Politecnico di Torino

III Facoltà di Ingegneria

Exercises and Homeworks for the course Integrated Systems Architecture

Master degree in Electronic Engineering

Group 07

Campagnoli Rafael, Fraval Adrien, Galasso Luigi

December 16, 2019

Contents

CHAPTER 1

Lab 2: Digital arithmetic and logic synthesis

1.1 Introduction and Objective

The purpose of this laboratory assignment was to study the floating point multiplier architecture. The architecture has been tested and then synthesized exploiting more than one possible implementation of the Stage 2 Significands multiplier. A further Synthesis have been performed exploring the possibility of retiming the architecture and optimizing it through particular Design Compiler commands. As last step, a MBE (Modified Booth Encoding) multiplier for unsigned data has been implemented and substituted in place of the behavioral multiplication. Afterwards, a final optimized synthesis has been done and all the timing and area results compared.

1.2 Floating Point Multiplier

The Floating Point Multiplier examined is a 4-stage pipelined multiplier with a 32-bit of parallelism. In order to test it, a TestBench was realized in Verilog and the fp_samples.hex file has been used for the multiplicands choice. The results obtained were stored in the prod.hex file. After that, two registers have been added to the input to store the 2 multiplicands in the VHDL file of stage 1, and then the same TestBench was used to confirm the correct behavior through a further simulation.

1.2.1 Flatten Hierarchy Synthesis

To speed up the Synthesis phase, the script *script.scr* was exploited. As a first Synthesis, the design compiler was forced to flatten the hierarchy (using the command $ungroup - all - flatten$) and then synthesize the architecture. The maximum frequency and the area have been found. In particular the total Cell Area is 4122.99 μm^2 and the critical Path Delay obtained is 1.40 ns therefore the maximum frequency achievable is 700 MHz.

1.2.2 Flatten Hierarchy CSA multiplier Synthesis

The second Synthesis was performed forcing the Design Compiler to flatten the hierarchy and to implement the Signifcands multiplier in Stage2 as a CSA multiplier. The command added was: $set\_implementation\ DW02\_mult/CSA[find\ cell\ I2/mult\_134]$. In that case the total Cell Area obtained was 4906.63 μm^2 and the critical Path Delay obtained is 3.89 ns therefore the maximum frequency achievable is 250 MHz.

1.2.3 Flatten Hierarchy PPARCH multiplier Synthesis

The third Synthesis was performed forcing the Design Compiler to flatten the hierarchy and to implement the Significands multiplier in Stage2 as a pparch multiplier. The command added was: $set\_implementation\ DW02\_mult/pparch[find\ cell\ I2/mult\_134]$. In that case the total Cell Area obtained was 4122.99 μm^2 and the critical Path Delay obtained is 1.40 ns therefore a maximum frequency achievable is 700 MHz. The results obtained are exactly the same as the first synthesis in which the Design Compiler was free to choose the implementation for the Significands multiplier.

1.3 Fine-grain Pipelining and optimization

In that phase one register was added at the output of the Significands multiplier and other required 1, 8 and 64-bit registers were added to host internal signals such that the timing of the whole Stage2 was correct. In order to confirm that a further simulation was performed and results compared with the previous one obtained. Then, the command *optimize registers* was issued after compile. This command allow the Design Compiler to retime the architecture. The reports generated were fg_timing.txt and fg_areatxt. The total Cell Area obtained was 5861.30 μm^2 and the critical Path Delay obtained is 0.87 ns therefore the maximum frequency achievable is 1.14 GHz. Then as a further step, the command *compile_ultra* was issued in place of *compile*. In that case too, he reports for timing(fg_ultra_timing.txt) and area(fg_ultra_area.txt) were generated. The total Cell Area obtained was 5347.92 μm^2 and the critical Path Delay obtained is 0.71 ns therefore the maximum frequency achievable is 1.4 GHz.

1.4 MBE multiplier for unsigned data

The aim of this laboratory activity is to realize a 32-bit 2-multiplicands unsigned multiplier. The general schema of the Multiplier is shown in Figure 1.1.

