ECE 5720 Modeling and Synthesis of Digital Systems Using Verilog

Worcester Polytechnic Institute

B-Term 2022

Homework Report 1: Simple RTL

Submitted by

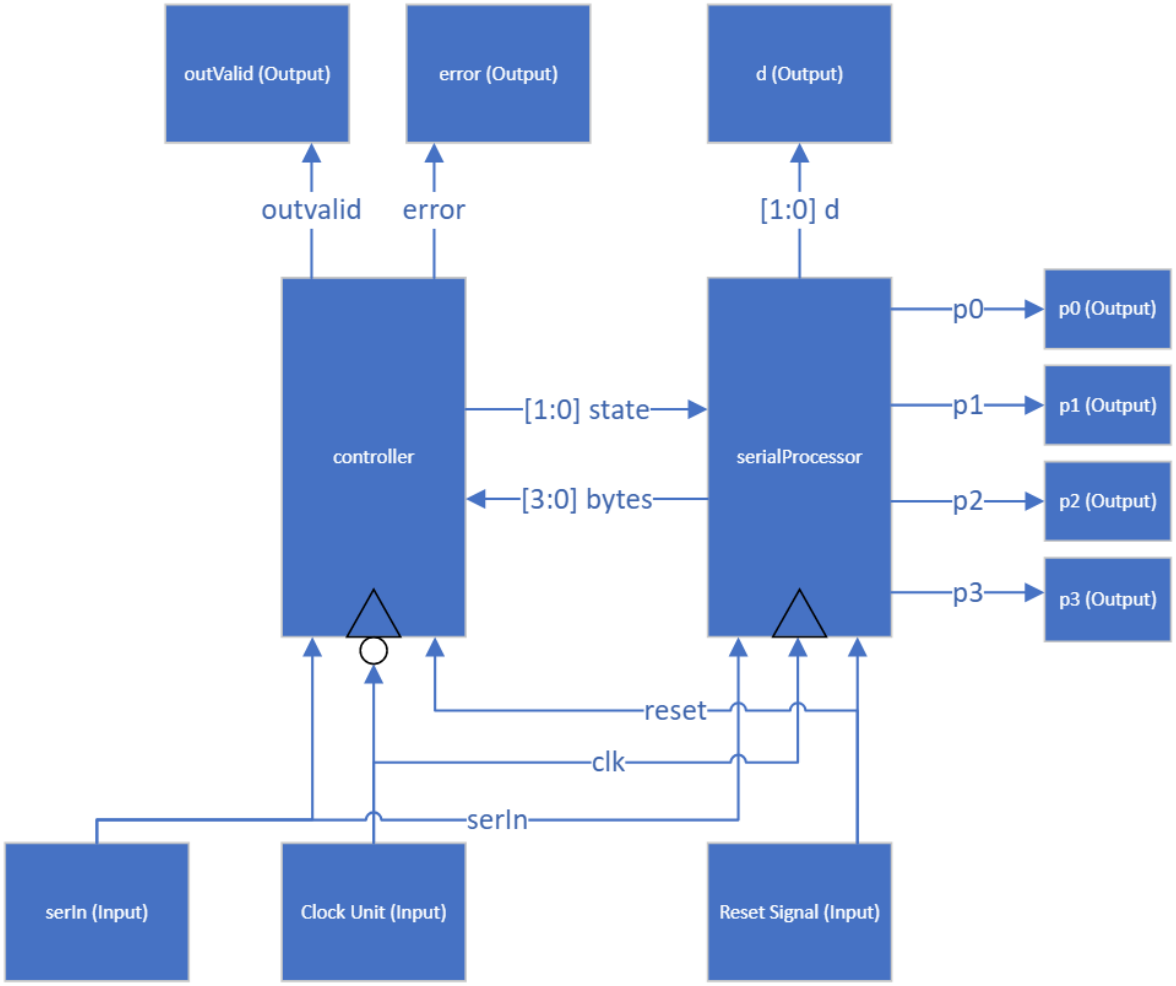
_____

Drew Solomon

_____

Professor Navabi

Datapath

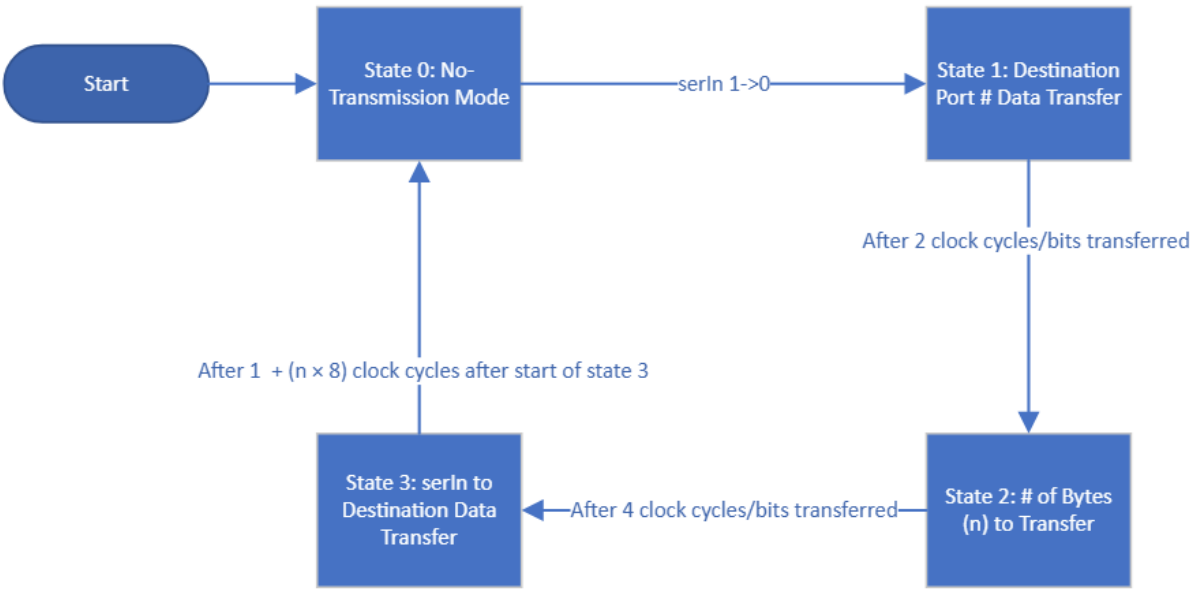


In order to achieve the MSSD functionality described in the project requirements, the project was organized into the datapath seen above. The only two modules present in the MSSD are the controller and a module named serialProcessor, which handles the incoming stream of serIn data differently depending on the current state of the system and the configuration data it receives before port data routing begins. serIn, clk, and reset are the inputs to the module, and route to both the controller and serialProcessor so they stay synchronized.

The controller and serialProcessor exchange data in the process of receiving a serIn data transmission, with the controller signaling changes in state to serialProcessor, and serialProcessor sending the number of bytes to be transmitted to the controller (to stay synchronized properly). serialProcessor's outputs all relate to the output ports 0-3 that the MSSD can redirect incoming serIn data to. d is a 2-bit binary output describing the currently activated port (the decimal value of the number corresponding to the port), and p0-p3 represent

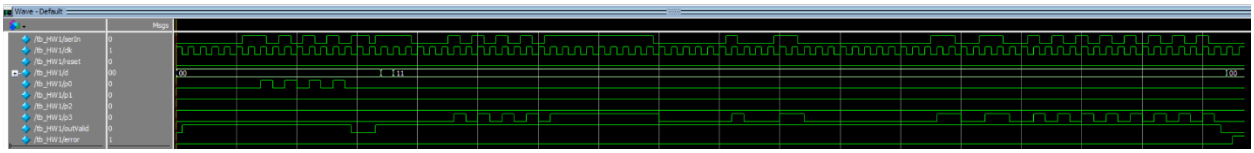
the output currently seen at each of the 4 ports. Only one of them should ever output high at a time, since their default state is 0 and only one can be activated a time. The controller outputs two flags as well: outvalid and error. outValid states if the output currently seen at the activated port is valid and can be trusted, and error states if there has been a transmission input error (will be triggered at the end of a serIn transmission if serIn does not return to 1 after the last bit is received).

Controller States



