Phase 1 mini project

Sequential 8x8 multiplier

Objective:

Build an 8x8 multiplier. The input to the multiplier consists of two 8-bit multiplicands (dataa and datab) and the output from the multiplier is 16-bit product (product8x8_out).

Additional outputs are a done bit (done_flag) and seven signals to drive a 7 segment display (seg_a, seg_b, seg_c, seg_d, seg_e, seg_f & seg_g).

This 8 x 8 multiplier requires four clock cycles to perform the full multiplication. During each cycle, a pair of 4-bit portion of the multiplicands is multiplied by a 4 x 4 multiplier. The multiplication result of these 4-bit slices is then accumulated. At the end of the four cycles (during the 5th cycle), the fully composed 16-bit product can be read at the output. The following equations illustrate the mathematical principles supporting this implementation:

result[15..0] = a[7..0] * b[7..0] =

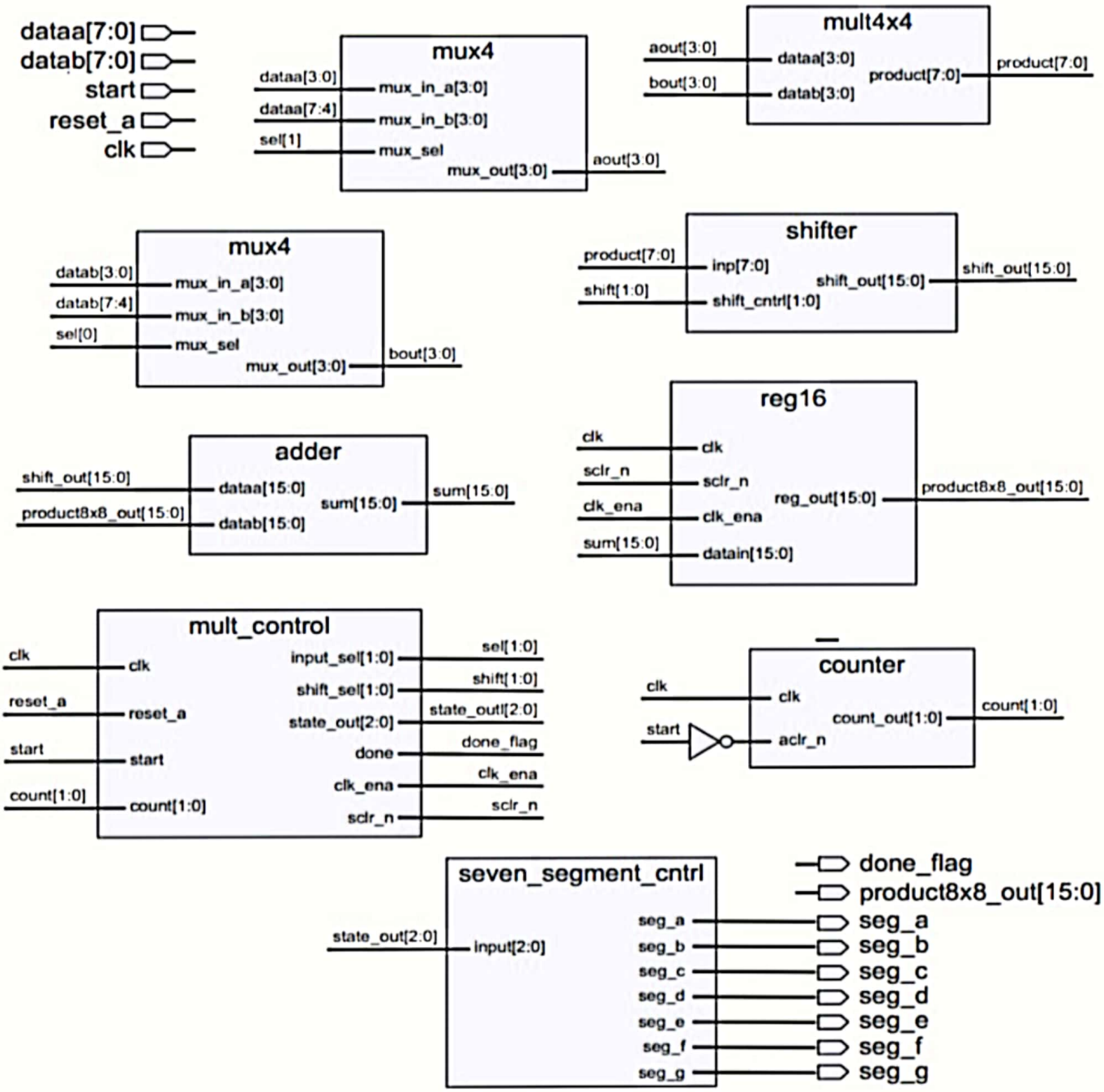
((a[7..4] * 2^4) + a[3..0] * 2^0) * ((b[7..4] * 2^4) + b[3..0] * 2^0) =

