Fast and accurate power estimation of FPGA DSP components based on high-level switching activity models

Ruzica Jevtic, Carlos Carreras and Gabriel Caffarena

When designing DSP circuits, it is important to predict their power consumption early in the design flow in order to reduce the repetition of time consuming design phases. High-level modelling is required for fast power estimation when a design is modified at the algorithm level. This paper presents a novel high-level analytical approach to estimate logic power consumption of arithmetic components implemented in FPGAs. In particular, models of adders and multipliers are presented in detail. The proposed methodology considers input signal correlation and glitching produced inside the component. It is based on an analytical computation of the switching activity in the component which takes into account the component architecture. The complete model can estimate the power consumption for any given clock frequency, signal statistics and operands' word-lengths. Compared to other proposed power estimation methods, the number of circuit simulations needed for characterizing the power model of the component is highly reduced. The accuracy of the model is within 10% of low-level power estimates given by the tool XPower, and it achieves better overall performance.

1 Introduction

As FPGAs are becoming a more common solution for DSP applications due to their potentials for reuse and dynamic configuration, their energy performance is emerging as one of the most important metrics that need to be considered during the design flow.

The dominant sources of power consumption in CMOS circuits are the charge and discharge of the node capacitances. For each node, power consumption is

$$P = \alpha \cdot C_l \cdot V_{dd}^2 \cdot f \tag{1}$$

where α (referred to as the switching activity) is the average number of $0 \leftrightarrow 1$ transitions in one clock-cycle, C_l is the load capacitance at the given node, V_{dd} is the power supply voltage, and f is the clock frequency.

In the case of FPGAs, two power terms V_{dd} and f are fixed for a given FPGA architecture and clock period of the design. The third power term, C_l, is considered to be constant in the case of DSP components implemented in look-up tables (LUTs) as it will be explained later. Hence, for a specific design, the product of all three parameters is considered to be a constant a and the power consumption of a node can be represented as:

$$P = \alpha \cdot a \tag{2}$$

Many accurate techniques for power estimation already exist at the logic and circuit levels. As they all need transistor or gate level circuit descriptions, the power estimation occurs late in the design process, thus leading to severe penalties in design time when constraints are not met. On the other hand, the methods for estimating power consumption at higher levels consider extensive module simulations for different input statistics as a step prior to high-level synthesis. As it is not possible to cover all possibilities for these variables in a reasonable time, a solution is sought in numerical methods, thus often resulting in not so accurate estimates. Another critical parameter is the word-length of the operands. Since the word-length optimization in DSP algorithms (Caffarena *et al.* 2006, Kum *et al.* 2001) has proven to provide significant cost savings, it is very important to have fast power estimates for components with any operands' word-lengths in order to see if the power constraints are met during architectural synthesis. However, as the number of combinations for input word-lengths is extremely high, a new set of simulations for the module characterization is necessary each time the module's parameters change.

In this paper, we present a methodology which has proven to overcome the above mentioned problems and is used for estimating power consumption of arithmetic components implemented in LUTs, in particular multipliers and adders. Although the use of embedded multipliers has become more popular, they remain unavailable in most HLS tools as the parameters used for their description are not directly compatible with the commonly used parameters and models of other FPGA components. Therefore, LUT based multipliers still remain as the standard way of implementing multiplication in HLS. The methodology is presented here for multipliers in detail, which is the more complex of the two types of components, to avoid unnecessary repetition of the steps taken in the approach. The specific details that apply to adders are presented afterwards. Unlike other proposed approaches, which are completely based on circuit and signal simulations, this approach is based on an analytical model that uses the operand's word-length and the signal statistics as parameters (Jevtic *et al.* 2007).

We propose three different power models for estimating power consumption in multipliers. We derive an expression for the switching activity of an array multiplier by using a word-level regional decomposition of the input signal based on its statistics. With the information about switching activity, a power model parameterized in terms of operand word-lengths and their signal statistics is constructed. Next, we include the FPGA implementation details of the multiplier into the expression for the switching activity to construct the second power model with enhanced characteristics. Finally, this model is improved by considering the effects of signal glitching.

Two types of models are proposed for estimating power consumption in adders. In the first one, we consider a ripple-carry adder. Following the methodology applied in the case of the array multiplier, an expression for the switching activity of this type of adder is derived by using a word-level regional decomposition of the input signal. In the second power model we include the glitching effects produced inside the component.

All proposed models are capable of producing fast and accurate estimates of logic power consumption in arithmetic components regardless of the word-lengths of the operands.

The paper is organized as follows. Section 2 highlights the related work done in the area of high-level power estimation. Section 3 presents some preliminaries of the work used for estimating signal transition activity from word-level statistics. In Section 4, the transition activity of a multiplier with and without glitching effects is computed. It is followed by the power model description for a ripple-carry adder given in Section 5. Experimental results are presented in Section 6. We conclude this paper in Section 7.

2 Previous research

Currently existing approaches based on the bit-level input signal statistics (Gupta *et al.* 2000, Shang *et al.* 2001) consider average bit level statistics which are found to be in direct relationship with power consumption. These statistics are introduced as variables in an equation which estimates the average power consumed by the module. Coefficients multiplying the variables in the equation are determined through extensive simulations. This model can be applied only to a specific component with fixed word-length, whereas in the approach proposed here the operand's word-lengths are used as variables in a single power model for that component.

