EE 330 HW II Solutions Spring 2024

Problen ! Av = - 9 mRL = - 3 = M(oxW) VEBRL = 3 So $\frac{UV}{L} = 3$ but $V_{EB} = +1.25 = \frac{3}{10^{4}10^{4}1}$. U = 3 U... Ler L= 1A, W= 2.4 M $\frac{1}{2} = 2.4$

Problem 2 Observe VGT=.8V, IGT-MAX= 200, MA, VFOU=.9V a) 12 - 0.8V > 200 MA = 0 R < 11.2V = 56KD200 AAb) $I_F = 500 - 090 = 1.23 A = 0 P = (1.23A)(.9b) = 1.11bb$ 40.2C) PGATE ~ (.80)(200MA) = 160MW



$$P_{Avg} = \frac{1}{8 \cdot 3m \text{ soc}} \int_{0}^{8 \cdot 3m \text{ soc}} (V_{P})(T_{L}) dt = \frac{1}{8 \cdot 3m \text{ soc}} \int_{0}^{1.61 \text{ mine}} (V_{R} \cdot 0) \delta t + \int_{1.61 \text{ max}}^{9.3 \text{ mod}} V_{R} \frac{1}{60} dt$$

$$= \frac{1}{8 \cdot 3m \text{ soc}} \int_{1.61 \text{ mod}}^{8 \cdot 3m \text{ soc}} (0.90) 4 \sin((2\pi \cdot 60t)) dt = 2.100$$

$$= \frac{3.6}{8 \cdot 3m \text{ soc}} \int_{1.61 \text{ mode}}^{8 \cdot 3m \text{ soc}} \sin((2\pi \cdot 60t)) dt = 2.100$$
C) Quadrants 1 d_{-3}
Problem 4 $V_{CT} = 2V$ want $\frac{R_{1}}{R_{1} + 10K} (100 \sin wt) = 20$
where $w = (2\pi) 60$ and $T_{S} = \frac{1}{60}$
Thus $\frac{R_{1}}{R_{1} + 10K} (100 \sin (2\pi (60)) \frac{1}{(8X60)}) = 2V$

$$\left(\frac{R_{1}}{R_{1} + 10K} (170) \sin((\pi/4)) = 2V$$
Solving obtain $R_{1} = 169.0$
Problem 5
$$\frac{V_{Vot}}{V_{Vot}}$$

$$\frac{V_{Vot}}{T_{0} = 100 \text{ soc}} (1 - \frac{V_{CS}^{2}}{V_{P}})^{2}$$

Vout = ID . R = 1004 . 6KR = 0.6V

Problem 6

$$g_{m} = \frac{\partial I_{D}}{\partial V_{cas}} = -2 \cdot \frac{I_{DSS}^{2}}{-V_{P}} \left(1 - \frac{V_{cas}}{V_{P}} \right) \left(1 - \lambda V_{OS} \right)$$

$$g_{n} = 2 \cdot \frac{I_{DSS}^{2} PO}{V_{P}} \left(\frac{W}{L} \right) \left(1 - \frac{V_{cas}}{V_{P}} \right) = \frac{2}{V_{P}} \frac{I_{DQ}}{V_{P}} \left(1 - \frac{V_{GSQ}}{V_{P}} \right)$$

$$g_{0} = \frac{2ID}{2V_{OS}} = \lambda \cdot I_{PSSPO} \left(\frac{W}{L} \right) \left(1 - \frac{V_{cas}}{V_{P}} \right)^{2}$$

7)

Picklem 7

Observe
$$U_{GSQ} = OV$$

 $I_{DR} = \frac{3G_{N} \cdot 10}{15} \cdot \left(1 - \frac{0}{7}\right)^{2} = 20nA$
 $S_{NR} = \frac{V_{out}}{15} \cdot \left(1 - \frac{0}{7}\right)^{2} = 20nA$
 $I_{DQ} = \frac{V_{out}R - (-5)}{50KR} = 20nA = 5V_{out} = -4V$
 $A_{V} = \frac{V_{out}}{V_{ih}} = -g_{M} \cdot S_{OKR} = \frac{1}{2V}$
 $+g_{M} = \frac{2}{V_{P}} I_{QQ}$

8)

a) This amplifier is unilateral. This can be determined by the lack of a y_{12} parameter, which means that $A_{VR} = 0$. Having an A_{VR} (or A_V with relabeled ports) of 0 is a property of unilateral amplifiers.

