

carry_lookahead_adder.sv


```

ucsd > ece111 > HW > Homework5 > Lab5 > carry_lookahead_adder > ≡ carry_lookahead_adder.sv
1  // 'include "fulladder.sv"
2  module carry_lookahead_adder#(parameter N=4)(
3      input logic[N-1:0] A, B,
4      input logic CIN,
5      output logic[N:0] result
6  );
7
8  // Internal signals for generate and propagate terms
9      logic [N-1:0] G; // Generate terms
10     logic [N-1:0] P; // Propagate terms
11     logic [N:0] C; // Carry terms
12
13     // Assign initial carry-in
14     assign C[0] = CIN;
15
16     // Generate and Propagate signals for each bit
17     genvar i;
18     generate
19         for (i = 0; i < N; i = i + 1) begin : gen_generate_propagate
20             assign G[i] = A[i] & B[i]; // Generate term
21             assign P[i] = A[i] ^ B[i]; // Propagate term
22         end
23     endgenerate
24
25     // Carry Lookahead Logic
26     generate
27         for (i = 1; i <= N; i = i + 1) begin : gen_carry
28             assign C[i] = G[i-1] | (P[i-1] & C[i-1]);
29         end
30     endgenerate
31
32     // Sum Calculation
33     generate
34         for (i = 0; i < N; i = i + 1) begin : gen_sum
35             assign result[i] = P[i] ^ C[i];
36         end
37     endgenerate
38
39     // Assign final carry-out as the MSB of result
40     assign result[N] = C[N];
41
42 endmodule : carry_lookahead_adder

```

Type	ID	Message
	332140	No Minimum Pulse width paths to report
	21076	High junction temperature operating condition is not set. Assuming a default value of '100'.
	21076	Low junction temperature operating condition is not set. Assuming a default value of '-40'.
	332142	No user constrained base clocks found in the design. Calling "derive_clocks -period 1.0"
	332096	The command derive_clocks did not find any clocks to derive. No clocks were created or changed.
	332068	No clocks defined in design.
	332154	The derive_clock_uncertainty command did not apply clock uncertainty to any clock-to-clock transfers.
	332102	Design is not fully constrained for setup requirements
	332102	Design is not fully constrained for hold requirements
		Quartus Prime Timing Analyzer was successful. 0 errors, 6 warnings

		Running Quartus Prime EDA Netlist Writer
		Command: quartus_eda --read_settings_files=off --write_settings_files=off carry_lookahead_adder -c carry_lookahead_adder
	18236	Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM_PA
	204019	Generated file carry_lookahead_adder.svo in folder "C:/Users/Ryo Andrew Onozuka/Documents/GitHub/notes/ucsd/ece111/HW/Homewor
		Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
	293000	Quartus Prime Full Compilation was successful. 0 errors, 16 warnings

