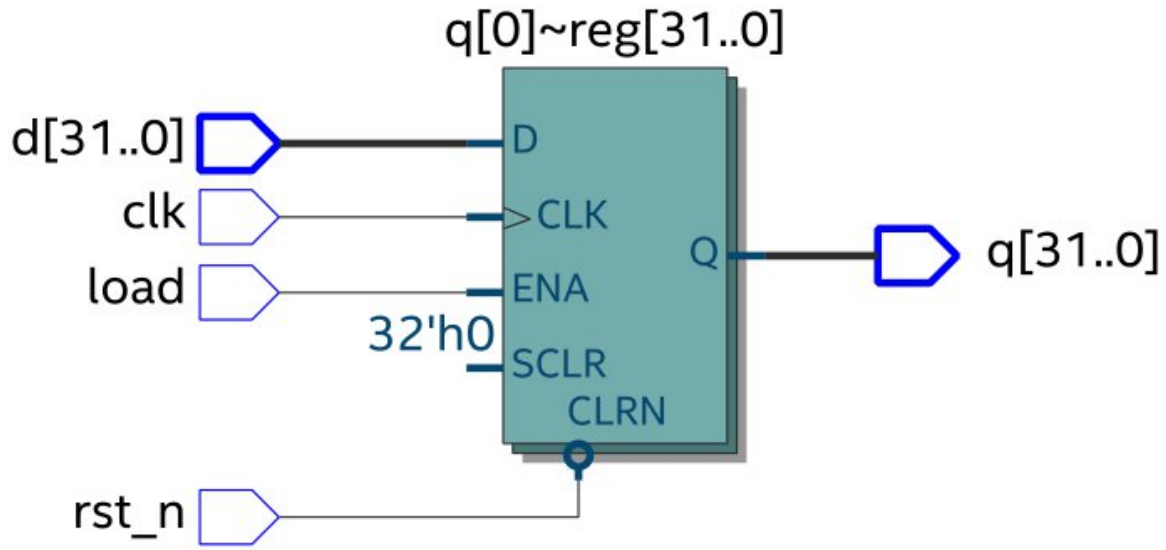
Muhammad Ahmed Qazi
CS-24045

Week #02

Reporting on tasks assigned

Question 01: Implement a 32-bit Register



Code: [reg32.sv]

Test-bench: [reg32_tb.sv]

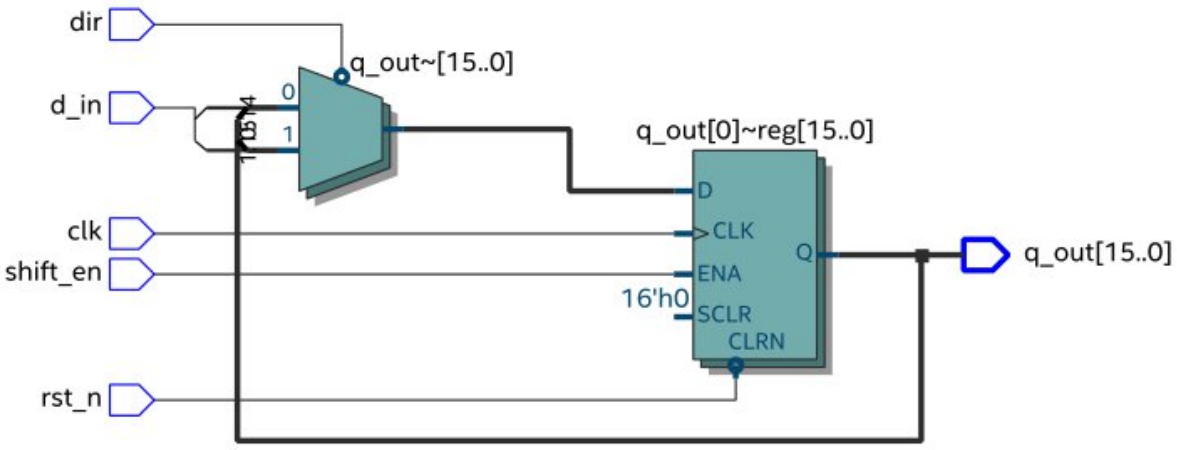
Output:

```
Reset released, q = 00000000 (expected 0)

After load, q = a5a5a5a5 (expected A5A5A5A5)

Hold test, q = a5a5a5a5 (expected A5A5A5A5)

After reset, q = 00000000 (expected 0)

Testbench completed.
```

