The University of Alabama in Huntsville
ECE Department
CPE 426 01
Final Exam
April 27, 2017

Name: _____

1. (5 points) Draw the transistor-level diagram of a CMOS inverter.

2. (5 points) If the NRE costs for FPGA and ASIC circuits are $50,000 and $3,500,000, respectively, and the cost of individual parts for FPGAand ASIC circuits are $63 and $14, respectively, what is the break-even manufacturing volume for these two types of circuits?

3. (1 point) A _____ provides stimuli and captures responses.

4 (1 point) A ___ _____ is a set of relational expressions that must be true for the chosen value of the variables.

5. (1 point) _____ are pieces of declarative code that check the relationships between design signals, either once or over a period of time.

6. (6 points) For the process given below, A, B, C, and D are all integers that have a value of 0 at time = 10 ns. If E changes from '0' to '1' at time 20 ns, specify all resulting changes. Indicate the time at which each change will occur, the signal/variable affected, and the value to which it will change.

```
process
  variable F : integer := 1; variable A : integer := 0;
begin
  wait on E;
  A := 1;
  F := A + 5;
  B <= E + 3 after 4 ns;
  C <= F + 2;
  D <= A + 5 after 12 ns;
  A := A + 5;
end process;
```

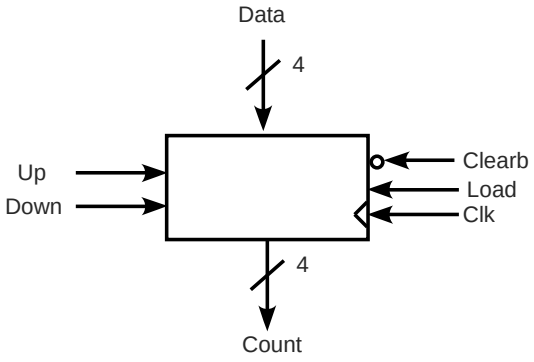
Scheduling Rules	Transport	Inertial
New before existing	Overwrite existing	Overwrite existing
New after existing	Append new	If $v_{new} = v_{existing}$, append new Elsif t_{new}-$t_{existing}$ > reject append new Else overwrite existing

Time	A	B	C	D	E	F
10 ns	0	0	0	0	0	1
20 ns	1	0	0	0	1	1

7. (1 point) _____ are primitives that are all the same height and varying widths.

8. (1 point) A _____ is a signal used in describing the interface of a VHDL model.

9. (15 points) Design a 4 bit counter that has an asynchronous clear input that is active low. It also has active high LOAD, UP and DOWN inputs. When data is loaded into the counter, counting proceeds from that loaded value. If UP and DOWN are both active, the counter counts up. (a) (5 points) Write an entity for the counter. (c) (11 points) Write an architecture for the counter



10. (10 points) Design a 2 to 4 decoder with enable. All outputs are tristated when the enable input = ‘0’. When the enable input = ‘1’, one of the four output D0, D1, D2, D3 is active based on the binary value of the two select inputs S1 and S0. (a) (2 points) Write a VHDL entity. (b) (4 points) Use concurrent signal assignments to implement the architecture. (c) (4 points) Use sequential statements to implement the architecture. Include any necessary library references.

11. (3 points) List three types of coverage used in determining the quality of a test scenario.

_____

_____

_____

12. (15 points) (4 points) Create a VHDL entity named adder_32.(b) (11 points) Create a VHDL architecture representing a structural model of the 32 bit adder using as many adder_8 components as are needed. You do not need to write an architecture for adder_8. Include any necessary library references.

```
entity adder_8 is
  port (a, b : in signed (7 downto 0);
        cin : std_logic;
        f : out signed (7 downto 0);
        cout : out std_logic);
end entity adder_8;
```

13. (1 point) _____ (True or False) All sequential statements are synthesized into sequential circuits.

14. (1 point) _____ (True/False) Operators may be overloaded in VHDL

15. (1 point)) _____ (True/False) Multiple assignments to a signal within a process can cause that signal to have multiple drivers.

16. (1 point). _____(True or False) A D flip-flop and a D latch have the same behavior.

17. (1 point) _____ (True or False) It is possible to make aggregate assignments in VHDL.

18. (10 points An ARM Advanced High-performance Bus (AHB) has the following signals.

Signal	Width	Direction	Description
HCLK	1	Output	Clock
HADDR	21	Output	Address
HWRITE	1	Output	Write flag: 1=write, 0=read
HTRANS	2	Output	Transaction type: 2'b00=IDLE, 2'b10=NONSEQ
HWDATA	8	Output	Write data
HRDATA	8	Input	Read data

Create a class to encapsulate the AHB transactions. In this class constrain:

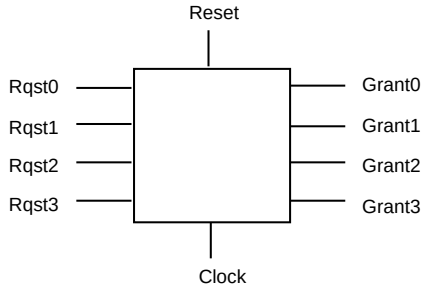
a. The address (HADDR) to be in the lower 5 addresses and upper 5 addresses each with probability 40% and the other addresses with probability 20%.
b. HTRANS to NONSEQ (HTRANS = 2'b10) and IDLE (HTRANS = 2'b00)..
c. All other AHB signals are randomized but unconstrained.

