MILESTONE 5

Datapath Unit Implementation

For Milestones 5 and 6, each group will be specified a complex digital system. The options are in Table 5.1. The list is not exhaustive. Your lecturer may have extra topics available. You can also propose your own topic.

Table 5.1: List of possible topics

Title	Level of Difficulty	Starting Point
Ones counter	1	In-book
GCD finder version 1	2	In-book
Serial adder	3	In-book
Binary multiplier	3	*http://freeusermanuals.com/backend/web/manuals/1523428835ASM_design_example_bin_mult.pdf*
GCD finder version 2	3	*http://esd.cs.ucr.edu/labs/gcd/gcd.html*
GCD finder version 3	3	*http://csg.csail.mit.edu/6.375/6_375_2006_www/handouts/lectures/L03-Verilog-Design-Examples.pdf*
Laser based distance measurer	3	*http://www2.engr.arizona.edu/~rlysecky/courses/ece274-08s/lectures/lecture13.pdf*
Serial binary-to-BCD converter	3	*https://www.digikey.com/eewiki/pages/viewpage.action?pageId=60030986*
UART transmitter	4	*http://electrotech99.blogspot.com/2011/05/design-and-implementation-of-uart.html*
Serial multiplier	4	*http://digsys.upc.edu/ed/CSD/prob/Ch3/P10/Prob3_10.html*
Booth multiplier	5	*http://www.cse.iitd.ernet.in/~neeraj/TA/cs316/2005-2006/project/final/mansi/*
Integer square root	5	*https://pdfs.semanticscholar.org/9b7e/07f3b5ed9c168f13206836a326bd9b91135a.pdf*
Serial divider	5	*https://www.csee.umbc.edu/portal/help/VHDL/samples/samples.shtml#div_ser*
UART receiver	5	*https://www.engr.siu.edu/haibo/ece428/notes/ece428_uart.pdf*

The steps for building the datapath for a naïve multiplier is shown in this handout. Follow the same procedure for the circuit assigned to you.

5.1 Problem Formulation

A naïve multiplier performs multiplication by repeated addition[1]. In this multiplier, multiplying 2 by 5 means performing the addition five times:

$$0+2+2+2+2+2=10$$

The algorithm for computing $P \leftarrow M \times N$ is as follows:

```
int naivemultiplier(int M, int N) {
  int P;

  P = 0;
  while( N != 0) {
    P = P + M;
    N = N - 1;
  }
  return P;
}
```

[1] A binary multiplier typically uses the binary system which can reduce hardware and execution time.

Table 5.2: Clock-by-clock countdown of Multiplier datapath.

Data			Signals		
Multiplicand (M)	**Multiplier** (N)	**Product** (P)	**Init**	**Work**	**Zero**
?	?	?	1	0	0
2	5	0	0	1	0
2	4	2	0	1	0
2	3	4	0	1	0
2	2	6	0	1	0
2	1	8	0	1	0
2	0	10	0	0	1

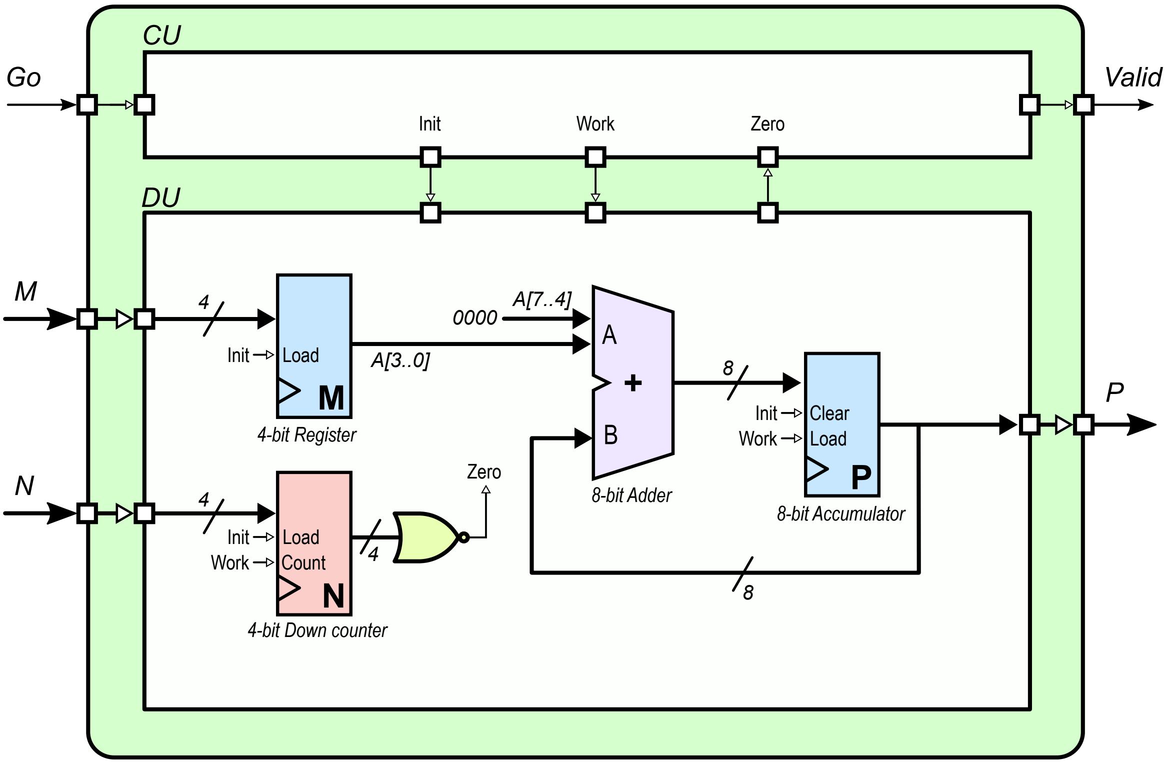

Figure 5.1: Naive multiplier DPU + CU.

