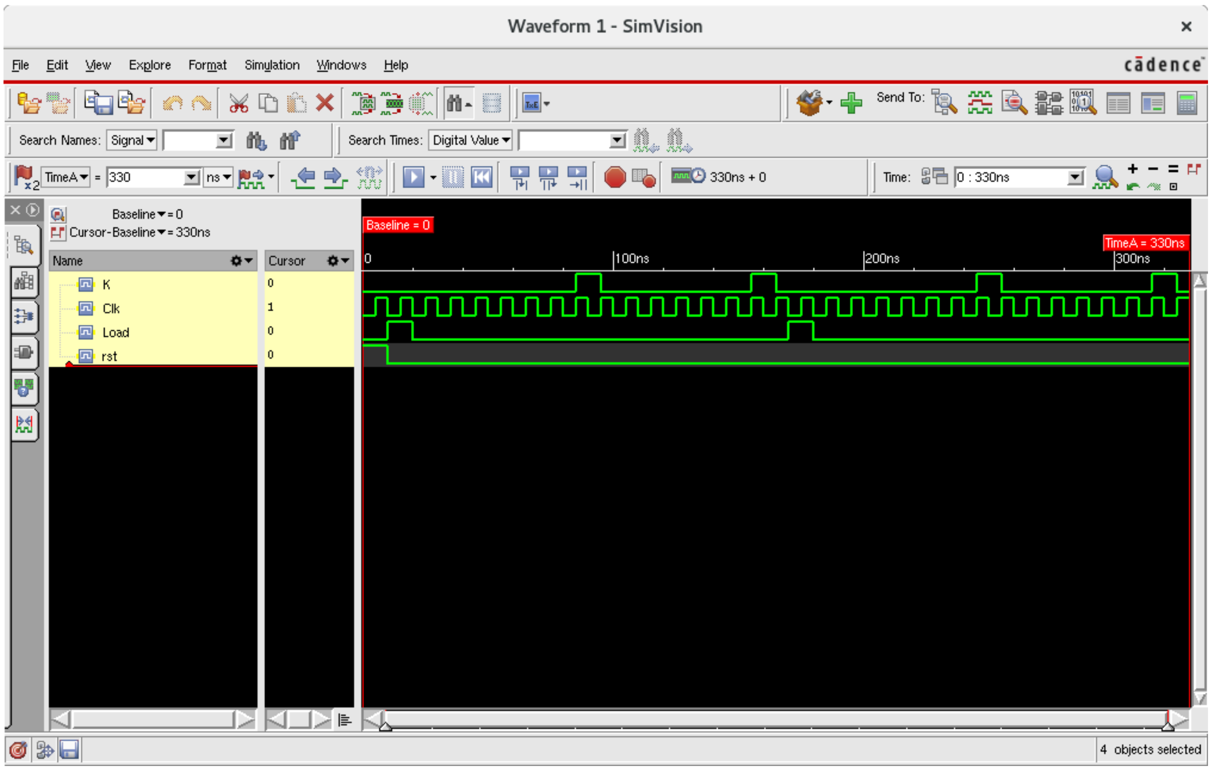
SD122 - Aula 24/25
Tema: Sistemas digitais com handshaking

Nome: Marcos Vinícius Barbosa
Data: 11/06/2025

```
module Counter #(parameter DATA_LENGTH = 4) (
      input Load, Clk, rst,
      output reg K
);

reg [DATA_LENGTH/2:0] count;

always @(posedge Clk, posedge rst) begin
      if (Load | rst) begin
      count <= 0;
      K <= 0;
      end else if (count == ((DATA_LENGTH * 2) - 2)) begin
      count <= 0;
      K <= 1;
      end else begin
      count <= count + 1;
      K <= 0;
      end
end

endmodule
```

```
module Counter_TB();

parameter DATA_LENGTH = 4;

reg Clk, Load, rst;
wire K;

Counter #(DATA_LENGTH) dut (
      .Load(Load),
      .Clk(Clk),
      .rst(rst),
      .K(K)
);

always #5 Clk = ~Clk;

initial begin
```

```
        Clk = 0;
        rst = 1;
        Load = 0;
        $monitor("Tempo: %t | Load=%b | K=%b", $time, Load, K);
        #10;
        rst = 0;
        Load = 1;
        #10;
        Load = 0;
        #150;
        Load = 1;
        #10;
        Load = 0;
        #150;
        $stop;
end

endmodule
```

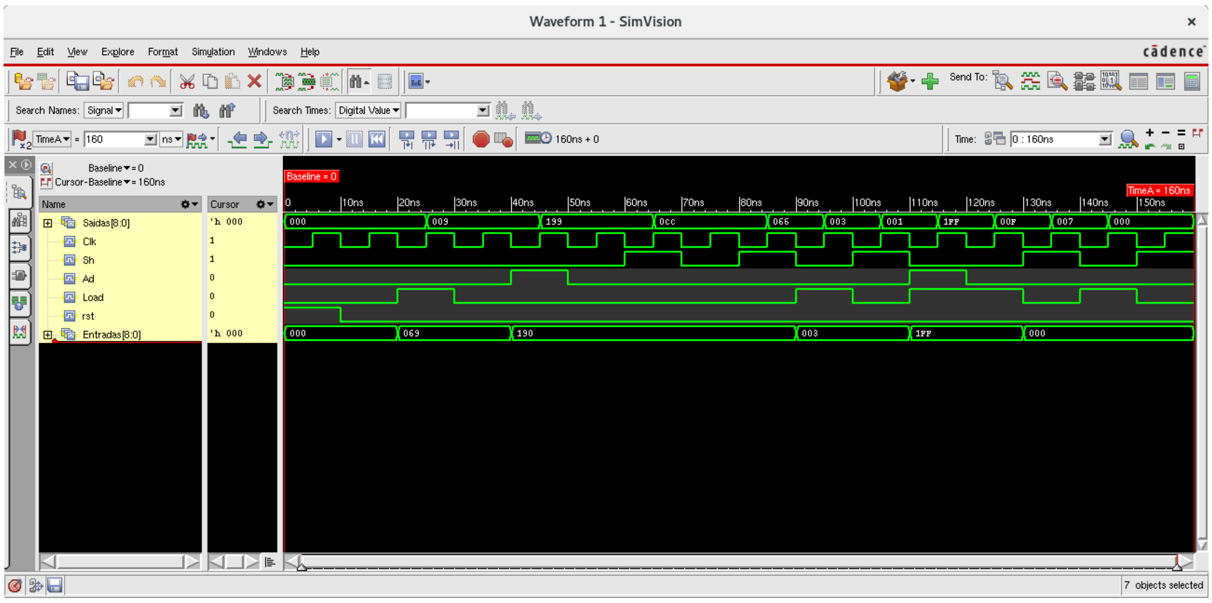


```
module ACC #(parameter DATA_LENGTH = 4) (
        input Load, Sh, Ad, Clk, rst,
        input  [2*DATA_LENGTH:0] Entradas,
