

**University of California at Berkeley**  
**College of Engineering**  
**Department of Electrical Engineering and Computer Sciences**

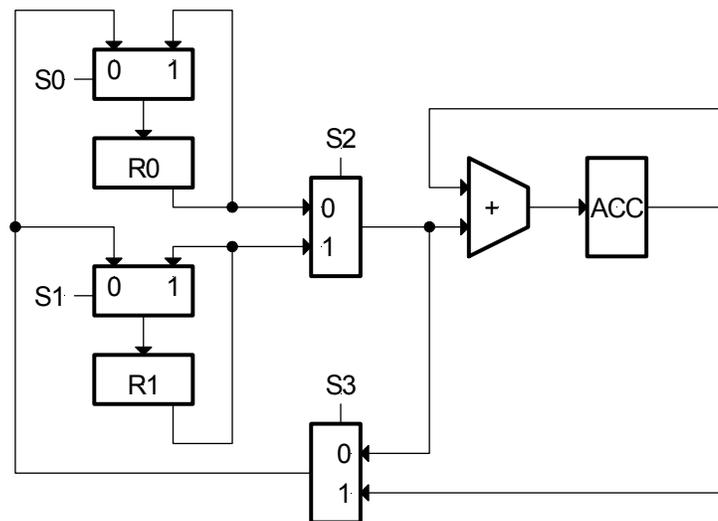
EECS150  
 Fall 2002

J. Wawrzynek

Homework #9

This homework is due on **Friday Nov 15<sup>th</sup> by 2pm**. Homework will be accepted in the EECS150 box outside 125 Cory Hall. Late homework will be penalized by 50%. No late homework will be accepted after the solution is posted.

1. From Mano: problems 5-12.
2. Convert the state transition diagram of figure P5-19 (in the problem section of Mano, chapter 5) from a Mealy machine (as shown) to a Moore machine. a) Draw your state transition diagram for the Moore machine. Remember a Moore machine may need additional states. b) Write the verilog description for both the Mealy version and the Moore version of the FSM.
3. The circuit shown below includes three registers (labeled R0, R1, and ACC) implemented with positive-edge triggered flip-flops, an n-bit adder, and 2-to-1 multiplexors.



The table below has one row for each clock cycle of interest. The table is filled in with the values of S0, S1, S2, S3, as applied to the circuit by a controller. These values are applied after the rising edge of the clock (so that they can be ready for

the *next* rising edge of the clock). Write the register transfer level (RTL) description for each cycle:

Cycle #	S0	S1	S2	S3
1	1	0	0	0
2	-	-	-	-
10	1	1	1	1
11	1	1	0	0
12	1	0	0	1
13	0	1	1	1
14	-	-	-	-

*These problems are based on the “List Processor Design Example” presented in class.*

4. Draw the state transition diagram for the FSM controller for architecture #4. Exactly how many clock cycles are needed to process a list with n nodes?
5. For the timing analysis, we assumed that the data-path registers did not have a reset input. Now, assume that the registers do have resets, but otherwise have the same delay characteristics. What would be the new maximum clock frequency for the optimized design of architecture #4?
6. Assume that we are allowed to restrict nodes to be aligned on multiples of 2 bytes. Redraw the new simplified data-path of architecture #4 to take advantage of this restriction. Rewrite the RTL description of the algorithm. Draw the associated FSM controller state transition diagram.
7. Now assume that we can restrict nodes to be aligned on multiples of 2 bytes *and* we can use a memory with a 16-bit wide output. Again redraw the new simplified data-path of architecture #4. Rewrite the RTL description of the algorithm. Draw the associated FSM controller state transition diagram.
8. Consider the design of a new list processor similar to the one presented in class. This circuit iteratively forms the sum of all the 8-bit integers stored in a linked-list structure starting at memory address 0. The integers are stored in 2 byte nodes (the first byte is a pointer to the next node and the second is the integer). Nodes are not restricted to be aligned on even multiples of 2 bytes.

Unlike the processor presented in class, this processor, in addition to forming the sum, stores to memory into each node the sum of all the nodes up to that point. Therefore, after completion, node  $i$  will contain the sum of the values of all the nodes from the first up to and including the  $i$ th. For example, the original list  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$  will be transformed into  $1 \rightarrow 3 \rightarrow 6 \rightarrow 10$ .

Your circuit includes a memory with an 8-bit data input/output port, a single adder, multiplexors, and registers of your choice.

- a) Draw a resource utilization chart for this list processor. Fill in the execution schedule showing at least three loop iterations. Attempt to create a schedule in such a way as to minimize 1) the total number of cycles needed to process a list, and 2) the number of cycles in the control loop. Use subscripts to distinguish different loop iterations.
- b) Based on your resource initialization chart, write the RTL description of your solution.