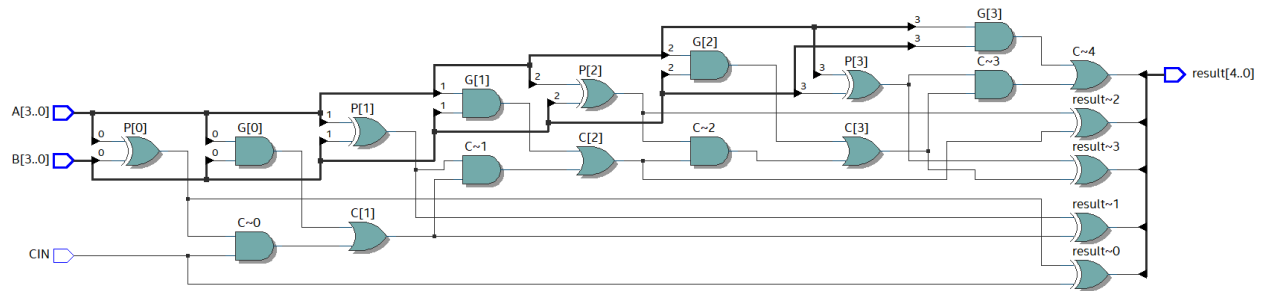
carry_lookahead_adder.sv Resource Usage Summaryucsd > ece111 > HW > Homework5 > Lab5 > carry_lookahead_adder >  carry_lookahead_adder-Resource Usage Summary.rpt

```

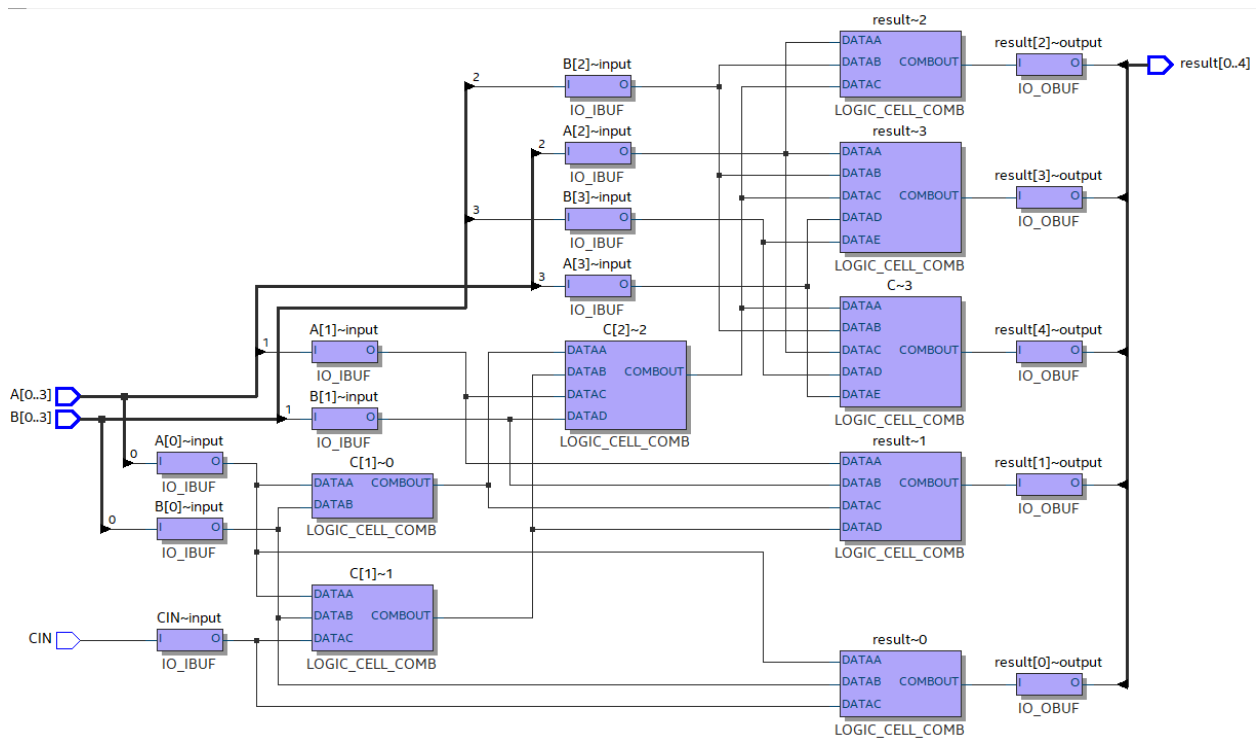
34 +-----+
35 ; Analysis & Synthesis Resource Usage Summary ;
36 +-----+
37 ; Resource ; Usage ;
38 +-----+
39 ; Estimated ALUTs Used ; 8 ;
40 ; -- Combinational ALUTs ; 8 ;
41 ; -- Memory ALUTs ; 0 ;
42 ; -- LUT_REGS ; 0 ;
43 ; Dedicated logic registers ; 0 ;
44 ; ; ;
45 ; Estimated ALUTs Unavailable ; 0 ;
46 ; -- Due to unpartnered combinational logic ; 0 ;
47 ; -- Due to Memory ALUTs ; 0 ;
48 ; ; ;
49 ; Total combinational functions ; 8 ;
50 ; Combinational ALUT usage by number of inputs ; ;
51 ; -- 7 input functions ; 0 ;