((a[7..4] * b[7..4]) * 2^8) +

((a[7..4] * b[3..0]) * 2^4) +

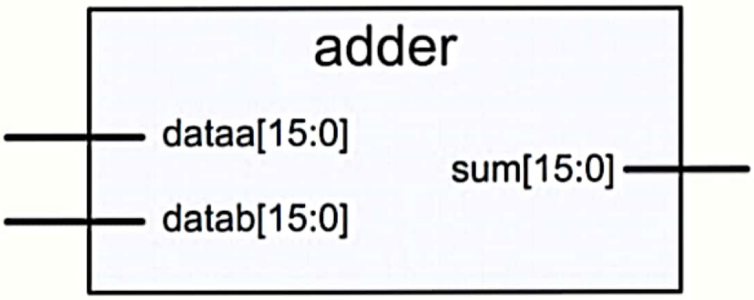
((a[3..0] * b[7..4]) * 2^4)+

((a[3..0] * b[3..0]) * 2^0)



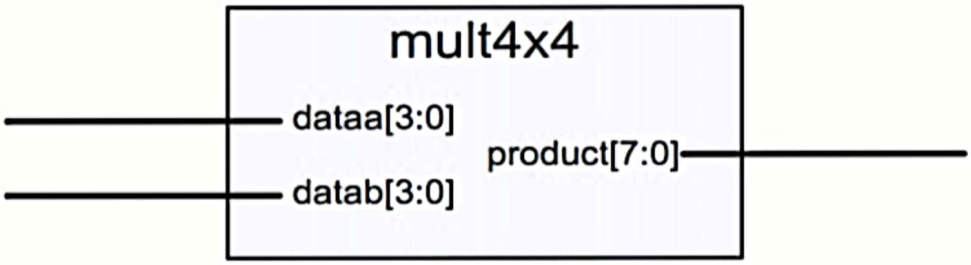
8x8 top level design block diagram

16-bit adder



- Write Verilog code to perform addition.
- Write its testbench.
- Put screenshots of testbench results and circuit schematic
  (hint: to show schematic in Quiestasim, type vsim -debugDB adder.v in Quiestasim transcript window).

4x4 multiplier

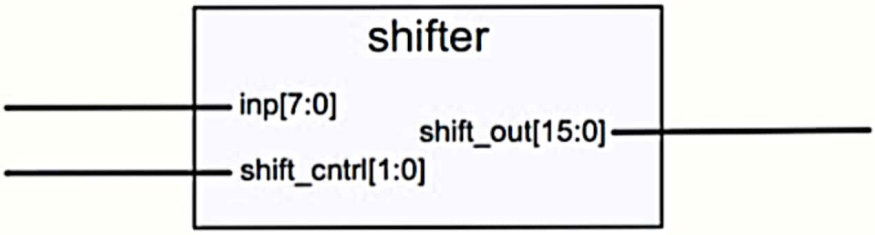


- Write Verilog code to perform multiplication.
- Write its testbench.
- Put screenshots of testbench results and circuit schematic
  (hint: to show schematic in Quiestasim, type vsim -debugDB mult4x4.v in Quiestasim transcript window).

