UNIVERSITY OF LIVERPOOL

Digital Systems Design ELEC373/473

Behavioural Modelling and Synthesis

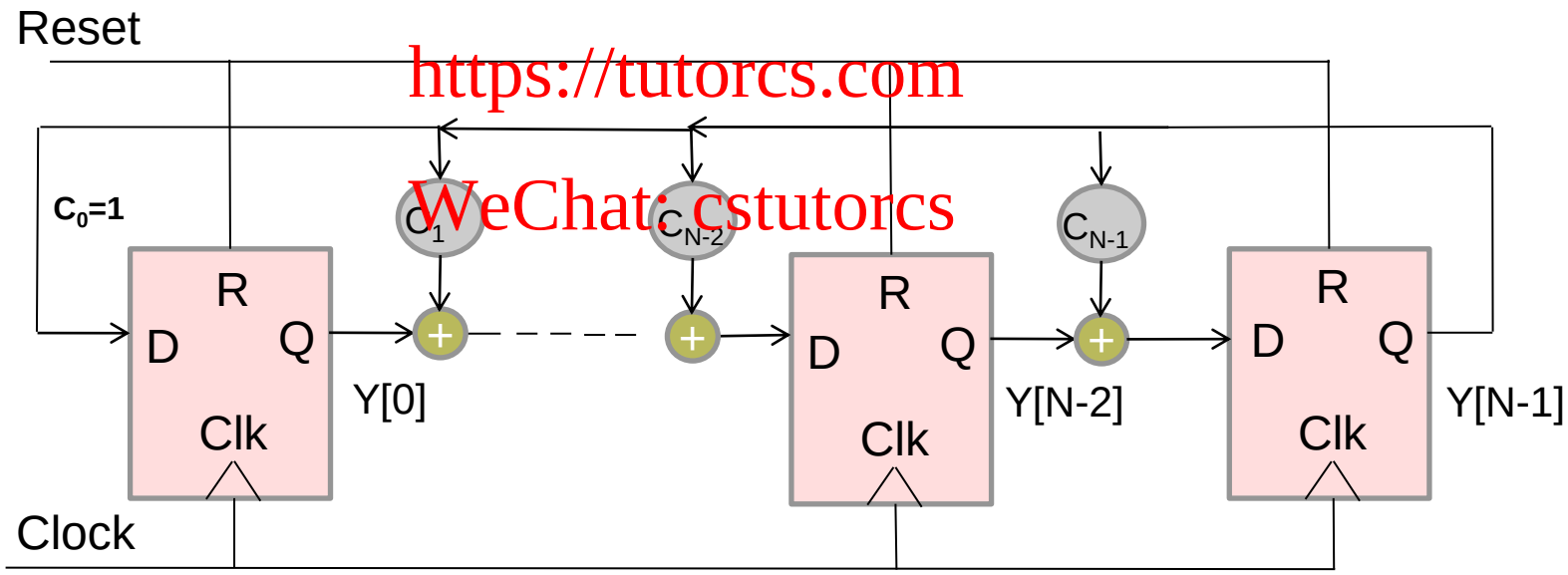
(Modelling Repetitive Algorithms)

Extracted from: Verilog HDL by M.D. Ciletti and IEEE Std 1364-2001 Version C

Prof J.S. Smith
Room A515;
E-mail: j.s.smith@liv.ac.uk

Dataflow Models of a Linear Feedback Shift Register (LFSR)

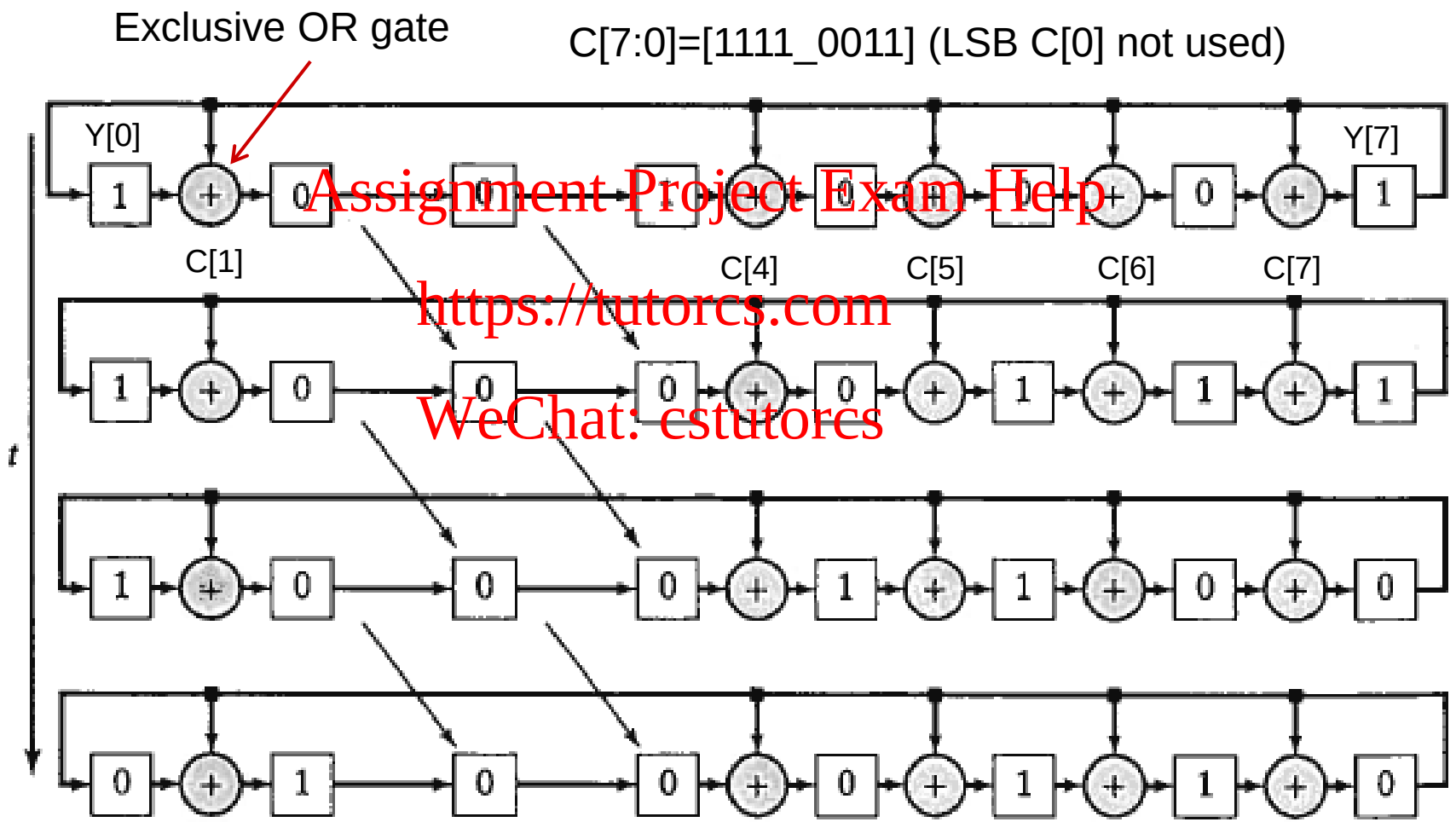
- LFSRs are commonly used in generating pseudo-random binary numbers and have many other applications like cyclic redundancy check (CRC).