b) This model can be developed using diagrams given in lecture slides

Unilateral amplifiers:



- · Thevenin equivalent output port often more standard
- R_{IN}, A_V, and R_{OUT} often used to characterize the two-port of amplifiers



Unilateral amplifier in terms of "amplifier" parameters

$$R_{IN} = \frac{1}{y_{11}}$$
 $A_V = -\frac{y_{21}}{y_{22}}$ $R_{OUT} = \frac{1}{y_{22}}$

From this description, $R_{IN} = \frac{1}{10^{-4} A/V} = 10k\Omega$, $A_V = -\frac{-10A/V}{0.1A/V} = 100$, $A_{VR} = \frac{0}{10^{-4}A/V} = 0$, and $R_{OUT} = \frac{1}{0.1A/V} = 10\Omega$.

c) While we found A_V in the previous step, this is not the actual gain of the given circuit. We'll perform some simple analysis to find the actual gain.



$$\frac{V_{out}}{V_{in}} = 98.5173$$

d) We'lll break these amplifiers up into stages and relate V_{IN} to V_{OUT} based on the intermediary node to find gain. R_{in} and R_{out} won't change for this amplifier



This leaves us with amplifier parameters $R_{in} = 10k\Omega$, $A_v = 9990.0099$, $A_{vr} = 0$, and $R_{out} = 10\Omega$

9&10) Four-Bit Adder Module Code

```
`timescale lns/lps
                                                   //nice simulation timescale
1
 2
 3
    module fourBitAdder(a, b, cin, en, out, cout); //instantiate module
            input cin, en;
 4
                                                   //instantiate 1 bit inputs
 5
             input [3:0] a;
                                                   //instantiate 4 bit inputs
 6
            input [3:0] b;
 7
            output cout;
                                                   //instantiate one bit output
 8
            output [3:0] out;
                                                   //instantiate four bit output
9
            reg [4:0] sum;
                                                   //instantiate a sum register
10
              assign out = sum[3:0];
                                                   //assign output to first four bits
11
             assign cout = sum[4];
                                                  //assign carry-out to last bit
    白白
                                                  //any time an input changes
12
              always 🤅 (*) begin
13
                                                   //if device is enabled
                     if (!en) begin
14
                                                   //calculate sum
                            sum <= a+b+cin;
    15
                     end
16
                     else begin
                                                   //if device is disabled
17
                            sum <= 0;
                                                   //output is zero
18
                     end
19
              end
20 endmodule
```