5.2 Identify and Construct the Building Blocks

Identify the components required to build a 4-bit × 4-bit multiplier:

- **Adder**
  Extend the adder from Milestone 1 from 4 bit to 8 bit.
- **M register**
  4-bit register with enable. This register must have 4-bit input, 4-bit output and Load control input.
- **N register**
  This “register” is actually a *4-bit down-counter* with load capability. The register must have 4-bit data input, Load control input, Count down enable input and Zero output. The Zero output is high when the counter reaches 0000. The actual value of the register is not important except for debugging.
- **P register**

8-bit register with enable. This register must have 8-bit input, 8-bit output and Load control input.

Design and build each component and test independently. This is a design course. As a digital designer, you are expected to be able to design each module yourself, individually. You are free to use **lpm** built-in modules, create your own module using schematics or write Verilog code to implement the modules.

5.3 Integration

1. Connect the datapath according to Fig. 5.2.
2. Test the datapath using sensible test data. Before you simulate, you should know what the expected outputs are. Ability to select input data and to verify circuit operation is also expected from all digital designers.
3. Save the datapath as a symbol file.

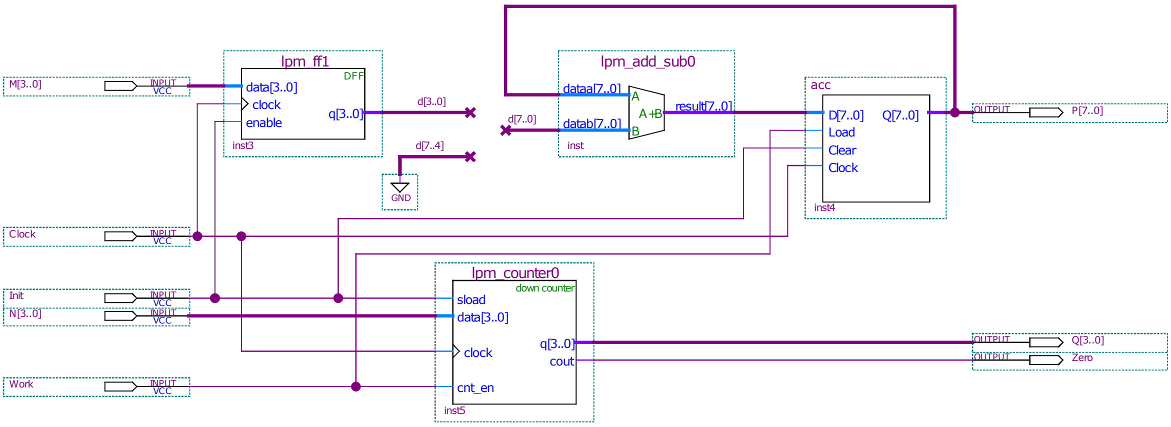

Figure 5.2: Naive multiplier datapath. The signal group Q[3..0] shows the current value of the counter. It is not used in the final system because the controller only needs to know whether the counter has reached zero (*cout*=*Zero*=1).

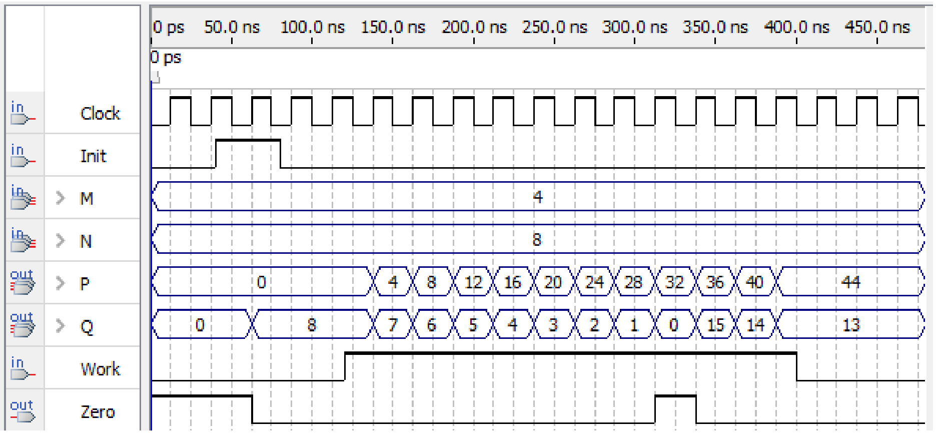

Figure 5.3: Simulation of naive multiplier datapath. The *Work* signal enables the counter to count down and the acccumulator to load a new value. When the controller is added later, *Work* can be stopped automatically when *Zero* = 1.

You have achieved Milestone 5 by correct simulation of the datapath unit.