19. (6 points) Given the following constraints, what are the solution probabilities?

```
Class MemTrans;
  Rand bit x;
  Rand bit [1:0] y;
  Constraint c_xy
  {
     y inside {[x:3]};
     solve x before y;
  }
Endclass
```

Solution	x	y	Probability
A	0	0	
B	0	1	
C	0	2	
D	0	3	
E	1	0	
F	1	1	
G	1	2	
H	1	3	

20. (15 points) An arbiter is a circuit that allows at most one subsystem at a time to use a shared resource. A four-way arbiter is shown below. Each subsystem sets its request signal to 1 when it wants to use the resource. When the arbiter sets the grant signal to 1, the subsystem uses the resource. The subsystem sets its request back to 0 when it has finished, and waits for grant to be 0 before starting a subsequent request. While a subsystem is granted use of the resource, other requests must wait, rather than pre-empting the active subsystem. Subsystems are granted requests in order, starting with 0, then 1, 2, 3 and back to 0. A subsystem is skipped if it has no pending request.



As part of creating a SystemVerilog testbench for this device, an interface and a packet for randomization has been created. The requests are being driven by the request device modeled in VHDL below. Each req_dev has an associated test program. The test program is responsible for issuing a signal to the req_dev to reset it and providing a time parameter. After reset, the requesting device raises its request line high and keeps it high for the amount of time provided by the test program. Since req_dev works on the positive edge of the clock, the test program will provide inputs to it on the negative edge of the clock. Complete the test0 program so that it repeats the following process 100 times.

1) waits a random number of clock cycles to set the reset for reqdev to 1.
2) then provides a random number of cycles that reqdev will hold a request high.
3) sets the reset to 0 after one clock cycle
4) waits one more cycle
5) waits for req_dev to lower request to 0
6) waits one more clock cycle

```
interface arbiter_if(input bit clk);
  logic [3:0] REQUEST, GRANT, GEN_REQ;
  logic RESET;
  int REQ_TIME3, REQ_TIME2, REQ_TIME1, REQ_TIME0;
endinterface


module top;
  bit clk;
  always #5ns clk = ~clk;
  arbiter_if arbiterif(clk);
  arbiter u1 (.CLK (clk), .REQ (arbiterif.REQUEST),
              .GRANT (arbiterif.GRANT), RESET (arbiterif.RESET));
  req_dev u2 (.GEN_REQ (arbiterif.GEN_REQ[3]), .REQ_TIME (arbiterif.REQ_TIME3),
              .REQUEST (arbiterif.REQUEST[3]), .GRANT (arbiterif.GRANT[3]),
              .RESET (arbiterif.RESET),.CLK (clk));
  req_dev u3 (.GEN_REQ (arbiterif.GEN_REQ[2]), .REQ_TIME (arbiterif.REQ_TIME2),
              .REQUEST (arbiterif.REQUEST[2]), .GRANT (arbiterif.GRANT[2]),
              .RESET (arbiterif.RESET), .CLK (clk));
  req_dev u4 (.GEN_REQ (arbiterif.GEN_REQ[1]), .REQ_TIME (arbiterif.REQ_TIME1),
              .REQUEST (arbiterif.REQUEST[1]), .GRANT (arbiterif.GRANT[1]),
              .RESET (arbiterif.RESET),.CLK (clk));
  req_dev u5 (.GEN_REQ (arbiterif.GEN_REQ[0]), .REQ_TIME (arbiterif.REQ_TIME0),
```

```
              .REQUEST (arbiterif.REQUEST[0]), .GRANT (arbiterif.GRANT[0]),
              .RESET (arbiterif.RESET), .CLK (clk));
  test3 t1(arbiterif);
  test2 t2(arbiterif);
  test1 t3(arbiterif);
  test0 t4(arbiterif);
  testr t5(arbiterif);
endmodule : top

package mine;
  class Packet_g;
    // The random variables
    rand int generate_t, request_t;
    // Limit the values
    constraint c1 {generate_t < 300; generate_t > 100;
                   request_t < 101; request_t > 0;}
  endclass : Packet_g
endpackage : mine

entity REQ_DEV is
  port(GEN_REQ : in std_logic;
       REQ_TIME : in integer range 0 to 100;
       REQUEST : out std_logic;
       GRANT : in std_logic;
       RESET, CLK : in std_logic);
end REQ_DEV;


import mine::*;
module test0(arbiter_if arbif);









endmodule : test0
```