2-1 multiplexer:

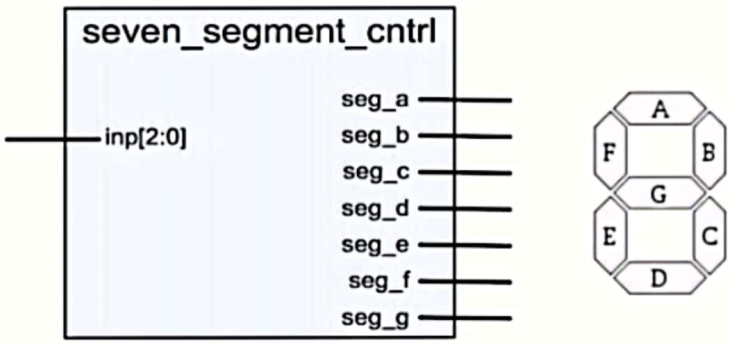
- Write Verilog code to perform 2-1 mux.
- Write its testbench.
- Put screenshots of testbench results and circuit schematic
  (hint: to show schematic in Quiestasim, type vsim -debugDB mux4.v in Quiestasim transcript window).

Shifter:



- Write Verilog code to perform 8-bit to 16-bit left shifter (hint: see equation in page 1)
- If shift_cntrl = 0 or 3, then no shift
- If shift_cntrl =1, then 4-bit shift left
- If shift_cntrl = 2, then 8-bit shift left
- Write its testbench.
- Put screenshots of testbench results and circuit schematic
  (hint: to show schematic in Quiestasim, type vsim -debugDB shifter.v in Quiestasim transcript window).

7-segment display encoder:

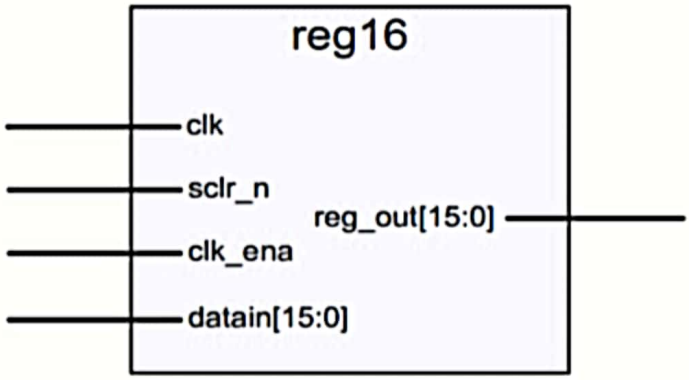


- Write Verilog code to perform encoder as following table:

Inputs	Outputs							LED Display
inp [2:0]	seg_a	seg_b	seg_c	seg_d	seg_e	seg_f	seg_g	
000	1	1	1	1	1	1	0	0
001	0	1	1	0	0	0	0	1
010	1	1	0	1	1	0	1	2
011	1	1	1	1	0	0	1	3
All other values	1	0	0	1	1	1	1	E

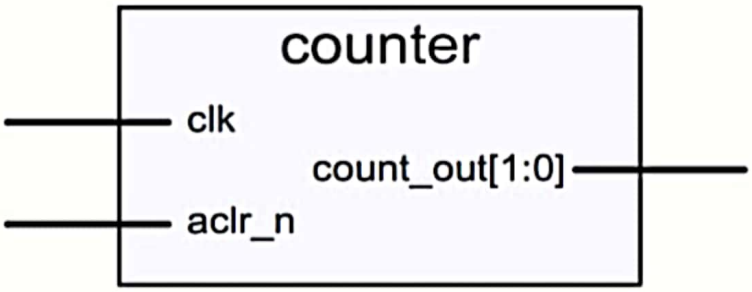
- Write its testbench.
- Put screenshots of testbench results and circuit schematic
  (hint: to show schematic in Quiestasim, type vsim -debugDB seven_segment_cntrl.v in Quiestasim transcript window).

Synchronous 16-bit register:



- Write Verilog code to perform synchronous 16-bit register where sclr_n is a reset signal and clk_ena is a clock enable signal.
- If clk_ena is high and sclr_n is low then the output of register is cleared.
- If clk_ena is high and sclr_n is high then output of register is set equal to its input.
- If clk_ena is low then do nothing.
- Write its testbench.
- Put screenshots of testbench results and circuit schematic
  (hint: to show schematic in Quiestasim, type vsim -debugDB reg16.v in Quiestasim transcript window).

2-bit Counter with asynchronous control:



- Write Verilog code to perform 2-bit counter where aclr_n is a reset signal
- If aclr_n is low, then counter goes to 00 immediately
- If aclr_n is high, output of counter increments by 1 on every rising edge clock.
- Write its testbench.
- Put screenshots of testbench results and circuit schematic