$C_x = 1$ means Y[x] is connected to the XOR gate (except for LSB).

$C_x = 0$ means there is no connection.

Data Movement in an LFSR

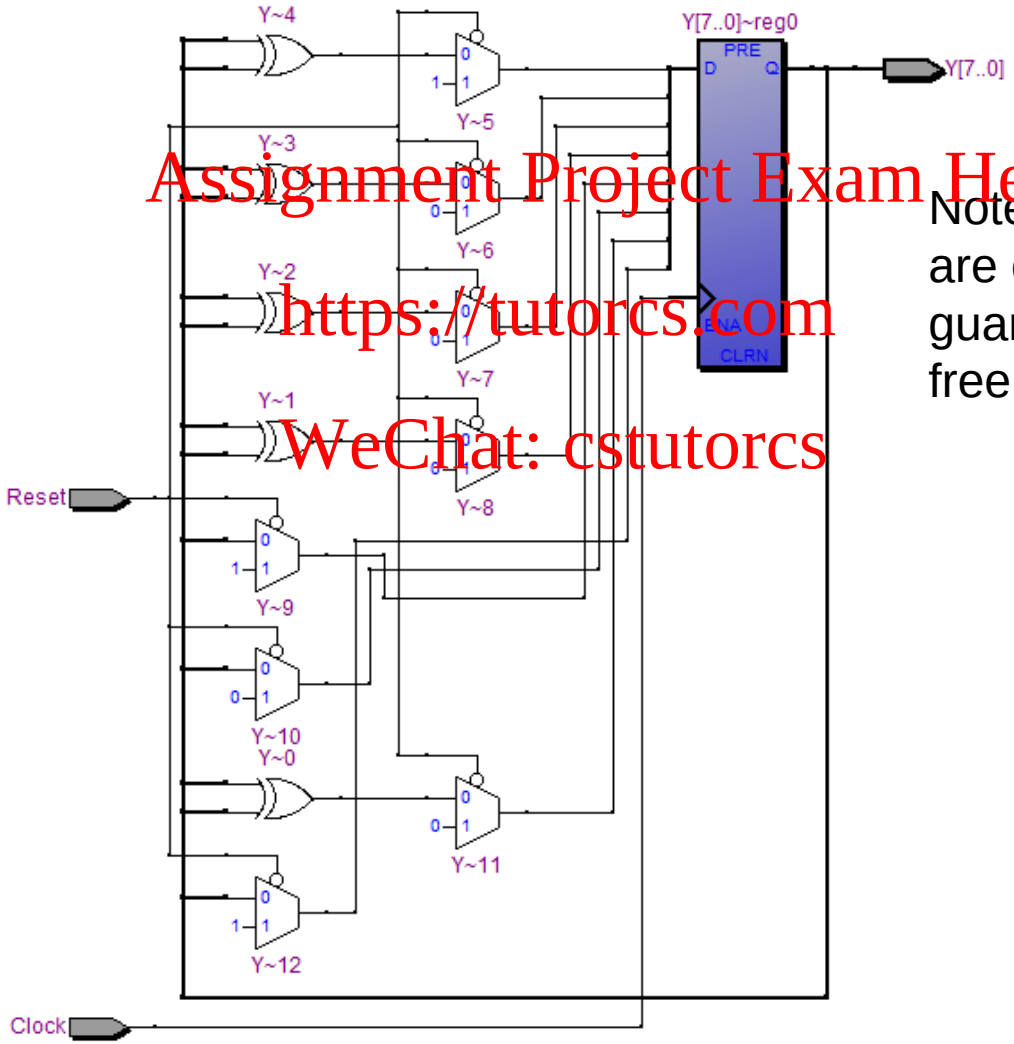


8-Cell LFSR using Nonblocking Assignments

```
1   module LFSR_RTL (Y, Clock, Reset);
2      parameter                  Length = 8;
3      parameter                  initial_state   =8'b1000_1001;
4      parameter [Length-1:0]     Tap_Coefficient =8'b1111_0011;
5
6      input                      Clock, Reset;
7      output    [Length-1:0]     Y;
8      reg       [Length-1:0]     Y;
9
10
11     always @ (posedge Clock)
12        if (Reset==0)  Y <= initial_state;
13           else
14              begin
15                 Y[0] <= Y[7];
16                 Y[1] <= Tap_Coefficient[1] ? Y[0] ^ Y[7] : Y[0];
17                 Y[2] <= Tap_Coefficient[2] ? Y[1] ^ Y[7] : Y[1];
