SystemC-based SoC Accelerator Design: The Multiplier

COE838: Systems-on-Chip Design
LAB-2a

1. Objectives

The purpose of this lab is to develop a SystemC-based multiplier accelerator for a System-on-Chip (SoC) design. Using the skills obtained in the previous lab, students are to code the modules, signals, and drivers necessary for designing and testing an 8x8 array multiplier unit.

2. Array Multiplier Overview

Figure 1 provides a block diagram for a 4x4 Array Multiplier unit. This overall unit accepts the 4-bit inputs A and B (A0:A3, B0:B3), and outputs an 8-bit product (P0:P7). According to Figure 1, the array multiplier consists of two main blocks: 1) Carry Save Adder (CSA) blocks- represented by square shape blocks, and 2) Carry Propagate Adder (CPA) blocks- represented by the rounded square blocks, present in the last row. Note that the least significant bits (LSB) of the multiplier's product (P0:P3) are produced by the CSA matrix, whereas the most significant bits (MSB - P4:P7) are output by the CPA matrix. Note that the final carry out bit located in the last row and column of the multiplier is not connected. Therefore this array multiplier is of type unsigned.

Figure 2(a) and 2(b) outline the internal functionality of the CSA and CPA blocks respectively. As seen in Figure 2(b), a CPA block consists of a simple full-adder, inputting the signals A, B and C, and outputting the carry (Co) and sum (So) bits. Students are therefore required to design the full-adder with all necessary logic gates to output a carry and sum bit.

A CSA unit differs slightly in comparison to a CPA. The CSA employs an AND gate on the inputs A and B, producing the output AB. The AB bit, Sum in (S), and the Carry in (C) bits are then used as input to the full-adder, producing the Co and So output bits. Notice that the first and last row inputs of the array multiplier (Figure 1) differ in comparison to the inner-most rows. Similarly, the most significant bit of the multiplier's column also differs in input stimuli when compared to the rest of the matrix.

Note that the order in which the rows and columns are calculated is vital. In particular, if a row or column depends on a previous row or column's calculation, this precedence and order must be maintained. Using C++ cout statements and generating waveforms may help in debugging your design. Therefore be sure to pay close attention to the connection and implementation details provided in Figure 1.

3. What To Hand In

Using the description and diagrams provided for a 4x4 array multiplier unit, students are to develop an 8x8 array multiplier unit using SystemC. Ensure that many test-case scenarios (average case, corner cases etc) are included in your testbench (or. sc_main.cpp) to guarantee the correct functionality of your multiplier.

This lab is due before the start of your lab session in week 4. Please print out and hand in all the SystemC code necessary for implementing the array multiplier design. You should include screenshot(s) of the

waveforms generated for all test cases. This lab is to be handed in with the University cover page, dated and signed. Your lab instructor may also ask you to demo your work and answer any related questions at the time of submission.

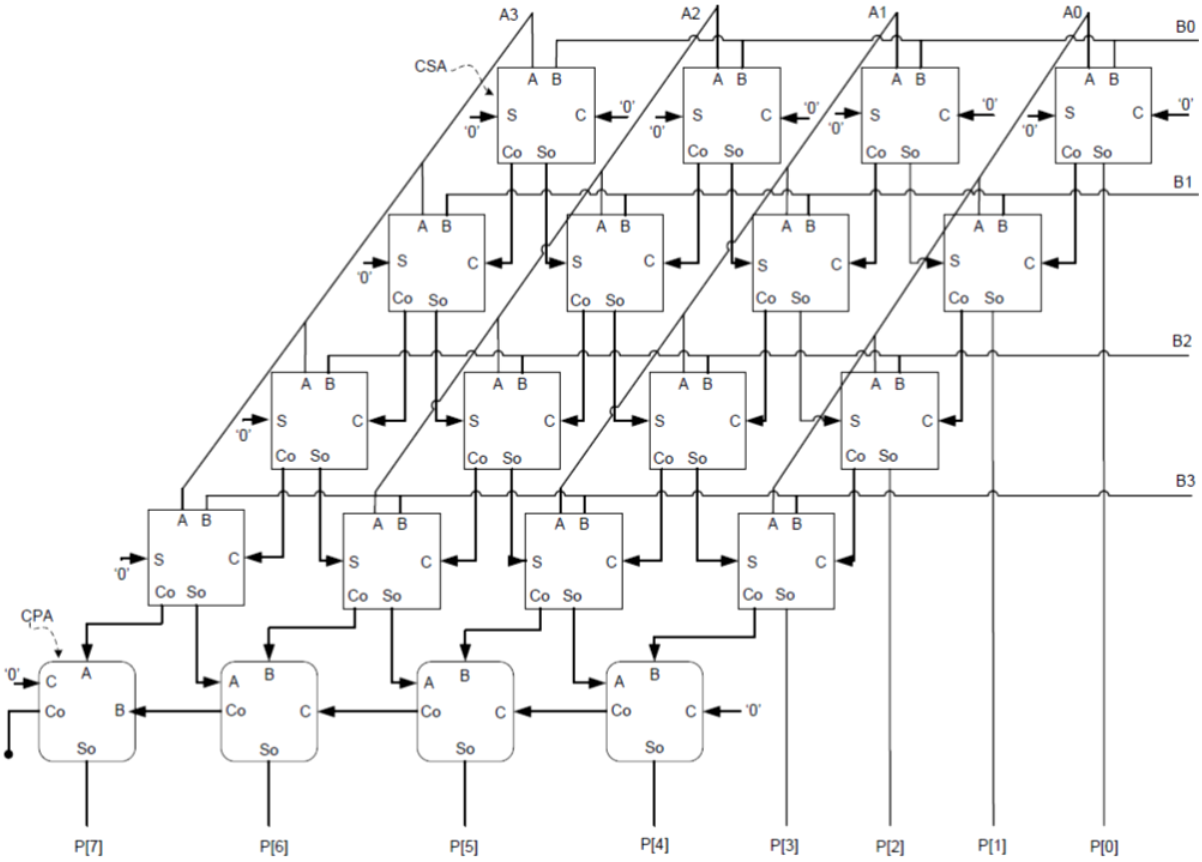

Figure 1: Array Multiplier Unit

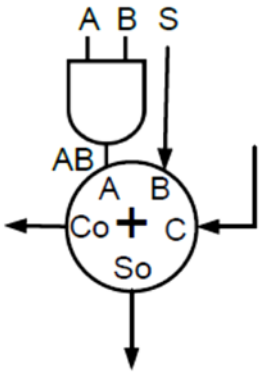

Figure 2(a): CSA Block

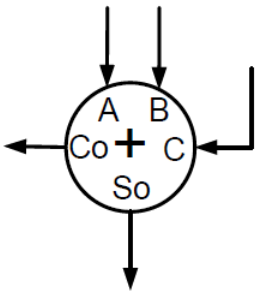

Figure 2(b): CPA Block