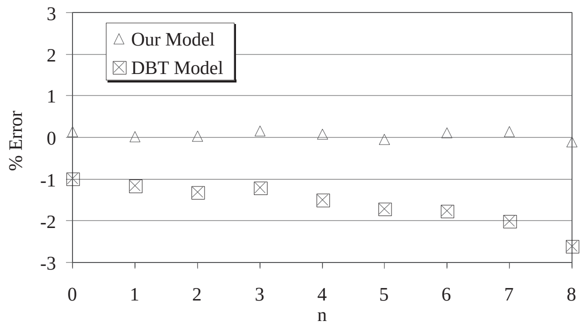


Figure 2. Activity estimation error as a percentage of actual activity for a stream of positive, uniformly random 16-bit data scaled by 2^{-n}.

acteristics of interest. Its consistent performance has been verified with real data such as speech samples, music samples, and data streams throughout a speech codec implementation. The $\frac{1}{3}$ and $\frac{3}{2}$ terms are approximations that have been rounded to the nearest simple fraction for model compactness. Other models presented in this section were found in a similar manner. There is a qualitative explanation for all of these models. From N, the average number of bits that are known to make $0 \to 0$ or $1 \to 1$ transitions are deducted and the average number of bits that are known to make $0 \to 1$ or $1 \to 0$ transitions are added. The resulting number of bits is multiplied by $1/4$, the probability that a bit makes a $0 \to 1$ transition when all four transitions are equally likely, and normalized by N. In terms of Average Hamming Distance (AHD), $\alpha_{0\to 1} \approx \frac{1}{2N} AHD$.

2.1. Comparison with the DBT model

Of all the existing activity models, the DBT model is the most practical for design analysis, and so we compare our model against it. The DBT model uses probabilistic parameters (mean, variance, and covariance) to characterize data-related activity and has been quite successful in high-level power estimation [6]. In this section, we show that our model does as well as the DBT model and, in fact, does better for some special cases. However, the strength of our model lies in the fact that the statistical parameters are intuitive and easy to use and better suited for making high-level design decisions.

Sign bit activity

Assume that $n_{CMD} < 1$ and $n_{IT} < 0.5$ for a practical stream of data encountered by an N-bit node. A stream of this type may contain significant amounts of downscaled data or data that does not require the full N-bit range of the representation. Since variations in n_{CMD} and n_{IT} are relatively insignificant at these low levels, a simpler model than (1) can be employed. The average $\alpha_{0\to 1}$ over all bits of an N-bit node in this case can be modeled as

$$\hat{\alpha}_{0\to 1} = \frac{1}{4N}\{N + (1-\alpha_{OS})(\frac{1}{3} - n_{IS}) + \alpha_{OS}(n_{IS} - \frac{1}{3})\} \quad (2)$$

Focusing on reducing α_{OS} may produce the greatest reduction in $\alpha_{0\to 1}$ for the least amount of work since α_{OS} has a scaling effect on the contributions of the other terms.

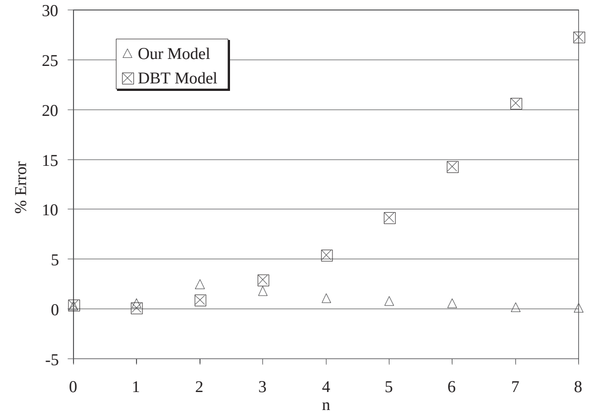


Figure 3. Activity estimation error as a percentage of actual activity for a stream of accumulation register input during accumulation of uniformly random 16-bit data scaled by 2^{-n}.

To demonstrate the accuracy of this model for activity related to sign bits, we derived the estimates for a data stream with 100,000 values that consists of uniform randomly distributed 16-bit values scaled by 2^{-n}. Figure 2 compares our model to the DBT model. Our model produces $\alpha_{0\to 1}$ estimates well within 1% of the actual value for all n shown. The DBT model is a little less exact than our model, however it remains within 3% of the actual value for all n shown. Good accuracy is expected of the DBT model in this case since the main effect it models is that of sign bit activity.