        output reg [2*DATA_LENGTH:0] Saidas
);
```

```
always @(posedge Clk, posedge rst) begin
      if (rst)
      Saidas <= 0;
      if (Load)
      Saidas <= {{(DATA_LENGTH){1'b0}}, Entradas[DATA_LENGTH-1:0]};
      if (Sh)
      Saidas <= Saidas >> 1;
      if (Ad)
      Saidas[2*DATA_LENGTH:DATA_LENGTH] <=
Entradas[2*DATA_LENGTH:DATA_LENGTH];
end

endmodule
```

```
module ACC_TB();

parameter DATA_LENGTH = 4;

reg Clk, Sh, Ad, Load, rst;
reg  [2*DATA_LENGTH:0] Entradas;
wire [2*DATA_LENGTH:0] Saidas;

ACC #(DATA_LENGTH) dut (
      .Load(Load), .Sh(Sh), .Ad(Ad), .Clk(Clk), .rst(rst),
      .Entradas(Entradas), .Saidas(Saidas)
);

initial begin
      Clk = 0;
      forever #5 Clk = ~Clk;
end

initial begin
      rst = 1;
      Load = 0;
      Sh = 0;
      Ad = 0;
      Entradas = 0;
      #10;

      rst = 0;
      #10;

      Entradas = 9'b001_101001;
      Load = 1;
```

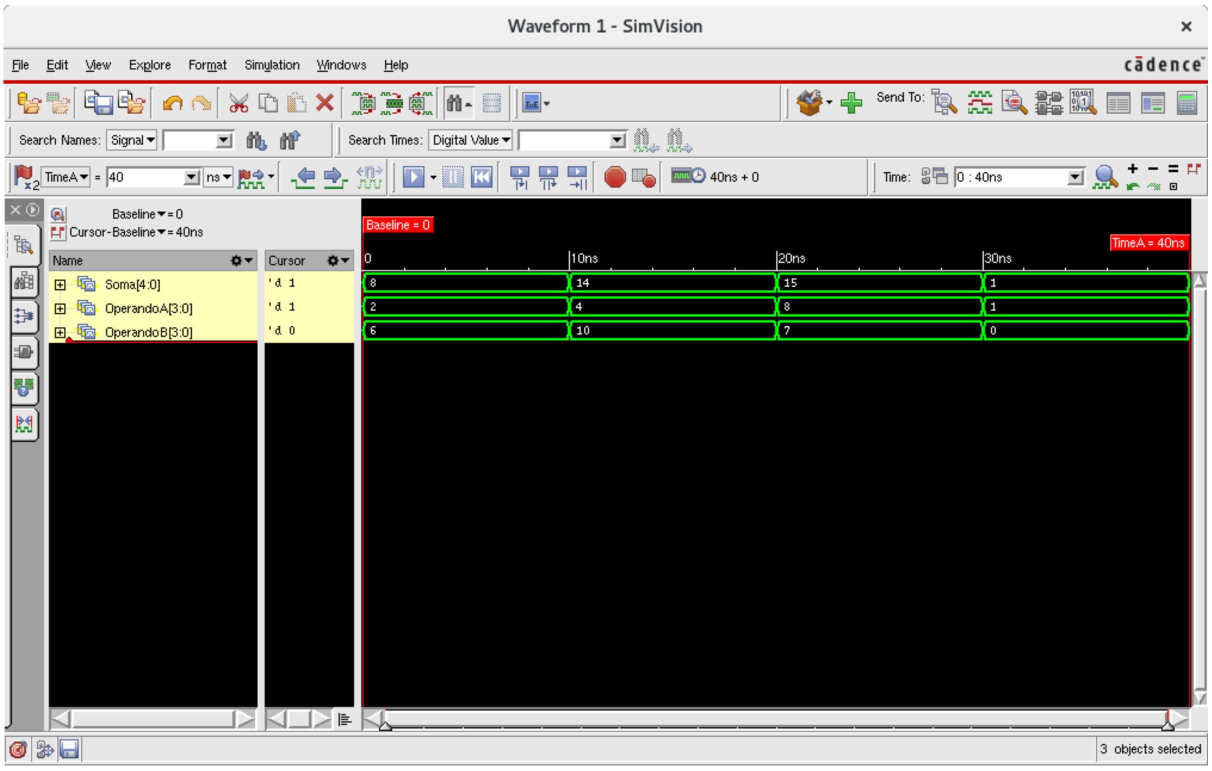
```verilog
      #10;

      Load = 0;
      #10;

      Entradas = 9'b110_010000;
      Ad = 1;
      #10;

      Ad = 0;
      #10;

      Sh = 1;
      #10;

      Sh = 0;
      #10;

      Sh = 1;
      #10;

      Sh = 0;

      Entradas = 9'b000_000011;
      Load = 1; Sh = 0; Ad = 0;
      #10 Load = 0; Sh = 1; Ad = 0;