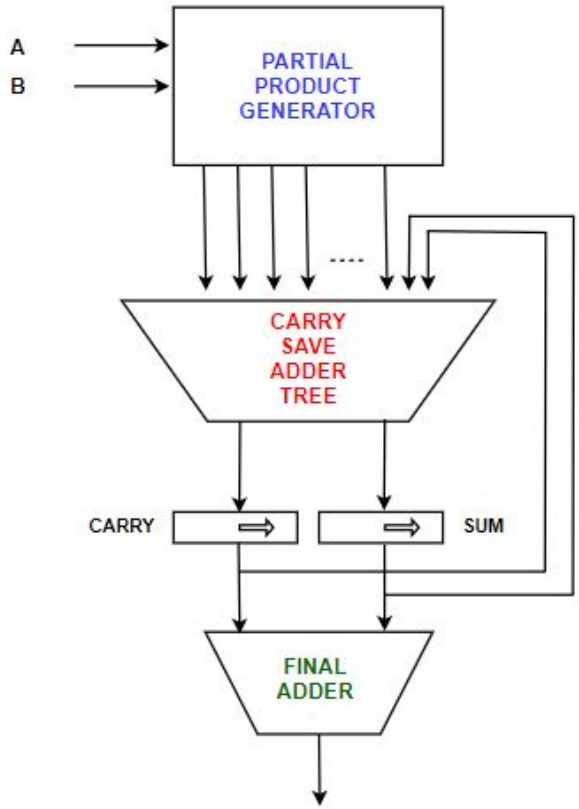

Figure 1.1: Multiplier

It is possible to observe the presence of three main blocks. From top to bottom in Figure are illustrated the Partial Product Generator, the Carry Save Adder(CSA) Tree and a Final Adder. The Partial Product Generator has been implemented following the MBE, i.e. Modified Booth Encoding, which requires a Radix-4 approach. In particular it produces half the partial product with respect to the Radix-2 solution. The multiplier has to be divided in 3-bit slices with one bit overlapped for two consecutive slices and considering the -1 bit as 0. Then each slice encodes the multiplicand according to the following expression represented in Figure 1.2:

$$q_j = \begin{cases} 0 & \text{if } \overline{(b_{2j} \oplus b_{2j-1})}\ \overline{(b_{2j+1} \oplus b_{2j})} \\ a & \text{if } b_{2j} \oplus b_{2j-1} \\ 2a & \text{if } \overline{(b_{2j} \oplus b_{2j-1})}\ (b_{2j+1} \oplus b_{2j}) \end{cases}$$

Figure 1.2: MBE partial product generation

The q_j value obtained is used to obtain the partial product $p_j = (b_{2j+1} \oplus q_j) + b_{2j+1}$. Then the Partial Product have been properly shifted. In addition to reduce the height (maximum number of

items to be added in any one column) can be reduced by one by combining the S term of the top partial product with the two leading ones of the same top partial product, which gives the final result, as shown in Figure 1.3 .