Approaches based on word-level signal statistics (Clarke *et al.* 2005, Landman *et al.* 1995) consider the variation in power consumption caused by the variation of the input signal variance, mean and correlation coefficients. The approach used in Clarke *et al.* 2005 generates first-order and second-order equations for adders and multipliers respectively. The only variable introduced in the equation is the operands' word-length. Therefore, this methodology does not model multipliers with operands of different sizes, while in the case of adders, where sign extension adjusts the operands' word-lengths, a considerable error is introduced as shown in the results that will be presented later.

The Dual-bit type method is presented in Landman *et al.* 1995. It describes a strategy for generating a black-box model of datapath power consumption at the architecture level. The technique accounts for a word-level signal broken into three regions: uncorrelated, correlated and sign data bits. Based on this methodology, a product of the capacitance and the switching activity is calculated for every signal region by using extensive simulations. A disadvantage of this model lies in the fact that every specific black-box component needs to be simulated before determining its power consumption, thus, increasing design time significantly.

The approach that considers a signal division into correlated, uncorrelated and sign regions has also been used in the power estimation method proposed here, but instead of measuring the transition activity of the sign bits, it has been estimated from word-level signal statistics as presented in Ramprasad *et al.* 1997 and Satyanarayana *et al.* 2000.

3 Word-level signal transition activity

In the signal model presented here, it is assumed that signals at different points in the system are stationary and considered to have zero-mean Gaussian distributions. It has been shown that dynamic power consumption in arithmetic components is affected to a greater extent by autocorrelation than by cross-correlation (Clarke *et al.* 2005). Therefore, we will consider only the effects of signals variances and autocorrelations on the power consumption models. As previously mentioned, we use the method described in Landman *et al.* 1995 to divide each signal word into three regions referred to as MSB, linear and LSB region. To demonstrate the applicability of this approach, we have plotted the bit transition activity in signal word versus their bit position in the word for different autocorrelations (see Fig. 1). All the signals have Gaussian distribution of variance σ^2. It can be seen that the bits on MSB positions demonstrate a high correlation and have a constant switching activity up to a certain limit BP1 and that the bits on the lowest positions have a switching activity of 0.5 as they behave as identically distributed inputs. The region in between can be approximated as a linear region. Instead of a word division into regions based upon the transition activity, we will divide it based upon their bit-level correlation ρ_i following the methodology described in Ramprasad *et al.* 1997. By definition $\rho_i = 0$ for $i < BP0$. It is assumed that $\rho_{BP1} = \rho$ where ρ represents the word-level temporal correlation. The expressions for the autocorrelation coefficient of all three regions are:

$$\rho_i = \begin{cases} 0 & i < BP0 \\ \frac{(i-BP0+1)\cdot\rho_{BP1}}{BP1-BP0} & BP0 \le i < BP1-1 \\ \rho_{BP1} & i \ge BP1-1 \end{cases} \tag{3}$$

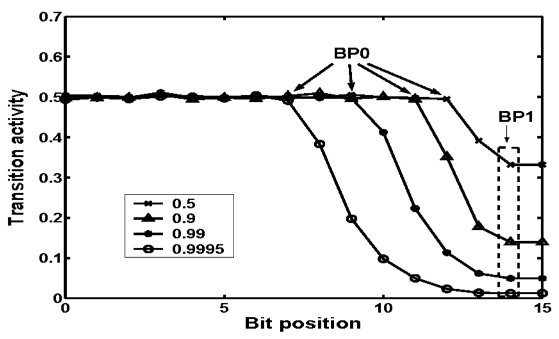


Figure 1. Bit transition activity vs. bit position in a word for different autocorrelations

In order to calculate the exact transition activity t_i we use the following relationship between the bit-level probability p_i (calculated as in Ramprasad *et al.* 1997) and the bit-level autocorrelation ρ_i as it follows:

$$t_i = 2 \cdot p_i \cdot (1 - p_i) \cdot (1 - \rho_i) \tag{4}$$

The expression derived for a breakpoint BP1 is given by:

$$BP1 = [\log_2(6\sigma)] \tag{5}$$

The expression derived for a breakpoint BP0 depends on the coefficients of the ARMA (autoregressive moving average) signal model and is to be computed here for signals that have zero-mean Gaussian distributions. An (N,M)-order ARMA model can be represented as

$$x(n) = \sum_{i=0}^{N} d_i \gamma(n-i) + \sum_{i=1}^{M} a_i x(n-i) \tag{6}$$

where the signal $\gamma(n)$ is a white (uncorrelated) noise source with zero mean, and $x(n)$ is the signal being generated. It is also possible to transform this IIR model into one that depends only on the inputs as shown below

$$x(n) = \sum_{i=0}^{\infty} h_i \gamma(n-i) \tag{7}$$

where h_i can be computed according to the following recursion:

$$h_k = d_k + \sum_{i=1}^{N} a_i h_{k-i} \tag{8}$$

where $h_k = 0$ for $k < 0$, and $h_0 = d_0$. The breakpoint BP0, for signal $x(n)$ in (7) is estimated as the maximum of the BP0's of the signals $h_i \gamma(n-i)$. Hence,

$$BP0 = [\log_2(h_{\max} \sigma_\gamma)] \tag{9}$$

where $h_{\max} = \max(|h_i|)$. We will show the method for computing the ARMA-model coefficients in (9) when a signal has a zero-mean Gaussian distribution. Equations (7) and (8) have been used as our starting point in finding a relation between the signal statistics and the coefficients of the ARMA signal model. Based on the signal generation model presented in Clarke *et al.* 2005, it can be shown that any given signal with a zero-mean Gaussian distribution of variance σ^2 and autocorrelation coefficient ρ can be expressed as an ARMA(0,1) model:

$$x(n) = d_0 \gamma(n) + a_1 x(n-1) \tag{10}$$

Computing the variance and autocorrelation of signal which is presented as in (10) leads to a system of two equations. The coefficients d_0 and a_1 are then obtained by solving this system and the resulting expressions for them are given as follows:

$$d_0 \cdot \sigma_\gamma = \sqrt{(1 - a_1^2)} \cdot \sigma_x \tag{11}$$

$$a_1 = \rho \tag{12}$$

Next, we have computed the coefficients for the ARMA(0,1) model. Combining (8) and (12) we obtain that

$$\begin{aligned} h_0 &= d_0 \\ h_1 &= a_1 d_0 = \rho \cdot d_0 \\ h_2 &= a_1^2 d_0 = \rho^2 d_0 \\ &\dots \end{aligned} \tag{13}$$

As the autocorrelation coefficient is always less than or equal to one, we obtain that the maximum of all h_i is h_0, i.e. d_0 . Finally, by replacing expressions (11) and (12) in (9), it becomes:

$$BP0 = \left[\log_2(\sqrt{1-\rho^2} \cdot \sigma_x)\right] \tag{14}$$

We have used this expression for the breakpoint BP0 in our approach as it gives better results than the one proposed in Landman *et al.* 1995.

4 Multiplier

4.1 *Array multiplier*

We consider a standard array multiplier whose structure is shown in Fig. 2a. Operand x has N bits and y has M bits. The multiplier consists of basic elements, namely AND gates and half-adder and full-adder cells. We consider that each type of element is implemented into the slice section composed of one LUT and logic gates. As all the LUTs in multiplier perform the same function, we assume that the capacitance being switched per each basic element is the same. Although within-die delay variability exists between different LUT pins (Sedcole *et al.* 2006), it is small enough so as to consider the constant capacitance assumption valid for estimation purposes. Thus, as already mentioned in introduction, the product of three power terms, V_{dd}^2, f, C_l, can be represented as a constant a. Our goal is to analytically calculate the total switching activity and with one-time measurement of the logic power of the multiplier obtain this constant. The module represented in Fig. 2a is regionally divided into 4 parts according to the input signal word decomposition as explained in Section 3. The linear region can be achieved by attributing the upper half of the bits in the linear region to the MSB region and the bottom half of the bits to the LSB region (Landman *et al.* 1995). Thus, the number of bits in MSB region is obtained from:

$$x = [(BP1 - BP0)/2 + (N - BP1)] \tag{15}$$

where N is the number of bits in a signal word and [ ] is the rounding operation.

The four parts of the multiplier exhibit different switching activities. For each part, the switching activity on the outputs of its basic elements is to be computed as a function of their input switching probabilities. As the observed signals have zero-mean Gaussian distributions and are encoded in two's complement, it is assumed that the probability of each bit of being equal to '1' is the same as the probability of being equal to '0'. In the case of non-zero Gaussian distributions, these probabilities would change for the sign bits (Clarke *et al.* 2006). However, in this paper we will consider only zero-mean Gaussian distributions. Although, these probabilities are assumed to be the same at the inputs of the multiplier, they change as the signals pass through the logic. With this effect taken into account, the switching activities of carry bits and outputs of the adder cells, as well as its probabilities of being '1' and '0', are to be computed as a function of the probabilities of the inputs and carry bits at the previous level. The methodology employed for computing the switching probability of the carry bit will be presented here as it is the most complex one. The calculation of the rest of the probabilities is obvious and only their expressions are provided. Here is the list of notations used in the equations:

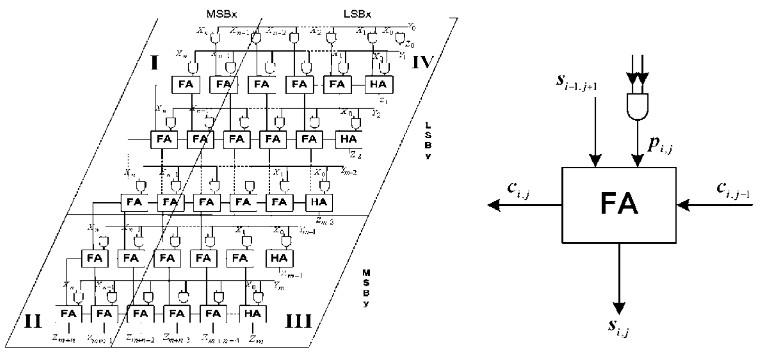


Figure 2. a) Regionally decomposed array multiplier, b) Full-adder cell

- p is the transition probability at the output of the AND gate which goes to the one of the inputs of the full-adder cell
- q is the transition probability at the output of the full- adder cell from the previous level (see Fig. 2b) which goes to the other input of the full-adder cell
- c is the transition probability of the carry bit
- s is the transition probability of the output of the full-adder cell
- p^0 and p^1 are the probabilities of input p of being '0' and '1' respectively
- q^0 and q^1 are the probabilities of input q of being '0' and '1' respectively
- c^0 and c^1 are the probabilities of the carry bit of being '0' and '1' respectively
- i and j are the row and column number of the full-adder cell

The carry bit from a previous cell is the third input to a full-adder cell. As q represents the transition probability at the output of the full-adder cell from the previous level, it is clear that $q_{i,j} = s_{i-1,j+1}$.

