Laboratory Exercise 11

Implementing Algorithms in Hardware

This is an exercise in using algorithmic state machine charts to implement algorithms as hardware circuits.

Background

Algorithmic State Machine (ASM) charts are an alternative representation for finite state machines, which allow designers to express larger state machines and circuits in a manner similar to a flow chart. An example of an ASM chart is shown in Figure 1. It represents a circuit that counts the number of bits set to 1 in an n-bit input A ($A = a_{n-1}a_{n-2}..a_1a_0$).

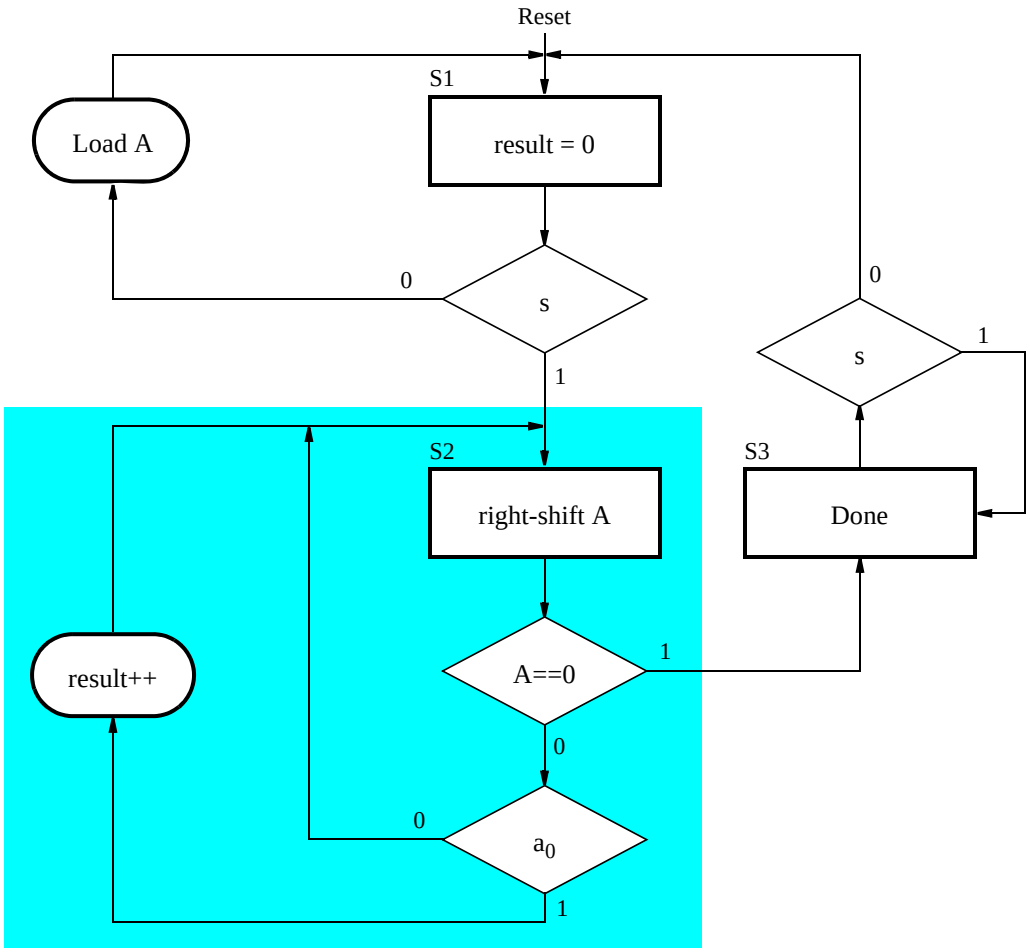

Figure 1: ASM chart for a bit counting circuit.

In this ASM chart, state S1 is the initial state where we load input into ~~shirt~~ register *A* and wait for the start (*s*) signal to begin operation. We then ~~counter~~ the number of 1's in *A* in state S2, and wait in state S3 when counting is completed.

The key distinction between ASM and flow charts is in what is known as *implied timing*. In contrast to a flow

chart, events that stem from a single state (rectangle) box in an ASM chart are considered to happen in the same clock cycle. Any synchronous elements, such as counters or registers, update their value when the next state is reached. Thus, the correct way to interpret the highlighted state in Figure 1 is as follows.

In state S2, the shift register *A* is enabled to shift contents at the next positive edge of the clock. Simultaneously, its present value is tested to check if it is equal to 0. If *A* is not 0, then we check if the least-significant bit of *A* (a_0) is 1. If it is, then the counter named *result* will be incremented at the next positive edge of the clock. If *A* is 0, then we proceed to state S3.

The implementation of the bit counting circuit consists of components controlled by an FSM that functions according to the ASM chart - we refer to these components as the *datapath*. The datapath components include a counter to store *result* and a shift register *A*.

In this exercise you will design and implement several circuits using ASM charts.

Part I

Implement the bit-counting circuit using the ASM chart shown in Figure 1 on a DE1 board. The inputs to your circuit should consist of an 8-bit input connected to slider switches SW_{7-0}, an asynchronous reset connected to KEY_0, and a start signal (s) connected to switch SW_8. Your circuit should display the number of 1s in the given 8-bit input value using red LEDs, and signal that the algorithm is finished by lighting up a green LED.

