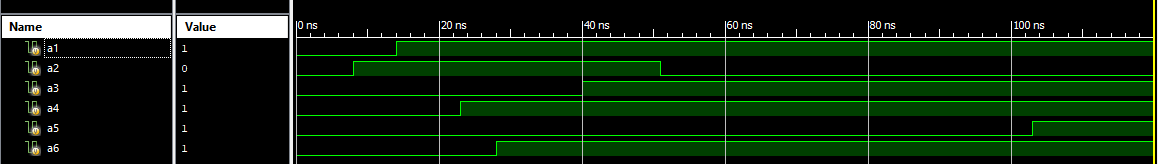
1

Q1)

1. Demo 1



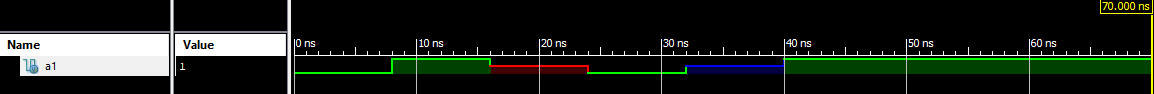
b)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Execution Time | A1 | A2 | A3 | A4 | A5 | A6 |
| 0 ns | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 ns | 0 | 1 | 0 | 0 | 0 | 0 |
| 14 ns | 1 | 1 | 0 | 0 | 0 | 0 |
| 22 ns | 1 | 1 | 0 | 1 | 0 | 0 |
| 28 ns | 1 | 1 | 0 | 1 | 0 | 1 |
| 40 ns | 1 | 1 | 1 | 1 | 0 | 1 |
| 51 ns | 1 | 0 | 1 | 1 | 0 | 1 |
| 103 ns | 1 | 0 | 1 | 1 | 1 | 1 |

c) Expected finish at 120 ps due to $finish statement.

d) The initial blocks are run concurrently since the changes happen in parallel and don’t wait for the other blocks.

e) Demo 2



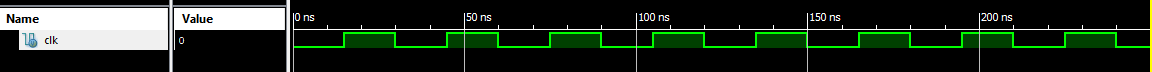
f)

|  |  |
| --- | --- |
| Execution Time | A1 |
| 0 ns | 0 |
| 8 ns | 1 |
| 16 ns | X |
| 24 ns | 0 |
| 32 ns | z |
| 40 ns | 1 |

Expected finish at 70 ps

1.2

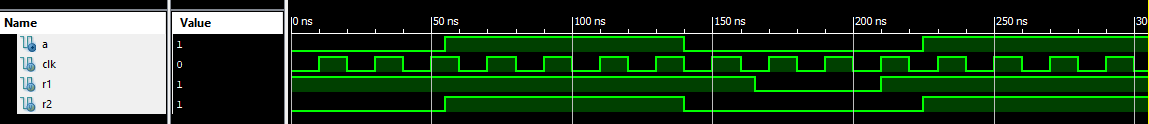
a) Demo 3



b) The measured period of the clock is 30 ps.

c) When the $finish is removed the program runs/loops forever until some external limit it reached, in this case is 1 000 000 ps.

d) Demo 4



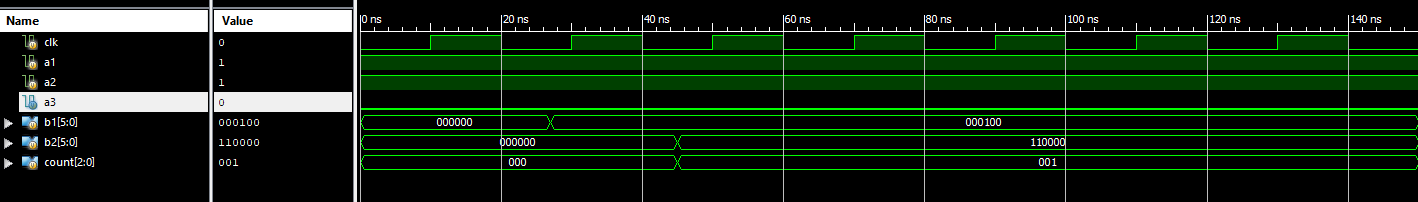
e)

|  |  |
| --- | --- |
| Execution Time | A |
| 0 ns | 0 |
| 60 ns | 1 |
| 140 ns | 0 |
| 225 ns | 1 |

f) “a” does not change on clock edges because it is not in an “always” procedure with that clock condition in the trigger list.

1.3

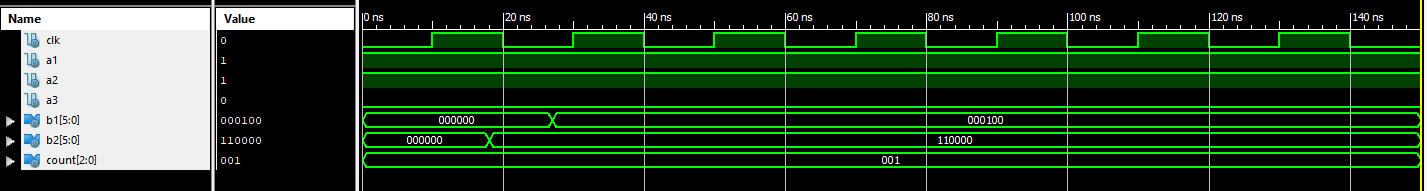
a) Demo 5



b)

|  |  |
| --- | --- |
| Execution Time | Statement |
| 0 ns | a1=1; a2=1; a3=0;  count=0;  b1=6'b0; b2=b1; |
| 27 ns | b1[2]=1'b1; |
| 45 ns | b2[5:3]={a1,a2,a3};  count=count+1; |

c) Demo 6



d)

