|  |
| --- |
| EXPERIMENT: A Datapath and Control Unit |

References

1. *Digital Design Using Digilent FPGA Boards – VHDL / Active-HDL Edition*
   1. Example 34b: *Multiplication using the Multiplication Operator*
   2. Example 51: *N-bit Counter*
   3. May be helpful: Example 64: *Datapaths and Control Units*

LAB

The pseudo-code below computes the factorial of an unsigned binary number.

Function Factorial(Num As Integer) As Integer

‘Assumes Num is zero or positive

Dim cnt as Integer

Dim Fact as Integer

Fact = 1

If Num = 0 then

Return (Fact)

End If

For cnt = 1 To Num

Fact = Fact \* cnt

Next cnt

Return (Fact)

End Function

Part 1

1. Design a 32-bit datapath for a special-purpose processor (SPP) that performs the algorithm listed in the pseudo-code above.
   1. Since there are eight 7-segment displays, the maximum answer from the factorial calculation that we can display on the board will be FFFFFFFF (which is 4,294,967,295 decimal). So, the maximum input that we can accept for which we can calculate the factorial and display the answer on the board would be 12 since 12! is 1C8CFC00 (479,001,600 decimal) and 13! is 17328CC00 (6,227,020,800 decimal) which would require 9 digits to display.
   2. This means that the entity should accept a 4-bit input even though the internal data path should be a 32-bit bus which means you will need to concatenate 28 bits of zeros to the beginning of the input into the rest of the circuit. Include in your design an *overflow* bit output which will be high if the input is invalid, that is, it will produce an output greater than FFFFFFFF; this *overflow* bit should go high whenever the input is too large. You may use the built-in multiplication operator *\** in the *IEEE.std\_logic\_unsigned.all* library instead of designing your own multiplier (refer to example 34b in the book). Implement the datapath using VHDL.
2. Design a state machine for a controller to control the datapath in #1 for the algorithm listed in the pseudo-code. The controller should remain in the *Start* state until an input signal, *go* is brought high. However, if the *overflow* bit from the datapath is high, the calculation should not occur. This forces the user to input a number that will provide a valid 32-bit result before your SPP will calculate the factorial. Implement the controller using VHDL.
3. Integrate the state machine and datapath to complete your digital circuit for computing the factorial into a single architecture *fact32*. Figure 1 should give you an idea of how the whole design should look. Simulate *fact32* with a testbench, calculating the factorial of 9.

4

num

*fact32*

go

clr

clk

*fact32\_dp*

*fact32\_ctrl*

ovfl

fact

32

Figure 1: *fact32*

**Deliverables:** fact32\_ctrl*.vhd*, fact32\_dp*.vhd*, fact32.*vhd*, fact32\_tb*.vhd* and functional simulation of fact32

Part 2

1. Design a top-level, *fact32\_top*, to test your design. Include the *x7segb8* component given to you on Moodle so that the output of *fact32* will display on the seven segment displays. Connect the overflow to an LED; it should be lit when there is an overflow. Input to the SPP should use 4 of the switches. Use *btn(0)* to clear the circuit, *btn(1)* to control the *go* signal. Drive the SPP using *mclk*, the 100 MHz clock from the board.
2. Implement your top-level design using VHDL and download *fact32\_top* to the Nexys board and demonstrate it to your lab instructor with valid inputs and inputs that are out of range.

**Deliverables:** fact32\_top*.vhd*