18                 Y[3] <= Tap_Coefficient[3] ? Y[2] ^ Y[7] : Y[2];
19                 Y[4] <= Tap_Coefficient[4] ? Y[3] ^ Y[7] : Y[3];
20                 Y[5] <= Tap_Coefficient[5] ? Y[4] ^ Y[7] : Y[4];
21                 Y[6] <= Tap_Coefficient[6] ? Y[5] ^ Y[7] : Y[5];
22                 Y[7] <= Tap_Coefficient[7] ? Y[6] ^ Y[7] : Y[6];
23              end
24  endmodule
```

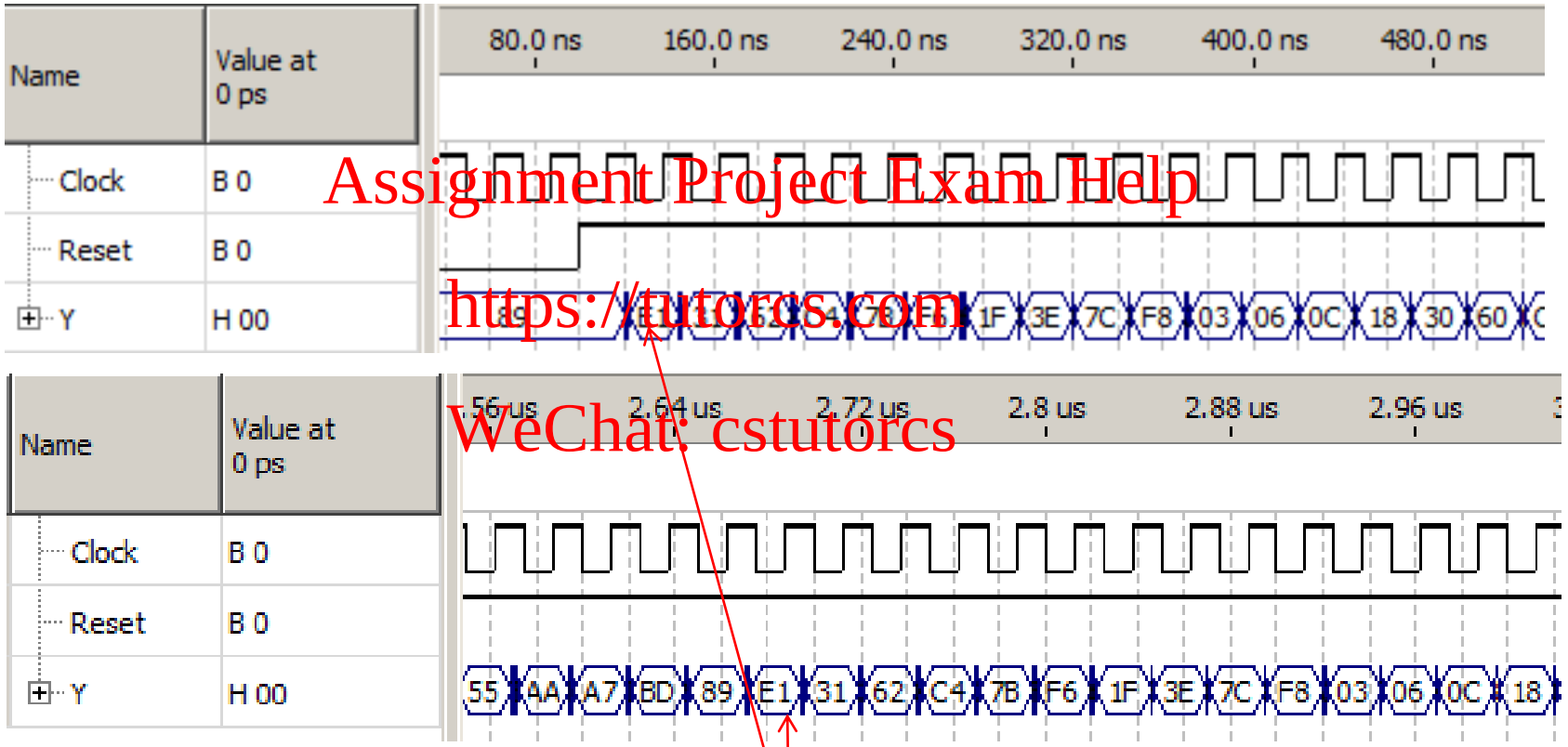

Note in this code everything is in the always@(posedge) block as the only output is the register value you can do this if there are only register outputs

Synthesised circuit



Note only register values are output. These are guaranteed to be glitch free.

Simulation Result



Could we go to 256 cycles?

The pattern is repeated after 2550 ns = 51 cycles X 50ns
This period can be increased to 255 cycles by changing the coefficients.