52 ; -- 6 input functions ; 0 ;
53 ; -- 5 input functions ; 2 ;
54 ; -- 4 input functions ; 2 ;
55 ; -- <=3 input functions ; 4 ;
56 ; ; ;
57 ; Combinational ALUTs by mode ; ;
58 ; -- normal mode ; 8 ;
59 ; -- extended LUT mode ; 0 ;
60 ; -- arithmetic mode ; 0 ;
61 ; -- shared arithmetic mode ; 0 ;
62 ; ; ;
63 ; Estimated ALUT/register pairs used ; 8 ;
64 ; ; ;
65 ; Total registers ; 0 ;
66 ; -- Dedicated logic registers ; 0 ;
67 ; -- I/O registers ; 0 ;
68 ; -- LUT_REGS ; 0 ;
69 ; ; ;
70 ; ; ;
71 ; I/O pins ; 14 ;
72 ; ; ;
73 ; DSP block 18-bit elements ; 0 ;
74 ; ; ;
75 ; Maximum fan-out node ; C[2]~2 ;
76 ; Maximum fan-out ; 3 ;
77 ; Total fan-out ; 48 ;
78 ; Average fan-out ; 1.33 ;
79 +-----+

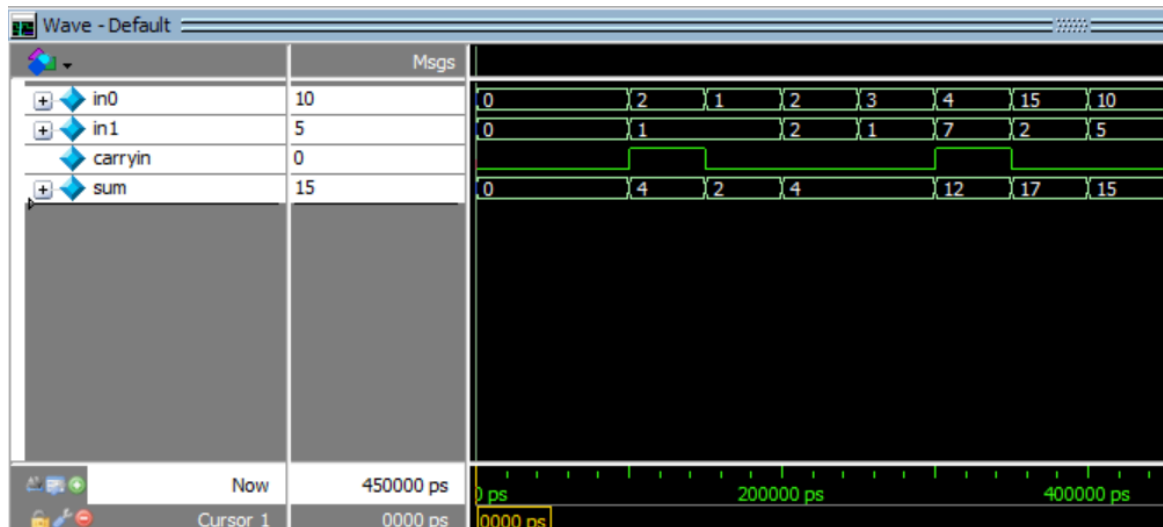
```

