PowerBit - Power aware arithmetic Bit-width optimization

Altaf Abdul Gaffar, Jonathan A. Clarke, George A. Constantinides

Department of Electrical and Electronic Engineering, Imperial College London
London SW7 2AZ, United Kingdom
{altaf.gaffar,jonathan.a.clarke,g.constantinides}@imperial.ac.uk

***Abstract*— In this paper we present a novel method reducing the dynamic power consumption in FPGA-based arithmetic circuits by optimizing the bit-widths of the signals inside the circuit. The proposed method is implemented in the tool PowerBit, which makes use of macro models parameterized by word-level signal statistics to estimate the circuit power consumption during the optimization process. The power models used take in to account the generation and propagation of signal glitches through the circuit. The bit-width optimization uses a static analysis technique which is capable of providing guaranteed accuracy in the design outputs. We show that, for sample designs implemented on FPGAs that improvements of over 10% are possible for multiple bit-width allocated designs optimized for power compared to designs allocated uniform bit-widths.**

I. Introduction

Optimizing the power consumption within a hardware circuit has become one of the main concerns of hardware designers in recent years. Methods which rely on the intuition of the designer alone fail when confronted with the increasing design size and complex nature of the power consumption within circuits. In addition there is a growing tendency to use high-level design descriptions to describe hardware designs at the word-level instead of the bit-level.

Optimizing the designs at the word-level enables the designer to better explore the design space available with respect to the design area, latency, throughput and power consumption. Word-length or bit-width optimization is one such high-level optimization technique which tries to allocate the optimum number of bits for the signals inside the design, while trying to minimize a design metric.

The power consumption in digital circuits can be divided into dynamic power and static power. The static power consumption refers to the power consumed in the circuit when the circuit is powered on but the signals in the circuit have zero activity, due to leakage currents, *etc*. Static power consumption is determined by the fabrication technology, circuit topology and transistor count of a circuit, all of which are determined by the FPGA manufacturer.

The dynamic power consumption, on the other hand, refers to the power consumed due to the transition activity in the signals inside the circuit. The average dynamic power P consumed in a particular capacitive element within a device can be calculated using (1), where: $n(K)$ is the number of transitions in the element during K clock cycles, C is the capacitance of the element and V_{dd} is the power supply voltage.

$$P = \frac{1}{2} C V_{dd}^2 \left[\lim_{K \to \infty} \frac{n(K)}{K} \right] \quad (1)$$

For the above equation to hold, the assumption is made that the number of transitions within a given period of time is an ergodic stochastic process, P is a constant value, and the above limit exists. By using this equation on every capacitive component in a design the dynamic power consumed by the device can be estimated. However this can be done only once the design has been placed and routed (to obtain the capacitance values), and then simulated at a low level (to obtain the activities of each signal) with test vectors which are expected to be typical inputs to the design. This is the approach taken by the XPower power estimation tool available from Xilinx [1].

The above approach, while able to provide good estimates for the power consumption in circuits, has a disadvantage in terms of its computational complexity. This becomes a major hurdle if this technique is used in conjunction with optimization routines, which require the rapid estimation of the power consumption of a circuit many times during the optimization process. To answer this particular requirement power estimation methods which operate on a high-level design description have been proposed [2], [3].

This paper makes the following contributions:

- the first bit-width optimization tool to target power optimization,
- demonstration of the operation of the proposed optimization tool through case studies, and,
- a quantification of the power savings achievable by bit-width optimization for power, compared to both optimal uniform bit-width, and multiple bit-width systems optimized for area.

Section II provides a brief summary of related work on power estimation and optimization of power consumption. Section III describes the bit-width optimization scheme used here and Section IV gives details with regards to the implementation and evaluation of PowerBit. In Section V four case studies are considered where the proposed method is applied and the results are evaluated.

II. Background

One method to reduce the power consumption in circuits is to customize the word-length or bit-width of the data-path, this can be thought of as an constrained optimization problem of allocating the optimum bit-widths to the signals in the design. There has been a considerable amount of research conducted on this problem, most of which [4]–[6] has focused on reducing the area occupied by the design in hardware. In these previous works macro-models parameterized by the signals bit-widths are used to estimate the design area during the optimization process.