Data bit activity

Assume that the data stream being considered had $n_{CMD} \geq 1$ and $n_{IT} < 0.5$. Such a stream of data can result, for example, at an accumulation register or for very slowly varying data with a large average magnitude. In this case $\alpha_{0\to 1}$ can be modeled as

$$\hat{\alpha}_{0\to 1} = \frac{1}{4N}\{N + (1-\alpha_{OS})(\frac{3}{2} - n_{CMD} - n_{IS}) + \alpha_{OS}(n_{IS} - \frac{1}{3})\} \quad (3)$$

Figure 3 shows the performance of our model compared to the DBT model for accumulation register input, where different scaling factors have been applied to the input data being accumulated. The data stream which consists of 100,000 values of uniform randomly distributed 16-bit values were scaled by 2^{-n} and accumulated. The DBT model stays within 10% of the actual value up to about $n = 5$, and degrades for $n > 5$. Thus, the DBT model works well for data with several CMD bits but performs poorly when large numbers of small values are accumulated. On the other hand, our model continues to produce reliable estimates for all values of n.

Truncation Bit Activity

Next, assume that the data stream being considered had $n_{CMD} < 1$ and $n_{IT} > 0.5$. A stream of this type can result, for example, if data is significantly truncated or up-scaled.

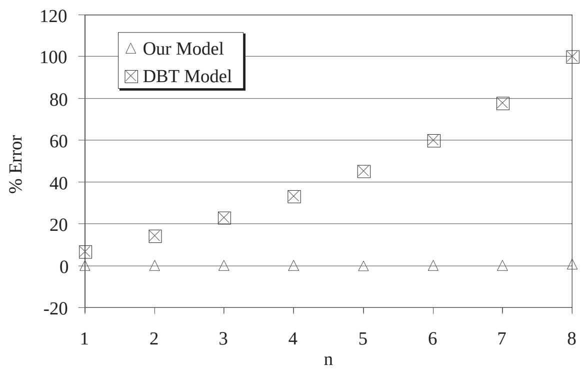


Figure 4. Activity estimation error as a percentage of actual activity for a stream of uniformly random 16-bit data truncated by n bits.

In this case $\alpha_{0\rightarrow 1}$ can be modeled as

$$\hat{\alpha}_{0\rightarrow 1} = \frac{1}{4N}\{N+\frac{1}{3}-n_{IT}+(1-\alpha_{OS})(\frac{1}{3}-n_{IS})+\alpha_{OS}(n_{IS}-\frac{1}{3})\} \quad (4)$$

Figure 4 shows the performance of our model compared to the DBT model for streams of 16-bit uniform randomly distributed data truncated by the same number of bits. Because the DBT model is not equipped to handle truncation bits, its estimates are poor for even two truncation bits. Though it is likely that an extension to the DBT model could handle truncation bits, these results show the importance of modeling them when they do exist.

3. Examples in High Level Synthesis

The previous sections gave general information about how data characteristics affect power dissipation in the data path. Since this information is in a form that is very intuitive, activity at functional unit inputs and buses can be predicted and understood with great ease. In this section, we illustrate the way in which the activity model parameters can be used to reduce the switching activity in data buses and inputs to functional units. We focus on data buses since they typically have significantly larger capacitance to switch. We also consider the inputs of the functional units since the activity of the nodes inside a functional unit is dependent upon the activity of the inputs, and so any reduction in the activity of the inputs translates to a power reduction inside the functional unit. We first describe how sign bit characteristics can be exploited by creative scheduling and allocation. Then we show with examples how data characteristics can be exploited in straight-line code, unrolled loops and independent blocks of code to reduce power consumption.

3.1. Power Reduction through Exploitation of Sign Bit Characteristics

As described in Section 2.1, if n_{CMD} and n_{IT} are not significant, then the average $0 \rightarrow 1$ transition activity per bit of an N-bit node is

$$\hat{\alpha}_{0\rightarrow 1} = \frac{1}{4N}\{N+(1-\alpha_{OS})(\frac{1}{3}-n_{IS})+\alpha_{OS}(n_{IS}-\frac{1}{3})\} \quad (5)$$

