

EECS150: Components and Design Techniques for Digital Systems

University of California
Dept. of Electrical Engineering and Computer Sciences

David E. Culler

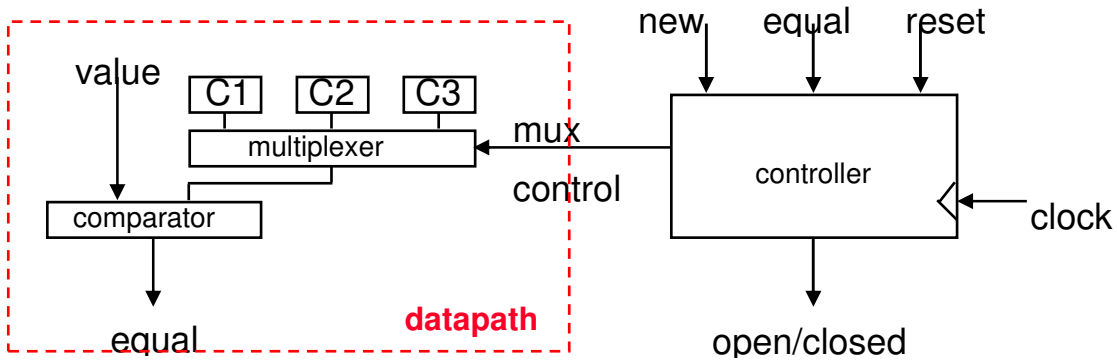
Fall 2004

Homework 4: Due Friday 10/1 @ 2 pm in box near 125 Cory

Problem 1: (YAM – yet another mux) Show that any n -input Boolean function can be implemented with an $2^{(n-1)}$ -input MUX and a single inverter. (You do not have the complemented inputs available.)

Problem 2. Explain in a couple of sentences the difference between a PLA and an FPGA. How might these differences in target technology change your design goals?

Problem 3. Consider the combo-lock design problem discussed in class and the book. (And a variant in lab. Phew!) You are given the following decomposition into datapath and controller. One difference from what we discussed in class, is that the mux control is pre-established. It is a two bit input to the mux, 01 selects C1, 10 selects C2, and 11 selects C3.

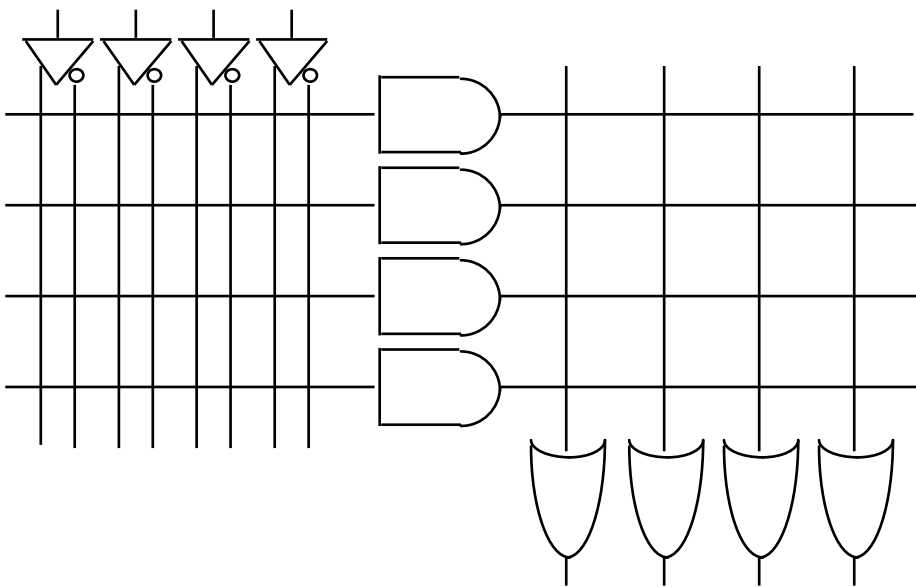


a. Starting from the state diagram on slide 7.42 and symbolic table shown below (which fixes one row), select concrete encodings for the state and output. Produce the concrete truth table defining the controller.

reset	new	equal	state	nextstate	mux	open/closed
1	-	-	-	S1	C1	closed
0	0	-	S1	S1	C1	closed
0	1	0	S1	ERR	-	closed
0	1	1	S1	S2	C2	closed
0	0	-	S2	S2	C2	closed
0	1	0	S2	ERR	-	closed
0	1	1	S2	S3	C3	closed

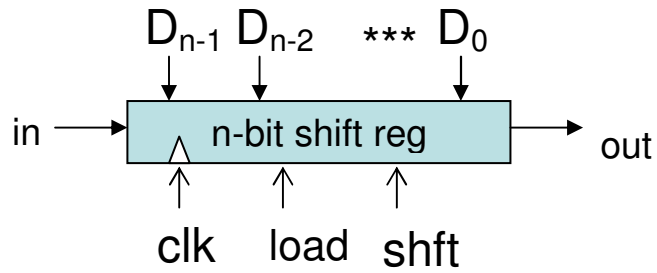
0	0	-	S3	S3	C3	closed
0	1	0	S3	ERR	-	closed
0	1	1	S3	OPEN	-	closed
0	-	-	OPEN	OPEN	-	open
0	-	-	ERR	ERR	-	closed

b. Implement this Moore FSM using D-flipflops for the state. Generate minimal SoP logic equations for the combinational logic. Implement this on PLAs. You may put marks on the utilized connects on a PLA template like the one below to show your answer. Do you end up with any shared terms?



c. Implement the datapath, the controller, and the composition of the two in verilog. Behavioral is fine for the two sub-modules. The verilog should be a clear specification of the design. It should have the state encodings and behavior. The C_i values can be fixed values.

Problem 4. This time you get to implement your FSM again, but using different components for the controller. Instead of starting with a bunch of flipflops, the state will be represented using a shift register. Its block diagram is shown below. As in class, when `load` is asserted the vector of data inputs is loaded into the register. When `SHFT` is asserted, the register shifts to the right, with input `in` is brought into the most significant bit. When neither are asserted, the register holds its n-bit value. The value of the least significant bit is available as `out`.



- a. Produce a new concrete state table with the encodings selected so that it will be really easy to implement on this device.
- b. Generate whatever logic is needed for your controller.

Problem 5. Implement the BYTE-bit adapter with rate matching described in slide 8.8. You may use any technique you like. Generate the symbolic and concrete tables. Show your implementation in verilog or PLAs or gates.

Problem 6. Katz problem 7.18

Problem 7. Katz problem 7.26