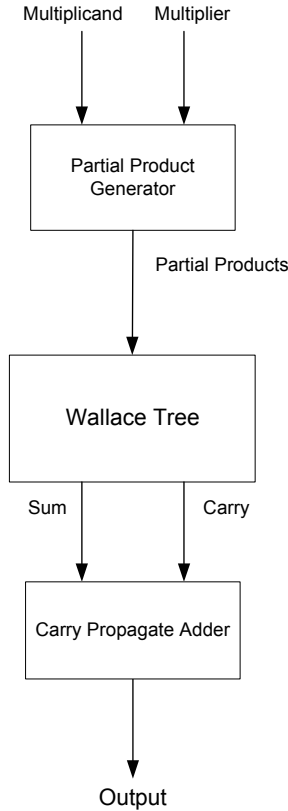


CSCE 611 - Digital Systems Design Using ASM

Assignment #6



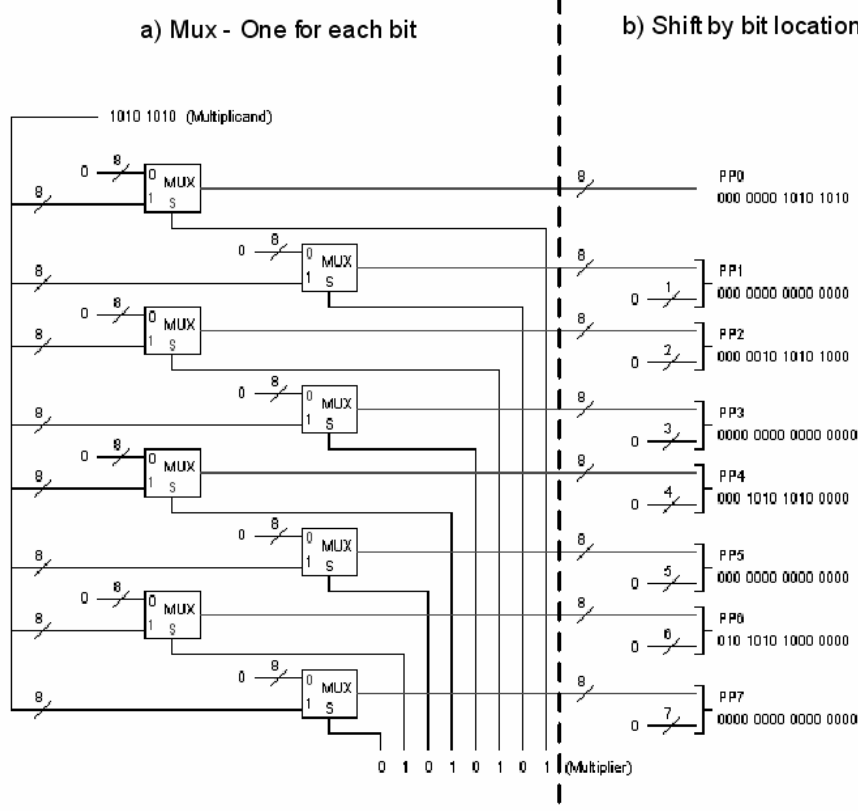
1. Description

The 8x8 Multiplier block is used to carry out unsigned integer arithmetic on two input operands, each of which is 8 bits in width. The Product produced is a 16-bit number. We decompose Multiplication into two parts: (1) generation of the partial products; and, (2) addition of the partial product terms to form the final product to be output. There are two architectures. Each of the architectures is to be modeled, simulated, synthesized and laid out on the Spartan-IIe FPGA. The results of the designs will be compared, looking at *effort distribution*, *design complexity* and *design size* (number of design elements, degree of reuse, etc.), and *design performance* (area, delay, number of cycles to complete the computation).