In [7] it is shown that it is possible to achieve improvements in the power consumption of a design as a side effect of optimizing the bit-widths in the design for area. Indeed, the non-uniform switching activity across bits in a word allows bit-width optimization to aggressively target power reduction. One of the difficulties faced when optimizing for power consumption has been lack of fast, high-level power estimation models. Techniques which rely on gate level simulations to estimate the power consumption are too slow to be used for bit-width optimization.

Due to the need for fast power estimations in optimization systems there has been much research [2], [8]–[12] in to the development of *macro-models* to estimate the power consumed in the functional units of the circuit. The macro-models operate by using word-level statistics of the signals to characterize the activity in them, and then relate this activity to the power consumed in components driven by the signals.

Macro models of individual circuit modules, such as arithmetic components, can account for glitching activity within a component by using characterized equations to model its power consumption. The equations used must be characterized with power measurements from low-level simulations of the component. These equations can be evaluated very quickly, but the loss in accuracy over low-level simulation can be significant if the power consuming behavior of the component is not well captured by the variables chosen to predict power consumption in the equations used.

For the work described in this paper, the macro-model presented in [3] is used to quickly estimate the power consumed in a circuit. The macro-model in [3] combines the macro-model in [2] with a macro-model for estimating the power when signals contain glitch activity. In [3] a technique based on *glitch-profiles* is used to model the propagation of the glitch activity throughout the circuit and to estimate the power consumption due to it.

III. Bit-width optimization

The goal of bit-width optimization is to find the optimal number of bits required to minimize a design metric, such as the area, latency or power consumption of a circuit, without exceeding user specified output error constraints. The bit-width optimization problem can be separated in to two sub-problems: *range analysis* and *precision analysis*.

Range analysis involves studying the data range of the computation and ensuring that the signals in the design have enough bits to accommodate this range. The range analysis setup of the bit-width analyzes the data flow graph of the circuit to determine the minimum number of most significant bits required in each signal to prevent an overflow from happening.

Precision analysis involves analyzing the sensitivity of the error at the outputs of a system to slight changes in the bit-widths of signals within the design. In precision analysis the bit-width analysis tries to find the minimum number of least significant bits for each signal while ensuring that the output of the design does not exceed a given error tolerance.

This paper uses the bit-width analysis technique described in [4], which is capable of providing guaranteed range safety and output error bounds. Affine arithmetic is used for both the range and precision analysis. In the precision analysis an error function is derived to model the effect of least significant bit truncation in the signals on the output via affine arithmetic. In precision analysis it is possible to have many different bit-width allocations which would satisfy the output error requirement. Therefore when performing the precision optimization a cost function to be minimized is used.

The precision analysis optimization problem can then be expressed as (2-3), where $W_0,\ldots,W_n$ are the bit-widths of the signals in the design, $f_{cost} \in \{area, latency, power\}$ is the design cost function to be minimized, f_{error} gives the error at the output of the design due to the selected bit-widths and $E_{req.}$ is the designer specified output error constraint.

$$\text{minimize}: \quad f_{cost}(W_0,\ldots,W_n) \tag{2}$$

$$\text{subject to}: \quad f_{error}(W_0,\ldots,W_n) \leq E_{req.} \tag{3}$$

In [4] the value of E_{req} is specified in terms of the output bit-width required to meet the error requirement.

Since the search space of possible solutions is very large and infeasible to search in a finite amount of time, it is not possible to find the global optimum solution to the precision analysis in most cases.