Now we will show the methodology for computing the transition probability of the outgoing carry bit of the full-adder cell. As the full-adder cell has three inputs, there are 2^3 combinations for its inputs in one clock cycle. For each combination, there are four possible events which could occur in the following clock cycle. In the first case, neither of the inputs changes. Hence, there will be no transition in the carry. In the second case, only one of the inputs makes a transition. In this case, for the combinations "000" and "111" there will be no change at the outgoing carry bit. For the combinations "001","010" and "100", if any of the zeros changes, the transition will occur at the carry also. Hence, the transition activity for this case is:

$$\begin{aligned}c_{i,j}^{A1} = {} & p_{i,j}^0 \cdot q_{i,j}^0 \cdot c_{i,j-1}^1 \cdot (1 - c_{i,j-1}) \cdot (p_{i,j} + q_{i,j} - 2 \cdot p_{i,j} \cdot q_{i,j}) + \\ & p_{i,j}^1 \cdot q_{i,j}^0 \cdot c_{i,j-1}^0 \cdot (1 - p_{i,j}) \cdot (q_{i,j} + c_{i,j-1} - 2 \cdot q_{i,j} \cdot c_{i,j-1}) + \\ & p_{i,j}^0 \cdot q_{i,j}^1 \cdot c_{i,j-1}^0 \cdot (1 - q_{i,j}) \cdot (p_{i,j} + c_{i,j-1} - 2 \cdot c_{i,j-1} \cdot p_{i,j})\end{aligned} \tag{16}$$

For the rest of the combinations "011","110"and "101" a change in the any of the ones, will produce a transition in the carry. Thus, the switching activity is

$$\begin{aligned}
&c_{i,j}^{A2} = p_{i,j}^0 \cdot q_{i,j}^1 \cdot c_{i,j-1}^1 \cdot (1 - p_{i,j}) \cdot (q_{i,j} + c_{i,j-1} - 2 \cdot q_{i,j} \cdot c_{i,j-1}) + \\
&p_{i,j}^1 \cdot q_{i,j}^0 \cdot c_{i,j-1}^1 \cdot (1 - q_{i,j}) \cdot (p_{i,j} + c_{i,j-1} - 2 \cdot c_{i,j-1} \cdot p_{i,j}) + \\
&p_{i,j}^1 \cdot q_{i,j}^1 \cdot c_{i,j-1}^0 \cdot (1 - c_{i,j-1}) \cdot (p_{i,j} + q_{i,j} - 2 \cdot q_{i,j} \cdot p_{i,j})
\end{aligned} \tag{17}$$

In the third case, two inputs change and one remains the same. In the fourth case, all input bits change. The methodology for computing transition probability in these cases is the same as in the second case. The probabilities of the carry and the output bit of the full-adder cell of being '0' are computed as:

$$\begin{aligned}
&c_{i,j}^0 = p_{i,j}^0 \cdot q_{i,j}^0 + c_{i,j-1}^0 \cdot (p_{i,j}^0 \cdot q_{i,j}^1 + p_{i,j}^1 \cdot q_{i,j}^0) \\
&s_{i,j}^0 = (p_{i,j}^0 \cdot q_{i,j}^0 + p_{i,j}^1 \cdot q_{i,j}^1) \cdot c_{i,j-1}^0 + (p_{i,j}^0 \cdot q_{i,j}^1 + p_{i,j}^1 \cdot q_{i,j}^0) \cdot c_{i,j-1}^1
\end{aligned} \tag{18}$$

The final expression for the carry is obtained by adding up the transition probabilities of all three cases.

Now, the only probability missing for the computation of the total switching activity is the probability at the output of the full-adder cell. It is given by:

$$\begin{aligned}
s_{i,j} = &(p_{i,j} q_{i,j} + (1 - p_{i,j}) \cdot (1 - q_{i,j})) \cdot c_{i,j-1} \\
&+ (p_{i,j} \cdot (1 - q_{i,j}) + q_{i,j} \cdot (1 - p_{i,j})) \cdot (1 - c_{i,j-1})
\end{aligned} \tag{19}$$

The total switching activity of the array multiplier consists of the switching activities of the carry-bits and outputs of the adder and multiplier cells. Hence, the final result for the switching activity is obtained from:

$$SW = \sum_{i=1}^{M-1} \sum_{j=1}^{N} (s_{i,j} + c_{i,j} + p_{i,j}) \tag{20}$$

This methodology has some points in common with the methodology described in Chou *et al.* 1994. However, in Chou *et al.* 1994, the method is applied to the whole module. Since the exact calculation of signal probability is NP-hard (Najm *et al.* 1995), in the case of very large circuits, a partitioning algorithm that limits the number of inputs to a module has to be employed. The approach proposed here applies the method for estimating switching activity only to a basic cell of a multiplier, thus providing a simple expression for the total switching activity of the module no matter its size.

The problem now reduces to the calculation of the constant a introduced in (2). It is obtained from the next equation:

$$P = a \cdot SW \tag{21}$$

P is the power obtained from the one-time low-level simulation and SW is the switching activity computed according to (20). Once the constant is computed, the power consumption of a multiplier with any given characteristics is estimated by using (21). It is clear that the model is parameterized in terms of operand's word lengths. Apart from being able to estimate the power consumption of a multiplier of any given size, the resulting model is also parameterized in terms of the input signal statistics. The statistics are expressed in terms of the transition probabilities of the input bits and the position of the MSB-LSB breakpoint.

