KYUNGPOOK NATIONAL UNIVERSITY

# COMP311-1 Logic Circuit Design Chap. 9

Instructor: Taigon Song (송대건)

2019 Fall

# 9.1 Procedural Continuous Assignments

## *9.1.1 assign and deassign (Don't use it anymore)*

- Procedural continuous assignments
  - Procedural statements which allow values of expressions to be driven continuously onto registers or nets for limited periods of time
- assign and deassign
  - Left-hand side to be register or a concatenation of registers only
  - Override the effect of regular procedural assignments

```
module edge_dff(q, qbar, d, clk, reset);
  output q,qbar;
  input d, clk, reset;
  reg q, qbar;

  always @(negedge clk)
    begin
      q = d;
      qbar = ~d;
  end
```

```
always@(negedge clk, posedge rst)
      if(rst)
            q<=0;
      else
            q<=d;
```

이런식으로 짜면되기때문에 assign문 안쓴다.

```
always @(reset)
  if(reset)
  begin
    assign q = 1'b0;
    assign qbar = 1'b1;
  end
  else
  begin
      deassign q;
      deassign qbar;
  end

endmodule
```

# 9.1 Procedural Continuous Assignments

## *9.1.2 force and release*

강제로 값을 넣어줄때 사용

- force and release
  - **Override** assignments on both registers and nets
  - Suggested to be used only for testbenches, or for debugging purposes

```
module stimulus;
...
...
//instantiate the d-flipflop
edge_dff dff(Q, Qbar, D, CLK, RESET);
...
...

initial
begin
    //these statements force value of 1 on dff.q between time 50 and
     //100, regardless of the actual output of the edge_dff.
    #50 force dff.q = 1'b1; //force value of q to 1 at time 50.
    #50 release dff.q;  //release the value of q at time 100.
end
...
...
endmodule
```

```
module top;
...
...
assign out = a & b & c;
...
initial
   #50 force out = a | b & c;
   #50 release out;
end
...
...
endmodule
```

# 9.2 Overriding Parameters

## *9.2.1 defparam Statement*

추천하지 않는 방법

- 'defparam' changes the parameter values
  - Not recommended anymore

탑오듈에서 파라미터 값 설정하는게 defparam인데 이제는 안쓰는 방법이다

**Example 9-2 Defparam Statement**

```
//Define a module hello_world
module hello_world;
parameter id_num = 0; //define a module identification number = 0

initial //display the module identification number
        $display("Displaying hello_world id number = %d", id_num);
endmodule


//define top-level module
module top;
//change parameter values in the instantiated modules
//Use defparam statement
defparam w1.id_num = 1, w2.id_num = 2;

//instantiate two hello_world modules
hello_world w1();
hello_world w2();

endmodule
```

  - Note, what is 'localparam'? (see chap.3)

# Data Types (Chap. 3)

*Integer, real, time, and register data types*

- Parameters
  - Similar as parameter definition in C/C++
  - Cannot be used as variables
  - Allows module instance customization
  - Discussed more on Chap. 9.

```
parameter port_id = 5; // Defines a constant port_id
parameter cache_line_width = 256; // Constant defines width of cache
line
parameter signed [15:0] WIDTH; // Fixed sign and range for parameter
                               // WIDTH
```

  - Local parameters
    - Used with localparam keyword
    - E.g., state encoding for a state machine
      - State encoding cannot be changed.

```
localparam state1 = 4'b0001,
           state2 = 4'b0010,
           state3 = 4'b0100,
           state4 = 4'b1000;
```

# 9.2 Overriding Parameters

## *9.2.2 Module_Instance Parameter Values*

- ANSI C style parameter declaration

**Example 9-3 ANSI C Style Parameter Declaration**

모듈 선언시 #(파라미터 선언) 해주고

```
//Define a module hello_world
module hello_world #(parameter id_num = 0) ;//ANSI C Style Parameter

initial //display the module identification number
        $display("Displaying hello_world id number = %d", id_num);
endmodule
```

```
// Verilog-2001
module adder
   #(parameter MSB=7, LSB=0)
   (output reg [MSB:LSB] sum;
    output reg co;
    input wire [MSB:LSB]a,b;
    input wire ci);
```

- This is also possible for parameter overriding

```
//define top-level module
module top;

//instantiate two hello_world modules; pass new parameter values
//Parameter value assignment by ordered list
hello_world #(1) w1; //pass value 1 to module w1

//Parameter value assignment by name
hello_world #(.id_num(2)) w2; //pass value 2 to id_num parameter
                              //for module w2

endmodule
```

모듈 생성할때 값을 넘겨준다.
예시
module mux21
#(parameter bits = 17)
input[bits - 1:0] a,b;
output[bits - 1:0] out;
이런식으로 선언