Case	Scheduled Order	$\alpha_{0\rightarrow 1}$
1	N[0],N[1],N[2], ...	Nominal
2	L[0],L[1],L[2], ...	Low
3	H[0],H[1],H[2], ...	High
4	H[0],H[2],H[4], ...	Low
5	H[1],H[3],H[5], ...	Low
6	H[0],N[0],H[1],N[1],H[2],N[2], ...	Nominal
7	L[0],H[0],L[1],H[1],L[2],H[2], ...	Nominal

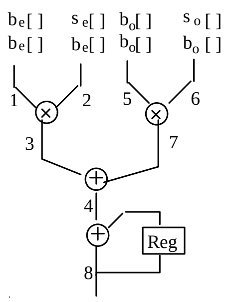
Table 1. Interesting scheduling options for three data streams allocated to the same 16-bit bus

When data is identified that could potentially have large n_{IS}, judicious scheduling and allocation can result in significant activity reductions. For example, take three vectors, L[], H[], and N[], of 16-bit fixed-point, two's complement data, each with $n_{IS} \approx \frac{N}{2}$. No other significant relationships exist between consecutive words in these arrays except that L[] has low α_{OS}, H[] has high α_{OS}, and N[] has $\alpha_{OS} \approx 0.5$ (which corresponds to nominal $\alpha_{0\rightarrow 1}$). In addition, assume that the same bus has been allocated to all of the data and no significant relationship exists between the arrays L[], H[], and N[].

Examples of some interesting scheduling possibilities along with the resulting $\alpha_{0\rightarrow 1}$ for the bus are shown in Table 1. Cases 1-3 show the effect of the different levels of α_{OS} on activity between consecutive array values. Cases 4 and 5 show that by breaking an array of high α_{OS} values into an array of even elements and an array of odd elements, the new arrays will actually have low α_{OS}, which can result in low $\alpha_{0\rightarrow 1}$. Case 6 shows that if nominal α_{OS} data is scheduled between high α_{OS} data, $\alpha_{0\rightarrow 1}$ will become nominal, or approximately 0.25. This schedule can be an effective way of reducing activity associated with a high α_{OS} array, if the array cannot be split into its even and odd elements. Case 7, on the other hand, show less desirable ways of reducing high α_{OS}, because the potential for lowering $\alpha_{0\rightarrow 1}$ even more by exploiting the low α_{OS} of the L[] array is lost. Thus we see that by characterizing data in terms of the activity model parameters, it is possible to find a near optimal schedule without directly considering all possibilities.

3.2. Power Reduction through Exploitation of Data Characteristics in Straight-Line Code

In high-level design, the amount of power reduction that can be achieved through scheduling and allocation is dependent on the algorithm design, speed constraints, resource availablility, and the ability to find power reduction opportunities. A very practical example from the VSELP speech codec implementation, namely, the code book search, will be used to illustrate the power reduction procedures. The processing that takes place during a codebook search is shown below in terms of C code. The reason so much processing is required for this computation is N and M are



Node	$\alpha_{0\to1}$	α_{OS}	n_{IS}	n_{CMD}	n_{IT}
1	0.20	0.22	5.72	0.65	0.36
2	0.19	0.24	6.18	0.65	1.24
3	0.13	0.17	11.8	0.37	8.01
4	0.13	0.15	10.83	0.42	7.98
5	0.20	0.22	5.76	0.64	0.38
6	0.19	0.24	6.19	0.65	1.25
7	0.13	0.17	11.87	0.36	8.03
8	0.12	0.04	7.70	2.72	7.93

Figure 5. Scheduling and allocation choice made without activity consideration

typically quite large, resulting in 2MN multiply-accumulate (MAC) operations per speech subframe.

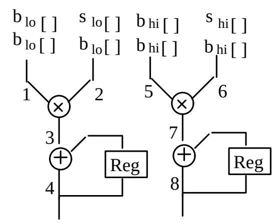
```
for(i = 0; i < N; i++)
{
    g[i]=0;
    for(j = 0; j < M; j++)
        g[i]=g[i]+b[i][j]*b[i][j];
    c[i]=0;
    for(j = 0; j < M; j++)
        c[i]=c[i]+s[j]*b[i][j];
}
```

One of the most obvious ways to map these operations onto an architecture that consists of two multipliers and two adders is shown in Figure 5 for one iteration of the i loop. All of the multiplications corresponding to even j indices are done by one multiplier, while all of the odd j-indexed multiplications are done by the other. In the figure, b_e[] and s_e[] are vectors of even j-indexed b[i][j] and s[j] values, respectively. Similarly, b_o[] and s_o[] are vectors of the odd values. The loop that calculates g[i] is computed first, followed by the c[i] loop. Model parameter values (α_{OS}, n_{IS}, n_{CMD}, n_{IT})and $\alpha_{0\to1}$ resulting from a typical speech signal input are shown in the table included in Figure 5. The speech segment is a male speaker uttering "Card games are fun to play."