carry_lookahead_adder.sv RTL viewer



carry_lookahead_adder.sv post mapping viewer



simulation wavelength results explanation:

```


Transcript
# Errors: 0, Warnings: 0
ModelSim> vsim work.carry_lookahead_adder_testbench
# vsim work.carry_lookahead_adder_testbench
# Start time: 11:33:08 on Nov 06, 2024
# Loading sv_std.std
# Loading work.carry_lookahead_adder_testbench
# Loading work.carry_lookahead_adder
add wave sim:/carry_lookahead_adder_testbench/*
VSIM 7> run -all
# time=0    A= 0    B= 0    CIN=0    result= 0
#
# time=100   A= 2    B= 1    CIN=1    result= 4
#
# time=150   A= 1    B= 1    CIN=0    result= 2
#
# time=200   A= 2    B= 2    CIN=0    result= 4
#
# time=250   A= 3    B= 1    CIN=0    result= 4
#
# time=300   A= 4    B= 7    CIN=1    result=12
#
# time=350   A=15    B= 2    CIN=0    result=17
#
# time=400   A=10    B= 5    CIN=0    result=15
#
VSIM 8>

```

The testbench for the carry lookahead adder initializes inputs A, B, and CIN to various values at different times to verify the functionality of the adder. At each step, it sets specific values for A, B, and CIN and observes the output result, which represents the sum of A, B, and CIN. The testbench uses delays (#50 or #100) between each input change to allow enough time for the computation and for observing results. The simulation output confirms the correct operation of the adder: for instance, when A=2, B=1, and CIN=1, the expected result of 2+1+1=4 is correctly output. Each subsequent test case follows this pattern, where the computed sum aligns with the expected result, such as 3+1=4, 4+7+1=12, and 15+2=17, validating the our carry lookahead adder model's ability to perform addition accurately across various input combinations.

booth_multiplier.sv

```

ucsd > ece111 > HW > Homework5 > Lab5 > booth_multiplier >  booth_multiplier.sv
1  `timescale 1ns/1ps
2  // include "carry_lookahead_adder.sv"
3
4  module booth_multiplier // Module start declaration
5  #(parameter N=4) // Parameter declaration
6  (
7      input clock, reset, start,
8      input logic signed [N-1:0] multiplicand, multiplier,
9      output logic signed [(2*N)-1:0] product,
10     output logic done
11 );
12
13 //Variable to store 2's complement of Multiplicand
14 logic [N:0] multiplicand_neg;
15
16 // Count variable for ADD/SHIFT stages
17 logic [$clog2(N)-1:0] count;
18
19 // Register to store Adder sum and multiplier
20 logic signed [(2*N)+1:0] shift_reg;
21
22 // Register to load multiplicand value
23 logic signed [N:0] load_reg_pos;
24 logic signed [N:0] load_reg_neg;
25
26 // wires to connect with carry lookahead adder
27 logic[N:0] add_operand1, add_operand2;
28 logic[N:0] sum;
29 logic cla_carry;
30
31 // next_state encoding and next_state variable
32 enum logic[2:0]{
33     IDLE           = 3'b000,
34     INITIALIZE     = 3'b001,
35     TEST           = 3'b010,
36     ADD            = 3'b011,
37     SHIFT_AND_COUNT = 3'b100,
38     DONE           = 3'b101
39 } next_state;
40
41 // Instantiate (N+1)-bit carry lookahead adder
42 //Use add_operand1, add_operand2, sum to connect carry lookahead adder
43 //Hint: Carry out from the adder is ignored in our calculations and output sum
44 //has same length as add_operand1 and add_operand2
45 // Tie CIN to '0'
46 carry_lookahead_adder #(N(N+1)) adder_inst(
47     .A(add_operand1),
48     .B(add_operand2),
49     .CIN(1'b0),
50     .result(sum)
51 );
52
53 // Create negative multiplicand value
54 assign multiplicand_neg = ~multiplicand;
55
56
57 // Control FSM for Signed Multiplier
58 // Use Single always block FSM approach
59 // Use *only* non-blocking assignment statements within always block
60 always_ff@(posedge clock, posedge reset) begin
61     if(reset) begin
62         count <= 0;
63         next_state <= IDLE;
64         load_reg_pos <= 0;
65         load_reg_neg <= 0;
66         shift_reg <= 0;
67     end
68     else begin
69         case(next_state)
70             // Wait for start signal
71             IDLE: begin
72                 if (start) begin
73                     next_state <= INITIALIZE;
74                 end
75             end