Source: Verilog를 이용한 효율적인 하드웨어 설계 Tip, IDEC

# 9.2 Overriding Parameters

## *9.2.2 Module_Instance Parameter Values*

- When having many parameters 사용예시

**Example 9-4 Module Instance Parameter Values**

```
//define module with delays
module bus_master;
parameter delay1 = 2;
parameter delay2 = 3;
parameter delay3 = 7;
...
<module internals>
...
endmodule

//top-level module; instantiates two bus_master modules
module top;

//Instantiate the modules with new delay values

//Parameter value assignment by ordered list
bus_master #(4, 5, 6) b1(); //b1: delay1 = 4, delay2 = 5, delay3 = 6
bus_master #(9, 4) b2(); //b2: delay1 = 9, delay2 = 4, delay3 =
7(default)

//Parameter value assignment by name
bus_master #(.delay2(4), delay3(7)) b3(); //b2: delay2 = 4, delay3 = 7
                                          //delay1=2 (default)
// It is recommended to use the parameter value assignment by name
// This minimizes the chance of error and parameters can be added
// or deleted without worrying about the order.

endmodule
```

# 9.3 Conditional compilation and Execution

## *9.3.1 Conditional Compilation*

- `ifdef, `ifndef, `else, `elsif, `endif

`define TEST 4'b1101

→ Macro (≈ global parameter)

모듈간 충돌을 막기 위해 사용
선언되있으면 그냥 사용
선언안되있으면 선언

**Example 9-5 Conditional Compilation**

```
//Conditional Compilation
//Example 1
'ifdef TEST //compile module test only if text macro TEST is defined
module test;
...
...
endmodule
'else //compile the module stimulus as default
module stimulus;
...
...
endmodule
'endif //completion of 'ifdef directive

//Example 2
module top;

bus_master b1(); //instantiate module unconditionally
'ifdef ADD_B2
   bus_master b2(); //b2 is instantiated conditionally if text macro
                    //ADD_B2 is defined
'elsif ADD_B3
   bus_master b3(); //b3 is instantiated conditionally if text macro
                    //ADD_B3 is defined
'else
   bus_master b4(); //b4 is instantiate by default
'endif

'ifndef IGNORE_B5
   bus_master b5(); //b5 is instantiated conditionally if text macro
                    //IGNORE_B5 is not defined
'endif
endmodule
```

# 9.3 Conditional compilation and Execution

## *9.3.2 Conditional Execution* SKIP

- This can be done using '$test$plusargs' and '$value$plusargs'
  - Can skip the compile process
  - Used only for behavioral statements

```
//Conditional execution
module test;
...
...
initial
begin
   if($test$plusargs("DISPLAY_VAR"))
       $display("Display = %b ", {a,b,c} ); //display only if flag is
set
   else
//Conditional execution
       $display("No Display"); //otherwise no display
end
endmodule
```

→ Run simulator with "+DISPLAY_VAR"

# 9.3 Conditional compilation and Execution

## *9.3.2 Conditional Execution* SKIP

- Using '$test$plusargs' and '$value$plusargs'

**Example 9-7 Conditional Execution with $value$plusargs**

```
//Conditional execution with $value$plusargs
module test;
reg [8*128-1:0] test_string;
integer clk_period;
...
...
initial
begin
   if($value$plusargs("testname=%s", test_string))
       $readmemh(test_string, vectors); //Read test vectors
   else
       //otherwise display error message
       $display("Test name option not specified");

   if($value$plusargs("clk_t=%d", clk_period))
       forever #(clk_period/2) clk = ~clk; //Set up clock
   else
       //otherwise display error message
       $display("Clock period option name not specified");

end

//For example, to invoke the above options invoke simulator with
//+testname=test1.vec +clk_t=10
//Test name = "test1.vec" and clk_period = 10
endmodule
```

# 9.4 Time Scales

- Usage: `timescale <reference_time_unit> / <time_precision>

**Example 9-8 Time Scales**

```
//Define a time scale for the module dummy1
//Reference time unit is 100 nanoseconds and precision is 1 ns
`timescale 100 ns / 1 ns

module dummy1;

reg toggle;

//initialize toggle
initial
  toggle = 1'b0;

//Flip the toggle register every 5 time units
//In this module 5 time units = 500 ns = .5 µs
always #5
    begin
        toggle = ~toggle;
        $display("%d , In %m toggle = %b ", $time, toggle);
    end

endmodule
```

```
//Define a time scale for the module dummy2
//Reference time unit is 1 microsecond and precision is 10 ns
`timescale 1 us / 10 ns

module dummy2;

reg toggle;

//initialize toggle
initial
   toggle = 1'b0;

 //Flip the toggle register every 5 time units
 //In this module 5 time units = 5 µs  = 5000 ns
 always #5
     begin
         toggle = ~toggle;
         $display("%d , In %m toggle = %b ", $time, toggle);
     end

 endmodule
```

