

# ACCURATE MODELS FOR ESTIMATING AREA AND POWER OF FPGA IMPLEMENTATIONS<sup>†</sup>

*Lanping Deng, Kanwaldeep Sobti, Chaitali Chakrabarti*

Department of Electrical Engineering  
Arizona State University, Tempe  
email: {ldeng2, ksobti, chaitali}@asu.edu

## ABSTRACT

This paper presents accurate area and power estimation models for implementations using FPGAs from the Xilinx Virtex-2Pro family. These models are designed to facilitate efficient design space exploration in an automated algorithm-architecture codesign framework. Detailed models for accurately estimating the number of slices, block RAMs and 18x18-bit multipliers for fixed point and floating-point IP cores have been developed. These models are also utilized to develop accurate power models that consider the effect of logic power, signal power, clock power and I/O power. In all cases, the model coefficients have been derived by using curve fitting or regression analysis. The modeling error for the IP cores is very small (average 0.95%). The error for fairly large examples such as floating point implementation of 8-point FFTs is also quite small; it is 1.87% for estimation of number of slices and 3.48% for estimation of power consumption.

**Keywords:** area and power models, FPGA implementation, IP core, regression analysis

## 1. INTRODUCTION

Reconfigurable hardware, especially field programmable gate arrays (FPGA), are widely used in digital signal processing and scientific computing applications. They provide an easy and cost-effective way to evaluate the algorithms from an implementation perspective. The process is cumbersome; the algorithm description in MATLAB or C has to be translated to hardware description language and then synthesized to get accurate area and power numbers. There is only a handful of automated tools that can support this translation for high accuracy scientific applications. Even then, the process is time consuming – it can take hours for large designs.

In this paper we describe a tool to provide fast and accurate estimates of area and power for FPGA based implementations. This tool is integrated with TANOR [1], which is an automated framework for translating MATLAB descriptions into VHDL. By adding the modeling aspect to the TANOR tool, the system-level designer is provided with fairly accurate hardware-related estimates to guide the algorithm design process.

The need for fast and accurate estimation of area and power has been recognized by many and there are several estimations tools for FPGA based implementations. Area models have been developed in [2], [3]. The area estimation method in [2] is based on lookup table (LUT) mapping and requires a logic-level netlist. The method in [3] is at a higher level and develops area models at the data-flow-graph

level. Unfortunately, the number of operations that are considered is quite limited. Power models have been developed in [4], [5], [6]. An RT-level power estimator which considers wire capacitance and switching activity have been considered in [4]. These estimates are made even better in [5] by considering short-circuit power and leakage power. Models for large parametrized IP cores have been presented in [6], [7]; [6] presents area models and detailed power model for fast Hadamard transform, and [7] presents area models for discrete Fourier transform IP cores.

The proposed modelling tool is quite general and is built for designs that utilize the parameterizable IP cores from Xilinx. These include fixed-point and floating-point IP cores for addition/subtraction, multiplication, square root, reciprocal, round and shift operations. The area models of the IP cores are represented in terms of models for number of slices, number of block RAMs (BRAM) and number of 18x18-bit multipliers. The model for the number of slices is obtained by curve fitting and linear regression. In a large design that consists of multiple IP cores, the total number of slices is estimated by an empirical model that also takes into account slice utilization. The power consumption is modelled into components that are related to area such as logic power, clock and signal power, and those that are not related to area such as static power and I/O power. Overall, the model is highly accurate. For designs with very high slice utilization, such as fully pipelined floating point implementation of 8-point FFTs, the error is only 1.87% for estimation of number of slices and 3.48% for estimation of power consumption.

The rest of the paper is organized as follows. Section 2 presents the detailed area models for the IP cores followed by models for system-level area and power consumption. Section 3 validates the proposed models for LUT intensive designs as well as FFT and DCT computations. Section 4 offers concluding remarks.

## 2. MODELLING PROCEDURE

This section presents our models to quickly and accurately estimate the area and power of implementations on a Xilinx Virtex-2Pro FPGA. The IP cores that are currently supported are listed in Tables 1 and 2. We first present models to estimate the area of each IP core followed by models to estimate the area and power consumption of the whole design.

In order to develop the models, the IP cores were synthesized using Xilinx ISE 8.2i and Synplify Pro 8.6.2. In each case, at least 50 configurations were synthesized and implemented, and curve fitting and non linear regression analysis [8] were used to develop the models of the number of slices. For each IP core, the estimation

<sup>†</sup> This work is supported by a grant from DARPA W911NF-05-1-0248

error is given by:

$$abs. value\left(\frac{Synthesis\_result - Estimated\_result}{Synthesis\_result}\right)$$

## 2.1. AREA MODELLING

The models for estimating area of both fixed point and floating point IP cores are represented in terms of number of slices, number of block RAMs and number of 18x18 multipliers.