      #10 Load = 1; Sh = 0; Ad = 1;

      // Overflow
      Entradas = 9'b1_11111111;
      #10 Load = 1; Sh = 0; Ad = 0;
      #10 Load = 0; Sh = 1; Ad = 0;

      // Underflow
      Entradas = 9'b0_00000000;
      #10 Load = 1; Sh = 0; Ad = 0;
      #10 Load = 0; Sh = 1; Ad = 0;

      #10;
      $stop;
end

endmodule
```



```
module Adder #(parameter DATA_LENGTH = 4) (
      input   [DATA_LENGTH-1:0] OperandoA, OperandoB,
      output  [DATA_LENGTH:0]   Soma
);

assign Soma = OperandoA + OperandoB;

endmodule
```

```
module Adder_TB();

parameter DATA_LENGTH = 4;

reg  [DATA_LENGTH-1:0] OperandoA;
reg  [DATA_LENGTH-1:0] OperandoB;
wire [DATA_LENGTH:0]   Soma;

Adder #(DATA_LENGTH) dut (
      .OperandoA(OperandoA),
      .OperandoB(OperandoB),
      .Soma(Soma)
);

initial begin
      OperandoA = 2;
      OperandoB = 6;
      #10;
      $display("%d somado com %d resulta em %d", OperandoA, OperandoB,
Soma);
```

```
      OperandoA = 4;
      OperandoB = 10;
      #10;
      $display("%d somado com %d resulta em %d", OperandoA, OperandoB,
Soma);

      OperandoA = 8;
      OperandoB = 7;
      #10;
      $display("%d somado com %d resulta em %d", OperandoA, OperandoB,
Soma);

      OperandoA = 1;
      OperandoB = 0;
      #10;
