Power analysis of multiplier blocks.

Suleyman Demirsoy
Andrew Dempster
Izzet Kale

Cavendish School of Computer Science

POWER ANALYSIS OF MULTIPLIER BLOCKS

Süleyman Sırrı Demirsoy, Andrew G. Dempster and İzzet Kale

University of Westminster, 115 New Cavendish Street, London W1W 6UW, United Kingdom
Tel: +44 20 7911 5000-3611(ext), E_mail: demirss@cmsa.westminster.ac.uk

ABSTRACT

In this study, three multiplier-blocks generated by different algorithms are analyzed for their power consumption via transition count based on their implementation on the Xilinx Virtex device. The high level Glitch-Path method, which is used for estimating the relative figures of transitions occurring at the outputs of the adders, has been refined for more accurate estimation and a new method GP Score is proposed. Several design issues are discussed regarding ways of reducing the transitions.

1. INTRODUCTION

Multiplier-Blocks based on the primitive operations (add, subtract and shift) lead to better utilization of the resources in the parallel implementation of digital filters [1]. When the FIR filter structure shown in Figure 1 has integer coefficients, the multiplier-block can be realized to generate product of the input x[n] and the coefficients, leading to less area and lower power consumption in the filter.

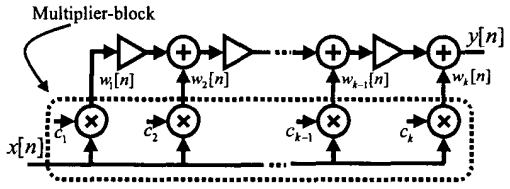

Figure 1 Transposed direct form FIR filter structure

There are several algorithms for generating multiplier-blocks. They all produce a graph with vertices representing two-input adders and edge values representing the amount of shifts. The numbers near the vertices are all odd numbers called fundamentals, which represent the partial products of the input at the output of that adder. Figure 2 shows two sample graphs that generate the product of 25.

The algorithm proposed by Bull and Horrocks (BH)[1] and its modified version (BHM)[2] looks for the closest match to the target value from a range of multiples of all fundamentals and add to graph and continue to do so until every coefficient value is formed. Reduced Adder Graph (RAG-n) [3] algorithm performs an exhaustive search for all possible structures that can be formed with one adder and then continue to do so until the targets are formed.

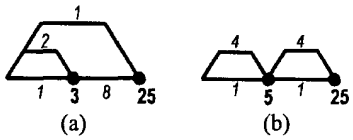

Figure 2 Two single- coefficient graphs for producing 25

The cost criteria of most algorithms are the number of adders in the graph [2][3]. The RAG-n algorithm gives the graph with lowest adders. However, it has been shown that fewer adders do not always imply less power [4]. This happens when the logic-depth, the longest path (measured in edges) from the input to any node in the graph, is bigger in comparison. BHM algorithm performs better than RAG-n for long word-length coefficients. A new algorithm, C1, which has been proposed in [5], aims to reduce the logic-depth of the multiplier-block by choosing the shallowest graph among the candidates even if this results in some additional adders.

In digital CMOS circuits the major source of power dissipation is due to transitions at the circuit nodes, and it is formulated as follows [6]:

$$P_{switching} = \alpha_{0\to1} C_L V_{dd}^2 f_{clk} \qquad (1)$$

where $\alpha_{0\to1}$ is the node transition activity factor (the average number of times the node makes a transition in one clock period), C_L is the load capacitance, V_{dd} is the supply voltage to the circuit and f_{clk} is the clock frequency of the circuit. It can be easily inferred that the number of transitions taking place in two circuits is an acceptable measure for the comparison of the relative power consumption provided that the number of nodes are not significantly different [6].

Glitch-Path (GP) count, a high-level tool to estimate the transition activity at the output nodes of the adders, was proposed in [7]. It relies on the fact that transitions generated by an adder output produce more transitions on the next adder stage when there is no pipelining. GP count for a node is formulated as follows:

$$GP^i_{output} = GP^i_{input\_1} + GP^i_{input\_2} + 1 \quad (2)$$

where $GP^i_{input\_1}$ and $GP^i_{input\_2}$ are the GP counts at the inputs of i^{th} node (adder).