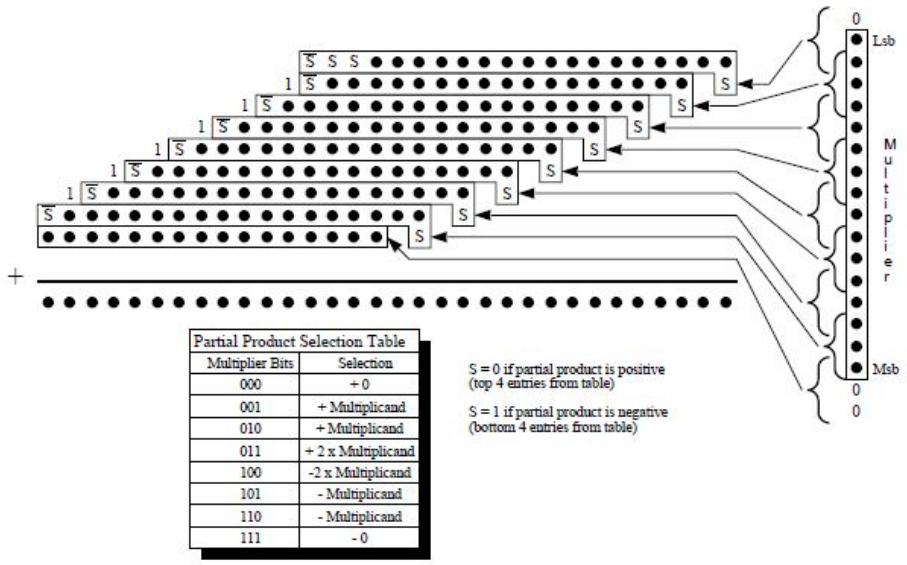

Figure 1.3: Complete 16 bit Booth 2 multiplication with height reduction

The Carry Save Adder Tree has been realized as a Dadda-tree. The Dadda reduction algorithm identifies the number of items that can be added, through the usage of Full Adder and Half Adder blocks. Dadda allocation is As-Late-As-Possible (ALAP), meaning that at each level j it is needed to allocate the minimum number of FAs and HAs such that the maximum number of operands at the next level (j - 1) is the one previously calculated. Each level has a specific maximum height calculated from the previous lower level height, exploiting the equation 1.1

$$d_{j+1} = \left\lfloor \frac{3}{2} d_j \right\rfloor \; ; j \mid d_j \geq \; n \;\&\; d_{j-1} < n \tag{1.1}$$

The Final Adder has to be a simple 2-operands (64 Bits) adder, so starting from the last level the heights obtained were: d_5=2, d_4=3, d_3=4, d_2=6, d_1=9, d_0=13.

Exploiting the dot representation the partial products were organized as in the Figure 1.4 .

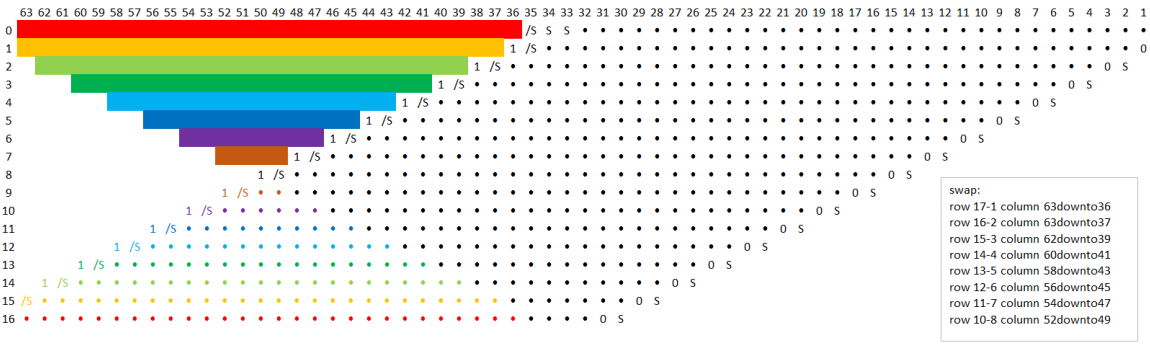

Figure 1.4: Complete 32 bit Booth 2 multiplication with height reduction

The Height Tree reduction follows the Figure 1.5 .

