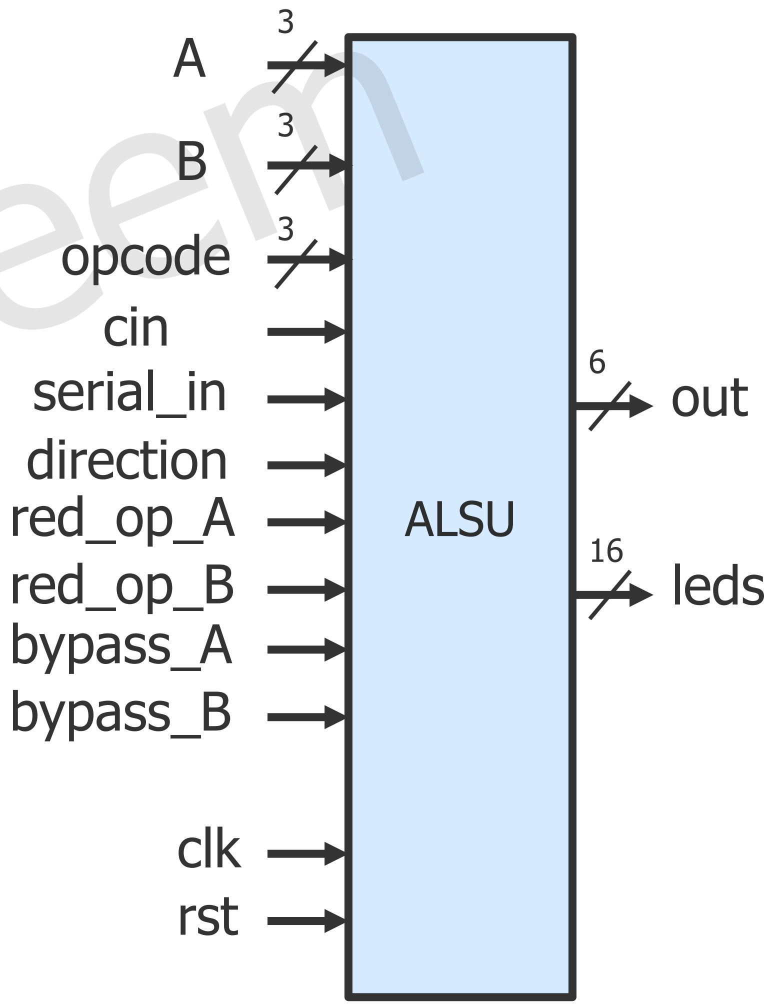
Arithmetic logic shift unit (ALSU)

- ALSU is a logic unit that can perform logical, arithmetic, and shift operations on input ports
- Input ports A and B have various operations that can take place depending on the value of the opcode
- Each input bit except for the clk and rst will be sampled at the rising edge before any processing so a D-FF is expected for each input bit at the design entry
- The output of the ALSU is registered and is available at the rising edge of the clock



Inputs

- All inputs except for the clk and rst will be sampled at the rising edge. The processing on the inputs shall only be done

Input	Width	Description
clk	1	Input clock
rst	1	Active high asynchronous reset
A	3	Input port A
B	3	Input port B
cin	1	Carry in bit, only valid to be used if the parameter FULL_ADDER is "ON"
serial_in	1	Serial in bit, used in shift operations only
red_op_A	1	When set to high, this indicates that reduction operation would be executed on A rather than bitwise operations on A and B when the opcode indicates AND and XOR operations
red_op_B	1	When set to high, this indicates that reduction operation would be executed on B rather than bitwise operations on A and B when the opcode indicates AND and XOR operations
opcode	3	Opcode has a separate table to describe the different operations executed
bypass_A	1	When set to high, this indicates that port A will be registered to the output ignoring the opcode operation
bypass_B	1	When set to high, this indicates that port B will be registered to the output ignoring the opcode operation
direction	1	The direction of the shift or rotation operation is left when this input is set to high; otherwise, it is right.

