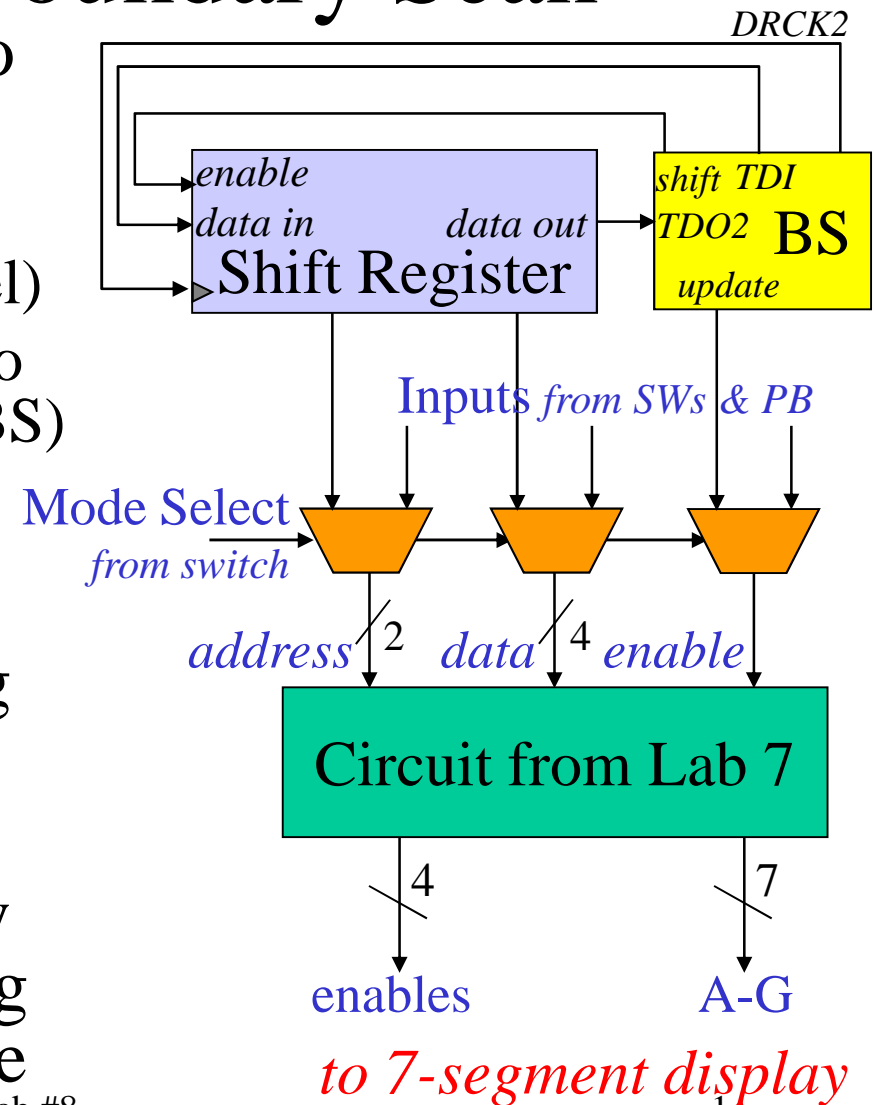


Hierarchical Modeling & Synthesis with access via Boundary Scan

- Write a new top-level VHDL to combine your models for
 - Hierarchical model from Lab 7
 - Multiplexers (need to write model)
 - 6-bit shift register that connects to internal BSCAN_SPARTAN3 (BS) port
- Use your universal counter/shift register model from Lab 5
- Simulate and verify (debugging where needed) design using ModelSim
- Synthesize, download, & verify design in Spartan 3 FPGA using Impact Boundary Scan interface