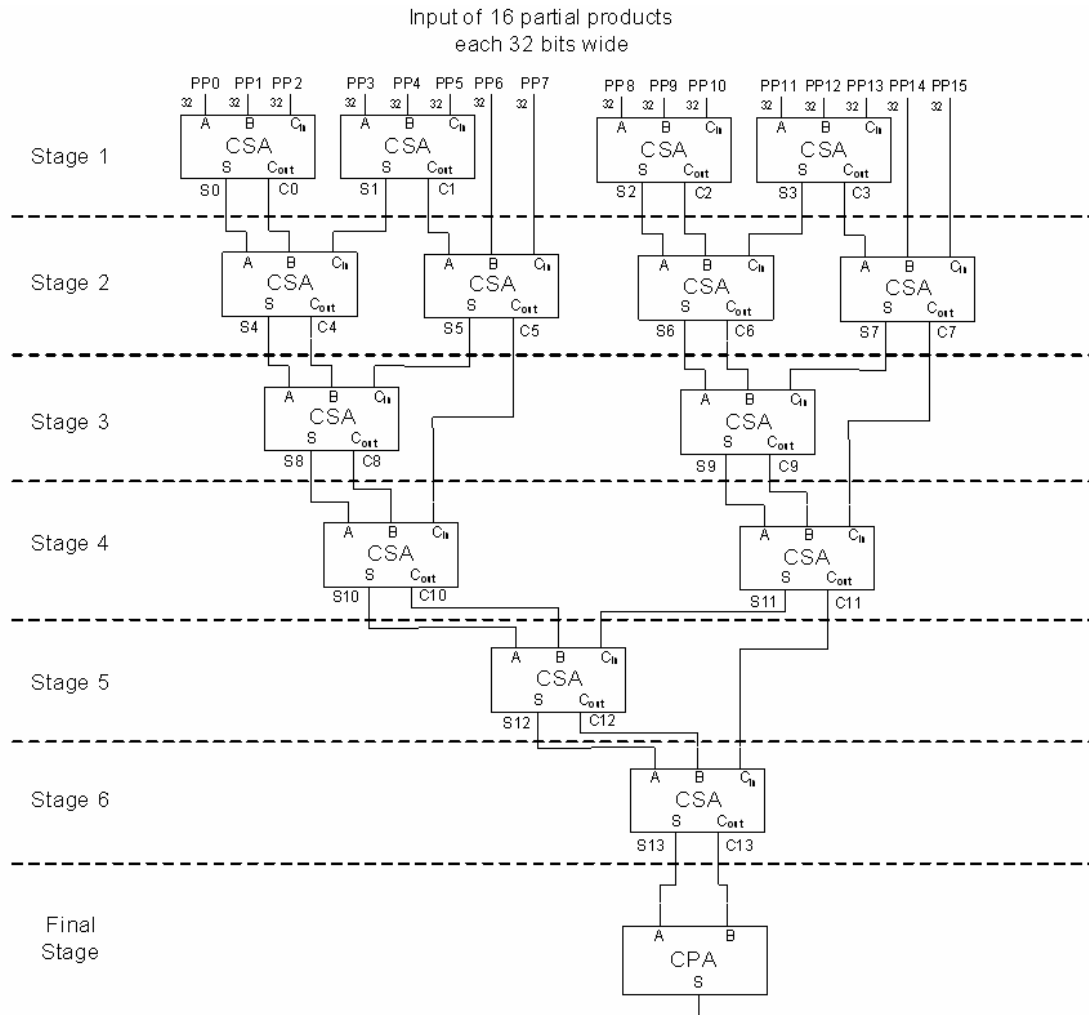
Create a design model in Nimbus to realize the functionality of the 8x8 unsigned integer multiplier (MUL_8). Complete all the deliverables required for assignments (Test Plan, Effort Distribution Worksheets, resultant design performance data, etc.).

2. Wallace MUL_16

We will create a regular architecture employing parallelism in the MULs and in ADDs of the MUL_8 unit. The architecture of this unit is shown in the following figure.



Adder delay for a Carry Save Adder (CSA) architecture is much less than other one. CSA has two-stage logic. It takes three operands, putting one on the Carry In and generates two outputs on the Sum and Carry Out. The (3, 2) reduction allows multi-operand addition to be done, which is faster than repeated 2-operand addition. Here, all of the partial products are added at the same time. See the topology in the figure.