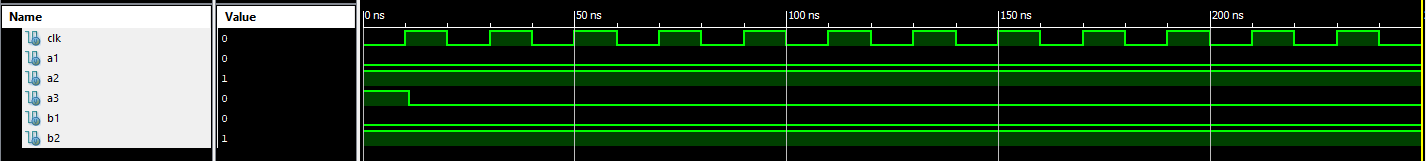
|  |  |
| --- | --- |
| Execution Time | Statement |
| 0 ns | a1=1; a2=1; a3=0;  count=0;  b1=6'b0; b2=b1;  count <= count+1; |
| 18 ns | b2[5:3] <= #18 {a1,a2,a3}; |
| 27 ns | b1[2] <= #27 1'b1; |

e) In demo 5, the statements in the procedure execute sequentially and in demo 6 the statements execute concurrently.

f) Blocking statements execute in sequence.

1.4

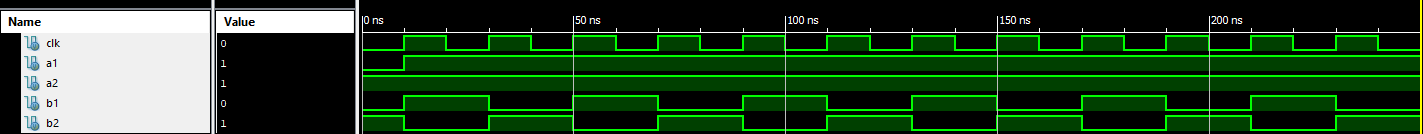
a) Demo 7



b) Due to non-blocking, “<=”, the value of reg 3 is assigned by the old value of reg 1.

c) No, due to non-blocking statements.

d) Demo 8

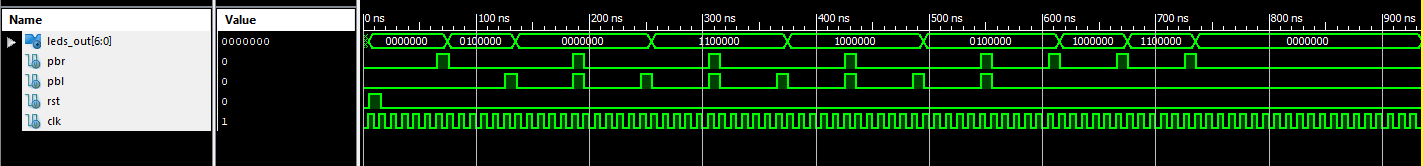


e) Non-blocking can do a toggle operation, since they sample the previous value.

f) Non-blocking statements are executed concurrently and sample the previous value if multiple statements altering shared registers are executed at the same delay.

Part 2

a)



Code if interested:

initial begin

// initialize all inputs

clk = 0; rst = 0;

pbl = 0; pbr = 0;

@(posedge clk); #1; // waits for a clock edge, then moves just past it

rst = 1; // put the system in reset

@(posedge clk); #1;

rst = 0; // remove the system from reset

//start count up once

repeat(5) @(posedge clk); #1;

pbr =1;

@(posedge clk); #1;

pbr = 0;

//start count down once

repeat(5) @(posedge clk); #1;

pbl =1;

@(posedge clk); #1;

pbl = 0;

// from s0 to s0

repeat(5) @(posedge clk); #1;

pbr = 1; pbl = 1;

@(posedge clk); #1;

pbr = 0; pbl = 0;

// from s0 to s3

repeat(5) @(posedge clk); #1;

pbl = 1;

@(posedge clk); #1;

pbr = 0; pbl = 0;

// from s3 to s3

repeat(5) @(posedge clk); #1;

pbr = 1; pbl = 1;

@(posedge clk); #1;

pbr = 0; pbl = 0;

// from s3 to s2

repeat(5) @(posedge clk); #1;

pbl = 1;

@(posedge clk); #1;

pbr = 0; pbl = 0;

// from s2 to s2

repeat(5) @(posedge clk); #1;

pbr = 1; pbl = 1;

@(posedge clk); #1;

pbr = 0; pbl = 0;

// from s2 to s1

repeat(5) @(posedge clk); #1;

pbl = 1;

@(posedge clk); #1;

pbr = 0; pbl = 0;

// from s1 to s1

repeat(5) @(posedge clk); #1;

pbr = 1; pbl = 1;

@(posedge clk); #1;

pbr = 0; pbl = 0;

// from s1 to s2

repeat(5) @(posedge clk); #1;

pbr = 1;

@(posedge clk); #1;

pbr = 0; pbl = 0;

// from s2 to s3

repeat(5) @(posedge clk); #1;

pbr = 1;

@(posedge clk); #1;

pbr = 0; pbl = 0;

// from s3 to s0

repeat(5) @(posedge clk); #1;

pbr = 1;

@(posedge clk); #1;

pbr = 0; pbl = 0;

// continue to test all states and all transitions

repeat(20) @(posedge clk);

$finish;

end