The total number of GPs in a graph is then defined as:

$$GP_{Total} = \sum_i GP^i_{output} \quad (3)$$

In this paper, a new method for power comparison estimation, GP Score, has been proposed based on the GP idea. Three multiplier-blocks generated by C1, BHM and RAG-n algorithms were implemented to measure their transition counts and verification of the GP Score. Section 2 is about the calculation of GP Score concept. Implementation details and timing simulations are explained in Section 3. A discussion about the results of all the experimental work undertaken and several design issues on low-power design of multiplier-blocks are given Section 4. Section 5 concludes the paper.

2. GP SCORE

The GP idea suggests that an adder would produce a GP plus the number of GP's coming to its inputs. This idea as we reported in the past [7] did not take into account the number of adder bits in an adder. Therefore a better and more representative way of predicting the glitch generation would be by considering the number of the adder cells deployed in the actual implementation.

Bearing these assumptions in mind, the procedure for calculating the GP score is as follows:

1) For an adder or subtractor, calculate the word-length of the output, n, by;
$$n = floor\,(\log_2(x)+1) + 8 \quad (4)$$
where x is partial product and 8 is the number of bits of the input. The *floor* function rounds the number to the nearest integer towards minus infinity.

2) Calculate the maximum number of zeros, e, that are padded to the end of the inputs (if any) by;
$$e=\log_2(max(v1,v2)) \quad (5)$$
where $v1$, $v2$ are edge values.

3) If the operation is a subtraction, the actual adder length, r, is;
$$r = n \quad (6a)$$
If the operation is addition, the adder length is;
$$r = n - e \quad (6b)$$
This is due to the fact that adder cells having one of their inputs permanently connected to zero will not be implemented.

4) If the shift value of one of the inputs is larger than the other inputs word-length, the transition of glitches along the carry chain will decrease. Therefore, the adder length, r, is modified to be;
$$r = (n - e+1) + e/2 \quad (6c)$$
for the subtractors. Figure 3 shows this effect on a subtractor where the transition counts of each bit is shown separately for the generation of 2047. It is implemented as 1x2048 - 1x1 where the input to the filter, shown by 1,is shifted for 11 bits and subtracted from itself.

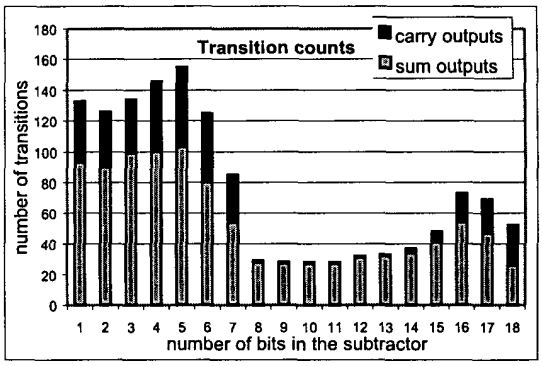

Figure 3 Transition counts for a subtractor that generates the product of input value by 2047.

5) For logic-depth values greater than 1, the GP score coming from the previous adders should be considered as the sum of the GP scores for the carry and sum outputs. Therefore a scaling factor 0.67, which has been empirically calculated from real experimental transition figures, should be applied to the GP scores that are affecting the next adder.

6) As the logic-depth increases, the amount of transitions generated by each adder cell also increases. This fact has been observed by considering the real timing data too. Therefore another empirically derived coefficient value of (k^α) is applied to the adder length where k is the logic-depth and α is a constant with value 0.55.

7) The resulting overall GP score formula from equations (4), (5) and (6) is:
$$GPS = r\,(k^\alpha) + 0.67(GPS_{input1}+GPS_{input2}) \quad (7)$$

which is a refined form of equation (2) by considering the details about the particular adder or subtractor.