Four-Bit Adder Testbench Code

<pre>module fourbitAdderTB(); //instantiate testbench reg cin, en; //create inputs reg [3:0] a; reg [3:0] b; vire cout; //create outputs vire [3:0] out; //create outputs fourbitAdder DUT(.a(a),.b(b),.cin(cin),.en(en),.out(out),.cout(cout)); initial clk = 0; initial clk = 0; initial clk = 0; initial en = 0; initial count_up = 1; //toggle clock every time unit initial count_up = 1; //increment do next clock cycle end else begin //if a is not maxed out a <= a + count_up; //increment a incr = 1; //increment a on next clock cycle end else begin //if b is to be incremented if (b == 4'bl000; //reset to zero incr = 0; //increment a on next clock cycle end else begin //if b is not maxed out b = 4'b0000; //reset to zero incr = 0; //increment a on next clock cycle end else begin //if b is not maxed out b <= b + count_up; //increment a incr = 0; //increment b incr = 0;</pre>	1		<pre>`timescale lns/lps</pre>		//nice simulation timescale
<pre>3 module fourbitAdderTB(); //intatatite testbench reg (1:0) a; 6 reg (1:0) b; 7 wire cout; //create outputs 8 wire (1:0) out; //create outputs 9 reg incr, count_up, clk; //create outputs 9 reg incr, count_up, clk; //create outputs 9 initial clk = 0; 11 initial clk = 0; 12 initial incr = 0; 13 initial incr = 0; 14 initial clm = 0; 15 initial incr = 0; 16 initial incr = 0; 17 always # iclk <= -clk; //toggle clock every time unit 18 initial count_up = 1; //create incrementing process 19 always @ (posedge clk) begin //if a is not be incremented 11 if (incr) begin //if a is not maxed out 22 always @ (posedge clk) begin //if a is not maxed out 23 incr = 1; //increment b on next clock cycle 24 end 25 end 26 end 27 end 28 end 29 end 29 end 20 end 20 end 20 end 20 end 20 end 20 end 21 incr = 0; //increment a incr = 1; //increment a 21 incr = 0; //increment a 22 end 23 end 24 end 25 end 26 end 26 end 27 incr = 0; //increment a 28 end 29 end 29 end 20 end 20 end 20 end 20 end 20 end 20 end 21 incr = 0; //increment a 22 end 23 end 24 end 25 end 26 end 26 end 27 incr = 0; //increment a 28 end 29 end 29 end 20 end 20 end 20 end 20 end 20 end 20 end 21 incr = 0; //increment b 22 end 23 end 24 end 25 end 26 end 26 end 27 incr = 0; //increment b 28 end 29 end 20 end 2</pre>	2				
<pre>4 reg cin, en; //create inputs 5 reg (3:0) a; 6 reg (3:0) b; 7 wire cout; //create outputs 9 reg incr, count_up, clk; //create outputs 9 reg incr, count_up, clk; //create outputs 9 reg incr, count_up, clk; //create outputs 10 fourBitAdder DUT(.a(a).b(b).cin(cin).en(en).out(out).cout(cout)); 11 initial clk = 0; 12 initial a = 0; 13 initial b = 0; 14 initial cln = 0; 15 initial en = 0; 16 initial en = 0; 17 always % (posedge clk) begin //ir or ising clock every time unit 18 initial cont_up = 1; //create incrementing process 19 always % (posedge clk) begin //if a is more enterted 10 if (a == 4'b000; //reset to zero 11 if (a == 4'b000; //reset to zero 12 incr = 1; //increment a on next clock cycle 13 end 14 else begin //if b is not maxed out 15 b = 4'b000; //reset to zero 16 incr = 0; //increment a 17 incr = 0; //increment a 18 incr = 0; //increment a 19 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 26 end 27 end 28 end 29 end 29 end 29 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 else begin //if b is not maxed out 25 end 26 end 26 end 27 end 28 end 29 end 29 end 29 end 20 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 26 end 27 end 28 end 29 end 29 end 29 end 20 end 20 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 24 end 25 end 26 end 26 end 27 end 28 end 29 end 29 end 29 end 20 end 20</pre>	3	Ę	<pre>module fourBitAdderTB();</pre>		//instantiate testbench