The following notations are used:  $FX$  for “fixed point format” with  $I$  for “integer bits” and  $F$  for “fraction bits”, and  $FP$  for “floating point format” with  $E$  for “exponent bits” and  $M$  for “mantissa bits”. In addition,  $in()$  and  $out()$  represent the number of input bits and output bits, respectively.

### 2.1.1. Fixed Point IP cores

For IP cores using fixed-point data, we have built models for Add/Sub, Square Root, Round, Fixed-to-Float, Reciprocal, Shifter, Multiplier and BRAM units.

**Number of Slices:** The number of occupied slices is one of the most important metrics for representing the area of a FPGA implementation. In Virtex-2Pro FPGA, a slice consists of two function generators configurable as 4-input lookup tables, 16-bit shift registers or 16-bit distributed SelectRAM+ memory, two storage elements configurable as D-flip flops or latches, carry logic, and arithmetic logic gates.

Let  $Slice()$  denote the estimate of the number of slices in an IP core. For fixed point cores, the relationship between the number of slices and the parameters is linear for Add/Sub, Round and Shifter; the relationship is nonlinear for Square Root, Fixed-to-Float and Reciprocal. We provide the expressions for estimating the number of slices for Adder and Square Root units as representative examples. Note that the actual number of slices is an integer obtained by rounding up the value obtained by these expressions.

$$Slice(add\_FX) = \max(in_1(I + F), in_2(I + F))$$

$$Slice(sqrt\_FX) = 0.56 \cdot [in(I) + 2.00 \cdot in(F)]^{1.8024} + 38.89$$

The model parameters and the average error for the number of slices for fixed point IP cores is given in Table 1. Note that the errors are very small for the individual IP cores.

**Table 1.** Fixed-point IP cores: Parameters and Errors in estimating the number of slices

FX cores	Model Parameters	Avg.Err
Add/Sub	$in_1(I, F), in_2(I, F)$	0.00%
Square Root	$in(I, F)$	0.10%
Round	$in(I)$	0.00%
Fixed-to-Float	$in(I, F)$	1.47%
Reciprocal	$divisor(I, F), out(F)$	1.58%
Shifter	$in(I, F), amount\ of\ shift$	0.00%

**Number of 18x18-bit Multipliers:** Virtex-2Pro FPGA provides 18x 18-bit multiplier (MULT18) blocks that are optimized for high-speed operations and low power consumption. These multipliers are used automatically by the synthesis tool to realize multiplications in HDL descriptions.

The MULT18 block supports two data input ports: 18-bit signed or 17-bit unsigned. The number of MULT18 blocks to implement the product of  $in_1(I, F)$  and  $in_2(I, F)$  is given by

$$M_{18}(mult\_FX) = \lceil \frac{in_1(I + F)}{17} \rceil \cdot \lceil \frac{in_2(I + F)}{17} \rceil$$

**Number of Block RAMs:** The Virtex-2Pro FPGA has a total of 444 18-Kb block SelectRAM+ (BRAM) resources. The BRAM has two types of configurations, the first type contains 2K x 9-b, 1K x 18-b and 512 x 36-b configurations with access to all 18-Kb memory locations, the other type contains 16K x 1-b, 8K x 2-b and 4K x 4-b configurations with access to 16-Kb memory locations.

In order to store a table with  $n$  entries and  $p$  bits per entry, a BRAM configuration with greater than  $n$  entries is selected and then multiple BRAMs of that type are utilized to accommodate the  $p$  bits per entry. For instance, if  $n = 1600$ ,  $p = 23$ , we choose the 2k x 9-b configuration and then utilize  $\lceil \frac{23}{9} \rceil = 3$  such BRAMs.

### 2.1.2. Floating Point IP cores

For IP cores using floating point data, we have built models for Add/Sub, Multiplier, Square Root, Reciprocal, Float-to-fixed and Shifter (see Table 2).

The relationship between the number of slices and the parameters is linear for Add/Sub and Shifter, the relationship is nonlinear for Reciprocal, Square Root and Float-to-Fixed, and is piecewise linear for Multiplier. We present the expressions for estimating the number of slices for Adder, Reciprocal and Multiplier units below.

$$Slice(add\_FP) = 5.40 \cdot E + 11.06 \cdot M + 51.20$$

$$Slice( recip\_FP) = 2.20 \cdot E + 3.94 \cdot M^{1.764} + 14.24$$

$$Slice(mult\_FP) = \begin{cases} 5.00 \cdot E + 2.67 \cdot M + 4.00 & \text{if } M \leq 17 \\ 3.66 \cdot E + 5.46 \cdot M + 24.82 & \text{if } M > 17 \end{cases}$$

The model parameters as well as the average errors for estimating the number of slices are given in Table 2. Note that the modeling error is fairly low except for the square root unit.