Outputs and parameters

Output	Width	Description
leds	16	All bits are blinking when an invalid operation takes place. Acts as a warning, otherwise it is set to low
out	6	Output of the ALSU

Parameter	Default value	Description
INPUT_PRIORITY	A	Priority is given to the port set by this parameter whenever there is a conflict. Conflicts can occur in two scenarios, red_op_A and red_op_B are both set to high or bypass_A and bypass_B are both set to high. Legal values for this parameter are A and B
FULL_ADDER	ON	When this parameter has value “ON” then cin input must be considered in the addition operation between A and B. Legal values for this parameter are ON and OFF

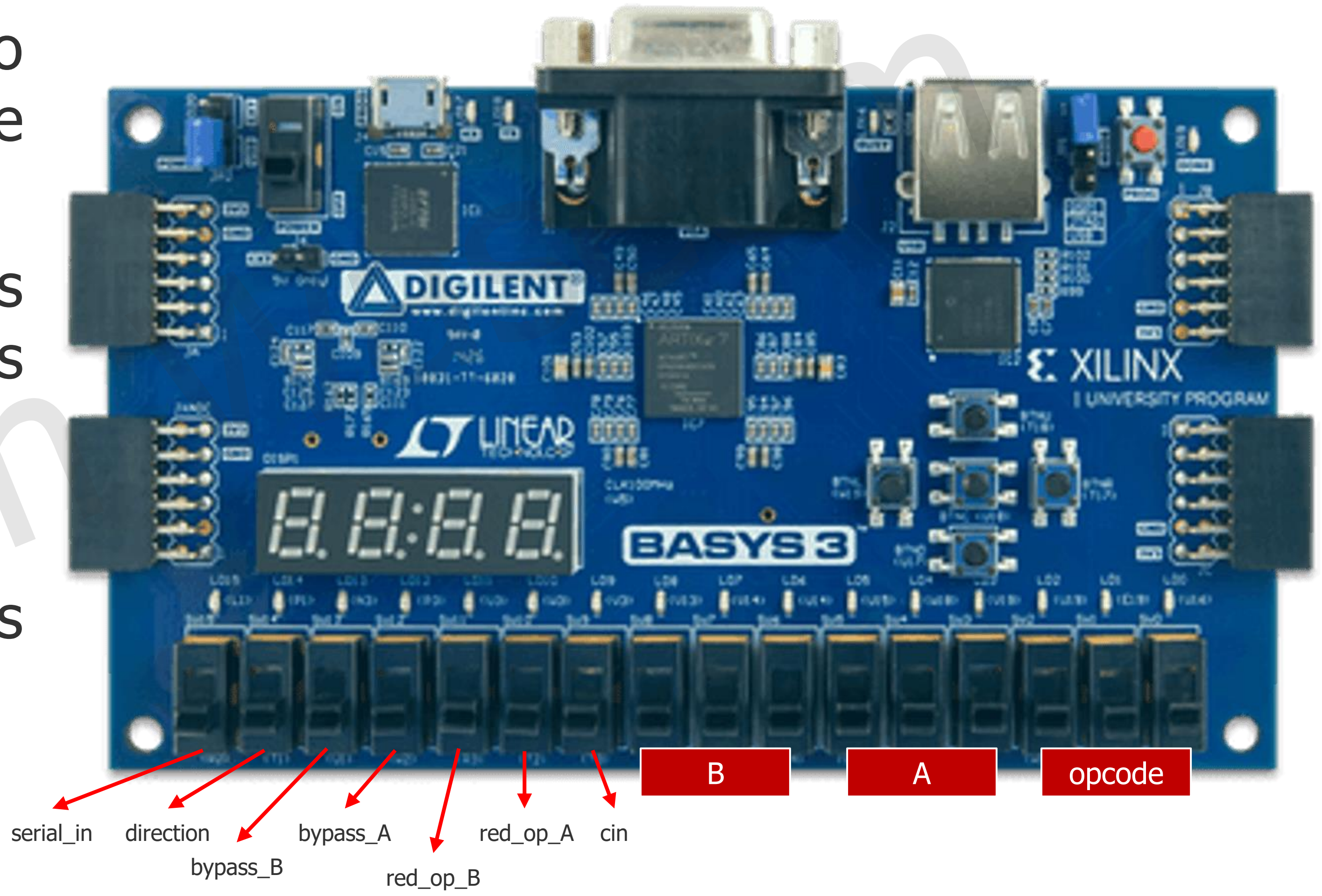
Opcodes & Handling invalid cases

- Invalid cases
  1. Opcode bits are set to 110 or 111
  2. red_op_A or red_op_B are set to high and the opcode is not AND or XOR operation
- Output when invalid cases occurs
  1. leds are blinking
  2. out bits are set to low