This formula operates only at top-level –on the multiplier-block and doesn't include or need any low-level data, which is not available to multiplier-block designers. It can easily be incorporated to the design data we have for comparing their relative power consumption. There is a point to remember that, this formula has been derived and used for 2's complement number representation only. It does not function properly for different number formats and alternative formulas need to be derived.

3. IMPLEMENTATION & SIMULATION

Three multiplier-blocks for the FIR filter with coefficients {-710, 327, 505, 582, 398, -35, -499, -662, -266, 699, 1943, 2987, 339, 2987, 1943, 699, -266, -662, -499, -35, 398, 582, 505, 327, -710} [5] have been designed using VHDL. The filter is a Remez design of order 24 with 12-bit coefficients. For the multiplier-block implementation, the even values are halved until an odd fundamental is found. The absolute value of these coefficients are generated by the multiplier-block and negated by exchanging the adder in the delay line with a subtractor.

Table 1 shows the structure of the multiplier-block generated by the BHM algorithm. Each row is an adder and the details about the input to that adder are given in the 2nd column. Last column gives the logic-depth up until that adder. The edge values are implemented by hard-wired shifting of the input. If the edge value is negative, a subtractor is generated instead of an adder. Bold partial products are the fundamentals of the coefficients. All adders and subtractors used were of ripple-carry type with optimized length for a particular product.

The VHDL implementation of the resulting filter has been hierarchically synthesized using the Leonardo Spectrum software [8]. Designs have been optimized for delay. They were implemented on the XILINX Virtex FPGA device; model BG432-4, with the Alliance tool [9]. The placer effort has been set to 2 and timing data was produced after the actual routing for back-annotated simulations.

Timing simulations were performed with the Modelsim simulator with 1ps precision [10]. The filters are excited with 512 uniformly distributed 8-bit random numbers using 2's complement representation. The transitions occurring at the sum and carry output of each adder in the multiplier blocks has were counted.

TABLE 1 Structure of the BHM multiplier-block

Partial product	**Input1xedge1 + input2xedge2**	Logic depth	Partial product	**Input1xedge1 + input2xedge2**	Logic depth
63	= 1x-1 + 1x64	1	**291**	= 35x1 + 1x256	3
505	= 63x8 + 1x1	2	**199**	= 17x8 + 63x1	2
17	= 1x16 + 1x1	1	**499**	= 5x-1 + 63x8	2
35	= 17x2 + 1x1	2	**331**	= 327x1 + 1x4	4
133	= 35x2 + 63x1	3	**699**	= 175x4 + 1x-1	4
175	= 35x4 + 35x1	3	243	= 499x1 + 1x-256	3
5	= 1x4 + 1x1	1	**1943**	= 243x8 + 1x-1	4
355	= 5x64 + 35x1	3	747	= 243x1 + 63x8	4
41	= 5x8 + 1x1	2	**2987**	= 747x4 + 1x-1	5
327	= 41x8 + 1x-1	3	**3395**	= 747x1 + 331x8	5

4. RESULTS

The results are presented in Table 2. According to the transition figures gathered from the timing simulations, the C1 design has the least amount of transition activity, whereas the RAG-n design has two times more transitions than the others, despite its adder-count figure. It is clearly seen that the logic-depth and GP counts are well correlated with the number of transitions. Our new GP Score metric came out to be the best indicator of transition activity among all measures when the individual ratios with transitions are considered.

Figure 4 shows the normalized ratios of GP score and GP counts to the actual transitions for all the adders in three multiplier blocks where 1 represents perfect estimation. The first 20 adders are for the BHM design and the next 19 adders are for the C1 design. The right-most point on the graph shows an adder with logic-depth 9 from RAG-n design. The standard deviation of the ratios came out as 0.07 for GP score and 0.21 for GP count. Maximum estimation errors for these designs are 60% for GP count and 20% GP score. It is clear from the graph that the GP Score ratio can be taken as a good indication of the transition figures for any adders in any design that uses carry-ripple adders with no pipelining.