**Table 2.** Floating-point IP cores: Parameters and Errors for estimating the number of slices

FP cores	Model Parameters	Avg.Err
Add/Sub	$E, M$	0.74%
Square Root	$E, M$	3.24%
Float-to-Fixed	$out(I, F)$	1.77%
Reciprocal	$E, M$	1.33%
Shifter	$E, M, amount\ of\ shift$	0.00%

The function to calculate the number of MULT18 blocks needed for floating point multiplication is shown below. Here both the inputs have the same number of mantissa bits.

$$M_{18}(mult\_FP) = \lceil \frac{M}{17} \rceil \cdot \lceil \frac{M}{17} \rceil$$

The number of BRAMs in floating point design is estimated in the same way as in the fixed point design.

## 2.2. Total Area Estimation

A typical design consists of multiple components, where each component consists of multiple IP cores, MULT18 blocks and BRAMs. The total number of BRAM modules ( $T_{BRAM}$ ) and the total number of MULT18 blocks ( $T_{M18}$ ) can be obtained by adding the number of these modules in each of the components. However, the number of slices of a complete design cannot be estimated as a sum of the number of slices in each component. This is because during synthesis, placement and routing, the Xilinx software automatically optimizes the design and the optimization procedure is not transparent to the user. For instance, in a small design, all the LUTs and flipflops in a

slice are not necessarily utilized. As the design gets larger, more of the LUTs and flipflops in a slice get utilized and so the number of slices do not scale proportionately. Thus there is a need to establish a relationship between slice utilization and the size of a design to correct this discrepancy.

We developed a simple empirical model based on evaluation of a large number of candidate designs. We calculated a scaling factor,  $\alpha$ , that is proportional to the number of slices in the design. The total number of slices,  $T_{slice}$  is then obtained by scaling the total number of slices obtained by adding the number of slices for each component in the design by  $\alpha$ .

$$T_{slice} = \alpha(\text{utilization}) \times \sum \text{Slice}()$$

Let  $S_{slice} = \sum \text{Slice}()$  be used to estimate the ‘‘slice utilization’’ of the design. Here we approximate the relationship between the scaling factor  $\alpha$  and  $S_{slice}$  by a damped pendulum function [8] as follows.

$$\alpha(\text{utilization}) = A \cdot e^{-\beta \cdot S_{slice}} \cdot \cos(w \cdot (S_{slice} - \phi)) + \theta,$$

where  $A = 2.374$ ,  $\beta = 0.0067$ ,  $w = 0.468$ ,  $\phi = 266.59$ , and  $\theta = 1.128$ .

### 2.3. Total Power Estimation

FPGA power consumption depends on the design, and is influenced by a number of factors such as clock frequency, activity rates, design density (number of interconnects), logic block and interconnect structure, power supply voltage levels and output loading[6]. In order to derive the power model, the power of the FPGA implementations is measured in Xpower. The clock frequency is set to 125MHz, the activity rate is set to 12.5%, the power supply voltages and output loading are set to the Xpower default values.

The estimated power is the sum of static power and dynamic power. The static power is set by Xpower and depends on the specific FPGA family. The dynamic power is the sum of logic power, input/output power, signal power and clock power.

In order to derive the power models, we group the power components into two categories: (i) those that are not dependent on the area of a design, including static power and input/output power, and (ii) those that are proportional to the area of a design, including logic power, signal power and clock power. The power models for the components that are proportional to the area of the design are derived by performing nonlinear regression analysis on the area and power data of the IP cores. In each case, at least 50 configurations were synthesized to obtain the model parameters.

#### 2.3.1. Power Components not related to Area

**Static Power:** Xpower has a default static power consumption which is set to 572.38 mW for a single Virtex-2Pro-100 device (in Xilinx ISE 8.2i).

**Input and Output Power:** When chip voltage, clock frequency and activity rate are fixed, the input power and the output power depends on the number of Input/Output Blocks (IOBs), which is proportional to the number of input/output pins in the design. The expressions for estimating the input and output powers are listed below.

$$Power_{input} = (\text{input\_pins}) \cdot 0.125 + 1$$

$$Power_{output} = (\text{output\_pins}) \cdot 2.25$$

The average error of input and output power estimates is 5.09% and 2.13%, respectively.