4.2 *Implementation details*

The previous calculation takes into account the structure of the array multiplier according to its definition. However, there are many different ways to implement it into an FPGA. In particular, we consider the Virtex-2 family of FPGAs from Xilinx. Xilinx IP Cores optimize the implementation of the array multiplier by transforming it into a row adder tree multiplier. This type of multiplier rearranges the adders of the array multiplier to equalize the number of adders that the result from each partial product must pass through (Andraka). As Virtex-2 and Spartan-III devices use 4-input LUTs, in the first optimization level the partial sum of two products is implemented into one LUT. This procedure optimizes two rows of the array multiplier by using only one row comprised of LUTs. The next level of the optimization compresses two LUT rows from the first optimization level into one using a similar methodology. Taking into account all these specific details of the multiplier implementation into the FPGA leads to a new expression for its switching activity. The methodology used for computing the switching activity is the same as in Section 4.1. and will not be repeated here. The final expression for the total switching activity when the word-length of the operand y is a power of two is:

$$SW = \sum_{i=1}^{\lceil \log_2 M \rceil} \sum_{j=1}^{l_j} \sum_{k=2^i-1}^{N+2^i-1} (s_{i,j,k} + c_{i,j,k}) \tag{22}$$

In the cases where y is not a power of two, the counters i and j of the two inner summations have slightly different values due to the parity of the number of LUT rows in each optimization level. This power model has the same features as the model described in the previous section as it is parameterized in terms of operands word lengths and input signal statistics.

4.3 *Introducing glitching effects in the model*

The switching activity at a node increases with glitching. The glitching is produced by the different signal delays entering the same logic component. As glitching represents additional activity at the output of the logic gate, it is directly proportional to the transition probabilities of its inputs. Therefore, we will consider that the most significant amount of glitching generated in the multiplier is produced at the most active regions of its inputs, ie. LSB regions. Hence, the glitching of the multiplier can be modeled as the sum of glitching produced by each cell belonging to the LSB region of the multiplier (region IV in Fig. 2a). This leads us to the following expression:

$$\begin{aligned}
&G = k \cdot ls_x \cdot \sum_{i=1}^{\lceil \log_2(ls_y) \rceil} m_i = k \cdot G^{'} \\
&m_i = \lceil m_{i-1}/2 \rceil \\
&m_1 = \lceil ls_y/2 \rceil
\end{aligned} \tag{23}$$

where G is the amount of glitching, ls_x and ls_y are the number of bits in the LSB region of the multiplicand and multiplier respectively and k is an empirically derived constant which represents the average glitching at the output of one LUT. When the position of the MSB-LSB breakpoint is known, the number of LSB bits in the operands is easily obtained. Hence, it is clear that the given model has the same features in terms of parameterization as the previous two. The final model for power estimation is given as follows:

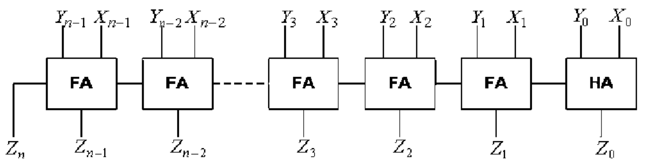
$$P = b \cdot (SW + k \cdot G^{'}) \tag{24}$$

Constant b can be obtained together with constant k which has been introduced into the expression for glitching.

4.4 *Summary of the power estimation procedure*

The complete power estimation procedure consists of the following steps:

(1) The number of optimization levels is determined according to the number of bits in the multiplier operand y;
(2) The number of MSB bits for the each operand is obtained from (15);
(3) The transition probabilities of all bits in the inputs are set to the values defined by (4);
(4) For each optimization level, the switching activity is calculated on the outputs of the partial sum generators using (16)-(19) and is added as in (22);
(5) Glitching presented in (23) is introduced into the final power model;



Y_{n-1} X_{n-1} Y_{n-2} X_{n-2} Y_3 X_3 Y_2 X_2 Y_1 X_1 Y_0 X_0
FA
FA
FA
FA
FA
HA
Z_n Z_{n-1} Z_{n-2} Z_3 Z_2 Z_1 Z_0

Figure 3. Ripple-carry adder

(6) Two low-level power measurements for different multiplier sizes using the same ρ are sufficient in order to determine coefficients b and k. As the factors SW, P and G' are known, the coefficients can be easily obtained.

5 Adder

5.1 *Ripple-carry adder*

The adder considered in this work is a ripple-carry adder consisting of full-adder cells as presented in Fig. 3. Module decomposition in a case of an adder depends on the breakpoint position of its input operands. There are two different cases as shown in Fig. 4. In both cases, operands have the same length as the shorter operand is always sign-extended until it reaches the length of the longer operand. One case corresponds to the situation where the MSB-LSB breakpoint of operand X lies in the MSB area of Y and another where the situation is vice versa. Thus, the adder decomposition consists of three parts: LSB-LSB, LSB-MSB and MSB-MSB. However, a special care has to be taken when calculating the switching activity of the middle part, LSB-MSB, as in the first case LSB part belongs to one operand and in second to another. There is a third special case when both breakpoints coincide, which results in only two adder regions.

The power model developed here considers a basic adder structure. However, it has been noticed that the adders implemented as Xilinx cores are based on carry-skip structure with fast carry chain conecting every second full-adder cell. The power model for this type of adder is currently under progress using the same methodology described here.