Opcode	Operation
000	AND
001	XOR
010	Addition
011	Multiplication
100	Shift output
101	Rotate output
110	Invalid opcode
111	Invalid opcode

XDC file

- You are required to write the XDC file for the basys3 board to connect the inputs to the switches as shown on the board
- "clk" is connected to W5 pin as suggested in the board's reference manual
- "rst" is connected to button U18
- "leds" are connected to the LEDs on the board



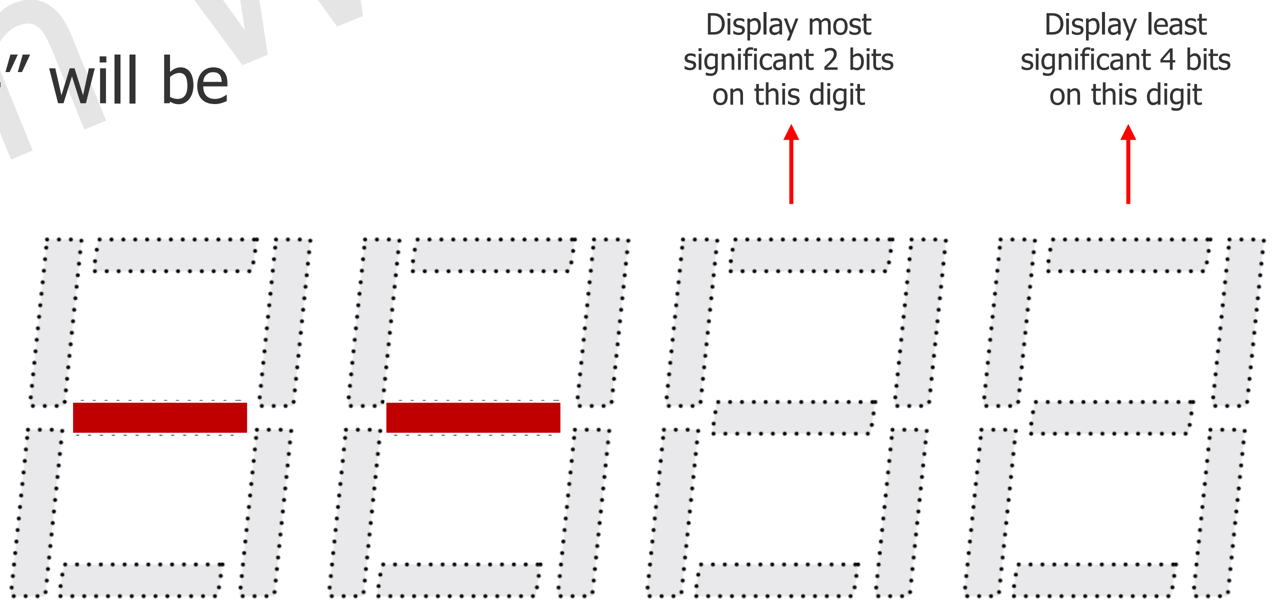
Bonus

- The output shall be displayed on the seven segment display
- Adjustments to the outputs
  - "out" port will not be output anymore and it will be an internal register
  - The output ports are shown below where two outputs are added to be connected to the seven segment display
  - You can learn more on how to drive the seven segment display in Basys reference manual and in this link

Output	Width	Description
leds	16	All bits are blinking when an invalid operation takes place. Acts as a warning
anode	4	Output to drive the anodes of the seven segment display
cathode	7	Output to drive the cathodes of the seven segment display