The activity data in this table shows lower than nominal activity for all nodes. The nature of the data is such that opposite sign activity is low for all nodes. A significant number of intersecting sign bits exist between consecutive values on all nodes due to downscaling that is introduced to avoid overflows during accumulation. Significant activity reduction at nodes 3, 4, 7, and 8 is due to truncation that is performed at the output registers of the multiplier to make the result only 24 bits wide. Nevertheless, activity can be further reduced by more thoughtful scheduling and allocation.

3.3. Power Reduction through Loop Unrolling

Similar to ILP exploitation, data characteristic exploitation opportunities can many times be identified from source code by using loop unrolling to increase the amount of straight-line code. Consider the g[i] loop (as studied in the previous section) unrolled by a factor of two. The C code for this case is shown below.



Node	$\alpha_{0\to1}$	α_{OS}	n_{IS}	n_{CMD}	n_{IT}
1	0.18	0.15	6.00	0.87	0.41
2	0.17	0.17	6.40	0.91	1.23
3	0.12	0.12	12.30	0.51	7.97
4	0.11	0.04	9.85	2.03	7.88
5	0.19	0.15	5.78	0.93	0.34
6	0.17	0.17	6.26	0.93	1.24
7	0.12	0.12	11.88	0.54	8.03
8	0.11	0.03	8.80	2.69	7.93

Figure 6. Scheduling and allocation choice made by considering unrolled loops

```
g[i]=0;
for(j = 0; j < M; j+ = 2)
{
    g[i]=g[i]+b[i][j]*b[i][j];
    g[i]=g[i]+b[i][j+1]*b[i][j+1];
}
```

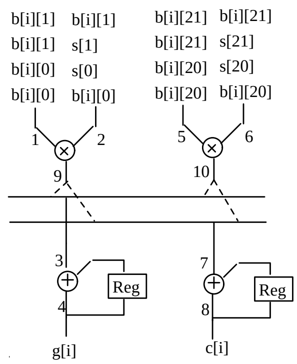
In this form, it can be seen quite easily that consecutive multiplications involve consecutive values from the b array, b[i][j] and b[i][j+1]. It can be determined either from the algorithm or simulation that these array elements tend to have low α_{OS} and significantly high n_{IS}. These characteristics are basically due to lowpass filtering and scaling. As a result, allocating the same multiplier to as many consecutive multiplications as possible will reduce activity at the inputs of that multiplier.

An implementation that accomplishes this is shown in Figure 6 for a single iteration of the i loop. The table in Figure 6 shows analysis parameters for this implementation resulting from the same sample speech signal again. In this figure, the b_{lo}[] and s_{lo}[] vectors are the lower M/2 j-indexed elements of b[i][j] and s[j], respectively, and the b_{hi}[] and s_{hi}[] vectors are the upper M/2 j-indexed elements of b[i][j] and s[j], respectively.

In this new implementation, activity is reduced at multiplier inputs by about 8% compared to the original implementation (see section 3.2). In addition, multiplier output activity is reduced by about 7%. This reduction is due to reduced α_{OS} at both inputs of each multiplier when computing c[i]. This results in reduced α_{OS} at the output of the multiplier, since the signs of the two multiplier inputs have no other strong relationship with each other. Thus, activity is reduced at the multiplier outputs. The reduction at the multiplier outputs, coupled with the reduction achieved in the previous section by exploiting unrolled code characteristics results in a 9% decrease in activity at adder inputs compared to the original implementation.

