

**ECE 368, Fall 2001, Instructor: Prof. Shantanu Dutt**

**Final Exam: Tue, Dec. 9, Time: 1-3 pm**

**Exam Format: Open Book and Notes, Total Points: 150**

**Important Note:** (1) Write your name on the top of this question sheet and submit along with your answer book

(2) You need to show all your work **clearly** in deriving the answers. Just writing down the final answers is not enough.

**Suggestion:** Begin by reading all questions and do those first that you think you know best.

## 1. Pipelining

Consider the computation given below using regular programming language statements:

```
for  $i = 1$  to 1000 do
```

```
begin
```

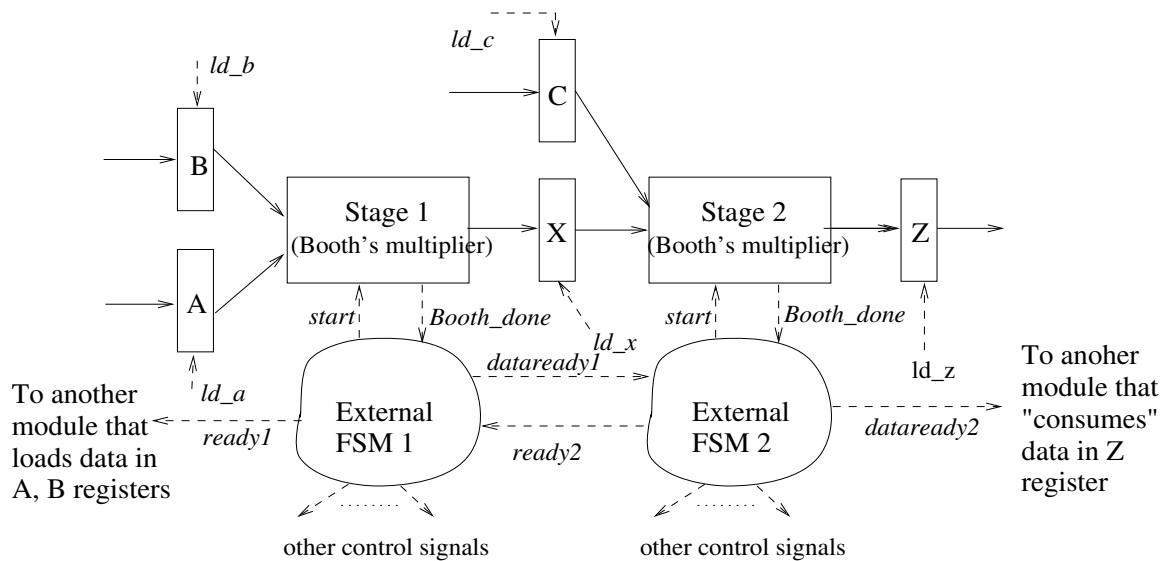
```
 $x = a(i) \times b(i)$ ; /*  $a(i), b(i)$  are input sequences of integers */
```

```
 $z(i) = x \times c(i)$ ; /*  $c(i)$  is another input sequence of integers */
```

```
/*  $z(i)$  is an output sequence of integers */
```

```
endfor
```

Assume that the input numbers  $a(i), b(i), c(i)$  are 16-bit numbers. The above computation needs to be **pipelined** using **two** Booth's multipliers, with each multiplier being a stage in the pipeline. Each Booth's multiplier hardware is composed of a datapath and a controlling FSM of the type you designed in your lab problem. The Booth's multiplier along with the FSM is enclosed in the box labeled "Booth's multiplier" shown in the figure below.



Since the number of clock cycles taken by a Booth multiplier to complete a multiplication is not predetermined, each stage can take a variable # of ccs to complete. Thus an **external FSM** is needed for **each stage** in order to synchronize data transfer correctly between the stages, as shown in the above figure.

- (a) Assuming that each Booth's multiplier takes  $2(\lceil n/3 \rceil) + n$  cc's<sup>1</sup> to complete a multiplication on the average, and that signaling between the external FSMs for each stage for correct data transfer takes an additional 3 cc's per stage, how many cc's would it take on the average to complete the above given computation (1000 iterations) in your pipelined design. **Show your calculation clearly starting with the initial general expression for a pipeline processing time.** **20**
- (b) What is the average throughput of the pipeline? **10**

---

<sup>1</sup>Strictly speaking this would be the time taken by a *Modified* Booth's multiplier—it does  $n/3$  subtractions or additions on the average, and we get the above function assuming 2 ccs for addition and 1cc for shifting. So in this problem you can assume that by Booth's multiplier we mean a Modified Booth's multiplier.