<pre>5 reg [3:0] a; reg [3:0] b; vire cout; //create outputs vire [3:0] out; //create outputs vire [3:0] out; //create outputs fourBitAdder DUT(.a(a),.b(b),.cin(cin),.en(en),.out(out),.cout(cout)); initial clk = 0; //give variables initial values initial a = 0; initial incr = 0; initial incr = 0; initial en = 0; always @ (posedge clk) begin //if with a clock every time unit initial count_up = 1; //create incrementing process always @ (posedge clk) begin //if a is to be incremented if (incr) begin //if a is not maxed out a = 4'b0000; //reset to zero incr = 1; //increment b on next clock cycle end else begin //if b is to be incremented if (incr) begin //if b is no thaxed out a <= a + count_up; //increment a incr = 1; //increment a on next clock cycle end else begin //if b is not maxed out b = 4'b0000; //reset to zero incr = 0; //increment a on next clock cycle end else begin //if b is not maxed out b = 4'b0000; //reset to zero incr = 0; //increment a on next clock cycle end else begin //if b is not maxed out b = 6 + count_up; //increment a incr = 0; //increment a on next clock cycle end else begin //if b is not maxed out incr = 0; //increment a on next clock cycle end else begin //if b is not maxed out incr = 0; //increment a on next clock cycle end else begin //if b is not maxed out incr = 0; //increment b incr = 0; //increment b</pre>	4		reg cin, en;		//create inputs
<pre>6 reg [3:0] b; 7 wire cout; //create outputs 9 reg incr, count_up, clk; //testbench-specific variables 10 forBitAdder DUT(.4(a),.b(b),.cin(cin),.en(en),.out(out)cout(cout)); 11 initial clk = 0; 12 initial b = 0; 13 initial incr = 0; 14 initial cn = 0; 15 initial en = 0; 16 initial en = 0; 17 always #1 clk <= -clk; //toggle clock every time unit 18 initial count_up = 1; //create incrementing process 19 B always #(posedge clk) begin //if a is to be incremented 11 initial count_up = 1; //ircset to zero 10 incr = 1; //increment b on next clock cycle 17 end 18 end 19 end 20 end 21 end 22 end 23 end 24 end 24 end 25 end 26 end 27 increment b on next clock cycle 28 end 29 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 27 increment a on next clock cycle 28 end 29 end 20 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 27 increment a on next clock cycle 28 end 29 end 29 end 20 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 27 increment a on next clock cycle 28 end 29 end 20 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 27 increment a on next clock cycle 28 end 29 end 20 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 24 end 25 end 26 end 27 increment a on next clock cycle 28 end 29 end 20 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 27 increment a on next clock cycle 28 end 29 end 20 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 27 increment a on next clock cycle 28 end 29 end 20 end 20 end 20 end 20 end 20 end 21 end 21 end 22 end 22 end 23 end 24 end 25 end 26 end 26 end 27 increment a on next clock cycle 28 end 29 end 20 end 21 end 21 end 22 end 23 end 24 end 25 end 26 end 27 end 28 end 29 end 20 end 20 e</pre>	5		reg [3:0] a;		
<pre>7 // // create outputs 8 // create outputs 9 // create outputs 9 // create outputs 9 // create outputs 10 // create outputs 11 // create outputs 12 // create outputs 13 // create outputs 14 // create outputs 15 // create outputs 16 // create outputs 17 // create outputs 18 // create outputs 19 // create outputs 19 // create outputs 10 // create outputs 10 // create outputs 11 // create outputs 12 // create outputs 13 // create outputs 14 // create outputs 15 // create outputs 16 // create outputs 17 // create outputs 18 // create outputs 19 // create outputs 19 // create outputs 19 // create outputs 10 // create outputs 11 // create outputs 12 // create outputs 13 // create outputs 14 // create outputs 15 // create outputs 16 // create outputs 17 // create outputs 18 // create outputs 19 // create outputs 10 // create outputs 10 // create outputs 11 // create outputs 12 // create outputs 13 // create outputs 14 // create outputs 15 // create outputs 16 // create outputs 17 // create outputs 18 // create outputs 19 // create outputs 10 // create outputs 11 // create outputs 12 // create outputs 13 // create outputs 14 // create outputs 15 // create outputs 16 // create outputs 17 // create outputs 18 // create outputs 19 // create