The controller consists of 4 total states, which change under the correct conditions on negative clock edges. The system starts in state 0, where the controller waits until serIn changes from 1 to 0. Once this occurs, state 1 is entered, and the next 2 positive edge clock ticks, serialProcessor will receive a bit from serIn that describes the port number to direct the data transmittion to. State 2 is then entered, where a similar operation as state 1 occurs. The next 4 positive edge ticks each have serialProcessor receive a bit from serIn describing how many bytes of data are about to be sent, which is used to determine how many clock ticks to receive data for before returning to state 0. State 3 is then entered, where the actual data transmission occurs. serIn is redirected to the active port for as many cycles as described in state 3. The system then returns to state 0 and awaits serIn to turn to 0 again, signaling a new data transmission is coming. If serIn does not return to the standby state of a constant 1 at this time, the error flag will go high.

Test Bench Results



This test bench consists of two data transmissions:

Trial	Port	Bytes	Data
1	0	1	10101010
2	3	8	10101010
			11111111
			10000001
			00011000
			00000000
			11001100
			10101010
			10101010

The first trial has serIn transmit 1 byte to port 0, and the second has serIn transmit 8 bytes to port 3. The routing of serIn to the ports can clearly be seen, with a mere half a clock tick delay. Having the controller and serialProcessor run on opposite clock edges helps prevent errors from input changes and data exchanges between the modules occurring on clock edges.

The d output signal follows the currently activated port, keeping as 0 until trial 2 when it switches to 2'b11, or port 3 in decimal. After trial 2, it returns to port 0, the default.

The outvalid signal goes high whenever a data transmission begins, and doesn't return to 0 until the transmittion is finished, as seen at the beginning and end of the waveform, and the middle between trial 1 and 2.

In the second trial, serIn was intentionally not returned to 1 after the data transmission finished so as to test the functionality of the error flag.