# 9.5 Useful System Tasks

## *9.5.1 File Output*

- Opening a file
  - Usage: <file_handle> = $fopen("<name_of_file>");

**Example 9-9 File Descriptors**

```
//Multichannel descriptor
integer handle1, handle2, handle3; //integers are 32-bit values

//standard output is open; descriptor = 32'h0000_0001 (bit 0 set)
initial
begin
   handle1 = $fopen("file1.out"); //handle1 = 32'h0000_0002 (bit 1 set)
   handle2 = $fopen("file2.out"); //handle2 = 32'h0000_0004 (bit 2 set)
   handle3 = $fopen("file3.out"); //handle3 = 32'h0000_0008 (bit 3 set)
end
```

- Writing to files - Usage:
  - $fdisplay(<file_descriptor>, p1, p2 ..., pn); //automatically add new line
  - $fmonitor(<file_descriptor>, p1, p2,..., pn);

```
//All handles defined in Example 9-9
//Writing to files
integer desc1, desc2, desc3; //three file descriptors
initial
begin
    desc1 = handle1 | 1; //bitwise or; desc1 = 32'h0000_0003
    $fdisplay(desc1, "Display 1");//write to files file1.out & stdout

    desc2 = handle2 | handle1; //desc2 = 32'h0000_0006
    $fdisplay(desc2, "Display 2");//write to files file1.out &
file2.out

    desc3 = handle3 ; //desc3 = 32'h0000_0008
    $fdisplay(desc3, "Display 3");//write to file file3.out only
end
```

# 9.5 Useful System Tasks

## *9.5.1 File Output*

- Closing files
  - Usage: $fclose(<file_handle>);

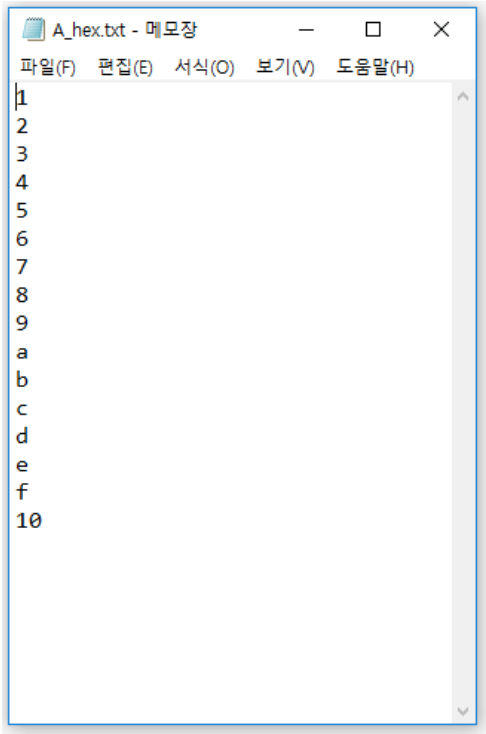
```
//Closing Files
$fclose(handle1);
```

# File I/O Example

## *Reading/writing a file in Verilog*

파이널 프로젝트때 파일 입출력 쓴다
한번 해보기

```
`timescale 1ns / 1ps

module tb();

    reg [7:0] A; //register declaration for storing each line of file.
    integer outfile0,outfile1,outfile2,outfile3; //file descriptors

initial begin

    //The $fopen function opens a file and returns a multi-channel descriptor
    //in the format of an unsized integer.
    outfile0=$fopen("A_hex.txt","r");   //"r" means reading and "w" means writing
    outfile1=$fopen("A_write_dec.txt","w");
    outfile2=$fopen("A_write_bin.txt","w");
    outfile3=$fopen("A_write_hex.txt","w");

    //read the contents of the file A_hex.txt as hexadecimal values into register "A".
    while (! $feof(outfile0)) begin //read until an "end of file" is reached.

        $fscanf(outfile0,"%h\n",A); //scan each line and get the value as an hexadecimal
    //Write the read value into text files.
        $fdisplay(outfile1,"%d",A); //write as decimal
        $fdisplay(outfile2,"%b",A); //write as binary
        $fdisplay(outfile3,"%h",A); //write as hexadecimal
        #10;

    end
    //once reading and writing is finished, close all the files.
    $fclose(outfile0);
    $fclose(outfile1);
    $fclose(outfile2);
    $fclose(outfile3);
    //wait and then stop the simulation.
    #300;
    $stop;
end

endmodule
```




Source:
http://verilogcodes.blogspot.com/2017/11/file-reading-and-writingline-by-line-in.html

# 9.5 Useful System Tasks

## *9.5.2 Displaying Hierarchy*

- Hierarchy at any level can be displayed by using “%m” option in any of the display tasks: $display, $write, $monitor, or $strobe

