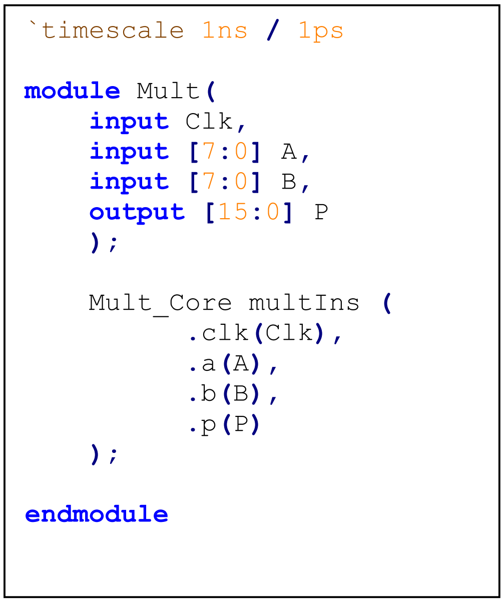
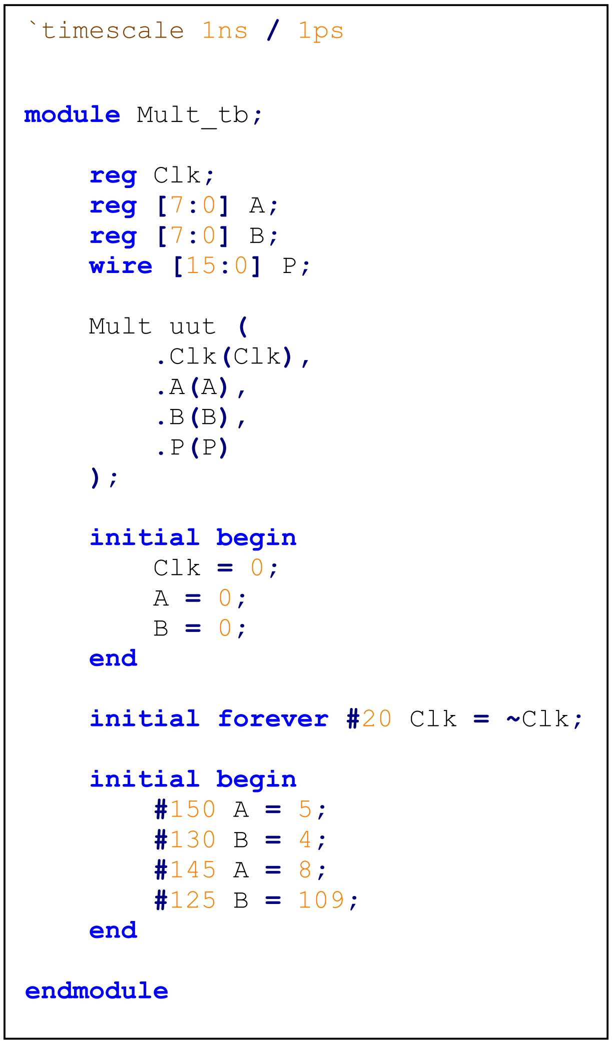
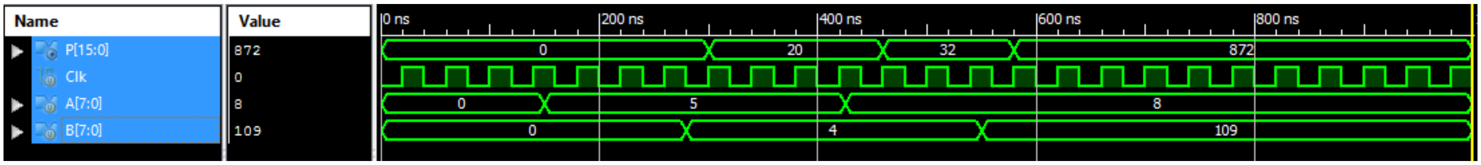
(۱



```
`timescale 1ns / 1ps

module Mult(
    input Clk,
    input [7:0] A,
    input [7:0] B,
    output [15:0] P
    );

    Mult_Core multIns (
          .clk(Clk),
          .a(A),
          .b(B),
          .p(P)
    );

endmodule
```



```
`timescale 1ns / 1ps


module Mult_tb;

    reg Clk;
    reg [7:0] A;
    reg [7:0] B;
    wire [15:0] P;

    Mult uut (
        .Clk(Clk),
        .A(A),
        .B(B),
        .P(P)
    );

    initial begin
        Clk = 0;
        A = 0;
        B = 0;
    end

    initial forever #20 Clk = ~Clk;

    initial begin
        #150 A = 5;
        #130 B = 4;
        #145 A = 8;
        #125 B = 109;
    end

endmodule
```



Name	Value
P[15:0]	872
Clk	0
A[7:0]	8
B[7:0]	109

۳)

```
`timescale 1ns / 1ps


module Convolution(
    clk,
    out
    );

    input clk;
    output reg [30:0] out;

    reg [3:0] h [15:0];
    reg [3:0] x [15:0];
    reg [30:0] IM_Result = 0;

    integer i;
    integer k;
    integer n;

    always @ (posedge clk) begin
        for (i = 0; i <= 15; i = i +
1) begin
            h[i] <= i % 4;
            x[i] <= i % 5;
        end
        for (n = 0; n < 31; n = n +
1) begin
            IM_Result = 0;
            for (k = 0; k <= n; k = k
+ 1) begin
                IM_Result <=
IM_Result + h[k] * x[n - k];
            end
            out[n] = IM_Result;
        end
    end


endmodule
```

```
`timescale 1ns / 1ps


module Con_tb;

    reg clk;
    wire [30:0] out;

    Convolution uut (
        .clk(clk),
        .out(out)
    );

    initial begin
        clk = 0;
    end

    initial forever #25 clk =
~clk;

endmodule
```