

ECEN 449: Microprocessor System Design
Department of Electrical and Computer Engineering
Texas A&M University

Assignment #1
Solutions

1. Suppose we have the following three designs to implement. Which style (among the choice of custom hardware, ASIC, reconfigurable hardware or software) will be the best style for each design? Assume that the reconfigurable hardware can run at up to 500MHz, while the ASIC can run at 1.5GHz, and the custom hardware can run at up to 4GHz.
 - (a) Design 1 involves a lookup table for an internet protocol (IP) router application. The functionality of this design is that it looks up IP addresses (32 bits) and returns the router interface (4 bits) for this address. Data is fed to this design serially at 1Gbit/second. Design 1 will be used by Cisco, which expects to sell it in high volumes since it will be used in routers for home use. This product must be developed within 9 months.
 - (b) Design 2 also involves a lookup table for an internet protocol (IP) router application. But the difference is that this design will be used in the internet backbone, for routing IP traffic between cities, and will be sold in low volumes. The functionality of this design is that it looks up IP addresses (32 bits) and returns the router interface (12 bits) for this address. Data is fed to this design serially at 10 Gbit/second. This product must be developed within 6 months.
 - (c) Design 3 is an automotive engine controller, which will be sold to GM and Toyota for all their cars. It coordinates all engine functionality, and must operate at a clock frequency of 2GHz. It must be developed in 4 years, for the automobiles developed for model years 2011 through 2020.

Solution. (a) ASIC: The IP addresses are 32 bits wide streaming in at 1 GBit/second serially therefore, the lookup table should be able to operate at 1/32 GHz or 31.25MHz. All design styles can satisfy this speed requirement. However, as this product must be developed within 9 months therefore, custom hardware approach cannot be used. Also, this product is expected to sell in high volumes. Therefore, for lower cost, ASIC is the better choice than reconfigurable hardware.

- (b) Reconfigurable hardware: The design has to lookup 32 bits IP addresses at 1/3.2 GHz or 312.5MHz. All design styles can satisfy this speed requirement however, it will not be cost effective to use ASIC and custom hardware approaches due to low volume manufacturing. Also, it will be difficult to develop this product within 6 months using ASIC and custom hardware approach. Therefore, reconfigurable hardware is the best choice.
- (c) Custom hardware: The design should be able to operate at 2GHz and only custom hardware can satisfy this requirement. The time available to develop this product is also sufficient to use custom hardware approach.

□

2. Consider the Verilog code fragment below:

```
'define BUS_SIZE 64; \\leading char is a backtick
module MYMODULE (A, b, C, d)
input a, B;
output C, d;
wire L1 L2 L3(1:4);
reg [3:4] BUS[1:16];
reg [4:3] NEWBUS[16:1];
integer X, Y;
wor #ARBITRATION_bit, \#Acknowledge_bit;
wand [0:15] VECTOR_SEQUENCE[0:15]; //16 bit vector sequence of width 16
'define BUS_SIZE 32; \\leading char is a backtick
\\ this ends all declarations
\\ for my module
/* Now the statements begin */

L3 = (3 * 2)' HF;
A = 1;
b = 0;
endmodule;
```

Indicate and fix all the syntactical errors in this code.

Solution. The errors are as follows:

- (a) Line 1: Comments start with two forward slashes (//).
- (b) Line 2: Need Semicolon at the end of module line.
- (c) Line 3: Input names are declared using wrong case.

- (d) Line 5: Square brackets are required to declare arrays.
- (e) Line 9: In Verilog, escaped identifiers (identifier whose first character is a backslash (\)) permit non alphanumeric characters in Verilog name. The escaped name includes all the characters following the backslash until the first white space character.
- (f) Line 11-13: Comments start with two forward slashes (//).
- (g) Line 15: An array cannot be assigned continuously. When assigning a wire, *assign* keyword should be used.
- (h) Line 16: When assigning a wire, *assign* keyword should be used.
- (i) Line 17: When assigning a wire, *assign* keyword should be used.
- (j) Line 18: Semicolon is not required after *endmodule*.

The corrected code is below:

```
'define BUS_SIZE 64; //leading char is a backtick
module try (A, b, C, d);
input A, b;
output C, d;
wire L1, L2, L3 [1:4];
reg [3:4] BUS[1:16];
reg [4:3] NEWBUS[16:1];
integer X, Y;
wor \#ARBITRATION_bit , \#Acknowledge_bit ;
wand [0:15] VECTOR_SEQUENCE[0:15]; //16 bit vector sequence of width 16
`define BUS_SIZE 32; // leading char is a backtick
// his ends all declarations
// or my module
/* Now the statements begin */
assign L3[1] = 1'b1;
assign A = 1;
assign b = 0;
endmodule
```

□

3. Write a Verilog module (behavioral) to implement a electronic thermometer. It receives an real numbered input voltage V such that $V = 0.025 \times T$, where T is the temperature in Celsius. The temperature range should be checked to be between 0 and 100 degrees Celsius (both values inclusive). If this range check fails, an output MINUSINFINITY should be driven (if the temperature is below 0 degrees Celsius), or the output PLUSINFINITY should be driven (if it is above 100 degrees Celsius). The design should sample the temperature value whenever there is an event on CLOCK (an input signal). When there is an event on PRINT (another input signal) the Fahrenheit value of the temperature

is calculated and written to the output RESULT (which is a binary number of appropriate length). For example, if the temperature is 15 degrees Fahrenheit, and if the length of RESULT was determined by you to be 5 bits, then the RESULT value would be 01111. Whenever the range check fails, RESULT is assigned all 1's.

Provide comments in your code.

Solution. The Verilog code for the module is as shown below:

```

module MYMODULE(CLOCK,PRINT,MINUSINFINITY,PLUSINFINITY,RESULT);
input CLOCK,PRINT;
output MINUSINFINITY, PLUSINFINITY;
output [7:0] RESULT; // 8 bits are required to express Temperature
reg MINUSINFINITY, PLUSINFINITY; // Need to hold their values
reg [7:0] RESULT; // Need to hold temperature value in Fahrenheit
real V,T; //Voltage can be assumed to be available inside the module
integer F; //Integer variable to hold temperature in Fahrenheit

always @(CLOCK)
begin
T = 40*V;
if ( T < 0 )
MINUSINFINITY = 1'b1;
else if ( T > 100 )
PLUSINFINITY = 1'b1;
end

always @(PRINT)
begin
if( T < 0 | T > 100)
RESULT = 8'b11111111; // Temperature is out of range
else
begin
F = $rtoi(T*1.8+32); //Compute temperature in Fahrenheit and convert it integer
RESULT = F;
end
end
endmodule

```

□