## 2. VHDL Code Modification

A module (a piece of hardware that does something useful) “something” is to be described in VHDL with the following specifications. It has two bit inputs  $start1$ ,  $start2$  and two integer inputs  $x$ ,  $y$ . Inputs  $start1$  and  $x$  are produced by one external module and inputs  $start2$  and  $y$  are produced by another external module. Each bit inputs are asserted (made = '1') for exactly 5 ns by the corresponding external module.  $start1$  becomes '1' when input data  $x$  is available and  $start2$  becomes '1' when input data  $y$  is available. There is no synchrony between the availability of data on the  $x$  and  $y$  inputs. Thus note in particular that  $start1$  and  $start2$  may or may not become '1' simultaneously, and in the case that are not '1' simultaneously, it is not known in which order they will become '1'.

Once any input signal ( $start1$ ,  $start2$ ,  $x$ ,  $y$ ) becomes valid or available, it remains valid until the *done* output signal is asserted by module “something” (see below).

Module “something” is supposed to produce the real output  $z = x^2 + y^2$  with a delay of 50 ns **after both input bit signals  $start1$ ,  $start2$  have become '1' in the current iteration. At the same time** that  $z$  is produced, module “something” also asserts an output bit signal *done* for exactly 5ns.

The following is the VHDL behavioral description of module “something”.

```
entity something is
```

```
port (start1, start2 :in bit; x, y: in integer; z: out real; done: out bit := 0);
```

```
end entity something;
```

```
architecture behav of something is
```

```
signal temp1, temp2: real;
```

```
begin
```

```
do_work: process is
```

```
begin
```

```
wait until (start1 = '1');
```

```
wait until (start2 = '1');
```

```
temp1 <= real(x * x); — real(integer_value) converts integer_value into a floating point representation
```

```
temp2 <= real(y * y);
```

```
z <= temp1 + temp2 after 50 ns;
```

```
done <= '1' after 50 ns;
```

```
wait for 50 ns;
```

```
done <= '0' after 5 ns; – this causes the pulse width of “done” to be 5ns
```

```
end process do_work;  
end architecture behav;
```

(a) **Ignoring syntax errors**, if any (no syntax error has been deliberately inserted, though inadvertently some might have crept in), in the above code, what are correctness/design problem(s) in the code (i.e., problem(s) that will cause it to function incorrectly)? Enumerate each problem clearly and explain what incorrect functioning it causes. **20**

(b) Rewrite the code so that the correctness/design problem(s) are fixed. **Comment the code clearly to explain the fixes.** **40**

### 3. VHDL Simulation

Consider the following VHDL code:

```
entity foo (a,b,c,d: in integer, Z: out integer := 0) is
end entity foo;
```

```
architecture behav of foo is
```

```
signal x, y, w : integer := 1;
```

```
begin
```

```
clock: process is
```

```
begin
```

```
clk <= '0','1' after 40 ns;
```

```
wait for 80 ns ;
```

```
end process clock
```

```
produce_output: process is
```

```
begin
```

– This is the beginning point of each iteration of the process.

– Let us call this point **point 1**.

– Note that a process loops forever from it's begin to its end statements.

```
wait until (clk'event and clk='1');
```

```
x <= a + 2*b;
```

```
y <= b*c;
```

– let us call this point **point 2**.

```
w <= a - d;
```

```
Z <= x - y + w after 25 ns;
```

```
end process produce_output;
```

```
end architecture behav;
```

Assume that the very first time that process “produce\_output” comes to **point 1**, it is the beginning of iteration 1. We will denote this combination of iteration and point as (**iter=1, pt=1**), and similarly the first time that **point 2** is reached as (**iter=1, pt=2**), and so on.

(a) What are the *simulation times* at: (**iter=1, pt=1**), (**iter=1, pt=2**), (**iter=2, pt=1**) and (**iter=2, pt=2**)?

Give the simulation times in the table provided on the last page. **20**

(b) How many iterations will it take for  $Z$  to get its final value after each set of values of inputs  $a, b, c, d$  is asserted? Assume that the inputs change every positive edge of the cc. **10**

(c) Suppose that at the positive edge of the first cc,  $a = 10, b = 20, c = 5, d = 40$ . Also assume that at the positive edge of every subsequent cc **i.e., starting with the second cc**, each of these input values increase by 5. What are the values of  $x, y, w, Z$  at the following points: **(iter=i, pt=1), (iter=i, pt=2)**, for  $i = 1$  to 3. Once again give the answer in the table provided below. **30**

Table for solution to part (a):

Iter, pt combination	Simulation Time
(iter=1, pt=1)	
(iter=1, pt=2)	
(iter=2, pt=1)	
(iter=2, pt=2)	

Table for solution to part (c):

Iter, pt combination	$x$	$y$	$w$	$Z$
(iter=1, pt=1)				
(iter=1, pt=2)				
(iter=2, pt=1)				
(iter=2, pt=2)				
(iter=3, pt=1)				
(iter=3, pt=2)				