      $display("%d somado com %d resulta em %d", OperandoA, OperandoB,
Soma);

      $stop;
end

endmodule
```



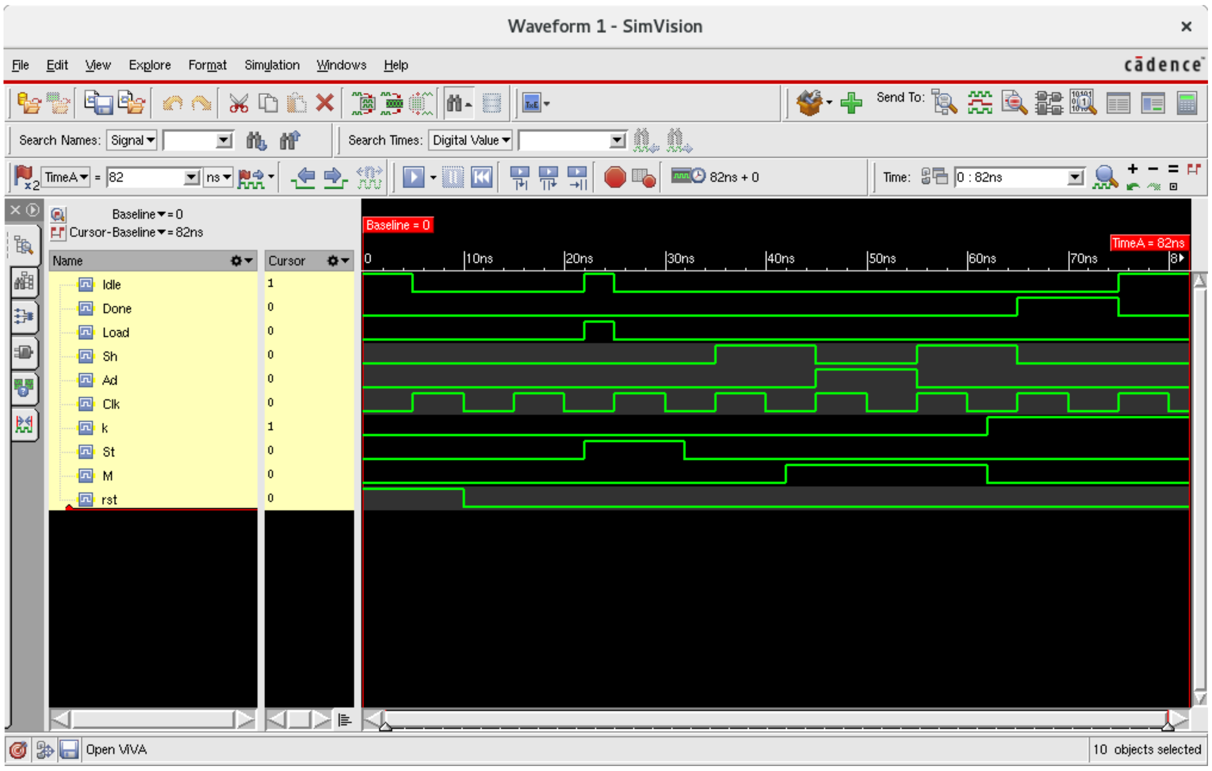
```
Baseline = 0
Cursor-Baseline = 40ns

Name            Cursor   0     10ns   20ns   30ns
Soma[4:0]       'd 1     8     14     15     1
OperandoA[3:0]  'd 1     2     4      8      1
OperandoB[3:0]  'd 0     6     10     7      0

Baseline = 0                          TimeA = 40ns
```

```
module CONTROL(
```

```verilog
      input  Clk, k, St, M, rst,
      output reg Idle, Done, Load, Sh, Ad
);

reg [1:0] state;
parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3;

always @(posedge Clk, posedge rst) begin
      if (rst) begin
      state <= S0;
      {Idle, Done, Load, Sh, Ad} <= 5'b00000;
      end else begin
      case (state)
            S0: if (St) state <= S1;
            S1: state <= S2;
            S2: if (k) state <= S3; else state <= S1;
            S3: state <= S0;
            default: state <= S0;
      endcase
      end
end

always @(*) begin
      case (state)
      S0: begin
            Ad    <= 0;
            Sh    <= 0;
            Load  <= 0;
            Idle  <= 1;
            Done  <= 0;
            if (St) Load <= 1;
      end
      S1: begin
            Sh    <= 0;
            Load  <= 0;