Looping Statements IEEE Std 1364-2001

- *forever*
  - *Continuously executes a statement.*
- *repeat*
  - *Executes a statement a fixed number of times. If the expression evaluates to unknown or* high impedance, it shall be treated as zero, and no statement shall be executed.
- *while*
  - *Executes a statement until an expression becomes false. If the expression starts out false,* the statement shall not be executed at all.

Looping Statements IEEE Std 1364-2001

- *for*
  - *Controls execution of its associated statement(s) by a three-step process, as follows:*

a) Executes an assignment normally used to initialize a variable that controls the number of loops executed.

b) Evaluates an expression—if the result is zero, the for-loop shall exit, and if it is not zero, the for-loop shall execute its associated statement(s) and then perform step c. If the expression evaluates to an unknown or high-impedance value, it shall be treated as zero.

c) Executes an assignment normally used to modify the value of the loop-control variable, then repeats step b.

8-cell LFSR using For loop

For loop can be used for describing repetitive algorithms.

```
1  module LFSR_RTL_for (Y, Clock, Reset);
2      parameter                Length = 8;
3      parameter                initial_state   =8'b1000_1001;
4      parameter [Length-1:0]   Tap_Coefficient =8'b1110_0011;
5
6      input                    Clock, Reset;
7      output     [Length-1:0]  Y;
8      reg        [Length-1:0]  Y;
9      integer                  Cell_ptr;
10
11
12     always @ (posedge Clock)
13        begin
14           if (Reset==0)  Y <= initial_state;
15              else
16                 begin
17                    Y[0] <= Y[Length-1];
18                    for (Cell_ptr = 1; Cell_ptr <= (Length-1); Cell_ptr = Cell_ptr +1)
19                    if(Tap_Coefficient[Cell_ptr] == 1)
20                       Y[Cell_ptr] <= Y[Cell_ptr-1] ^ Y[Length-1];
21                    else
22                       Y[Cell_ptr] <= Y[Cell_ptr-1];
23                 end
24        end
25 endmodule
```


initial_statement executes once

If the control expression is true the following statement(s) will be executed.

If the control expression is false the loop terminates.

index_statement

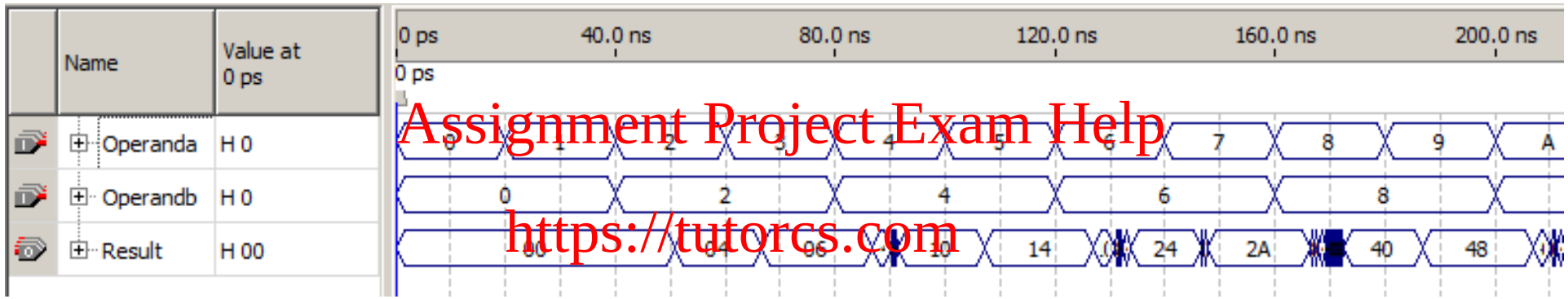
Repeat Example: Combinational Multiplier

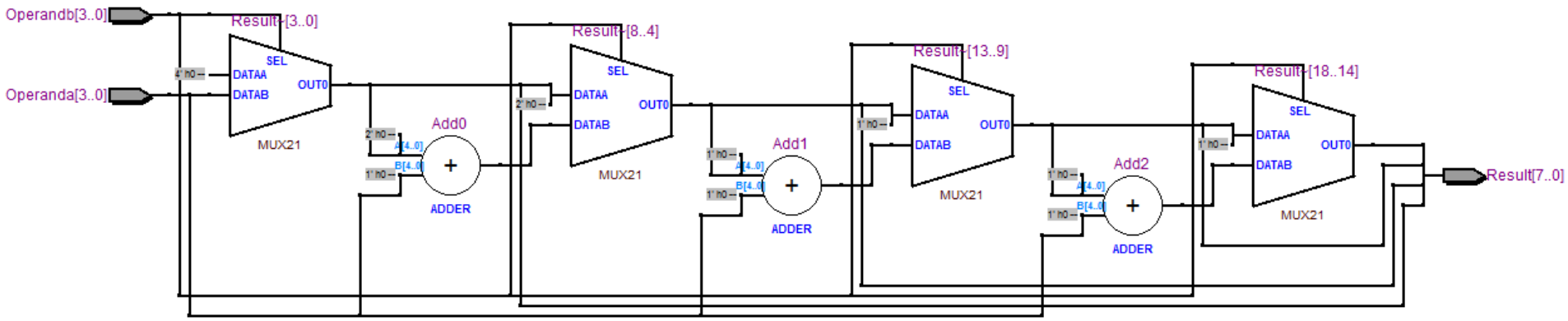
```
1  module Multiplier (Result, Operanda, Operandb);
2
3     parameter   SIZE = 4;
4     parameter   LONGSIZE = 8;
5
6     output   [LONGSIZE-1:0] Result;
7     input    [SIZE-1:0] Operanda, Operandb;
8
9     reg      [LONGSIZE-1:0] shift_opa, shift_opb, Result;
10
11    always @ (Operanda, Operandb)
12       begin
13          /* The left 4 bits of shift_opa and shift_opb are assinged to 0
14             while their right 4 bits are assigned to Operanda and Operanndb */
15          shift_opa = Operanda;
16          shift_opb = Operandb;
17          Result = 0;
18
19          repeat (SIZE)
20             begin
21                if (shift_opb[0])
22                   Result = Result + shift_opa;
23                shift_opa = shift_opa << 1;
24                shift_opb = shift_opb >> 1;
25             end
26       end
27 endmodule
```


Will this produce a combinational or sequential design?