outputs 10 // create outputs 11 // create outputs 12 // create outputs 13 // create outputs 14 // create outputs 15 // create outputs 16 // create outputs 17 // create outputs 18 // create outputs 19 // create outputs 10 // create outputs 11 // create outputs 12 // create outputs 13 // create outputs 14 // create outputs 15 // create outputs 16 // create outputs 17 // create outputs 18 // create outputs 19 // create outputs 10 // create outputs 11 // create outputs 12 // create outputs 13 // create outputs 14 // create outputs 14 // create outputs 15 // create outputs 16 // create outputs 17 // create outputs 18 // create outputs 19 // create outputs 10 // create outputs 11 // create outputs 12 // create outputs 14 // create outputs 15 // create outputs 16 // create outputs 17 //</pre>	6		reg [3:0] b;		
<pre>% vire [3:0] out; reg incr. count_up, clk; //testbench-specific variables fourBitAdder DUT(.a(a),.b(b),.cin(cin),.en(en),.out(out),.cout(cout)); initial clk = 0; initial a = 0; initial b = 0; initial incr = 0; initial incr = 0; initial incr = 0; initial count_up = 1; //toggle clock every time unit initial count_up = 1; //toreate incrementing process always % (posedge clk) begin //if a is to be incremented</pre>	7		wire cout;		//create outputs
<pre>9 reg incr, count_up, clk; //testbench-specific variables 10 11 11 11 11 11 11 11 11 11 11 11 11</pre>	8		wire [3:0] out;		
<pre>10 forBitAdder DT(.a(a),.b(b),.cin(cin),.en(en),.out(out),.cout(cout)); 11 initial cin = D; 12 initial a = 0; 13 initial a = 0; 14 initial cin = 0; 15 initial incr = 0; 16 initial en = 0; 17 always # (posedge clk) begin //if a is to be incremented 18 if (incr) begin //if a is not maxed out 20 incr = 1; //increment b on next clock cycle 21 end 22 end 23 end 24 end 25 end 26 end 27 incr = 1; //increment b on next clock cycle 28 end 29 end 29 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 27 incr = 0; //if b is not maxed out 28 end 29 end 29 end 20 end 20 end 20 end 21 end 22 end 23 incr = 0; //if b is not maxed out 24 end 25 end 26 end 27 increment b on next clock cycle 28 end 29 end 29 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 27 increment a on next clock cycle 28 end 29 end 29 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 26 end 27 increment a on next clock cycle 28 end 29 end 29 end 20 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 27 increment a on next clock cycle 28 end 29 end 29 end 20 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 26 end 27 increment a on next clock cycle 28 end 29 end 29 end 20 end 20 end 20 end 20 end 20 end 20 end 21 end 22 end 23 end 24 end 25 end 26 end 26 end 27 increment a on next clock cycle 28 end 29 end 29 end 20 end</pre>	9		reg incr, count u	up, clk;	//testbench-specific variables
<pre>initial clk = 0; //give variables initial values initial a = 0; initial b = 0; initial cin = 0; initial cin = 0; initial en = 0; initial en = 0; initial count_up = 1; //toggle clock every time unit initial count_up = 1; //increment b connext clock cycle incr = 1; //increment b connext clock cycle end else begin //if a is not maxed out a <= a + count_up; //increment a incr = 1; //increment b connext clock cycle end else if (incr) begin //if b is to be incremented if (b == 4 +blull) begin //if b is not maxed out b = 4 +bould); //reset to zero incr = 0; //increment a on next clock cycle end else begin //if b is not maxed out b <= b + count_up; //increment a incr = 0; //increment b end end end end if (a==0 && b==4 +blull) begin //in comespecific input combination cin <= -cin; //toggle carry-in bit end end end end end end end end end end</pre>	10		fourBitAdder DUT	(.a(a),.b(b),.cin(cin),.en(en),	.out(out),.cout(cout));
<pre>12 13 14 14 15 14 14 16 17 16 17 17 17 17 17 18 16 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 17 17 18 18 17 17 18 18 17 17 18 18 17 17 18 18 17 17 18 18 17 17 18 18 17 17 18 18 18 1 18 18 1 18 18 18 1 1 1 1</pre>	11		initial clk = 0;		//give variables initial values