```

```

76 // Load Multiplicand and Multiplier in a load register and a shift register
77 INITIALIZE: begin
78     // load multiplicand to load_reg_pos
79     load_reg_pos <= (multiplicand[N-1], multiplicand);
80     // load multiplicand_neg[N:0] to load_reg_neg
81     load_reg_neg <= (multiplicand_neg[N-1], multiplicand_neg);
82
83     shift_reg <= {1'b0, {N{1'b0}}}, multiplier, 1'b0;
84     next_state <= TEST;
85     count <= 0;
86 end
87
88 // Check shift register LSB and based on that perform ADD/Shift operation
89 // If last 2 LSB='01' then perform ADD with positive multiplicand followed by Right Shift by 1
90 // If last 2 LSB='10' then perform ADD with negative multiplicand followed by Right Shift by 1
91 // If last 2 LSB='00' then perform Right Shift by 1
92 // If last 2 LSB='11' then perform Right Shift by 1
93 TEST: begin
94     if(shift_reg[1:0] == 2'b01) begin
95         // Pass positive Multiplicand to carry lookahead adder input
96         // Pass previous adder output value after shift to add with Multiplicand
97         // move to add state
98         add_operand1 <= load_reg_pos;
99         add_operand2 <= shift_reg[(2*N):N+1];
100         next_state <= ADD;
101     end
102     else if(shift_reg[1:0] == 2'b10) begin
103         // Pass negative Multiplicand to carry lookahead adder input
104         // Pass previous adder output value after shift to add with Multiplicand
105         // move to add state
106         add_operand1 <= load_reg_neg;
107         add_operand2 <= shift_reg[(2*N):N+1];
108         next_state <= ADD;
109     end
110     else begin
111         // assign add_operand1 to 0, Since no add operation to be perform pass 0 to carry lookadder input
112         // Pass previous adder output value after shift to add with Multiplicand
113         // move to shift and increment count state
114         add_operand1 <= {(N+1){1'b0}};
115         add_operand2 <= shift_reg[(2*N):N+1];
116         next_state <= SHIFT_AND_COUNT;
117     end
118 end
119
120 ADD: begin
121     shift_reg <= (sum, shift_reg[N:0]); // Load shift register : Output sum from Adder which includes carry and retain previous lower bit of shift register
122     // Move to shift and increment count state
123     next_state <= SHIFT_AND_COUNT;
124 end
125
126 SHIFT_AND_COUNT: begin
127     shift_reg <= (shift_reg >> 1); // Right Arithmetic shift entire shift register by 1 position
128     // Increment count
129     count <= count + 1;
130
131     if(count == N-1) begin // If 'N' times SHIFT operation performed then move to Done state else go back to Test state
132         next_state <= DONE;
133     end
134     else begin
135         next_state <= TEST;
136     end
137 end
138
139 DONE: begin
140     next_state <= IDLE; // Wait for right shift value to be available. This is the final product value.
141 end
142 endcase
143 end
144 end
145
146 // Generate done=1 when FSM reaches DONE state
147 assign done = (next_state == DONE) ? 1 : 0;
148
149 // Generate Product in DONE state by loading shift_reg value to it
150 assign product = (next_state == DONE) ? {shift_reg[(2*N)], shift_reg[(2*N):1]} : 0;
151
152 endmodule: booth_multiplier

```

Type	ID	Message
	332140	No Recovery paths to report
	332140	No Removal paths to report
	332146	worst-case minimum pulse width slack is -2.846
		Analyzing Fast 900mV -40C Model
	332123	Deriving Clock Uncertainty. Please refer to report_sdc in the Timing Analyzer to see clock uncertainties.
	332146	worst-case setup slack is 0.296
	332146	worst-case hold slack is 0.113
	332140	No Recovery paths to report
	332140	No Removal paths to report
	332148	Timing requirements not met
	332146	worst-case minimum pulse width slack is -2.846
	21076	High junction temperature operating condition is not set. Assuming a default value of '100'.
	21076	Low junction temperature operating condition is not set. Assuming a default value of '-40'.
	332123	Deriving Clock Uncertainty. Please refer to report_sdc in the Timing Analyzer to see clock uncertainties.
	332102	Design is not fully constrained for hold requirements
	332102	Design is not fully constrained for hold requirements
		Quartus Prime Timing Analyzer was successful. 0 errors, 5 warnings
		Running Quartus Prime EDA Netlist Writer
		Command: quartus_edu --read_settings_files=off --write_settings_files=off booth_multiplier -c booth_multiplier
	18236	Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM_PA
	204019	Generated file booth_multiplier.svo in folder 'C:/Users/Ryo Andrew Onozuka/Documents/GitHub/notes/ucsd/ece111/HW/Homework5/Lal
		Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
	293000	Quartus Prime Full Compilation was successful. 0 errors, 19 warnings