Operanda	1010	X
Operandb	1101	
Shift_opa	00001010	+
	00000000	
	00101000	
	01010000	
Result	10000010	

Simulation and Synthesis





No latches were synthesized. Combinational circuit with Adders and Multiplexers.

Counting 1’s in a 7 bit value

The next few examples are going to count the number of 1’s in an 7 bit value:

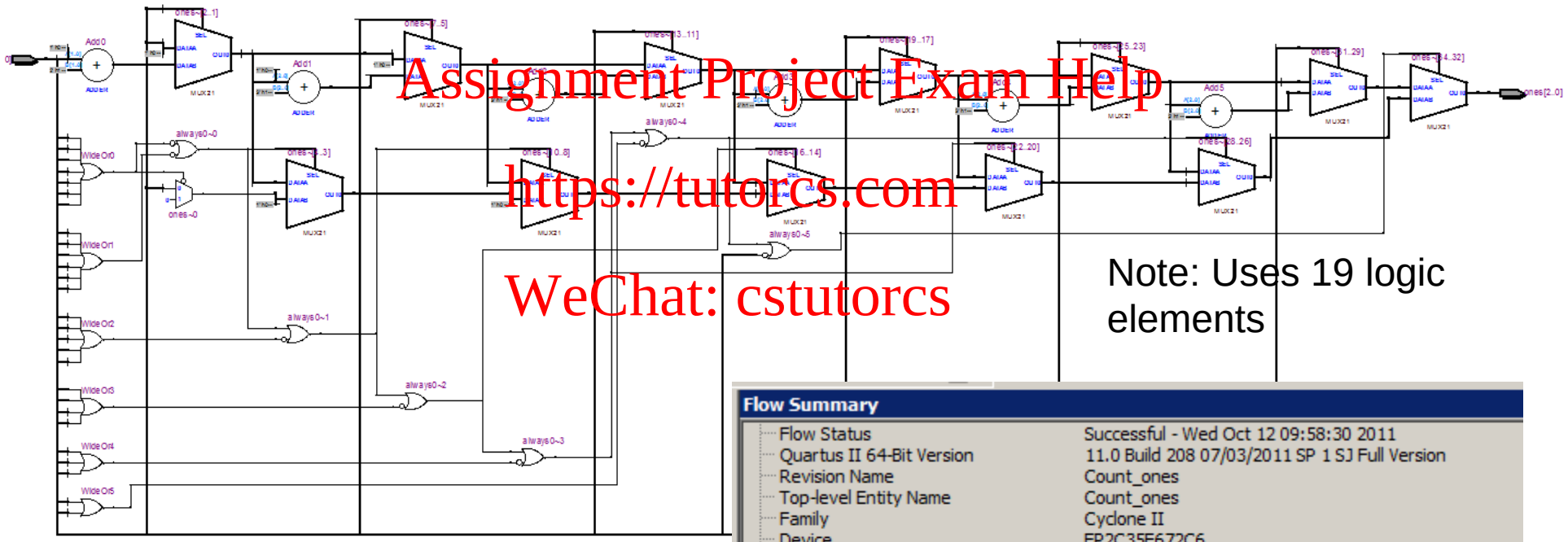
i.e. 1010101B = 4 1’s

"While" Example: Counting the Number of Ones in an Input Register

```
1  module Count_ones (ones, rega);
2
3      output   [2:0] ones;
4      input    [6:0] rega;
5
6      reg      [6:0] temp_reg;
7      reg      [2:0] ones;
8
9      always @ (rega)
10     begin
11        ones = 0;
12        temp_reg = rega;
13        while (temp_reg)
14           begin
15              if (temp_reg[0]) ones = ones + 1;
16              temp_reg = temp_reg >> 1;
17           end;
18     end
19 endmodule
```


Will this produce a combinational or sequential design?