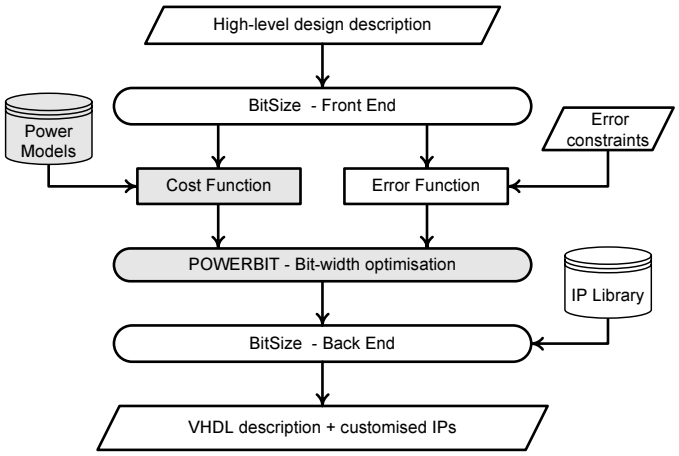

Fig. 1. PowerBit operational flow implemented within BitSize, shaded boxes are operations specific to PowerBit.

The optimization problem described in equations (2) and (3) is a discrete non-linear optimization, to solve this in [4] Simulated Annealing is selected. Simulated Annealing is selected due to its ability to avoid local minima when optimizing.

In [4], the first step of the precision optimization is to find the *uniform solution*. This the set of minimum values for $W_0,\ldots,W_n$ which satisfy inequality (3) such that $W_0 = W_1 = \cdots = W_n$. Starting with this initial solution an improved solution, the *non-uniform solution*, which reduces the cost function in (2) is found using a method based on Simulated Annealing.

IV. Tool implementation

The PowerBit system is implemented within the BitSize [4] bit-width analysis framework. Figure 1 illustrates the operation of the PowerBit system. A high-level description of the hardware design is taken as input to the system and from this description the power cost function and the error function are derived. The power cost function depends on the hardware power models.

These derived functions are used by PowerBit to produce a bit-width optimized internal design description. This optimized internal design description is next converted to a hardware implementation description in VHDL by the BitSize back-end stage; the back-end also instantiates any IP components required using Xilinx CoreGen.

The produced VHDL design descriptions are first synthesized to net-list level descriptions, for this the Synplify Pro FPGA compiler is used. Place and route is performed using the Xilinx tools, targeting the Xilinx Virtex2 1000 device.

In order to evaluate the power consumption reductions produced in designs optimized by PowerBit, a post place and route VHDL description of the net-list is generated along with the routing delay information, and simulated using the ModelSim HDL simulator to ascertain the activity levels of the signals within the circuit. This information is in turn used by the Xilinx XPower tool to estimate the logic power consumption of the design.

Although we acknowledge that the base-line we have chosen for our comparison Xilinx XPower is known to be inaccurate [13], it has the advantage in being able to provide us with breakdown of the power consumption in terms of the components in the circuit.

V. Evaluation

This section considers four case studies that illustrate the operation of PowerBit and demonstrate the power consumption reductions

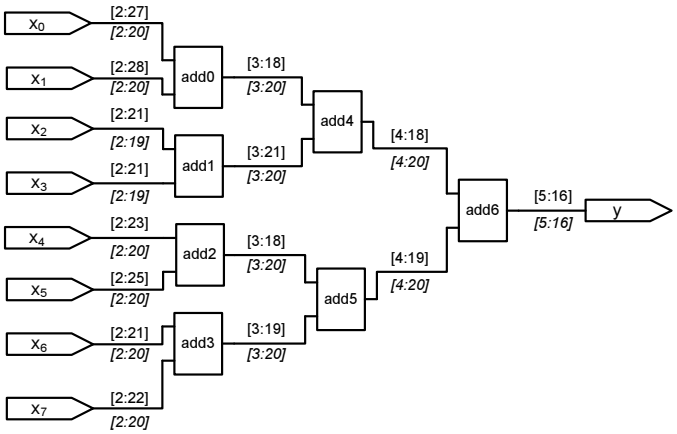

Fig. 2. Data-flow graph for cascaded adders, the edges of the graph is annotated with the signal bit-widths ([Int. BW : Fract. BW]) when optimized using a power based (normal text) and area based (italic text) cost function.

it can achieve. The PowerBit system is implemented within the BitSize [4] bit-width analysis framework. A high-level description of the hardware design is taken as input to the system and from this description the power cost function and the error function are derived. The power cost function depends on the hardware power models.