Table 3 shows the details about the implementation of an adders and a subtractor for product 35 in two different designs. The adder is from BHM design. One of its inputs is from the output of the adder for product 17 and the other one is connected to the input of the filter. The subtractor is from C1 design and used for product 35 too. Both of its inputs are connected to the output of the adder for product 5. As seen from the table, transition activities are significantly different even though their logic-depth are the same. One reason for this is the difference of number of actually implemented adder/subtractor cells in the designs. Subtractors have almost always more bits implemented than the adders. Another reason is the

amount of transition activity occurring at the inputs of the adder/subtractor cells. Both of these facts are covered by the idea of the GP score and the outcome of the GP Score is in good correlation with the transition activity as seen from the Table 3.

TABLE 2 Results

Design	Adder count	Logic depth	GP	GP Score	Transitions
BHM	20	5	77	1140	1037407
C1	19	4	67	864	944440
RAG-n	18	9	140	1798	1766134

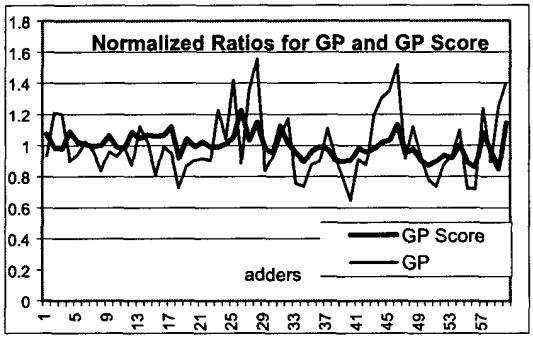

Figure 4 Normalized ratios of GP Score and GP count to the actual transitions for all the adders in the designs.

TABLE 3 Comparison of two implementations from two different designs for their transition activity

Design	Product	Formation	Logic depth	Transitions	GP Score
BHM	35	= 17x2 + 1x1	2	26219	27
C1	35	= 5x-1 + 5x8	2	39962	35

5. CONCLUSION

A new high-level method called GP score ratio has been proposed and tested. Three multiplier-blocks generated by the RAG-n, BHM and C1 algorithms were implemented on a XILINX Virtex device and their transition activity was observed. The C1 design was found to be slightly better than BHM. The RAG-n design had the most transition activity even though it has the least amount of adders. Our novel GP Score metric was found to be a good indicator of transition activities of the adders with 20% maximum estimation error when compared to 60% error of GP count. Future work will focus on the power estimation of multiplier-blocks with carry-save adders.

6. REFERENCES

[1] Bull D.R. and D.H Horrocks, "Primitive operator digital filters", *IEE Proceedings-G,* vol. 138, no.3, June 1991, pp. 401-412

[2] A G Dempster and M D Macleod, "Constant integer multiplication using minimum adders", IEE Proceedings - Circuits, Devices and Systems, vol 141, no 5, pp407-413, Oct 1994

[3] Dempster A.G. and Macleod M.D., "Use of minimum-adder multiplier-blocks in FIR digital filters", *IEEE Trans. CAS-II,* vol. 42, no. 9, pp. 569-577, November 1995

[4] Horrocks D.H. and Wongsuwan Y., "Reduced Complexity Primitive Operator FIR filters for low power dissipation", *Proc ECCTD '99* , Stresa, Italy, pp273-276, 1999

[5] Dempster A.G., S.S. Demirsoy and I. Kale "Designing Multiplier Blocks with Low Logic Depth", *submitted to IEEE ISCAS'2002 conference.*

[6] Chandrakasan A.P. and R.W. Brodersen, "Minimising power consumption in digital CMOS circuits", *Proc. IEEE,* vol. 83, pp. 498-523, April 1995.

[7] Demirsoy S. S., A. Dempster and I. Kale, "Transition analysis in multiplier-block based FIR filter structures", *IEEE Int. Conf. on Elect., Circuit & Systems (ICECS),* Lebanon, December 2000.

[8] http://www.exemplar.com/products/leonardospectrum.html

[9] http://www.xilinx.com/

[10] http://www.modelsim.com