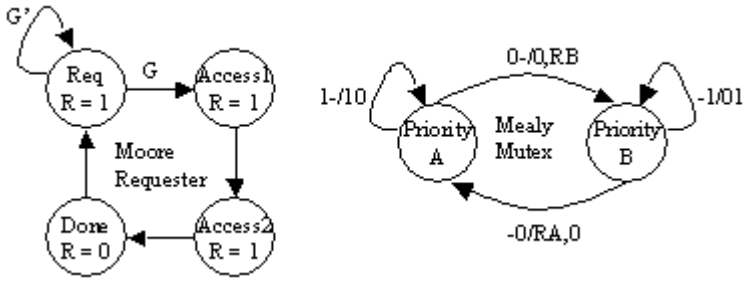
CSE370 Autumn '99
Assignment 8 draft
Distributed 11/23/99
Due 12/8/99

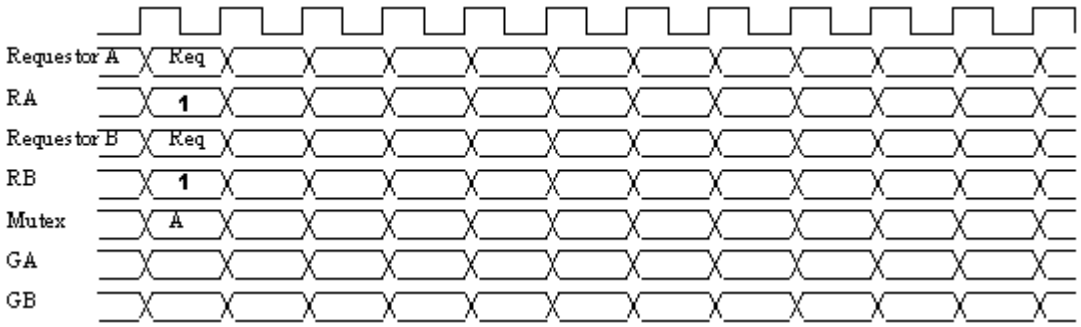
Reading: Katz 8 except 8.3, 9 except 9.3.4, 10.2, and 11 for reference.

Problem 1.
Moore v Mealy, Communicating State Machines:



Part A. Complete the design of the Mealy Mutex state machine above using D-flip-flops. If Tsu = 2ns and Th = 1ns for the D-ff, what are the setup and hold times for RA and RB?

Part B. Without considering detailed timing issues, draw the timing diagram for a system consisting of two instances of the Moore Requester and one Mealy Mutex. Your timing diagram should show all of the signal values, and the states of all three machines. The initial states are shown. What is the utilization of the hypothetical resource (Access Cycles/Total Cycles).?



Problem 2.
State Machine Analysis: Katz 8.7

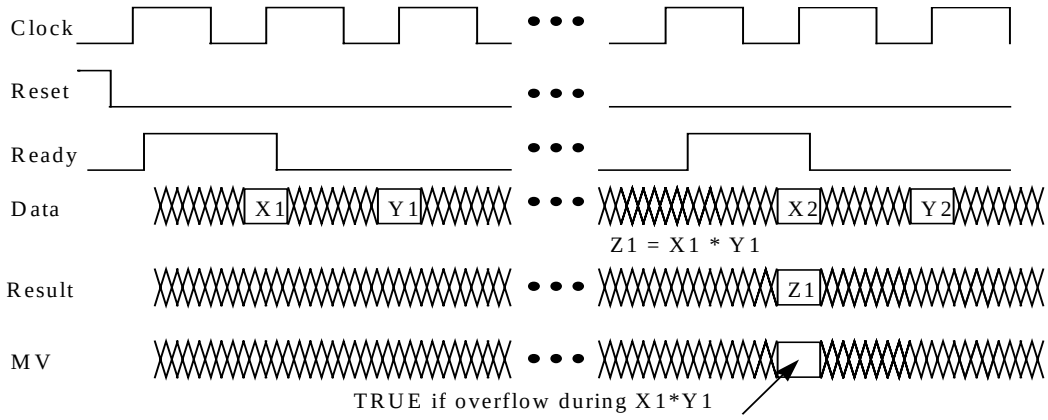
Problem 3.
State Minimization: Katz 9.5

Problem 4: Datapath and Control for 8-Bit Multiplier
The following is a specification for a sequential 8-bit multiply unit. The computation cycle begins when your system asserts the "Ready" output indicating that the previous computation is complete and that new operands will be accepted on the next two clock edges. The system should be self-starting so that no reset signal is needed. It is okay if the system requires a few clock cycles to reach the initial ready state.

The inputs to the system are:
Clock;
Data[7:0];

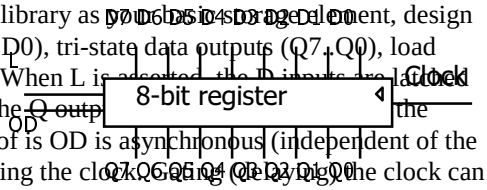
The outputs are:
Result[7;0] ; Is valid when Ready is asserted
MV ; Asserted, with Ready, to indicate that multiplication overflow has occurred
Ready ; Indicates that new result is available, and that new data will be read for next multiply.

The interface to the multiplier is defined by the following timing diagram:



Note that the above timing diagram shows both inputs and outputs of the system. The system will always read the operands for the multiply from the input Data on the two rising edges immediately following the assertion of the Ready output. The outcome of the previous operation is valid on the Result outputs at the first rising edge after assertion of the Ready signal. MV is asserted along with Ready if an overflow has occurred during the computation, in which case Result is invalid.

Part A. Register Design
Using the "D Flip Flop wo /RSQ/" device from DW primlog library as your basic storage element, design an 8-bit register with the following features: data inputs (D7..D0), tri-state data outputs (Q7..Q0), load enable (L), output disable (OD), and of course a clock input. When L is asserted, the D inputs are latched into the register on the rising edge, when the OD is asserted the Q outputs are disabled, and the currently stored data is asserted on the Q outputs. The effect of is OD is asynchronous (independent of the clock). Your load enable should be implemented without gating the clock. Gating the clock can lead to timing problems. Turn in your schematic and design explanation.



Part B. Datapath Design
Draw a block diagram of the datapath required to implement a sequential 8-bit "shift-and-add" multiply algorithm using the ALU from assignment 6 as one of the major components* and the registers from Part A. Because of the tri-state outputs, you should not need any multiplexors to implement your datapath. Name your registers, list all the control points for the datapath, and describe their function. Your control points should include such items as opcode for the ALU, and load and output controls for the registers.
The ALU must be modified so that SHR sets the C condition code if A[0] = 1. You can download my modified ALU from the website under Assignment 8, or you can modify your own. The opcodes for my ALU can be found in the A6 solutions (see the spreadsheet).

Part C. State Diagram

Draw a state diagram for the controller, providing the following information for each state: A symbolic name, a pseudo-Verilog description of the state indicating control point assignments and the next state function.

Part D. Complete the Design

Implement the state machine by following these steps:

- Choose a state assignment (encoding),
- Design the output and next state logic functions.
- Enter the design into design works using DW gates or Verilog
- Verification and turn-in: TBD