#### 2.3.2. Power Components related to Area

The majority of the dynamic power comes from logic power, clock power and signal power, which is greatly affected by the area of the actual implementation. We develop separate models for implementations that need block RAM and those that do not. In each case, we derive the model coefficients using regression analysis.

##### Implementations with BRAMs:

**Logic Power:** Logic power is a function of the number of slices, block RAMs and MULT18.

$$Power_{Logic} = \alpha_1 \cdot (T_{slice})^{c_1} + \alpha_2 \cdot (T_{M18})^{c_2} + \alpha_3 \cdot (T_{BrAm})^{c_3} + c_4$$

where  $\alpha_1, \alpha_2, \alpha_3, c_1, c_2, c_3$  are constants listed in Table 3.

**Signal Power / Clock Power:** Signal power is proportional to the number and length of nets over which signal switching occurs. Clock power depends on the distribution of the clock nets, which depends on the chip area. We put these two components together since they are both dependent on the total area of the implementations

$$Power_{signal}(\text{clock}) = \alpha_1 \cdot (c_1 \cdot T_{slice} + c_2 \cdot T_{M18} + T_{BrAm})^{c_3} + c_4$$

where  $\alpha_1, c_1, c_2, c_3$  and  $c_4$  are constants listed in Table 3.

**Table 3.** Coefficients for power models of implementations with BRAM

Power	$\alpha_1$	$\alpha_2$	$\alpha_3$	$c_1$	$c_2$	$c_3$	$c_4$
Logic	0.033	-1.28e-7	-5.93	1.180	4.213	0.827	-1.26
Signal	0.420	n/a	n/a	0.122	-0.423	1.085	9.46
Clock	0.179	n/a	n/a	0.198	1.239	1.056	47.47

##### Implementations without BRAMs:

In designs where no BRAMs are needed, the logic power, signal power and clock power can be estimated using similar formula; the actual value of the coefficients are quite different as shown in Table 4.

$$Power_{Logic}(\text{signal/clock}) = (c_1 \cdot T_{slice} + T_{M18})^{c_2} + c_3$$

**Table 4.** Coefficients for power models of implementations without BRAM

Power	$c_1$	$c_2$	$c_3$
Logic	0.073	0.963	-1.647
Signal	0.082	1.032	9.250
Clock	4.231	0.588	-18.729

## 3. EXPERIMENTAL RESULTS

In this section, we compare the area and power estimates using our models and those obtained by the actual synthesis followed by P&R for a representative set of examples. Two groups of experiments are performed, one is for implementations that require BRAMs such as lookup table based function evaluations and the other is for implementations that require no BRAM such as 1D Discrete Fourier Transform, 1D Discrete Cosine Transform and FIR filter.

### 3.1. Lookup Table based Implementations

We choose two functions that are important in scientific computations and are implemented using a combination of lookup tables and

interpolation schemes [9]. The two functions are  $J_0(x)$  which is the Bessel function of the first kind with zero order, and  $e^{-x}$  which is the exponential function. We use TANOR tool [1] to generate the HDL of these two functions when implemented in fixed point format. The configuration labeled  $(n, X, Y)$  corresponds to the case where  $n$  is the degree of Taylor series,  $X$  is the maximum number of total bits and  $Y$  is the maximum number of fraction bits in fixed point format.

**Table 5.** Resource estimation results for  $J_0(x)$  and  $e^{-x}$  functions

Config.	Slices			Total Power (mW)		
	Estim.	Synth.	Err.	Estim.	Synth.	Err.
$J_0(3,64,5)$	297	307	3.26%	397	369	7.80%
$J_0(3,64,10)$	418	441	5.22%	417	406	2.86%
$J_0(3,64,15)$	779	759	2.64%	508	482	5.50%
$J_0(3,64,20)$	1258	1113	12.99%	629	583	7.99%
$J_0(9,64,5)$	1143	1135	0.70%	586	590	0.66%
$J_0(9,64,10)$	1580	1844	14.30%	707	772	8.35%
$J_0(9,64,15)$	2057	2437	14.97%	845	945	10.09%
$J_0(9,64,20)$	4779	4315	10.57%	1589	1437	10.62%
exp(3,64,5)	187	194	3.61%	368	339	8.50%
exp(3,64,10)	291	273	6.58%	391	363	7.79%
exp(3,64,15)	517	465	11.18%	445	424	5.00%
exp(3,64,20)	805	722	11.45%	518	481	7.87%

The comparison results are shown in Table 5. For the 12 configurations shown here, the average error is 8.20% for the number of slices and 6.98% for the total power. The results of block RAM and 18x18-bit Multipliers are not shown because there is no mismatch between the estimated and synthesized results, as expected.