<pre>13 initial b = 0; 14 initial cin = 0; 15 initial incr = 0; 16 initial cont_up = 1; //create incrementing process 17 always # lotk <= -clk; //toggle clock every time unit 18 initial count_up = 1; //create incrementing process 19 cl always @ (posedge clk) begin //if a is to be incremented 20 cl if (lincr) begin //if a is not maxed out 21 cl a = 4'b0000; //reset to zero 22 incr = 1; //increment b on next clock cycle 23 end 24 end 25 end 26 end 27 end 28 end 29 end 30 cl end 30 cl end 31 cl if (b == 4'b1111) begin //if b is to be incremented 31 cl if (b == 4'b1111) begin //if b is to be incremented 32 end 33 end 34 end 35 cl end 35 cl end 36 end 36 end 37 incr = 0; //increment a on next clock cycle 38 end 39 end 30 end 30 end 30 end 31 end 32 end 33 end 34 end 35 end 35 end 36 end 37 incr = 0; //increment a on next clock cycle 38 end 39 end 40 end 40 end 41 end 42 end 42 end 43 end 44 end 44 end 45 end 45 end 46 end 46 end 47 if (a==0 66 b==4'b1111 66 cin) begin //on one specific input combination 41 end 42 end 43 end 44 end 44 end 45 end 45 end 46 end 46 end 47 if (a==0 66 b==4'b1111 66 cin) begin //on one specific input combination 40 end 41 end 41 end 42 end 43 end 44 end 44 end 44 end 45 end 45 end 46 end 46 end 47 increment a on next clock cycle 48 end 49 end 40 end 40 end 40 end 40 end 40 end 41 end 41 end 41 end 41 end 41 end 41 end 42 end 42 end 43 end 44 end 44 end 45 end 45 end 46 end 46 end 46 end 47 end 47 end 47 end 48 end 48 end 49 end 40 end 40 end 40 end 40 end 40 end 40 end 41 e</pre>	12		initial a = 0;		
<pre>14 15 16 17 16 17 17 18 18 19 19 19 19 19 19 19 10 10 11 11 11 11 11 11 11 11 11 11 11</pre>	13		initial b = 0;		
<pre>15 16 17 18 17 18 18 19 19 19 19 19 20 20 20 20 21 21 21 22 21 22 22 23 24 24 25 25 25 26 26 27 27 28 29 29 29 20 20 20 20 20 20 20 20 21 21 22 22 23 24 24 25 2 25 25 26 26 26 27 26 27 27 27 28 27 28 29 29 29 29 29 20 20 20 20 20 21 21 21 22 22 23 2 2 24 2 25 2 25 2 26 27 27 27 28 28 2 29 29 29 20 20 20 20 20 20 21 21 21 22 22 23 2 2 2 2 2 2 2 2 2 2 2</pre>	14		initial cin = 0;		
<pre>16 17 18 18 19 19 19 19 19 19 10 10 10 10 10 10 10 1 1 1 1</pre>	15		initial incr = 0		
<pre>17 17 18 18 19 19 19 19 19 10 10 10 10 10 10 10 10 10 10 10 10 10</pre>	16		initial en = 0;		
<pre>18 19 19 19 19 19 19 19 10 10 10 10 10 10 10 10 10 10 10 10 10</pre>	17		always #1 clk <=	-clk;	//toggle clock every time unit
<pre>19 always @ (posedge clk) begin //on rising clock edge 20 if (lincr) begin //if a is to be incremented 21 if (a == 4'b0000; //if a is maxed out 22 a a = 4'b0000; //reset to zero 23 incr = 1; //increment b on next clock cycle 24 end 25 end 26 end 27 incr = 1; //increment a 27 incr = 1; //increment b on next clock cycle 28 end 29 end 29 end 20 else if (incr) begin //if b is to be incremented 29 incr = 0; //increment a on next clock cycle 20 end 20 else begin //if b is not maxed out 20 incr = 0; //increment a on next clock cycle 31 end 32 end 33 end 44 end 40 end 40 end 40 end 41 if (a==0 66 b==4'b111) begin //increment a on next clock cycle 43 end 44 end 45 end 46 end 46 end 47 //increment a on next clock cycle 48 end 49 end 40 end 40 end 40 end 40 end 40 end 41 if (a==0 66 b==4'b111) begin //on one specific input combination 41 of (a==0 66 b==4'b111 66 cin) begin //on one specific input combination 41 end 43 end 44 end 44 end 45 end 46 end 47 //increment a on next clock cycle 48 end 49 end 40 end 40 end 40 end 40 end 40 end 41 if (a==0 66 b==4'b111 begin //on one specific input combination 41 on cin <= -cin; //toggle carry-in bit 42 end 43 end 44 end 44 end 45 end 46 end 46 end 47 //increment a on pecific input combination 48 end 49 end 40 end 40 end 40 end 40 end 40 end 41 if (a==0 66 b==4'b111 66 cin) begin //on one specific input combination 41 on cin <= -en; //toggle enable</pre>	18		initial count up	= 1;	//create incrementing process
<pre>20</pre>	19	Ē	always @ (posedge	e clk) begin	//on rising clock edge
<pre>21 if (a == 4'bll1) begin //if a is maxed out 22 a</pre>	20	F	if (linc)	r) begin	//if a is to be incremented