Waveform: [Reg32 Waveform PDF]

Muhammad Ahmed Qazi
CS-24045

Question 02: Implement a Synchronous Up/Down Counter



Code: [counter.sv]

Test-bench: [counter_tb.sv]

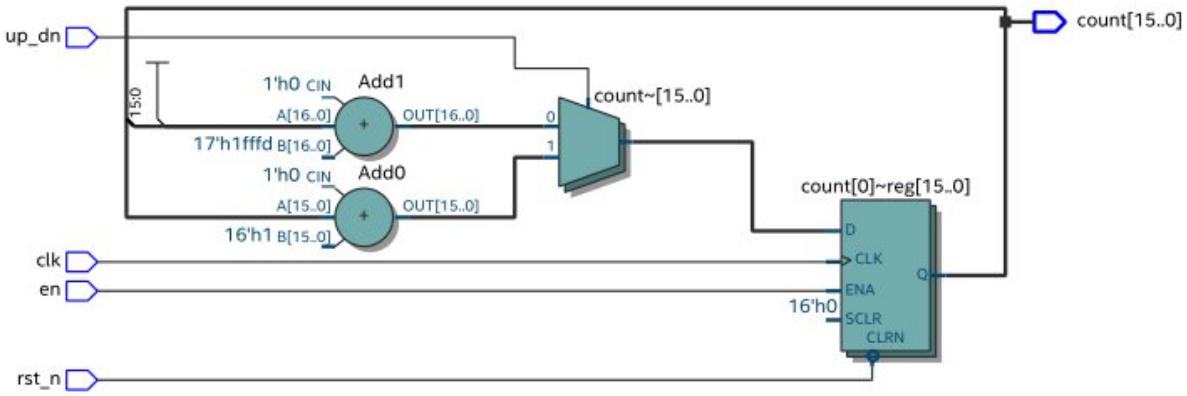
Output:

```
T=5000 | RESET | count=0 (expected 0)

T=15000 | PASS | en=0 up_dn=0 | count=0

T=25000 | PASS | en=1 up_dn=1 | count=0

T=35000 | PASS | en=1 up_dn=1 | count=1

T=45000 | PASS | en=1 up_dn=1 | count=2

T=55000 | PASS | en=1 up_dn=1 | count=3

T=65000 | PASS | en=1 up_dn=1 | count=4

T=75000 | PASS | en=0 up_dn=1 | count=5

T=85000 | PASS | en=0 up_dn=1 | count=5

T=95000 | PASS | en=0 up_dn=1 | count=5

T=105000 | PASS | en=1 up_dn=0 | count=5

T=115000 | PASS | en=1 up_dn=0 | count=4

T=125000 | PASS | en=1 up_dn=0 | count=3

- counter_tb.sv:122: Verilog $finish

T=135000 | PASS | en=1 up_dn=0 | count=2
```

Waveform: [Counter Waveform PDF]

Muhammad Ahmed Qazi
CS-24045

Question 03: Implement a Synchronous Bi-directional Shift Register



Code: [shift_reg.sv]

Test-bench: [shift_reg_tb.sv]