3.4. Power Reduction through Parallel Execution of Independent Blocks

In addition to identifying data characteristics in existing straight-line code and unrolled loops, it is important to identify data relationships between independently executable blocks of source code. Consider the case where both the g[i] and c[i] calculation loops are combined and unrolled by a factor of two:



Node	$\alpha_{0\to 1}$	α_{OS}	n_{IS}	n_{CMD}	n_{IT}
1	0.09	0.07	6.25	5.18	0.20
2	0.21	0.33	1.01	0.58	1.11
3	0.12	0.07	11.21	0.48	7.53
4	0.12	0.04	6.46	4.35	3.33
5	0.09	0.07	6.02	5.33	0.17
6	0.21	0.32	5.87	0.58	1.04
7	0.14	0.25	12.88	0.49	7.96
8	0.11	0.05	9.04	3.01	9.93
9	0.15	0.29	11.75	0.39	7.98
10	0.15	0.29	10.87	0.40	7.63

Figure 7. Scheduling and allocation choice made by considering independent blocks

```
for(i = 0; i < N; i++)
{
g[i]=0;
c[i]=0;
    for(j = 0; j < M; j++)
    {
        g[i]=g[i]+b[i][j]*b[i][j];
        g[i]=g[i]+b[i][j+1]*b[i][j+1];
        c[i]=c[i]+s[j]*b[i][j];
        c[i]=c[i]+s[j+1]*b[i][j+1];
    }
}
```

In this form, it can be seen by inspection that the same iteration of g[i] and c[i] calculations involve the same b[i][j] value. As a result, if these calculations are performed consecutively using the same multiplier, activity at one of the inputs can be significantly reduced, because the same b[i][j] value will be used. However, the real trick of taking advantage of this reduction is keeping activity on the other multiplier input low. This can be accomplished by performing the MAC operations in the following order:

```
g[i]=g[i]+b[i][j]*b[i][j];
c[i]=c[i]+s[j]*b[i][j];
c[i]=c[i]+s[j+1]*b[i][j+1];
g[i]=g[i]+b[i][j+1]*b[i][j+1];
```

The corresponding implementation is shown in Figure 7. Here, the first four pairs of inputs are shown for each multiplier with the very first inputs appearing at the bottom of the list. The table of parameters for the speech signal is also included in Figure 7. From this table it can be seen that activity is reduced by almost 23% at multiplier inputs over the original implementation. A tradeoff of this design is that control is a little more complicated due to the demultiplexing that must go on to accumulate c[i] and g[i] values on separate adders, even though the products involved come from shared multipliers. In addition, activity at the outputs of the multipliers is actually larger for this implementation. This activity is not propagated throughout the rest of the architecture, however, because of the demultiplexing that takes place. Also, activity reduction at the adder inputs is only about 5% compared with the 9% reduction achieved with the design based on loop unrolling only. However, none of these tradeoffs are expected to significantly negate the power saved from the 23% reduction in activity at the multiplier inputs.

4. Conclusions

This paper presents a high-level methodology for exploiting data characteristics to reduce power consumption of static CMOS circuits. The approach aids in achieving efficient switching activity reduction without excluding good design possibilities. At the root of this procedure is a model, based on four statistical parameters, that relates switching activity to data characteristics. The importance of this model is that it is more intuitive and general than previous models and, therefore, simplifies activity analysis. This paper demonstrates the use of this model in reducing switching activity in buses and functional unit inputs.

References

[1] A. Raghunathan and N. Jha, "Behavioral synthesis for low power", in Proc. ICCD'94, Oct. 1994, pp. 318-322.

[2] D. Shin and K. Choi, "Low Power High Level Synthesis by Increasing Data Correlation", *Proc of the Int Symp on Low Power Electronics and Design*, 1997, pp. 62-67.

[3] A. Chatterjee and R. Roy, "Synthesis of Low Power Linear DSP Circuits using Activity Metrics", *Proc of the IEEE Int Conf on VLSI Design*, 1994, pp. 265-270.

[4] A. Chandrakasan and R. Brodersen, "Minimizing power consumption in digital CMOS circuits," Proc. IEEE, Apr. 1995, vol. 83, pp. 498-523.

[5] E. Musoll and J. Cortadella, "High-level synthesis techniques for reducing the activity of functional units", in Proc. ISLPD, Apr. 1995, pp. 99-104.

[6] P. Landman and J. Rabaey, "Architectural Power Analysis: The Dual Bit Type Method", *IEEE Trans on VLSI Systems*, 3(2):173-187, 1995.

[7] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "Analytical Estimation of Signal Transition Activity from Word-Level Statistics", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 16, July 1997, pp. 718-733.

[8] A. Raghunathan, S. Dey, and N. Jha, "Register-Transfer Level Estimation Techniques for Switching Activity and Power Consumption", in Proc. ICCAD, Nov. 1996, pp. 158-165.

[9] R. Henning and C. Chakrabarti, "Activity Models for use in Low Power, High-Level Synthesis," Proc of Int. Conf. on Acoustics, Speech and Signal Processing, March 1999.