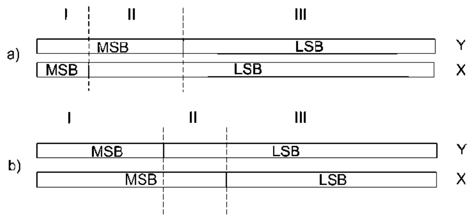


Figure 4. Module decomposition in function of MSB-LSB breakpoint of the longer operand

As the basic cell of the adder is a full-adder cell, the computation of the transition probabilities of the carry bit and the output bit has already been explained in the Section 4.1. The total switching activity of the adder is obtained by summing the switching activities of the carry bits and output bits.

5.2 *Glitching model*

In order to include the glitching effects, we used a method similar to the one described in the Section 4.3. As in the case of a multiplier, we consider that the most significant amount of glitching generated in the adder is produced at the LSB regions of its inputs. Hence, the glitching of the two-input adder can be expressed as the sum of glitching produced by each cell whose inputs belong to the LSB region of the adder (region III in Fig. 4). This leads us to the following expression for the amount of glitching produced inside the adder:

$$G = k \cdot min(ls_x, ls_y) \tag{25}$$

where G is the amount of glitching, ls_x and ls_y are the number of bits in the LSB region of the operands and k is an empirically derived constant which represents the average glitching at the output of one LUT as described in the previous section. The construction of the power model is equivalent to the approach proposed for the multiplier, already summarized in 4.4.

6 Results

The following experiments have been performed to verify the proposed models of the word-level switching activities of arithmetic components implemented in FPGAs. Two types of experiments have been considered - one where the word-lengths of both input signals are the same while the size of the component is varied together with the input signal statistics, and the other where one of the input word-lengths is varied while the signal statistics remain fixed. The experiments have been performed on multipliers and adders implemented as IP Cores in Xilinx Virtex-2 XC2V2000-5 devices. The design frequency used in all experiments was set to 100 MHz. Different autocorrelation values between 0 and 0.9995 were used in the experiments. The signals used for experiments had zero-mean Gaussian distributions. The test-benches for arithmetic components with input word-lengths smaller than 55 bits were generated using Matlab whereas the 64-bit numbers were generated using functions provided in GNU Multiple Precision Arithmetic Library (GMP). Each operand signal word was divided into regions according to equations (5) and (14). All the estimated values have been compared to low level power estimated values obtained from

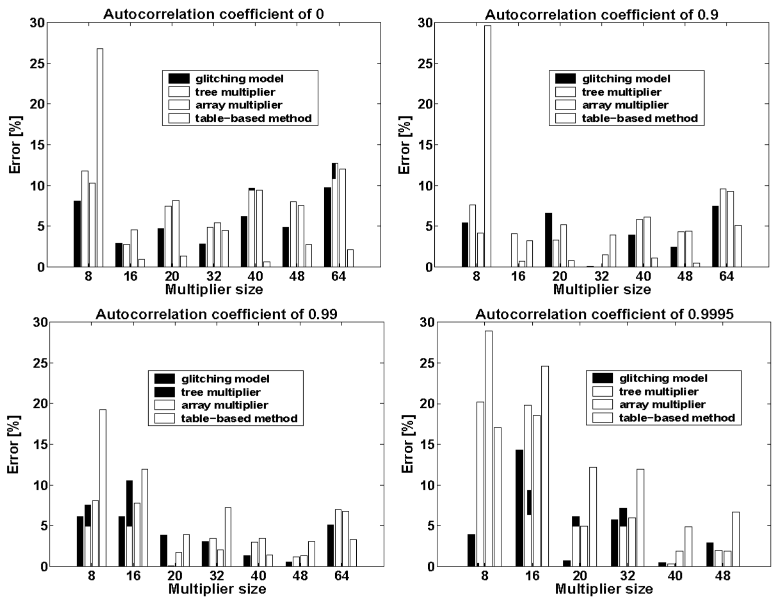


Figure 5. Error performance of multipliers for various autocorrelation coefficients

the Xilinx tool XPower. We believe that the estimates provided by XPower are sufficiently accurate when power consumption of DSP blocks implemented in LUTs is considered as this tool is created by the FPGA vendors themselves. Power estimation values are given for the Xilinx cores and thus, refer only to the DSP component structure which is used for the implementation of cores in Xilinx FPGAs.

The first set of experiments assumes multipliers and adders with operands with the same word-lengths. Results are compared to those presented in Clarke *et al.* 2005, as they also refer to arithmetic blocks implemented in LUTs. The results in Clarke *et al.* 2005 relate to tables of coefficients used to obtain the power consumption, and a clock frequency is required to perform the appropriate computations. Since such information is not included in their work, we assume a frequency of 25MHz for multipliers and 100MHz for adders, as these are the ones providing the results that are closer to the values obtained from XPower. Input word-lengths are varied between 8 and 64 bits. The errors obtained for multipliers are given in Fig. 5. Four models are taken into consideration: 1) the model that considers both glitching effects and implementation details; 2) the model for a row adder tree multiplier; 3) the model for an array multiplier; and 4) the model described in Clarke *et al.* 2005 to which we will refer to as table-based method from now on. It can be observed that in most cases the estimate provided by the model with glitching effects is accurate up to 10% of the value obtained by XPower. Besides, it clearly outperforms the