Synthesised Circuit

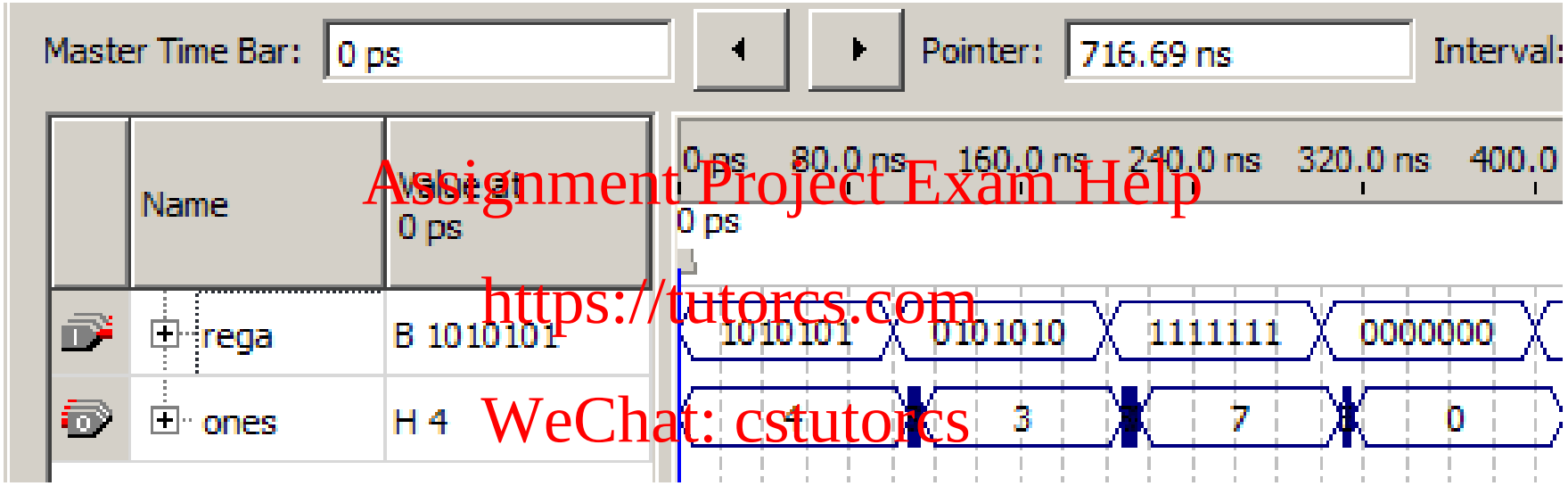


Note: Uses 19 logic elements

Flow Summary	
Flow Status	Successful - Wed Oct 12 09:58:30 2011
Quartus II 64-Bit Version	11.0 Build 208 07/03/2011 SP 1 SJ Full Version
Revision Name	Count_ones
Top-level Entity Name	Count_ones
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	19 / 33,216 (< 1 %)
Total combinational functions	19 / 33,216 (< 1 %)
Dedicated logic registers	0 / 33,216 (0 %)
Total registers	0
Total pins	10 / 475 (2 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

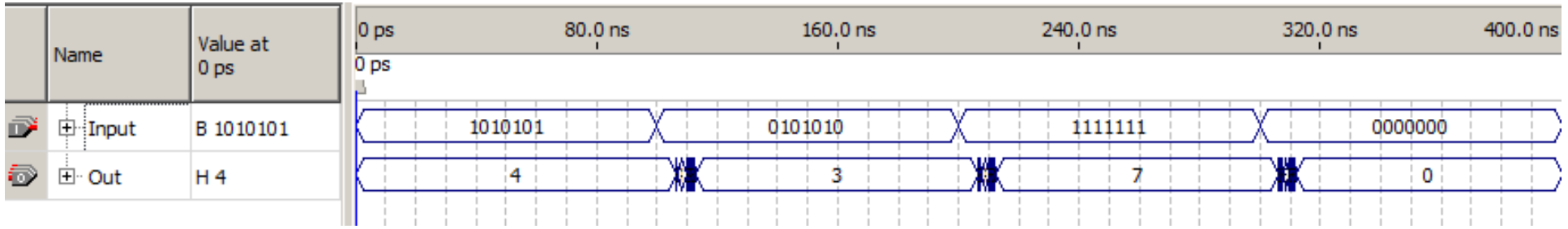
No latches synthesised, all combinational logic.

Simulation Results

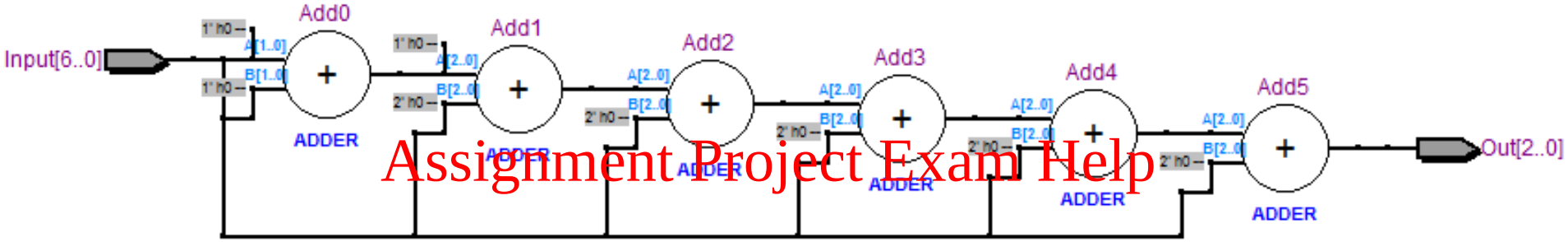


Counting the Number of Ones (more efficient approach)

```
1  module NumberOfOnes ( Out, Input);
2      output    [2:0] Out;
3      input     [6:0] Input;
4
5      assign Out = Input[0]+Input[1]+Input[2]+
6                   Input[3]+Input[4]+Input[5]+Input[6];
7  endmodule
```