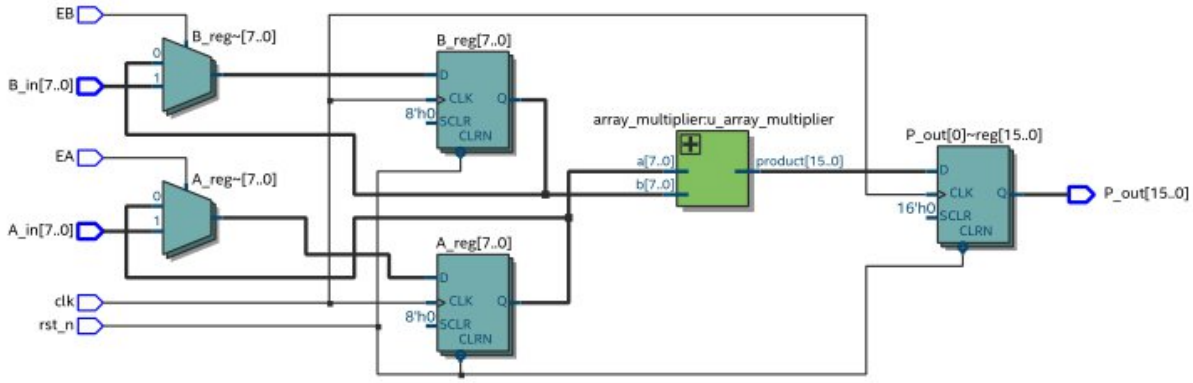
Output:

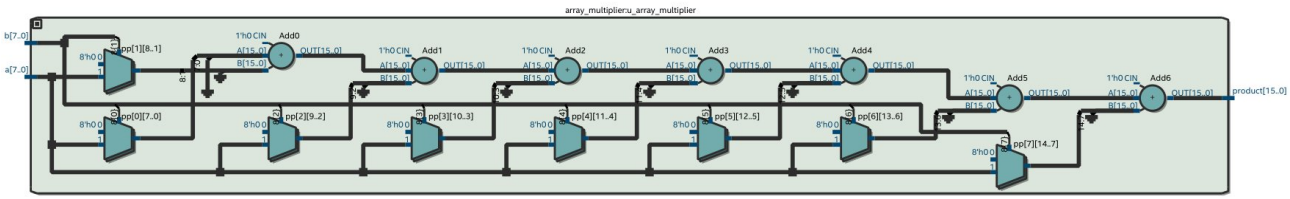
```
T=5000 | RESET | q_out=00000000 (expected 0)
T=15000 | PASS | en=0 dir=0 din=0 | q_out=00000000
T=25000 | PASS | en=1 dir=0 din=1 | q_out=00000000
T=35000 | PASS | en=1 dir=0 din=0 | q_out=00000001
T=45000 | PASS | en=1 dir=0 din=1 | q_out=00000010
T=55000 | PASS | en=1 dir=0 din=1 | q_out=00000101
T=65000 | PASS | en=0 dir=0 din=1 | q_out=00001011
T=75000 | PASS | en=0 dir=0 din=1 | q_out=00001011
T=85000 | PASS | en=1 dir=1 din=1 | q_out=00001011
T=95000 | PASS | en=1 dir=1 din=1 | q_out=10000101
T=105000 | PASS | en=1 dir=1 din=0 | q_out=11000010
T=115000 | PASS | en=1 dir=1 din=0 | q_out=01100001
- shift_reg_tb.sv:141: Verilog $finish
T=125000 | PASS | en=1 dir=1 din=0 | q_out=00110000
```

Waveform: [Shift Register Waveform PDF]

Muhammad Ahmed Qazi
CS-24045

Question 04: Implement a Parameterisable Array Multiplier





Code: [array_multiplier.sv]

Test-bench: [tb_multiplier.sv]

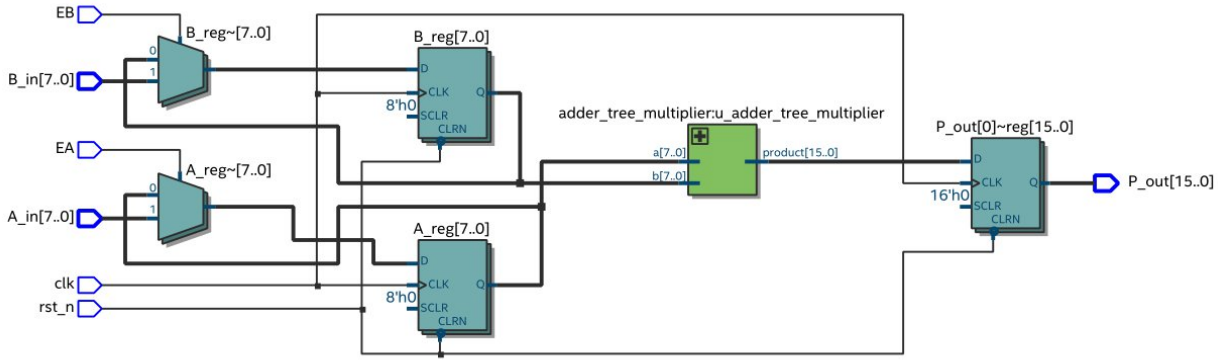
Output:

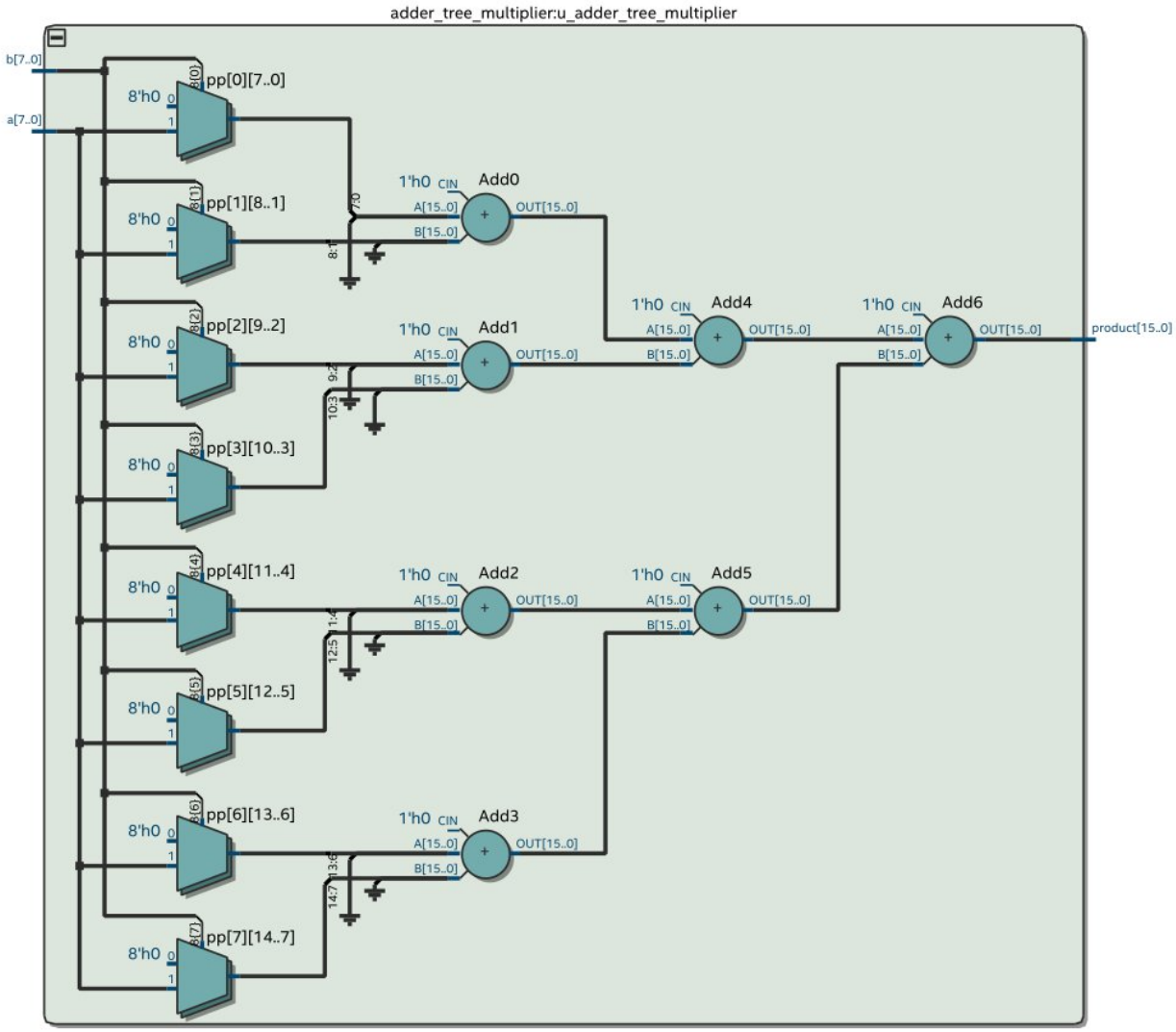
```
[PASS] Test 1: A= 10, B= 15 |
Expected=  150, Got=  150

[PASS] Test 2: A=  0, B=255 |
Expected=    0, Got=    0

[PASS] Test 3: A=255, B=255 |
Expected=65025, Got=65025

[PASS] Test 4: A=128, B=128 |
Expected=16384, Got=16384

- tb_multiplier.sv:136: Verilog
$finish
```