In order to evaluate the power consumption reductions produced in designs optimized by PowerBit, a post place and route net-list description of the design is simulated and the net activity information derived is in turn used by Xilinx XPower tool to estimate the logic power consumption of the design.

Although we acknowledge that the base-line we have chosen for our comparison Xilinx XPower is known to be inaccurate [13], it has the advantage in being able to provide us with breakdown of the power consumption in terms of the components in the circuit

A. Cascaded Addition

For the first case study we consider a simple cascaded addition circuit, to demonstrate the operation of the PowerBit optimization system. Figure 2 shows the data flow graph for the circuit with the edges annotated with the optimized signal bit-widths.

Figure 3 shows the component-wise breakdown of the logic power inside the cascaded adder circuit. Adders that are located deeper down the cascade consume more logic power than those at the top. As shown in Figure 2, PowerBit allocated more fraction bits to adders at the top of the cascade than to those further down the cascade. In Figure 3 we see that though the adders at the top of the cascade may consume slightly more power after the power optimization, those at the bottom consume much less, providing an overall reduction in the logic power consumption.

B. Matrix Multiplication

The 2×2 matrix multiplication using Strassen's algorithm [14] is considered, which is commonly used as a basic processing element for large matrix multiplications. We assume the elements of the input matrices are over [0,1].

$$\begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix}$$

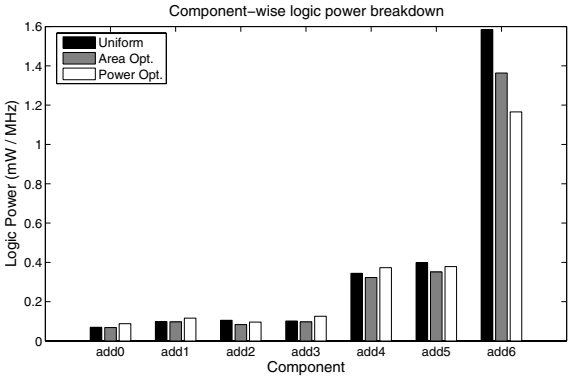

Fig. 3. Component-wise breakdown of the logic power consumption for the cascaded adder for different optimization schemes. The output error requirement is set to 16 bits.

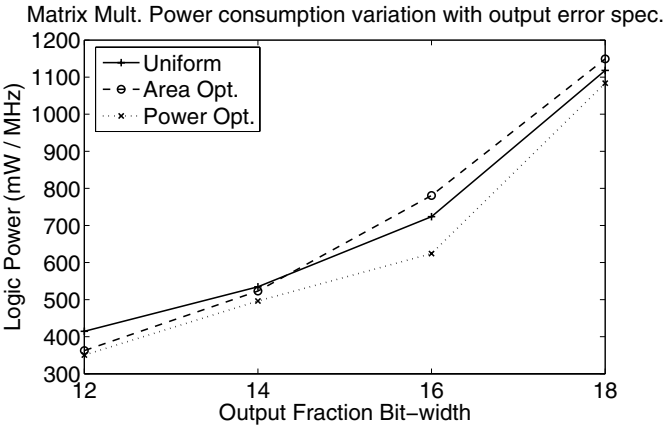

Fig. 4. Variation in Logic Power consumption with output error specification for matrix multiplication implementations.