Part II

We wish to implement a binary search algorithm, which searches through an array to locate an 8-bit value *A* specified via switches SW_{7-0}. A block diagram for the circuit is shown in Figure 2.

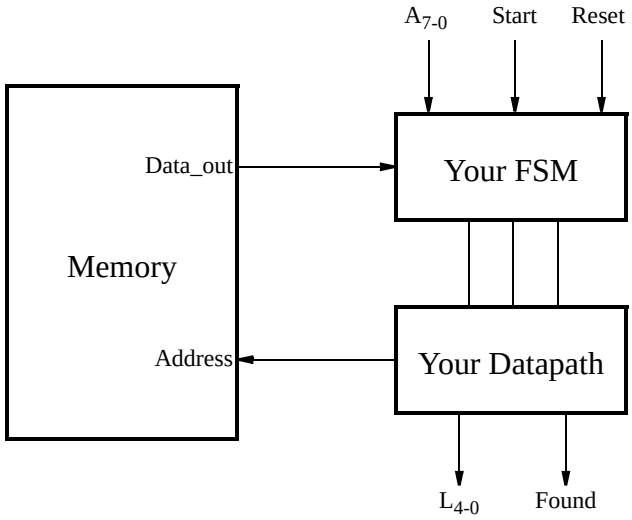

Figure 2: A block diagram for a circuit that performs a binary search.

The binary search algorithm works on a sorted array. Rather than comparing each value in the array to the one being sought, we first look at the middle element and compare the sought value to the middle element. If the middle element has a greater value, then we know that the element we seek must be in the first half of the array. Otherwise, the value we seek must be in the other half of the array. By applying this approach recursively, we can locate the sought element in only a few steps.

In this circuit, the array is stored in an on-chip memory instantiated using MegaWizard Plug-In Manager. To create the approriate memory block, use the the RAM: 1-PORT module from the MegaWizard Plug-In Manager as shown in Figure 3.

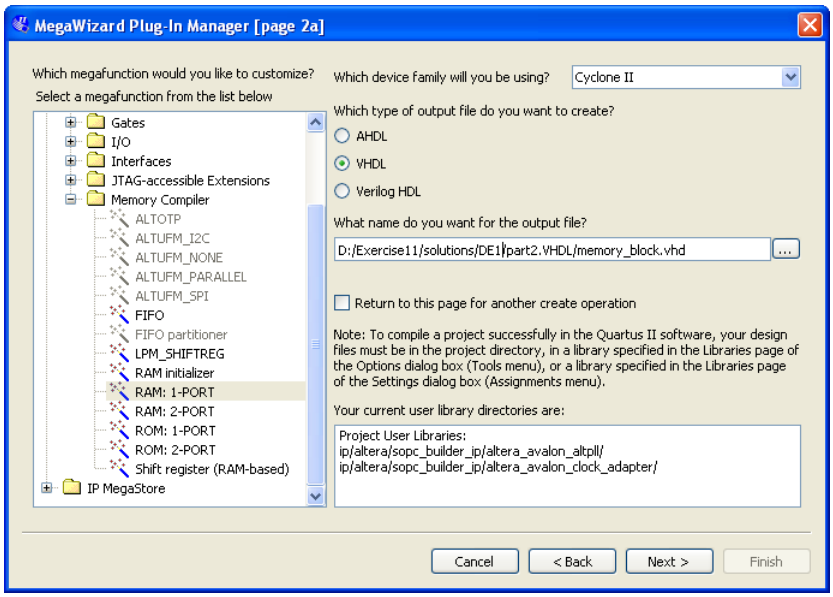

Figure 3: Single-port memory selection using MegaWizard Plug-In Manager.

In the window in Figure 3, specify the VHDL output file to be memory_block.vhd. When creating the memory block, you should also specify a memory initilization file to be my_array.mif, so that the memory contents can be set to contain an ordered array of numbers.

The circuit should produce a 5-bit value L that specifies the address in the memory where the number A is located. In addition, a signal *Found* should be set high to indicate that the number A was found in the memory, and set low otherwise.

Perform the following steps:

1. Create an ASM chart for the binary search algorithm. Keep in mind that it takes two clock cycles for the data to be read from memory. You may assume that the array has a fixed size of 32 elements.

2. Implement the FSM and the datapath for your circuit.

3. Connect your FSM and datapath to the memory block as shown in Figure 2.

4. Include in your project the necessary pin assignments to implement your circuit on the DE1 board. Use switch SW_8 to drive the processor's *Run* input, use SW_7 to SW_0 to specify the value to be searched, use KEY_0 for *Resetn*, and use the board's 50 MHz clock signal as the *Clock* input. Connect $LEDR_4$ to $LEDR_0$ to show the address in memory of the number A, and $LEDG_0$ for the *Found* signal.

5. Create a file called my_array.mif and fill it with an ordered set of 32 eight-bit integer numbers. You can do this in Quartus II by choosing File > New... from the main menu and selecting Memory Initialization File. This will open a memory file editor, where the contents of the memory may be specified. After this file is created and/or modified, your design needs to be fully recompiled, and downloaded onto the DE1 board for the changes to take effect.

Preparation

The recommended preparation for this exercise is to write VHDL code for Parts I and II.

Copyright ©2011 Altera Corporation.