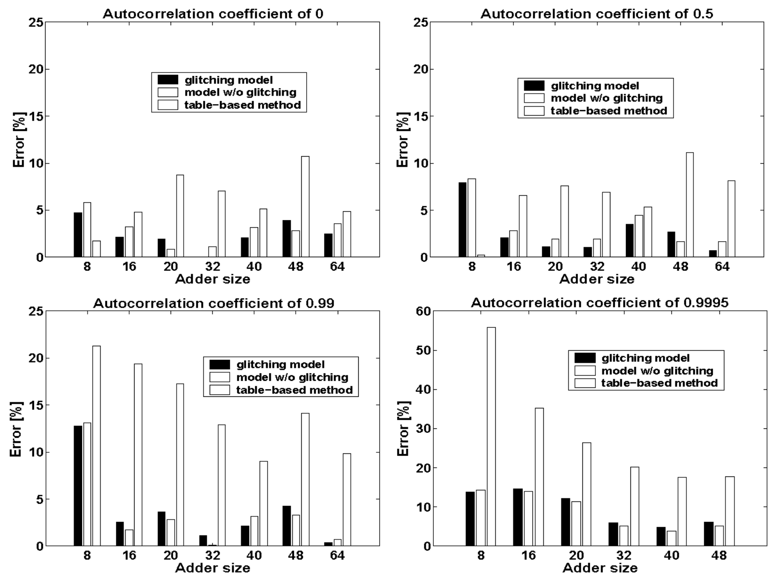


Figure 6. Error performance of adders for various autocorrelation coefficients

models that do not consider glitching effects. The table-based method gives good results when considering large word-lengths, while the error is greater than 20% for 8-bit multipliers. We believe that this is due to the quadratic nature of this model which is too simple for modelling the power consumption of multipliers.

Next, in Fig. 6 we give the errors obtained for adders. The results are given for three different power consumption models: 1) the model that considers glitching effects; 2) the model for a ripple-carry adder and 3) the table-based method. Again, it can be observed that in most cases the estimate provided by the model with glitching effects is accurate up to 10% of the measured value and that the error of the table-based method increases substantially for small word-lengths.

The second set of experiments evaluates the error performance for DSP components where one of the inputs is first set to 48 bits while varying the other from 8 to 40 bits and then set to 32 bits while varying the other from 8 to 20 bits. As already mentioned, to our knowledge, this is the first work that provides accurate estimations for DSP components with different operand sizes, without the need of a different power model construction. First, the error performance is given for the three proposed models for the multipliers. The results are shown in Fig. 7. It can be seen again that the model considering glitching effects outperforms the other models. When comparing the "tree multiplier" and the "array multiplier" model, it can be noted that the latter has larger maximum error, but a better error performance. This can be explained by the

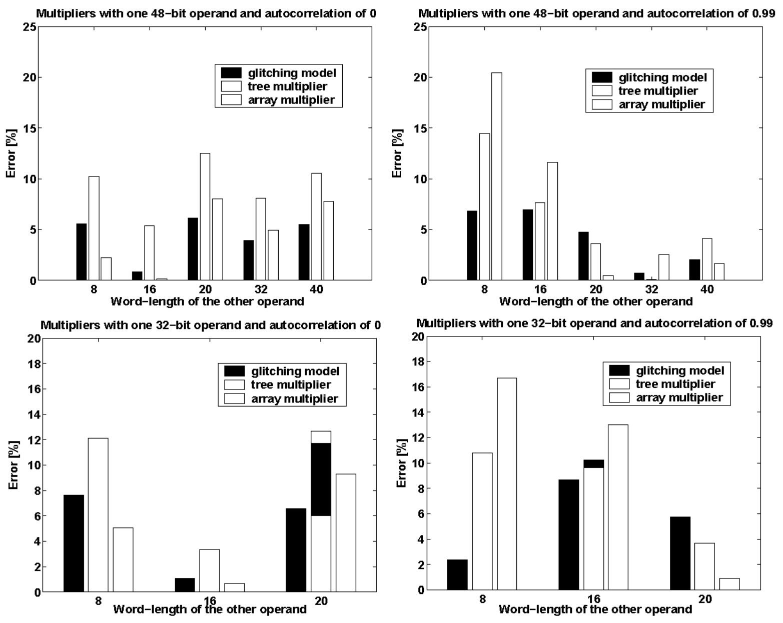


Figure 7. Error performance for multipliers with operands of different sizes

fact that this model considers separately the transition activities at the outputs of the AND gates and the outputs of the full-adder cells. The transition activities of these elements in turn contribute to the glitching activity of the multiplier, so the "array multiplier" model indirectly takes into account some of the glitching effects, whereas this does not occur in the "tree multiplier" model. Next, we give the errors obtained for adders in Fig. 8. We consider the same power models as in Fig. 6. It can be seen that the models proposed here are highly accurate, but behave similarly in many cases. This is probably due to the existence of only one logic level in the case of adder components, having as a consequence a smaller effect of glitching on the total power consumption. On the other hand, table-based method gives high errors, especially for the adders where the other operand is sign-extended for a large number of bits. This is due to a simplicity of the equations described in table-based method. As they depend only on the autocorrelation coefficient and the size of the adder, the same power value is obtained for any adder of a specific size regardless of the number of the sign bits of its operands.