**Example 9-10 Displaying Hierarchy**

```
//Displaying hierarchy information
module M;
...
initial
    $display("Displaying in %m");
endmodule

//instantiate module M
module top;
...
M  m1();
M  m2();
//Displaying hierarchy information
M  m3();
endmodule
```

이렇게 표시된다

```
Displaying in top.m1
Displaying in top.m2
Displaying in top.m3
```

# 9.5 Useful System Tasks

## *9.5.3 Strobing*

- Very similar to $display
  - When having $display in the same concurrent time, $display may be executed in non-deterministic way
  - When having $strobe in the same concurrent time, it is always executed after all other statements on the same time are executed

    같은 시간에서 다른 모든 동작이 다 된 후 실행시킬때 사용

```
//Strobing
always @(posedge clock)
begin
   a = b;
   c = d;
end

always @(posedge clock)
   $strobe("Displaying a = %b, c = %b", a, c); // display values at
posedge
```

# 9.5 Useful System Tasks

## *9.5.4 Random Number Generation*

- Usage:
  - $random;
  - $random(<seed>);

꼭 한번 해보기!
차이 알기

**Example 9-12 Random Number Generation**

```
//Generate random numbers and apply them to a simple ROM
module test;
integer r_seed;
reg [31:0] addr;//input to ROM
wire [31:0] data;//output from ROM
...
...
ROM rom1(data, addr);

initial
   r_seed = 2; //arbitrarily define the seed as 2.

always @(posedge clock)
   addr = $random(r_seed); //generates random numbers
...
<check output of ROM against expected results>
...
...
endmodule
```

```
reg [23:0] rand1, rand2;
rand1 = $random % 60; //Generates a random number between -59 and 59
rand2 = {$random} % 60; //Addition of concatenation operator to
                        //$random generates a positive value between
                        //0 and 59.
```

# 9.5 Useful System Tasks

## *9.5.5 Initializing Memory from File*

다음주에함

- Usage:
  - $readmemb("<file_name>", <memory_name>);
  - $readmemb("<file_name>", <memory_name>, <start_addr>);
  - $readmemb("<file_name>", <memory_name>, <start_addr>,<finish_addr>);
    - Identical syntax for $readmemh.

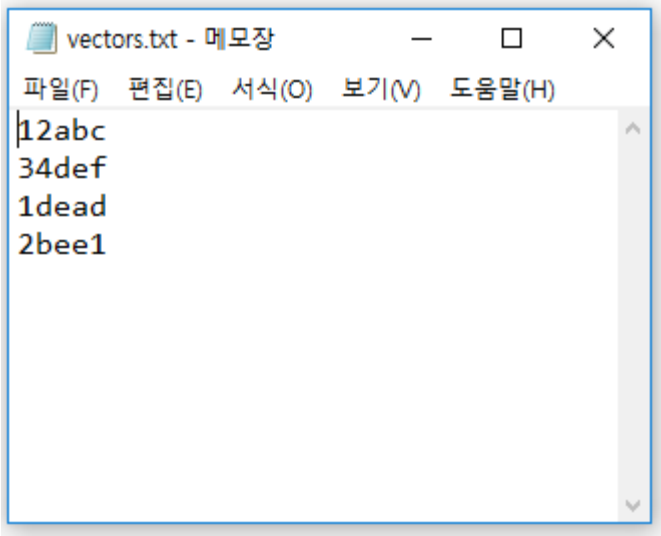
**Example 9-14 Initializing Memory**

```
module test;

reg [7:0] memory[0:7]; //declare an 8-byte memory
integer i;

initial
begin
  //read memory file init.dat. address locations given in memory
  $readmemb("init.dat", memory);
module test;
  //display contents of initialized memory
  for(i=0; i < 8; i = i + 1)
    $display("Memory [%0d] = %b", i, memory[i]);
end

endmodule
```

# Example

*Initialize memory*

```
module file_readmemh;
  /* Declare a array 4 word deep 20 locations wide for 20/4 = 5 hexadecimal words */
  reg [19:0] data [0:3];

  // initialize the hexadecimal reads from the vectors.txt file
  initial $readmemh("vectors.txt", data);

  /* declare an integer for the conditional
  statement to read values from test file */
  integer i;

  /*read and display the values from the text file on screen*/
  initial begin
     $display("rdata:");
     for (i=0; i < 4; i=i+1)
       $display("%d:%b",i,data[i]);
  end

endmodule
```




http://fullchipdesign.com/readmemh.htm

# In-class assignment

*Design a 0-9 decimal counter*

- Directions:
  - Create one 0-15 counter, then think how this could be modified to a 0-9 counter
  - Input/outputs
    - Output [3:0] count
    - Input rst, on, clk