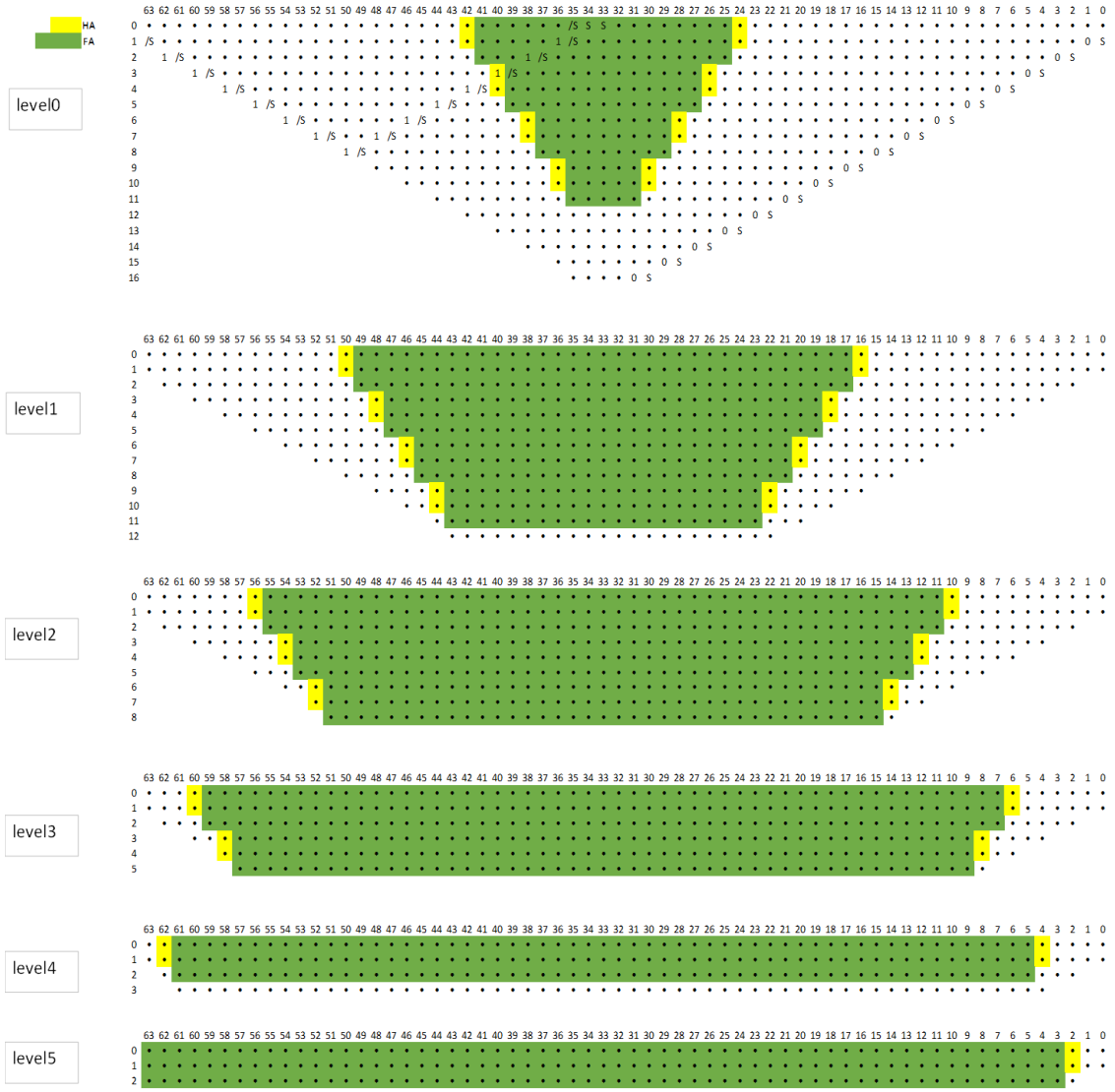

Figure 1.5: Complete 32 bit Booth 2 multiplication with height reduction

The complete architecture was written in VHDL code, dividing the project in the three main blocks and following the organization described. For what concern the Final Adder it has been realized as a Ripple Carry Adder. The architecture has been implemented in a not generic way and for shortness it is not reported here the code. The completed structure has been then tested, confirming the correct expected behavior.

1.4.1 Floating Point Pipelined optimized multiplier Simulation and Synthesis

The Multiplier previously realized has been inserted in place of the behavioural operator "$*$" in the Significands multiplier in Stage2 of the Floating point multiplier. Some simulations were performed and the same result of the behavioral version were obtained, as they are reported in the file $MBE\_prod\_reg.txt$. A final Synthesis has been performed and the total Cell Area obtained was 7678.88 μm^2 and the critical Path Delay obtained is 0.77 ns therefore a maximum frequency achievable is 1.29 GHz.

1.4.2 Comparisons and Conclusions

In the following table 1.1 are reported the results related to the synthesis of the same pipelined floating point 32-bit multiplier, for which the commands *compile_ultra* and *optimizeregisters* have been issued only for the MBE architecture and the Design Compiler chosen architecture.These architectures have been forced to be used in place of the behavioral " $*$ " of the Significands Multiplier in Stage2:

Architecture	Compile Ultra	Critical Path Delay	Cell Area
CSA	no	3.89 ns	4906.63 μm^2
PPARCH	no	1.40 ns	4122.99 μm^2
MBE	yes	0.77 ns	7678.88 μm^2
Design Compiler choice	yes	0.71 ns	5347.92 μm^2

Table 1.1: Reports

As it can be seen by the table, even if it respects the correct behavior of the multiplier, the MBE does not allow to achieve a smaller Critical Path Delay with respect to the Design Compiler Choice, and in addition it has an higher Area occupation. One of possible reason for that result is the usage of a not optimized Final Adder. Possible improvements of the architecture in terms of speed would be possible exploiting, for example of a faster adder as the CLA,.i.e Carry LookaHead Adder.