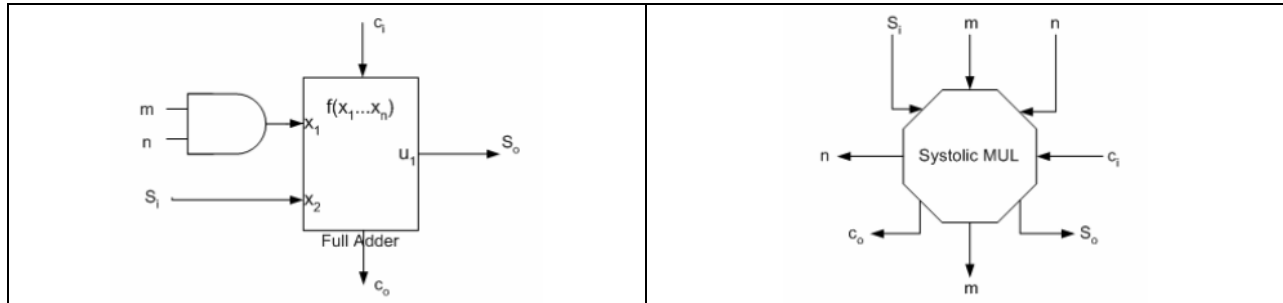


This Wallace Tree employs a set of Carry-Save Adder (CSA) units, which are basically regular Full-Adder (FA) units organized as shown in the topology. The carry-in and out lines are used to feed operands. The final Carry Propagate Adder (CPA) operates as we'd expect a two-input FA to operate. (Our 8-bit model will not have so many partial product terms, and therefore, will not have a tree of this depth).

The Full Adder unit of the CPA can be realized directly using the ASM ADD macro, but the CSA adder must be realized using more primitive macro-functions. This is because we are using the inputs and outputs differently than the purpose specified for the existing ADD macro. The MUX units in the previous figure that perform the calculation of the individual partial product terms can also be realized using copies of the NMUX macro-function.

2. Systolic MUL_8

In the following two figures, below, we have the basic element for a parallel MUL unit.



The basic unit employs a Full-Adder (FA) unit, as does the Wallace MUL architectures. The difference is that this architecture uses only the FA, which also has a gated AND input, and uses the Carry in and out signals in a novel way to achieve the MUL functionality. It is a variation of a theme: multiplication by shifting and adding.

The second figure shows the configuration of the individual cellular unit, which comprises how the cells are interconnected to realize the Multiplier. (See *Week 12 Notes* for more discussion on this architecture). The A operand bits feed into the cellular array rows, and the B operand bits feed into the columns. Each bit propagates into the design (a-bits feed from left to right, b-bits feed downward) and is available in all cells of its row or column. Carry signals propagate from right to left. The Sum of an individual addition from a FA stage feeds diagonally into the S input to the cell just below and to the right of the one above it—this provides one of the two operands for that next-stage FA unit.

3. **Model Verification:** You will need to devise a robust and reasonably complete test plan for this model, as encapsulated in your test harness. Please use the Test Planning Worksheet (you'll need to use 3-4 test cases to define your test plan). Define a single collection of test data that you can use for all the MUL architectures you will be examining. As you simulate each of your architectures, you will need to assess (and include in your report) how each model performs relative to this benchmark data set, in terms of latency, elapsed cycle time to complete the MUL operation, and whether each of your models generates consistent product results.
4. **Model Realization:** Once you have tested your models, you will need to synthesize them in Synopsys and run them through Xilinx to place & route for the Spartan-IIe device. At the end, you will have data for two different styles of MUL architectures.

Remember, this is a combinational logic circuit model, so there is no sensitivity to the clock. However, for the real circuit, we might want to insert staging registers in order to speed the computation up in both MUL_8 architectures. Creating a synchronous sequential control circuit is not required; however, extra points will be given if you decide to do it. But remember that the project will need to be completed on time.

Graduate students: you will also need to carry out these activities for a third model, namely, an 8-bit Serial-Parallel multiplier model created as an ASM model, as discussed in our text. It will have 4-6 states, have a counter loop, and a test whether to add the current partial product value, and then shift one of the operands, multiplying each bit in turn. You can find this in the text. You will have results for 3 different 8x8 MUL models as part of your submission for this Assignment.

Please do your own work. Please collect accurate Effort Distribution data. Pace yourselves. Enjoy.

References:

1. Lewin, D., and D. Protheroe, *Design of Logic Systems*, 2nd. Ed., London, Chapman and Hall Publishers, 1992.
2. Davis, J.P., H.A. Wake, C. LeeHaug, J. Branton, and N. Namilae, "Design versus Programming in Custom Computing Machine Applications: Experiences Using the Algorithmic State Machine Method", Katz, R. (ed.), *Proceedings MAPLD-2004 Military Applications of Programmable Logic Devices*, NASA Office of Logic Programming, 2004.
3. Hayes, J.P, *Computer Architecture and Organization*, 3rd. ed., New York, McGraw-Hill Publishers, Inc., 2002.
4. Carpinelli, J.D., *Computer Systems Organization and Architecture*, Reading, MA, Addison Wesley Longman, Inc., 2001.
5. Wolf, 2004.