7 Conclusion

We have presented a high-level analytical approach to estimate logic power consumption of DSP components implemented in FPGAs in the presence of

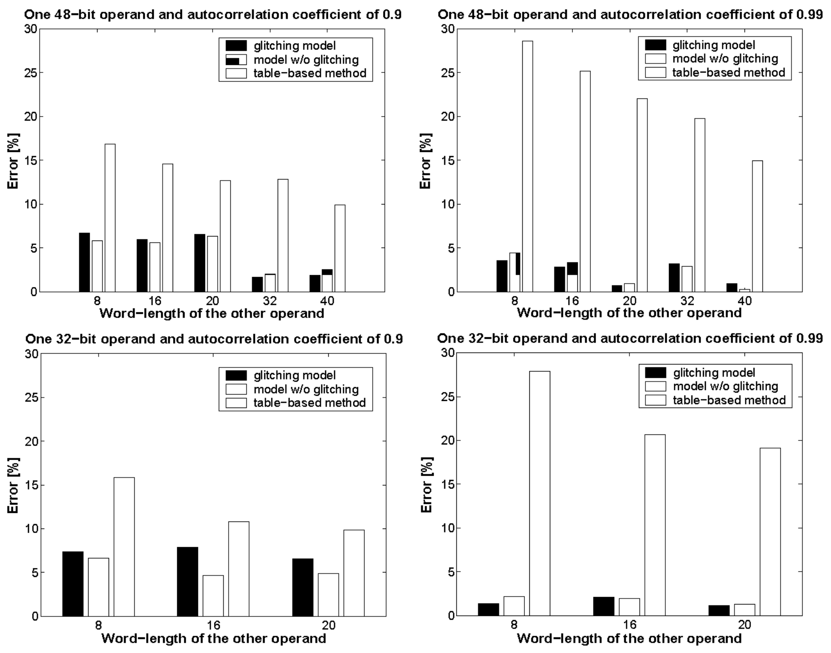


Figure 8. Error performance for adders with operands of different sizes

glitching and correlation. The proposed methodology is based on an analytical model for the switching activity of the component and its structural description. Both multipliers and adders implemented in LUTs have been modelled in detail. We have constructed three different models for estimating power consumption in multipliers using the proposed approach: one considering the basic structure of an array multiplier, another including the structural details of its FPGA implementation, and a third one including the glitching effects too. We have also used the same approach to build two power models for estimating power consumption in adders: one considering ripple-carry adder and the other representing an improved version of the first power model as it includes the glitching produced inside the component. All power models are parameterized in terms of complexity factors such as word-lengths and signal statistics of both operands. They are based on fast and fewer simulations and are capable of giving very fast power estimates in the order of miliseconds. We have shown that the accuracy of the proposed models is within 10% of low-level power estimates given by the tool XPower over a wide range of these parameters. The model that accounts for the glitching activity clearly provides more accurate results than the other proposed models.

Acknowledgement This work was supported in part by the Spanish Ministry of Education and Science under project TEC2006-13067-C03-03/TCM.

REFERENCES

Andraka Consulting Group: Multiplication in FPGAs, available at http://www.fpga-guru.com/multipli.htm

G. Caffarena, G.A. Constantinides, P.Y.K. Cheung, C. Carreras and O. Nieto-Taladriz,“Optimal Combined Word-length Allocation and Architectural Synthesis of Digital Signal Processing Circuits”, IEEE Trans. on Circuits and Systems II, Express Briefs, vol. 53, no. 5, (May 2006) 339-343

T. Chou, K. Roy and S. Prasad “Estimation of Circuit Activity Considering Signal Correlations and Simulteneous Switching”, Proc. of the 1994 IEEE/ACM Int. Conference on Computer-aided Design, (1994) 300-303

J.A. Clarke, A.A. Gaffar and G.A. Constantinides, “Parameterized logic power consumption models for FPGA-based arithmetic”, Proc. FPL (2005) 626-629

J.A. Clarke, A.A. Gaffar, G.A. Constantinides and P.Y.K. Cheung, “Fast word-level power models for synthesis of FPGA-based arithmetic”, Proc. ISCAS (2006) 1299-1302

GNU MP library available at http://www.swox.com/gmp/

S. Gupta and F.N. Najm: “Power Modeling for High Level Power Estimation”, IEEE Trans. VLSI Syst., vol.8, (Feb. 2000) 18-29

R. Jevtic, C. Carreras and G. Caffarena:“Switching Activity Models for Power Estimation in FPGA Multipliers”, Lecture Notes in Computer Science (Springer), vol. 4419, (March 2007) 201-213

K.I. Kum and W. Sung, “Combined word-length optimization and high-level synthesis of digital signal processing systems”, IEEE Trans. Circuits Syst., vol. 20, no. 8, (Aug. 2001) 921-930

P. Landman and J. Rabaey, “Architectural Power Analysis: The dual bit type method”, IEEE Trans. On VLSI Systems, vol. 3, no.2, (1995) 173-187

F.N. Najm, “Transition Density, A Stochastic Measure of Activity in Digital Circuits”, ACM/IEEE Design Automation Conf., (1991) 644-649

S. Ramprasad, N. Shanbhag and I.N. Hajj, “Analytical Estimation of Signal Transition Activity from Word-Level Statistics”, IEEE Trans. On CAD of Integrated Circuits and Systems, vol. 16, no. 7, (July 1997) 718-733

J. Satyanarayana and K.K. Parhi, “Theoretical Analysis of Word-Level Switching Activity in the Presence of Glitching and Correlation”, IEEE Trans. On VLSI Systems, vol. 8, no. 2, (Apr. 2000) 148-159

P. Sedcole and P.Y.K. Cheung, “Within-die Delay Variability in 90nm FPGAs and Beyond”, Proc. FPT (2006) 97-104

L. Shang and N.K. Jha, “High-level Power Modeling of CPLDs and FPGAs”, in Proc. of the Int. Conf. on Comp. Design. IEEE Computer Society, (2001) 46-53

Xilinx Inc. www.xilinx.com