The four quadrants of the result matrix can be calculated as follows:

$$\begin{aligned} y_{00} &= p_0 + p_3 - p_4 + p_6 \\ y_{01} &= p_2 + p_4 \\ y_{10} &= p_1 + p_3 \\ y_{11} &= p_0 + p_2 - p_1 + p_5 \end{aligned}$$

$$\begin{aligned} p_0 &= (a_{00} + a_{11})(b_{00} + b_{11}) \\ p_1 &= (a_{10} + a_{11})b_{00} \\ p_2 &= a_{00}(b_{01} - b_{11}) \\ p_3 &= a_{11}(b_{10} - b_{00}) \\ p_4 &= (a_{00} + a_{01})b_{11} \\ p_5 &= (a_{10} - a_{00})(b_{00} + b_{01}) \\ p_6 &= (a_{01} - a_{11})(b_{10} + b_{11}) \end{aligned}$$

Figure 4 shows the variation in the logic power consumption of the matrix multiplication implementation with increasing output error specification. The uniform solution consumes the most power, the power cost function based non-uniform solution gives a reduction between 3% and 15% on the uniform solution power. When the area cost function is used the power consumption tends to increase by 3% to 8% for certain output error specifications, as the area cost function is unable to distinguish between adders at the top and bottom of the computation data flow graph.

C. B-Splines

We examine uniform cubic B-splines, commonly used for image warping applications. The B-spline basis functions, B_0, B_1, B_2 and

B_3, are defined by

$$B_0(u) = \frac{(1-u)^3}{6} \qquad B_2(u) = \frac{-3u^3+3u^2+3u+1}{6}$$
$$B_1(u) = \frac{3u^3-6u^2+4}{6} \qquad B_3(u) = \frac{-u^3}{6}q$$

where $u \in [0,1]$. For the implementation of this design, optimizations including shifts instead of multiplications and sharing common intermediate results are carried out.

D. Discrete Cosine Transform (DCT)

We consider the 8×8 discrete cosine transform (DCT) implemented according to [15]. A vector of input data $x_{0...7}$ can be transformed to DCT coefficients $y_{0...7}$ by

$$\begin{bmatrix} y_0 \\ y_2 \\ y_4 \\ y_6 \end{bmatrix} = \begin{bmatrix} c_0 & c_0 & c_0 & c_0 \\ c_2 & c_5 & -c_5 & c_2 \\ c_0 & -c_0 & -c_0 & c_0 \\ c_5 & -c_2 & c_2 & -c_5 \end{bmatrix} \begin{bmatrix} x_0+x_7 \\ x_1+x_6 \\ x_2+x_5 \\ x_3+x_4 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_3 \\ y_5 \\ y_7 \end{bmatrix} = \begin{bmatrix} c_1 & c_3 & c_4 & c_6 \\ c_3 & -c_6 & -c_1 & -c_4 \\ c_4 & -c_1 & c_6 & c_0 \\ c_6 & -c_4 & c_3 & -c_1 \end{bmatrix} \begin{bmatrix} x_0-x_7 \\ x_1-x_6 \\ x_2-x_5 \\ x_3-x_4 \end{bmatrix}$$

where $c_{0...7}$ are trigonometric constants. We use 8-bit unsigned integers for the elements in the input vector $x_{0...7}$.

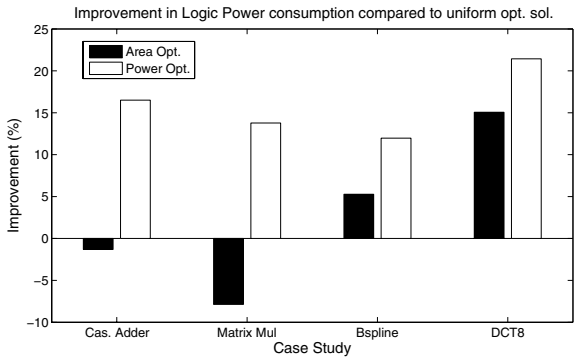

Fig. 5. Logic Power consumption improvements achieved over the uniform solution, when optimizing different cost functions. The output error requirement is set at 16 bits.

E. Results Summary

Figure 5 shows the improvements in logic power consumption achieved when the non-uniform solution is used as opposed to to the optimal uniform solution. For the non-uniform solution two different cost functions based on area and power are used. The power based cost function is capable of providing a reduction in logic power of over 10% in all the four case studies considered. In the case of the DCT8 example this reduction is up to 20%.