Synthesised Design

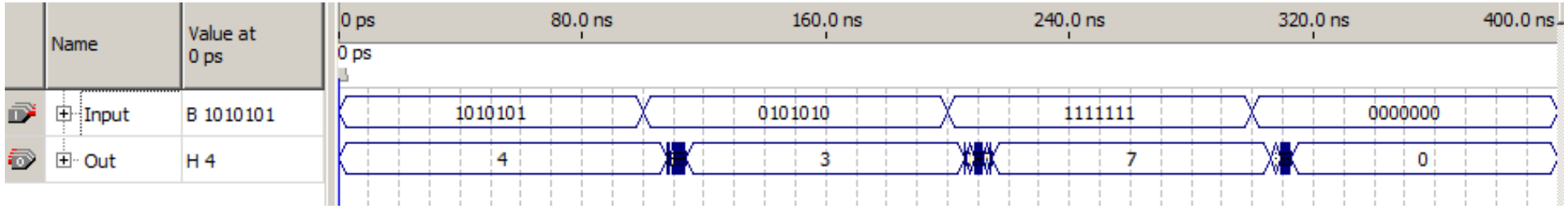


Flow Summary	
Flow Status	Successful - Wed Oct 12 11:44:32 2011
Quartus II 64-Bit Version	11.0 Build 208 07/03/2011 SP 1 SJ Full Version
Revision Name	NumberOfOnes
Top-level Entity Name	NumberOfOnes
Family	Cyclone II
Total logic elements	8 / 33,216 (< 1 %)
Total combinational functions	8 / 33,216 (< 1 %)
Dedicated logic registers	0 / 33,216 (0 %)
Total registers	0
Total pins	10 / 475 (2 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)
Device	EP2C35F672C6
Timing Models	Final

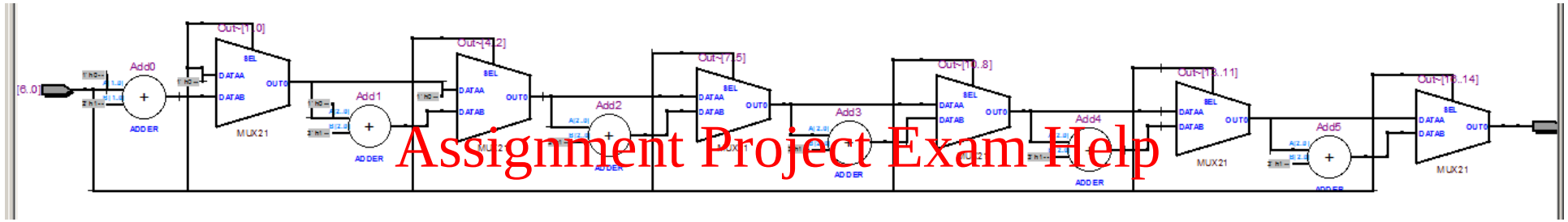
Note: 8 logic elements

Counting the Number of Ones (using a for loop)

```
1  module NumberOfOnes_for( Out, Input);
2     parameter   size = 7;
3
4     output    [2:0] Out;
5     input     [size-1:0] Input;
6
7     reg       [2:0] Out;
8     integer   Count;
9
10    always @ (Input)
11       begin
12          Out = 0;
13          for (Count = 0; Count < size; Count = Count + 1)
14             if (Input[Count])
15                Out = Out + 1;
16       end
17 endmodule
```

Synthesised Design



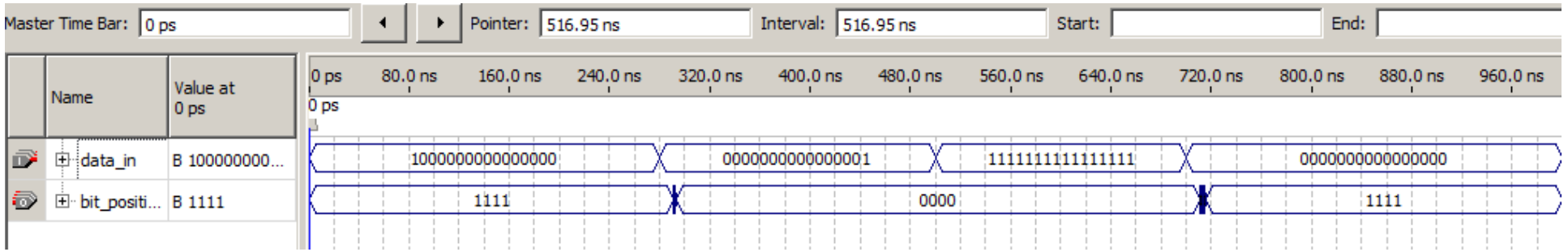
Flow Summary	
Flow Status	Successful - Wed Oct 12 11:46:42 2011
Quartus II 64-Bit Version	11.0 Build 208 07/03/2011 SP 1 SJ Full Version
Revision Name	NumberOfOnes_for
Top-level Entity Name	NumberOfOnes_for
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	9 / 33,216 (< 1 %)
Total combinational functions	9 / 33,216 (< 1 %)
Dedicated logic registers	0 / 33,216 (0 %)
Total registers	0
Total pins	10 / 475 (2 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Note: 9 Logic Elements

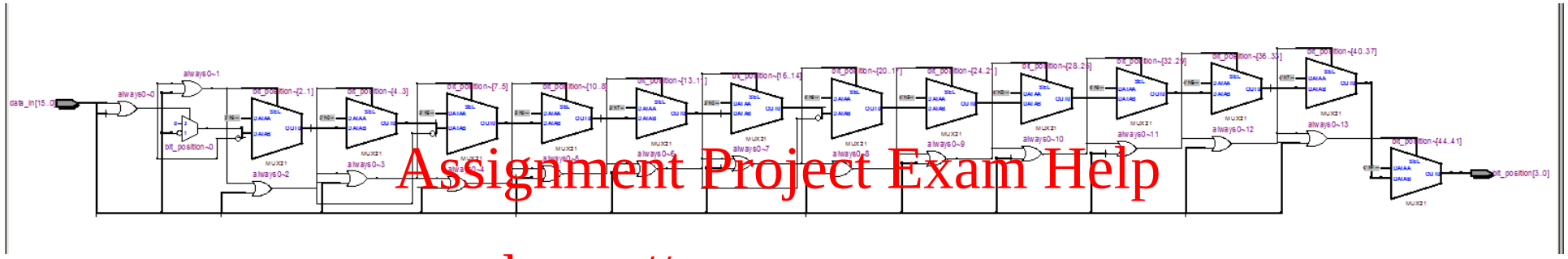
Disable Statement
Example: Position of the First '1'