Waveform: [Array Multiplier Waveform PDF]

Muhammad Ahmed Qazi
CS-24045

Question 05: Implement a 8x8 Adder Tree Multiplier





Code: [adder_tree_multiplier.sv]

Test-bench: [tb_multiplier.sv]

Muhammad Ahmed Qazi
CS-24045

Output:

```
[PASS] Test 1: A= 10, B= 15 |
Expected=  150, Got=  150

[PASS] Test 2: A=  0, B=255 |
Expected=    0, Got=    0

[PASS] Test 3: A=255, B=255 |
Expected=65025, Got=65025

[PASS] Test 4: A=128, B=128 |
Expected=16384, Got=16384

- tb_multiplier.sv:136: Verilog
$finish
```

Waveform: [Adder Tree Multiplier Waveform PDF]

Muhammad Ahmed Qazi
CS-24045

Question 06: Write an Assertion-Based Test-bench for the 32-bit Register

Code: [reg32_tb.sv]

Assertions included:

Reset Behaviour

```
    // Reset behavior (active-low: reset=0 means in reset)
    property reset_behavior;
        @(posedge clk)
            !reset |=> (q == 32'b0);  // Check NEXT cycle after reset goes low
    endproperty

    assert property (reset_behavior)
        else $error("RESET FAILED: q is not zero when reset is active");
```

Load Behaviour

```
    // Load behavior - check one cycle AFTER load
    property load_behavior;
        @(posedge clk)
            (reset && $past(load)) |-> (q == $past(d));  // Check when load was high PREVIOUS cycle
    endproperty

    assert property (load_behavior)
        else $error("LOAD FAILED: q ≠ previous d");
```

Hold Behaviour

```
    // Hold behavior - should hold when no load in previous cycle
    property hold_behavior;
        @(posedge clk)
            (reset && $past(!load)) |-> (q == $past(q));  // Check when load was low PREVIOUS cycle
    endproperty

    assert property (hold_behavior)
        else $error("HOLD FAILED: q changed without load in previous cycle");
```

Output:

```
Reset released, q = 00000000 (expected 0)

After load, q = a5a5a5a5 (expected A5A5A5A5)

Hold test, q = a5a5a5a5 (expected A5A5A5A5)

After reset, q = 00000000 (expected 0)

Testbench completed.
```

Muhammad Ahmed Qazi
CS-24045

Question 07: Write Layered Test-benches

1. Layered Test-bench for Synchronous Bi-directional Shift Register

Code: shift_reg_tb.sv

Layers included:

- Parameters & Signals
- DUT Instantiation
- Clock Generator
- Reset Task
- Driver Tasks
- Reference Model
- Monitor + Checker
- Test Scenario

Test-bench Output

2. Layered Test-bench for Synchronous Up/Down Counter

Code: counter_tb.sv

Layers included:

- Parameters & Signals
- DUT Instantiation
- Clock Generator
- Reset Task
- Driver Tasks
- Reference Model
- Monitor + Checker
- Test Scenario

Test-bench Output

3. Layered Test-bench for Array & Adder Tree Multipliers

Code: tb_multiplier.sv *(the same test-bench is used to test both array and adder-tree multiplier modules)*

Layers included:

- Signal Declarations
- Queue Declarations
- Printing Waveform
- DUT Instantiation
- Clock Generation
- Reset Sequence
- Driver Process (Stimulus Application)
- Scoreboard Process (Self-Checking)

Test-bench Output