booth_multiplier.sv Resource Usage Summary

ucsd > ece111 > HW > Homework5 > Lab5 > booth_multiplier > booth_multiplier-Resource Usage Summary.rpt

```

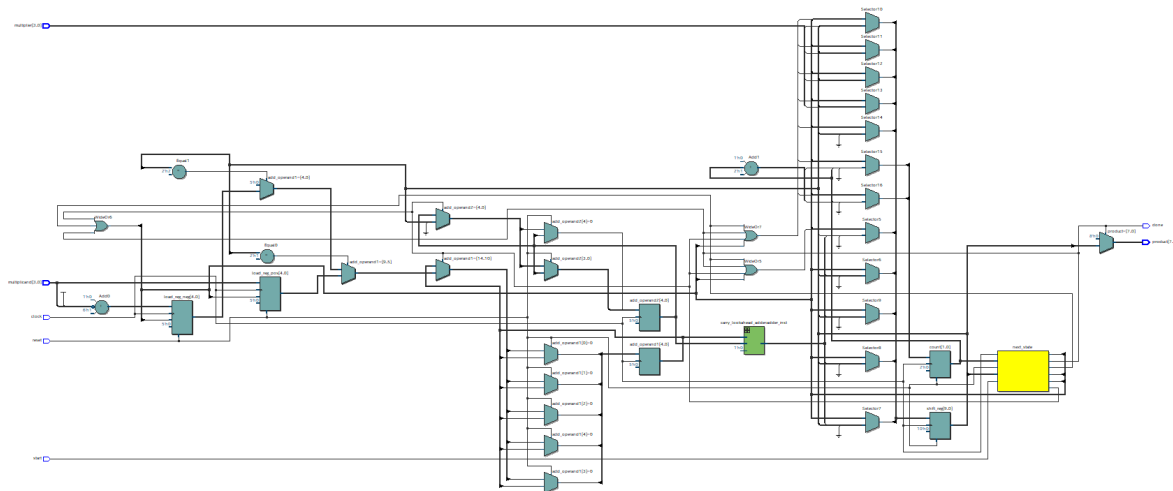
34  +-----+
35  ; Analysis & Synthesis Resource Usage Summary ;
36  +-----+
37  ; Resource ; Usage ;
38  +-----+
39  ; Estimated ALUTs Used ; 43 ;
40  ; -- Combinational ALUTs ; 43 ;
41  ; -- Memory ALUTs ; 0 ;
42  ; -- LUT_REGS ; 0 ;
43  ; Dedicated logic registers ; 35 ;
44  ; ; ;
45  ; Estimated ALUTs Unavailable ; 1 ;
46  ; -- Due to unpartnered combinational logic ; 1 ;
47  ; -- Due to Memory ALUTs ; 0 ;
48  ; ; ;
49  ; Total combinational functions ; 43 ;
50  ; Combinational ALUT usage by number of inputs ; ;
51  ; -- 7 input functions ; 1 ;
52  ; -- 6 input functions ; 2 ;
53  ; -- 5 input functions ; 3 ;
54  ; -- 4 input functions ; 13 ;
55  ; -- <=3 input functions ; 24 ;
56  ; ; ;
57  ; Combinational ALUTs by mode ; ;
58  ; -- normal mode ; 42 ;
59  ; -- extended LUT mode ; 1 ;
60  ; -- arithmetic mode ; 0 ;
61  ; -- shared arithmetic mode ; 0 ;
62  ; ; ;
63  ; Estimated ALUT/register pairs used ; 48 ;
64  ; ; ;
65  ; Total registers ; 35 ;
66  ; -- Dedicated logic registers ; 35 ;
67  ; -- I/O registers ; 0 ;
68  ; -- LUT_REGS ; 0 ;
69  ; ; ;
70  ; ; ;
71  ; I/O pins ; 20 ;
72  ; ; ;
73  ; DSP block 18-bit elements ; 0 ;
74  ; ; ;
75  ; Maximum fan-out node ; clock~input ;
76  ; Maximum fan-out ; 35 ;
77  ; Total fan-out ; 295 ;
78  ; Average fan-out ; 2.50 ;
79  +-----+
80