### 3.2. Transform Computations

In this section, we choose two algorithms which are widely used in digital signal processing: the 8-point FFT (FFT8) and the 8-point DCT (DCT8). The FFT8 is implemented in floating point; the notation  $(E, M)$  stands for  $E$  exponent bits and  $M$  mantissa bits. We also pipelined two FFT8 implementations on a single FPGA to check our models for situations of extremely large slice utilization (99%). The DCT8 is implemented in fixed point; the notation  $(8, 16)$  means each input/output port is 8 bits and the internal variables are 16 bits.

**Table 6.** Resource estimation results for FFT8 and DCT8

Config.	Slices			Total Power (mW)		
	Estim.	Synth.	Err.	Estim.	Synth.	Err.
FFT(8,16)	16515	16808	1.74%	4835	4922	1.75%
FFT(8,18)	18112	18377	1.44%	5230	5244	0.26%
FFT(8,20)	19462	19486	0.12%	5554	5572	0.33%
FFT(8,23)	21480	21622	0.65%	6036	6040	0.06%
2 FFT(8,16)	33030	34219	3.47%	7838	7438	5.38%
2 FFT(8,18)	36225	37389	3.11%	8535	7971	6.98%
2 FFT(8,20)	38925	39680	1.89%	9092	8502	6.94%
2 FFT(8,23)	42961	44066	2.51%	9924	9353	6.01%
DCT(8,16)	431	421	2.38%	896	893	0.36%

Table 6 lists the estimated values, the actual values and the modeling error for both the number of slices and the power consumption of the two transform examples. For the FFT8 and DCT8 configurations, the average error is 1.92% for the number of slices and 3.14% for the total power. Note that the error here is lower than the LUT based implementations presented in Table 5. This is because the

estimate for the number of slices is a lot more accurate when the design is large and a significant portion of the FPGA slices is utilized. Since the estimate of the power consumption is closely related to the estimate of the number of slices, this translates to a fairly accurate estimate of the power consumption.

## 4. CONCLUSION

In this paper we have presented area and power estimation models for IP core based FPGA implementations. These models were developed to speed up the algorithm-architecture co-exploration for systems that have to meet area/power/accuracy requirements. The models consist of parameterized functions that estimate the resource (number of slices, MULT18, BRAM) for IP cores, and use of these functions in models to accurately estimate system-level area and power consumption. The models have been derived using curve fitting and non-linear regression methods. The average error for fairly large designs (FFT8) is only 1.87% for area estimation and 3.48% for power estimation. Further more, the time required to generate these estimates is of the order of microseconds, as compared to minutes or even hours for designs which undergo actual synthesis followed by P&R. While all the results presented in this paper are for Xilinx Virtex-2Pro-100 FPGA, the method can be applied to many other FPGA platforms as well.

## 5. REFERENCES

- [1] J. Kim and et al., "TANOR: A tool for accelerating N-body simulations on reconfigurable platform," in *Proceedings of FPL'07: International Conference On Field Programmable Logic And Applications*, 2007, pp. 68 – 73.
- [2] M. Xu and F. Kurdahi, "Area and timing estimation for lookup table based FPGAs," in *Proceedings of EDTC'96: the European conference on Design and Test*, 1996, pp. 151–157.
- [3] R. Enzler and et al., "High-level area and performance estimation of hardware building blocks on FPGAs," in *Proceedings of FPL'00: International Workshop on Field-Programmable Logic and Applications*, 2000, pp. 525–534.
- [4] D. Chen and et al., "Low-power high-level synthesis for FPGA architectures," in *Proceedings of ISLPED'03: International Symposium on Low Power Electronics and Design*, 2003, pp. 134–139.
- [5] K. K. W. Poon and et al., "A flexible power model for FPGAs," in *Proceedings of FPL'02: International Conference on Field-Programmable Logic and Applications*, 2002, pp. 312–321.
- [6] A. Amira and S. Chandrasekaran, "Power modeling and efficient FPGA implementation of FHT for signal processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, pp. 286–295, March 2007.
- [7] P. A. Milder and et al., "Fast and accurate resource estimation of automatically generated custom DFT IP cores," in *Proceedings of the International Symposium on Field Programmable Gate Arrays*, 2006, pp. 211–220.
- [8] P. H. Sherrod, "Nonlinear Regression Analysis Program," 2005, <http://www.nlreg.com/NLREG.pdf>.
- [9] K. Sobti and et al., "Efficient function evaluations with lookup tables for structured matrix operations," in *Proceedings of SiPS'07: IEEE Workshop on Signal Processing Systems*, 2007, pp. 463 – 468.