<pre>22 23 24 24 25 25 26 26 27 29 26 27 27 28 29 29 29 29 29 20 29 20 20 20 20 20 21 21 22 22 22 21 22 23 24 24 25 25 25 25 25 25 25 25 25 25 25 25 25</pre>	21	F	1	if (a == 4'bllll) begin	//if a is maxed out
<pre>23 23 24 24 25 24 24 25 24 24 25 26 27 27 27 28 27 27 28 29 29 29 29 29 29 29 29 29 29 29 29 29</pre>	22	Т		a = 4'b0000;	//reset to zero
<pre>24 end 25 e end 26 else begin //if a is not maxed out 27 a a <= a + count_up; //increment a 27 a incr = 1; //increment b on next clock cycle 28 end 29 end 29 end 30 else if (incr) begin //if b is to be incremented 31 if (b == 4'bl11) begin //if b is maxed out 32 b a else begin //if b is not maxed out 34 end 35 e end 36 else begin //if b is not maxed out 36 b <= b + count_up; //increment b 37 incr = 0; //increment b 39 end 40 end 41 end 43 end 43 end 44 end 44 end 45 end 45 end 45 end 46 if (a==0 && b==4'bl111 && f(a) begin //on one specific input combination 46 if (a==0 && b==4'bl111 && f(a) begin //on one specific input combination 47 end 48 end 49 end 40 end 40 end 40 end 40 end 41 end 43 end 44 end 44 end 45 end 45 end 46 if (a==0 && b==4'bl111 && f(a) begin //on one specific input combination 46 end 47 end 48 end 49 end 40 end 40 end 40 end 40 end 41 end 41 end 42 end 43 end 44 end 45 end 45 end 46 end 47 end 48 end 49 end 40 end 40 end 40 end 40 end 41 end 41 end 42 end 43 end 44 end 44 end 45 end 45 end 46 end 47 end 48 end 49 end 40 end 40 end 40 end 40 end 40 end 41 end 41 end 41 end 42 end 43 end 44 end 45 end 45 end 46 end 47 end 47 end 48 end 49 end 40 end 40 end 40 end 40 end 40 end 41 end 41 end 41 end 42 end 43 end 44 end 44 end 45 end 46 end 47 end 48 end 49 end 40 end 40 end 40 end 40 end 40 end 41 end 41 end 41 end 41 end 41 end 41 end 41 end 41 end 42 end 43 end 44 end 44 end 45 end 46 end 47 end 48 end 48 end 49 end 40 end 40 end 40 end 40 end 40 end 41 end</pre>	23			incr = 1;	//increment b on next clock cvcle
<pre>25</pre>	24			end	
<pre>26</pre>	25	Ŀ		else begin	//if a is not maxed out
<pre>27 incr = 1; //increment b on next clock cycle 28 end 29 end 30 else if (incr) begin //if b is to be incremented 31 if (b == 4'bl1ll) begin //if b is maxed out 32 incr = 0; //increment a on next clock cycle 33 end 44 end 45 end 46 end 47 end 48 end 49 end 49 end 40 end 40 if (a==0 && b==4'bl1ll) begin //on one specific input combination 41 end 43 end 44 end 45 end 45 end 46 end 47 end 47 end 47 end 48 end 49 end 49 end 40 end 40 end 40 end 40 end 40 end 41 end 41 end 43 end 44 end 45 end 46 end 47 end 47 end 48 end 49 end 49 end 40 end 40 end 40 end 40 end 40 end 41 end 41 end 43 end 44 end 45 end 46 end 47 end 46 end 47 end 48 end 49 end 49 end 40 end 40 end 40 end 40 end 40 end 40 end 41 end 41 end 41 end 42 end 43 end 44 end 44 end 45 end 45 end 46 end 46 end 47 end 47 end 47 end 48 end 49 end 40 end 40 end 40 end 40 end 40 end 41 end 4</pre>	26	Т		a <= a + count up;	//increment a
<pre>end end end else if (incr) begin //if b is to be incremented if (b == 4'b111) begin //if b is maxed out b = 4'b0000; //reset to zero incr = 0; //increment a on next clock cycle end else begin //if b is not maxed out b <= b + count_up; //increment b incr = 0; //increment b incr = 0; //increment a on next clock cycle end end end if (a==0 && b==4'b111) begin //on one specific input combination cin <= -cin; //toggle carry-in bit end if (a==0 && b==4'b111 && cin) begin //on one specific input combination en <= -en; //toggle enable end</pre>	27			incr = 1:	//increment b on next clock cycle
<pre>end else if (incr) begin //if b is to be incremented if (b == 4'blll) begin //if b is maxed out b = 4'b0000; //reset to zero incr = 0; //increment a on next clock cycle end else begin //if b is not maxed out b <= b + count_up; //increment b incr = 0; //increment b incr = 0; //increment a on next clock cycle end end end end if (a==0 && b==4'blll) begin //on one specific input combination cin <= -cin; //toggle carry-in bit end if (a==0 && b==4'blll && cin) begin //on one specific input combination en <= -en; //toggle enable end</pre>	28			end	