```

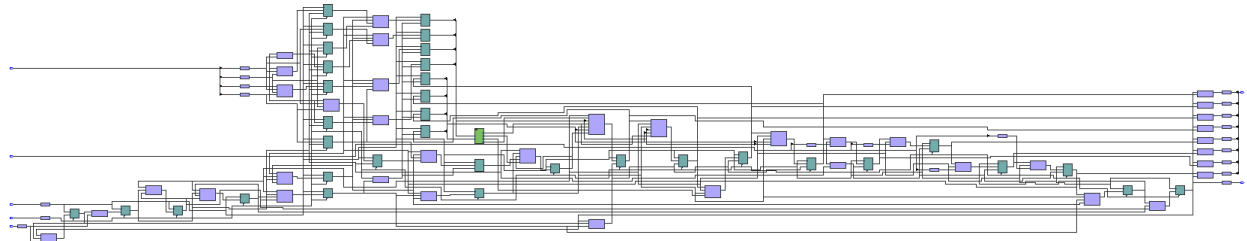
Andrew Onozuka A16760043

ECE 111 HW5 | 11/6/2024

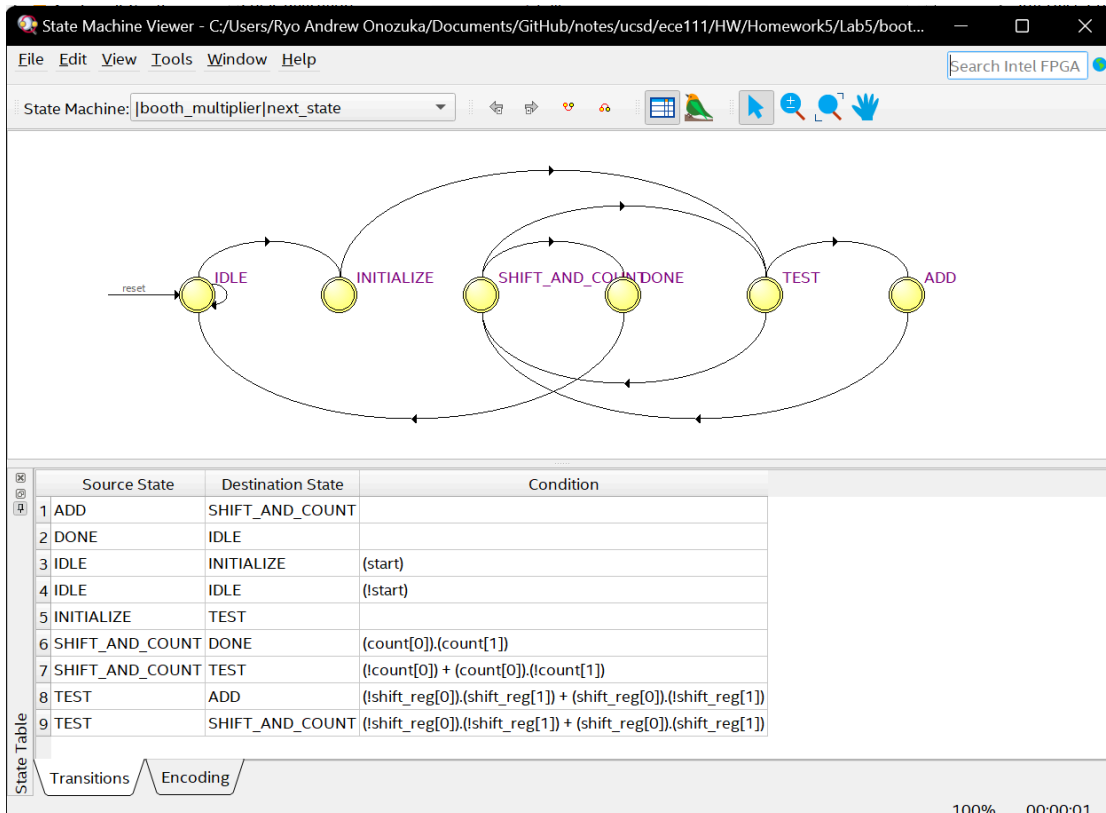
booth_multiplier.sv RTL viewer



booth_multiplier.sv post mapping viewer



booth_multiplier.sv state machine viewer

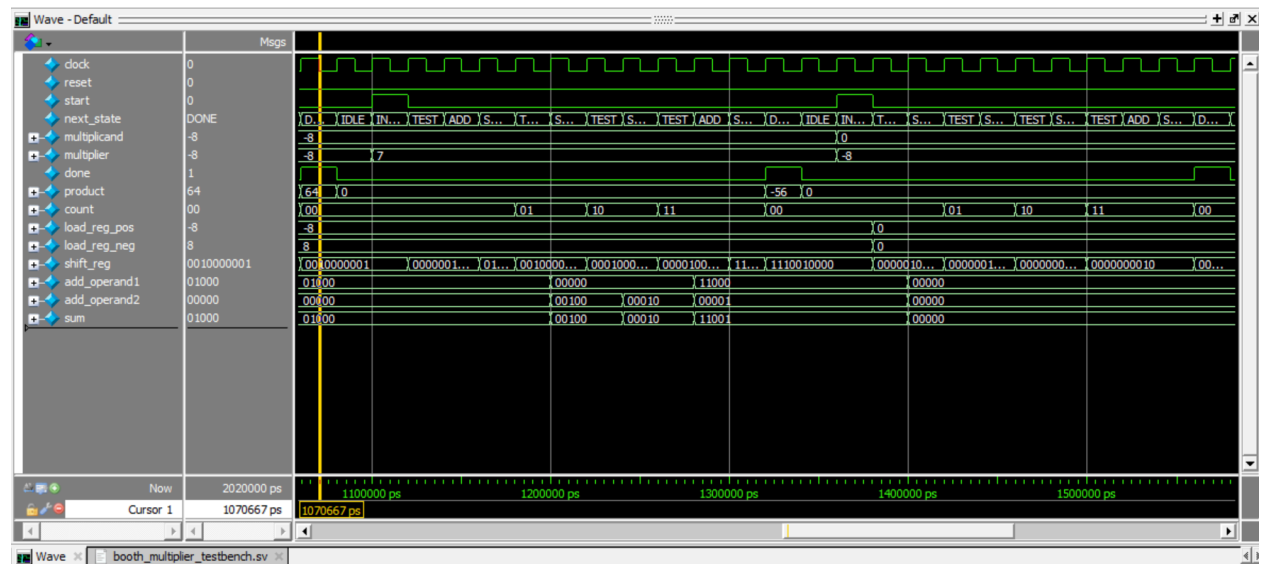


simulation transcript:

```

Transcript
# time=1060000 Multiplicand= -8 Multiplier= -8 Product= 64 Done=1 Correct Result
# time=1320000 Multiplicand= -8 Multiplier= 7 Product= -56 Done=1 Correct Result
# time=1560000 Multiplicand= 0 Multiplier= -8 Product= 0 Done=1 Correct Result
# time=1780000 Multiplicand= -8 Multiplier= 0 Product= 0 Done=1 Correct Result
#
# ** Note: $finish : C:/Users/Ryo Andrew Onozuka/Documents/GitHub/notes/ucsd/ecell1/HW/Homework5/Lab5/booth_multiplier/booth_multiplier_testbench.sv(190)
# Time: 2020 ns Iteration: 0 Instance: /booth_multiplier_testbench
# Break in Module booth_multiplier_testbench at C:/Users/Ryo Andrew Onozuka/Documents/GitHub/notes/ucsd/ecell1/HW/Homework5/Lab5/booth_multiplier/booth_multiplier_testbench.sv line 190
VSIM 5>

```

simulation wavelength results explanation:

This waveform demonstrates the behavior of a Booth multiplier as it processes a sequence of signed multiplication operations. Key signals such as multiplicand, multiplier, product, next_state, and various internal registers (shift_reg, add_operand1, add_operand2, and sum) are displayed, showing the step-by-step computation and state transitions. The next_state signal progresses through different stages (e.g., IDLE, INITIALIZE, TEST, ADD, SHIFT_AND_COUNT, and DONE), indicating the control flow of the Booth multiplication algorithm. Also see the state machine shown above.

For each multiplication operation, multiplicand and multiplier values are loaded, and the computation proceeds as the count variable increments and the shift_reg is updated. Notably, in the first computation (multiplicand = -8, multiplier = -8), the product correctly reaches 64 when next_state enters DONE, confirming the accuracy of the result. Similarly, subsequent operations with different combinations of positive, negative, and zero inputs (multiplicand = -8, multiplier = 7, and others) produce the expected results, as indicated by the correct product values and the final Done=1 assertion, verifying successful completion for each case.

This waveform effectively illustrates how the Booth multiplier module handles control flow and internal arithmetic operations, providing a visual confirmation of its correctness in computing signed products.