Boundary Scan Interface

- The FPGA has two internal Boundary Scan ports to access the programmable logic and routing resources
 - Boundary Scan is a serial interface and will require a serial to parallel conversion to communicate with the register file and display controller
 - Use a mode switch to select between the switch & push button interface from Lab 7 and the new Boundary Scan interface circuitry
- Specifications for design:
 - A bank of seven 2-to-1 MUXs controlled by a common select input
 - Select from remaining switches on Spartan 3 PCB
 - For 6-bit serial-to-parallel shift register (Lab#5) use
 - Active high enable (connects to SHIFT signal from BSCAN module)
 - Serial data in (connects to TDI signal from BSCAN module)
 - Serial data out (connects to TDO1 or TDO2 signal to BSCAN module)
 - Parallel data out (connects to MUXs)

Boundary Scan Interface

- Pre-lab assignment:
 - Write VHDL models for
 - Multiplexer bank
 - Top level model that connects
 - ✓ the Lab 6 circuit with MUXs
 - ✓ shift register with instantiation and connection to BSCAN_SPARTAN3 module (referred to as BSCAN module here) and to Multiplexers
 - Determine the FPGA pin number for mode select pin
 - Determine FPGA pin numbers to connect BSCAN module signals to LEDs
 - you can then watch BS in action
 - ✓ DRCK2, SEL2, Shift, Update, TDI, TDO2 *or*
 - ✓ DRCK1, SEL1, Shift, Update, TDI, TDO1
 - » depending on which BS port you decide to connect to
 - Read
 - “Spartan 3 Configuration” (pages 169-183) from class web page
 - Gefu’s tutorial on “Communicating with BS” on class web page

Instantiating BSCAN module

- Include the following library and package to access BSCAN module

```
library UNISIM;  
use UNISIM.vcomponents.all;
```
- Include the following instantiation of BSCAN module
 - Note: you do not declare the component
 - Note: you do not need to include unused connections
 - Note: you should bring BS signals to LEDs to watch the operation of the BS interface in conjunction with movement through the TAP state diagram
 - Choose your connections to either BS User Port #2 (in red), or User Port #1 (in blue) plus the common connections (in green)

```
BSCAN_SPARTAN3_inst : BSCAN_SPARTAN3
```

```
port map (
```

```
  CAPTURE => CAPTURE, -- CAPTURE output from TAP controller (not used in your design)  
  DRCK1 => DRCK1,    -- Data register output for USER1 functions (clock for shift register – bring to LED)  
  DRCK2 => DRCK2,    -- Data register output for USER2 functions (clock for shift register – bring to LED)  
  RESET => RESET,    -- Reset output from TAP controller (not used in your design)  
  SEL1 => SEL1,      -- USER1 active output (not used in your design but bring it out to an LED)  
  SEL2 => SEL2,      -- USER2 active output (not used in your design but bring it out to an LED)  
  SHIFT => SHIFT,    -- SHIFT output from TAP controller (enable for shift register – bring it out to an LED)  
  TDI => TDI,        -- TDI output from TAP controller (input data to shift register – bring it out to an LED)  
  UPDATE => UPDATE, -- UPDATE output from TAP controller (write enable to Lab 6 circuit)  
  TDO1 => TDO1,      -- Data input for USER1 function (output data from shift register – bring it to an LED)  
  TDO2 => TDO2       -- Data input for USER2 function (output data from shift register – bring it to an LED)
```