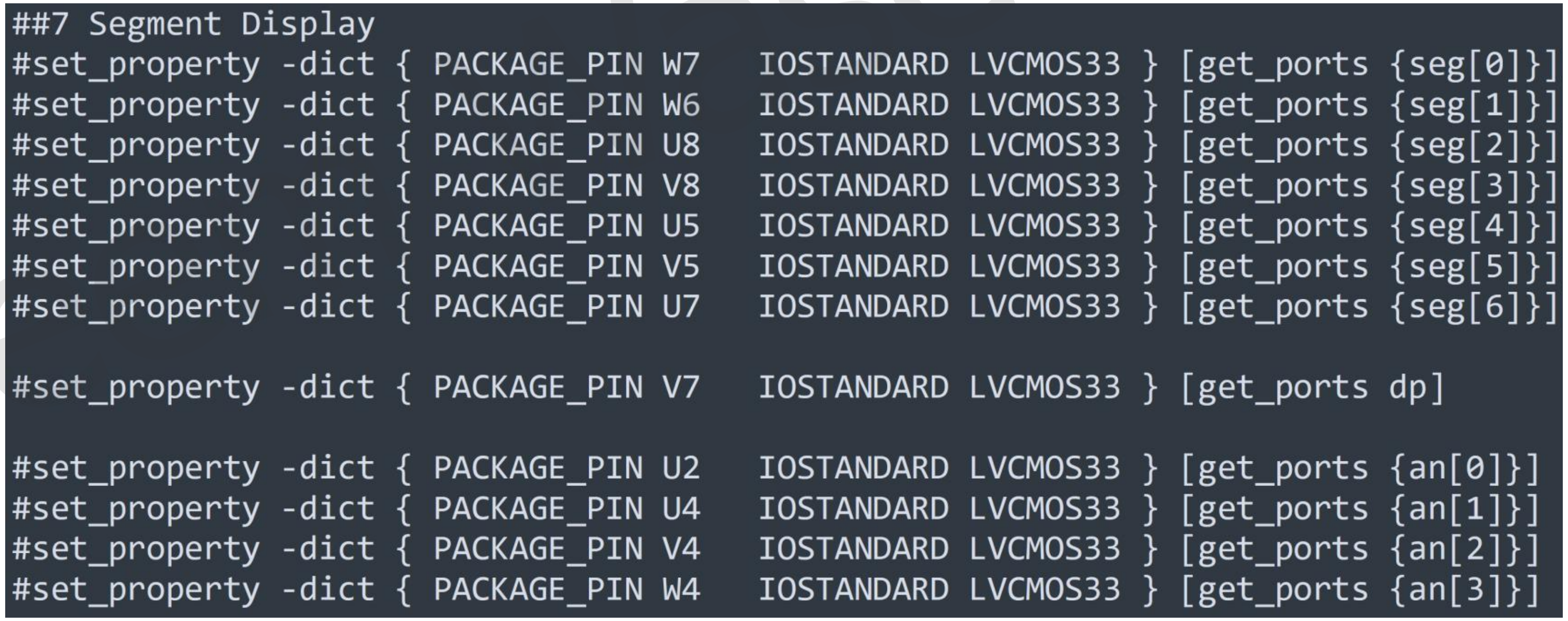
Bonus – seven segment display

- The least significant 4 bits of the internal register "out" which is the ALSU output will be displayed on the right most digit
- The most significant 2 bits of the internal register "out" will be displayed on the second digit on the right
- The 2 digits on the left will be displaying a dash in the middle of them as they are not used
- If invalid case occurs then "E404" will be displayed on the 4 digits


Display most significant 2 bits on this digit
Display least significant 4 bits on this digit

Bonus – XDC File

- Add the seven segment display constraints in the XDC file
  - Connect the anode pins
  - Connect the cathode pins
  - Ignore DP pin



```
##7 Segment Display
#set_property -dict { PACKAGE_PIN W7   IOSTANDARD LVCMOS33 } [get_ports {seg[0]}]
#set_property -dict { PACKAGE_PIN W6   IOSTANDARD LVCMOS33 } [get_ports {seg[1]}]
#set_property -dict { PACKAGE_PIN U8   IOSTANDARD LVCMOS33 } [get_ports {seg[2]}]
#set_property -dict { PACKAGE_PIN V8   IOSTANDARD LVCMOS33 } [get_ports {seg[3]}]
#set_property -dict { PACKAGE_PIN U5   IOSTANDARD LVCMOS33 } [get_ports {seg[4]}]
#set_property -dict { PACKAGE_PIN V5   IOSTANDARD LVCMOS33 } [get_ports {seg[5]}]
#set_property -dict { PACKAGE_PIN U7   IOSTANDARD LVCMOS33 } [get_ports {seg[6]}]

#set_property -dict { PACKAGE_PIN V7   IOSTANDARD LVCMOS33 } [get_ports dp]

#set_property -dict { PACKAGE_PIN U2   IOSTANDARD LVCMOS33 } [get_ports {an[0]}]
#set_property -dict { PACKAGE_PIN U4   IOSTANDARD LVCMOS33 } [get_ports {an[1]}]
#set_property -dict { PACKAGE_PIN V4   IOSTANDARD LVCMOS33 } [get_ports {an[2]}]
#set_property -dict { PACKAGE_PIN W4   IOSTANDARD LVCMOS33 } [get_ports {an[3]}]
```