<pre>30 = else if (incr) begin //if b is to be incremented 31 = if (b == 4'blll) begin //if b is maxed out 32 b = 4'b0000; //reset to zero 33 incr = 0; //increment a on next clock cycle 44 end 35 = else begin //if b is not maxed out 36 b <= b + count_up; //increment b 37 incr = 0; //increment b 38 end 39 end 40 = if (a==0 && b==4'blll) begin //on one specific input combination 41 cin <= -cin; //toggle carry-in bit 42 end 43 = if (a==0 && b==4'bllll && cin) begin //on one specific input combination 44 end 45 end</pre>	29		end		
<pre>31</pre>	30	Ē	else if	(incr) begin	//if b is to be incremented
<pre>32 b = 4'b000; //reset to zero 33 incr = 0; //increment a on next clock cycle 34 end 35 else begin //if b is not maxed out 36 b <= b + count_up; //increment b 37 incr = 0; //increment b 39 end 40 end 40 end 41 cin (a==0 && b==4'blll) begin //on one specific input combination 41 cin <= -cin; //toggle carry-in bit 42 end 43 end 44 end 45 end 45 end 46 end 47 end 46 end 47 end 46 end 47 end 48 end 49 end 40 en</pre>	31	금		if (b == 4'bllll) begin	//if b is maxed out
<pre>33 33 33 incr = 0; //increment a on next clock cycle 34 35 end 35 else begin //if b is not maxed out 36 37 b <= b + count_up; //increment b 37 and 38 end 39 end 40 end 40 end 41 f (a==0 && b==4'blll) begin //on one specific input combination 41 cin <= -cin; //toggle carry-in bit 42 end 43 end 44 end 45 e</pre>	32	Т		b = 4'b0000;	//reset to zero
<pre>34 end 35 end 36 else begin //if b is not maxed out 36 b <= b + count_up; //increment b 37 incr = 0; //increment b 38 end 39 end 40 end 40 end 41 cin <= -cin; //togle carry-in bit 42 end 43 end 44 end 45 end 45 end 46 b==4'bllll && cin) begin //on one specific input combination 46 end 47 end 48 end 49 end 49 end 40 end 40 end 40 end 40 end 40 end 41 //on one specific input combination 42 end 43 end 44 end 45 end 46 b==4'bllll && cin) begin //on one specific input combination 46 end 47 end 48 end 49 end 40 end 4</pre>	33			incr = 0;	//increment a on next clock cvcle
<pre>35 = else begin //if b is not maxed out 36</pre>	34			end	
36 b <= b + count_up;	35	Ŀ		else begin	//if b is not maxed out
<pre>37 incr = 0; //increment a on next clock cycle 38 end 39 end 40 end 40 end 41 cin <= -cin; //toggle carry-in bit 42 end 43 end 43 end 44 end 45 end 45 end 46 end 47 end 46 end 47 end 46 end 47 end 48 end 49 end 49 end 40 end 40</pre>	36	Т		b <= b + count up;	//increment b
<pre>38 end 39 end 40 fight end 41 cin <= -cin; //toggle carry-in bit 42 end 43 fight end 43 fight end 44 end 45 end 45 end 46 b==4'bllll && cin begin 47 end 48 end 49 end 40 end 40 fight end 41 fight end 41 fight end 41 fight end 41 fight end 41 fight end 41 fight end 42 fight end 43 fight end 44 fight end 44 fight end 44 fight end 45 fight end 46 fight end 47 fight end 48 fight end 49 fight end 40 fight</pre>	37			incr = 0;	//increment a on next clock cvcle
<pre>39 - end 40 = if (a==0 && b==4'blll) begin //on one specific input combination 41 cin <= -cin; //toggle carry-in bit 42 - end 43 = if (a==0 && b==4'blll && cin) begin //on one specific input combination 44 end //toggle enable 45 - end</pre>	38			end	· · · · · · · · · · · · · · · · · · ·
40 if (a==0 && b==4'blll) begin //on one specific input combination 41 cin <= -cin;	39		end		
41 cin <= -cin;	40	Ŀ	if (a==0	<pre>&& b==4'bllll) begin</pre>	//on one specific input combination
<pre>42 - end 43 □ if (a==0 && b==4'bllll && cin) begin //on one specific input combination 44 en <= -en; //toggle enable 45 - end</pre>	41	T		cin <= -cin:	//toggle carry-in bit
43 if (a==0 && b==4'bllll && cin) begin //on one specific input combination 44 en <= -en;	42		end		,,,
44 en <= -en; //toggle enable 45 end	43	F	if (a==0	66 b==4'bllll 66 cin) begin	//on one specific input combination
45 - end	44	Т		en <= -en:	//toggle enable
	45		end		
46 L end	46		end		
47 endmodule	47		endmodule		

Four Bit Adder Waveforms



Proves function by

- Adds A+B+C_{IN}
- Output is sum and COUT
- Output is zero when en is high