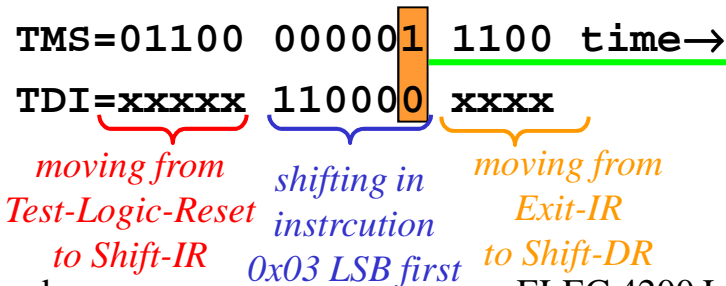
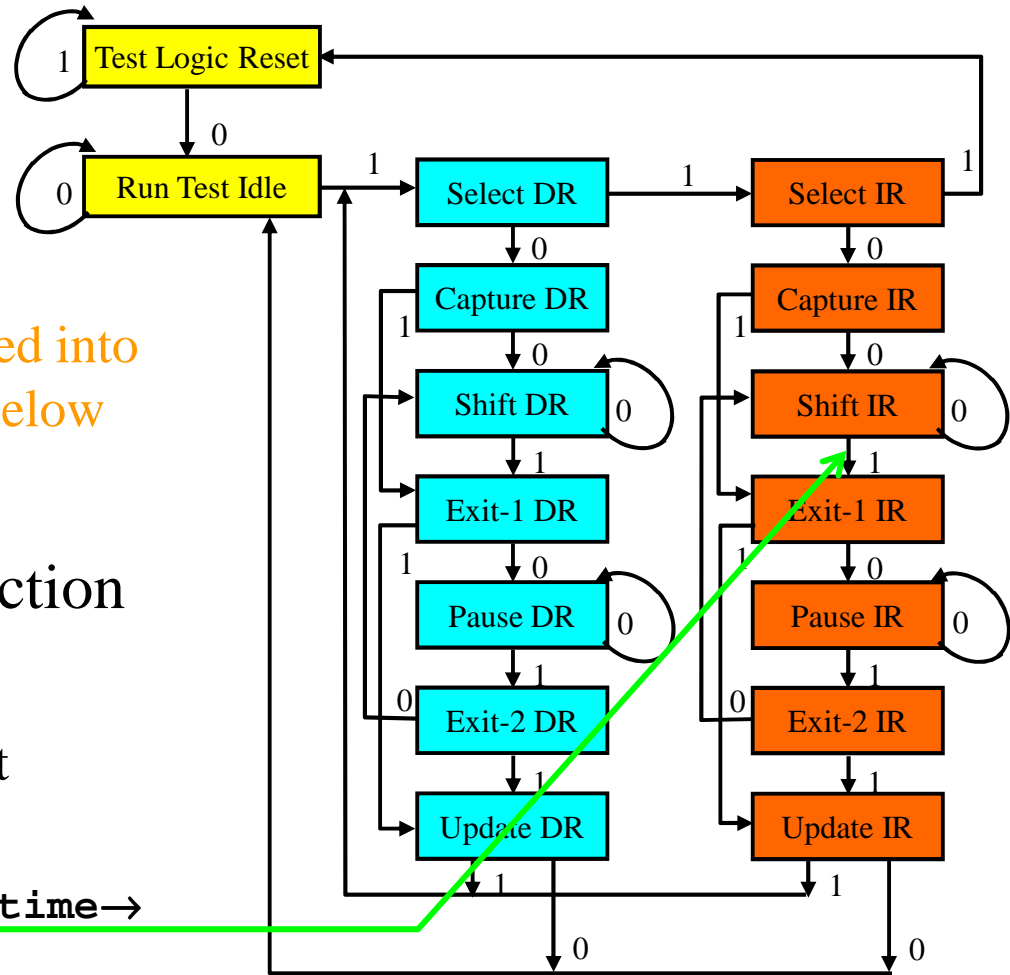
```
); C. E. Stroud
```