The area based optimization is also capable of providing some reduction in the power consumption, as a side effect of area optimization, but in certain cases such as the cascaded addition and matrix multiplication this can result in increases in the power consumption.

Area based optimization is capable of distinguishing only between the type of the arithmetic component, the power based optimization on the other hand is capable of not only distinguishing between different component types, but is also capable of distinguishing between components of the same type based on their location within a computation chain.

VI. Conclusion

In this paper we presented a method to reduce the logic power consumption on FPGA based arithmetic circuits by optimizing the bit-width of the data-path. PowerBit, a tool developed and implemented by us demonstrates this technique. This paper builds upon our previous work involving bit-width optimization and macro-level power modelling. Several case studies were used to illustrate the additional power reductions possible when a power based cost function is used as opposed to an area based cost function when optimizing for the non-uniform solution. We have demonstrated that power reductions of over 10% are possible for the non-uniform solution compared to the uniform solution and improvements of up to 20% in certain cases compared to area cost function based solutions. While this framework allows for logic power and the power consumed by routing wires within components to be targeted by bit-width optimization, the power consumed in routing wires between components is currently unaccounted for. Future work will address this issue.

Acknowledgments

The authors would like to acknowledge the support of Synplicity, Inc., Xilinx, Celoxica and the EPSRC under grant numbers EP/C512596/1 and EP/C549481/1.

References

[1] "Xilinx. http://www.xilinx.com/. Accessed on: 01/03/2006."

[2] 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," in *Proc. IEEE Symp. on Circuits and Systems*, Kos, Greece, 2006.

[3] A. A. Gaffar, J. A. Clarke, and G. A. Constantinides, "Modeling of glitch effects in FPGA based arithmetic circuits," in *Proc. IEEE International Conference on Field-Programmable Technology*, 2006.

[4] D.-U. Lee, A. A. Gaffar, R. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides, "Accuracy guaranteed bit-width optimisation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1990–2000, October 2006.

[5] K.-I. Kum, J. Kang, and W. Sung, "AUTOSCALER for C: an optimizing floating-point to integer C program converter for fixed-point digital signal processors," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 9, pp. 840–848, 2000.

[6] C. Shi and R. Brodersen, "Automated fixed-point data-type optimization tool for signal processing and communication systems," in *Proc. ACM/IEEE Design Automation Conf.*, 2004, pp. 478–483.

[7] G. A. Constantinides, "Perturbation analysis for word-length optimization," in *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines*, 2003.

[8] P. Landman and J. Rabaey, "Architectural power analysis: The dual bit type method," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 3, no. 2, pp. 173–187, 1995.

[9] S. Bobba, I. N. Hajj, and N. R. Shanbhag, "Analytical expressions for power dissipation of macro-blocks in dsp architectures," in *Proc. of the 12th International Conference on VLSI Design*. IEEE Computer Society, 1999, pp. 358–365.

[10] S. Ramprasad, N. Shandhag, and I. Hajj, "Analytical estimation of signal transition activity from word-level statistics," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 16, no. 7, pp. 718–733, 1997.

[11] L. Shang and N. Jha, "High-level power modeling of CPLDs and FPGAs," in *Proc. International Conference on Computer Design*, 2001, pp. 46–51.

[12] T. Jiang, X. Tang, and P. Banerjee, "Macro-models for high level area and power estimation on FPGAs," in *Proc. Great Lakes Symposium on VLSI*. ACM Press, 2004, pp. 162–165.

[13] S. J. Wilton, S.-S. Ang, and W. Luk, "The Impact of Pipelining on Energy per Operation in Field-Programmable Gate Arrays," in *Proc. Field-Programmable Logic and Applications*, ser. LNCS, vol. 3203, 2004, pp. 719–728.

[14] V. Strassen, "Gaussian elimination is not optimal," *Numerische Mathematik*, vol. 13, pp. 354–356, 1969.

[15] A. Madisetti and A.N. Willson Jr., "A 100 MHz 2-D 8×8 DCT/IDCT processor for HDTV applications," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 5, no. 2, pp. 158–165, 1995.