The *disable* statement is used to prematurely terminate a named block of procedural statements.

```
1   module find_first_one( bit_position, data_in);
2
3       output    [3:0]   bit_position; // four bits will identify
4       input     [15:0]    data_in;        // the position
5
6       reg       [3:0]   bit_position;
7
8       always @ (data_in)                  // rerun when input data changes
9          begin: search_for_1              // named Block
10            for (bit_position = 0; bit_position < 15; bit_position = bit_position + 1)
11               if (data_in[bit_position]==1)
12                  disable search_for_1;
13         end
14  endmodule
```

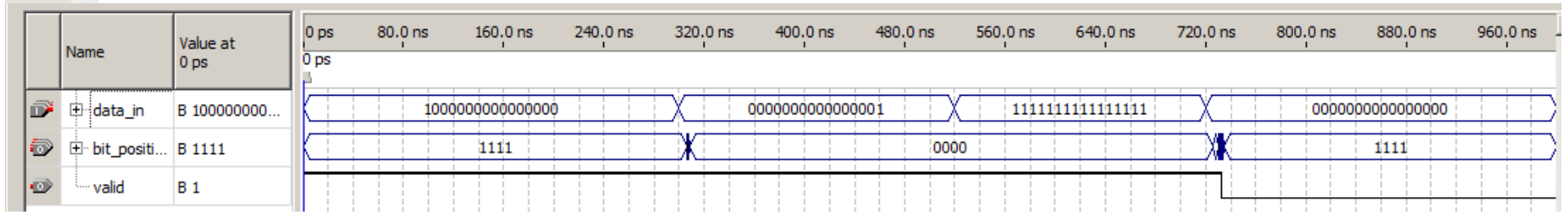
Synthesised Circuit



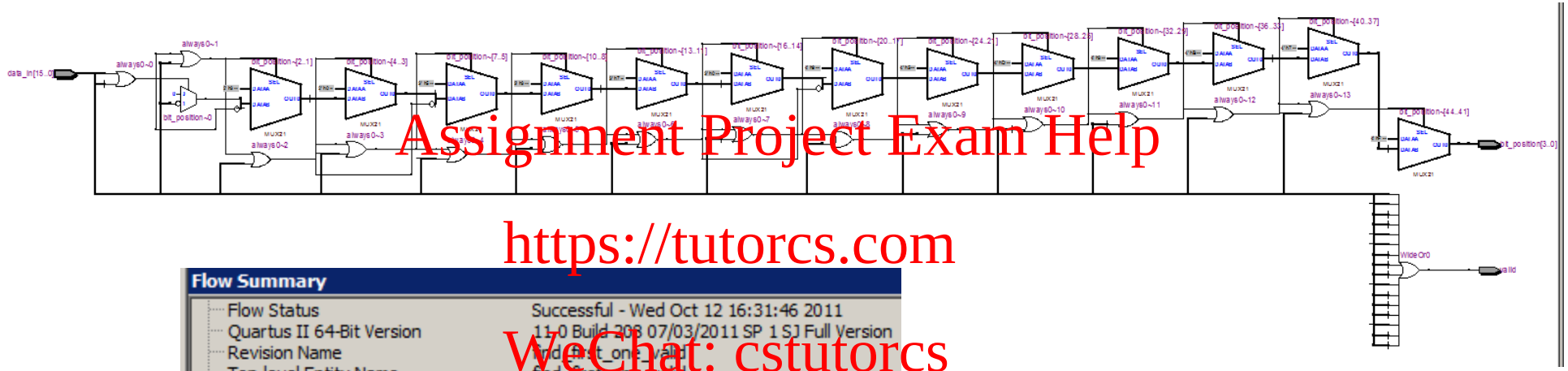
Flow Summary	
Flow Status	Successful - Wed Oct 12 16:17:37 2011
Quartus II 64-Bit Version	11.0 Build 208 07/03/2011 SP 1 SJ Full Version
Revision Name	find_first_one
Top-level Entity Name	find_first_one
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	14 / 33,216 (< 1 %)
Total combinational functions	14 / 33,216 (< 1 %)
Dedicated logic registers	0 / 33,216 (0 %)
Total registers	0
Total pins	20 / 475 (4 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Position of the First ‘1’ with Valid

```
1   module find_first_one_valid( bit_position, valid, data_in);
2
3       output    [3:0]    bit_position;  // four bits will identify
4       input     [15:0]   data_in;       // the position
5       output             valid;
6
7       reg       [3:0]    bit_position;
8
9       assign    valid = |data_in;
10
11      always @ (data_in)                // rerun when input data changes
12         begin: search_for_1            // named Block
13            for (bit_position = 0; bit_position < 15; bit_position = bit_position + 1)
14               if (data_in[bit_position]==1)
15                  disable search_for_1;
16         end
17  endmodule
```

Synthesised Circuit



Flow Summary

Flow Status	Successful - Wed Oct 12 16:31:46 2011
Quartus II 64-Bit Version	11.0 Build 208 07/03/2011 SP 1 SJ Full Version
Revision Name	find_first_one_valid
Top-level Entity Name	find_first_one_valid
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	17 / 33,216 (< 1 %)
Total combinational functions	17 / 33,216 (< 1 %)
Dedicated logic registers	0 / 33,216 (0 %)
Total registers	0
Total pins	21 / 475 (4 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)