ELEC 4200 Lab #8

4

Instructions for BSCAN Access

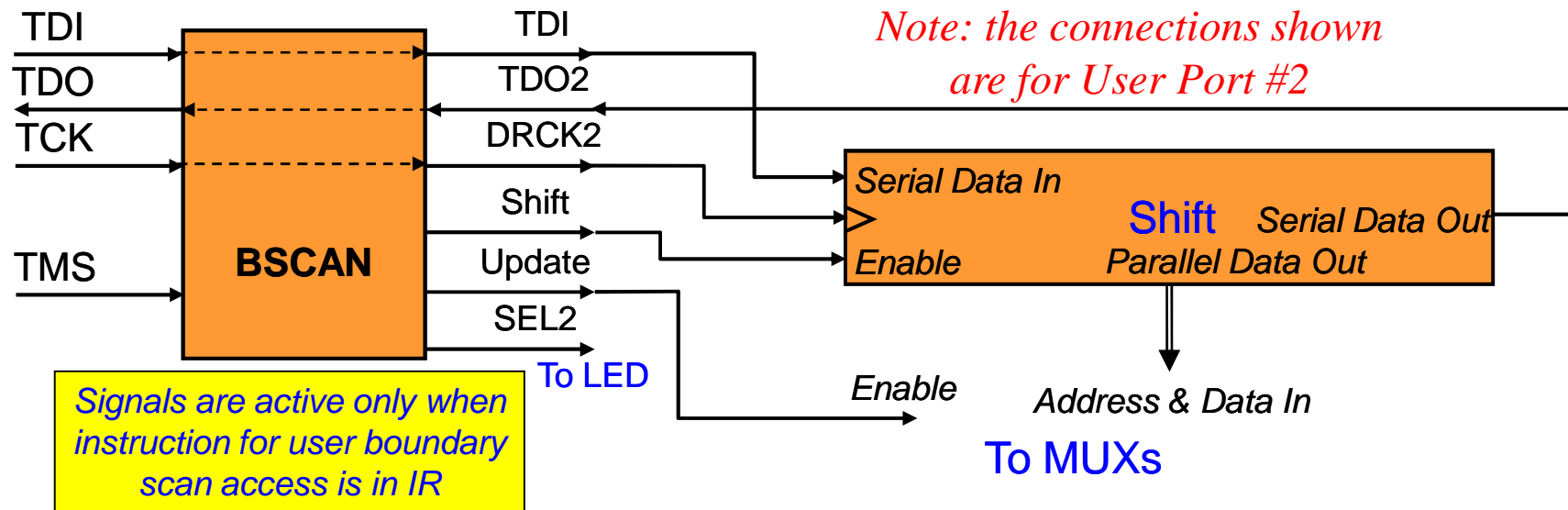
- 6-bit instruction
 - User port #1: 0x02
 - User port #2: 0x03
 - note that this is what is shifted into the IR in the time example below
 - Shifted into IR LSB first
 - Exit IR on last bit of instruction
 - You are now in Shift-DR
 - ready to access BSCAN port



Note: transitions on rising edge of TCK based on TMS value

Boundary Scan Communications

- As you serially shift in address and data on TDI via the USER2 port of the BSCAN module, the previous values you sent to the register file will shift out on TDO
 - As a sanity check, you can continue to shift in data and see it come out 6 clock cycles later in TDO in the Impact GUI
- As you shift in the last bit of your address and data bring TMS high to Exit-DR and the next clock with TMS high will activate Update and load your data into the appropriate register in your register file
 - You can go back to Shift-DR to shift in a new set of address and data values for another register and repeat the process as long as you like without changing IR



Boundary Scan Interface

- Lab exercise:
 - Show your pre-lab work to the GTA at the beginning of the lab session
 - Simulate your VHDL models and verify your design using ModelSim, debug as necessary
 - Note that you will “force” the outputs of the BSCAN module during your simulation since you will not be able to simulate the module in ModelSim
 - Synthesize your design for the Spartan 3S200 FPGA using ISE
 - Open the synthesis report file and record #Slices, #LUTs, and #FF/latches
 - Test and debug your circuit using the Impact Boundary Scan interface to move through the TAP state diagram and communicate with your internal circuitry
 - Demonstrate your working circuit (& your ability to communicate with BS) to the GTA

Boundary Scan Interface

- Post-lab: Turn in your lab report at the beginning of the next lab session, including:
 - Verified parameterized VHDL model
 - Indicate which BS User Port you chose
 - Simulation results
 - A logic diagram showing interconnection and hierarchy of your VHDL models with internal signal names
 - Synthesis Report results (#slices, #LUTs, and #FFs/latches)
 - Pre-lab work, including
 - FPGA pin numbers and PCB functions used for synthesis and verification