            Idle  <= 0;
            Done  <= 0;
            if (M) Ad <= 1; else Ad <= 0;
      end
      S2: begin
            Sh    <= 1;
            Ad    <= 0;
            Load  <= 0;
            Idle  <= 0;
            Done  <= 0;
      end
      S3: begin
            Ad    <= 0;
            Sh    <= 0;
            Load  <= 0;
            Idle  <= 0;
            Done  <= 1;
      end
      endcase
end

endmodule
```

```
module CONTROL_TB();

reg Clk, k, St, M, rst;
wire Idle, Done, Load, Sh, Ad;

CONTROL dut (
      .Clk(Clk),
      .k(k),
      .St(St),
      .M(M),
      .rst(rst),
      .Idle(Idle),
      .Done(Done),
      .Load(Load),
      .Sh(Sh),
      .Ad(Ad)
);

always #5 Clk = ~Clk;

initial begin
      rst = 1;
      Clk = 0;
      k = 0;
      St = 0;
      M = 0;
      #10 rst = 0;
      #12;
      $monitor("Tempo: %t | St=%b k=%b M=%b | Idle=%b Load=%b Ad=%b
Sh=%b Done=%b", $time, St, k, M, Idle, Load, Ad, Sh, Done);

      St = 1;
      #10;
```

```
        St = 0;
        #10;

        repeat (2) begin
        k = 0;
        M = 1;
        #10;
        end

        k = 1;
        M = 0;
        #10;

        #10;
        $stop;
end

endmodule
```



Name	Cursor
Idle	1
Done	0
Load	0
Sh	0
Ad	0
Clk	0
k	1
St	0
M	0
rst	0

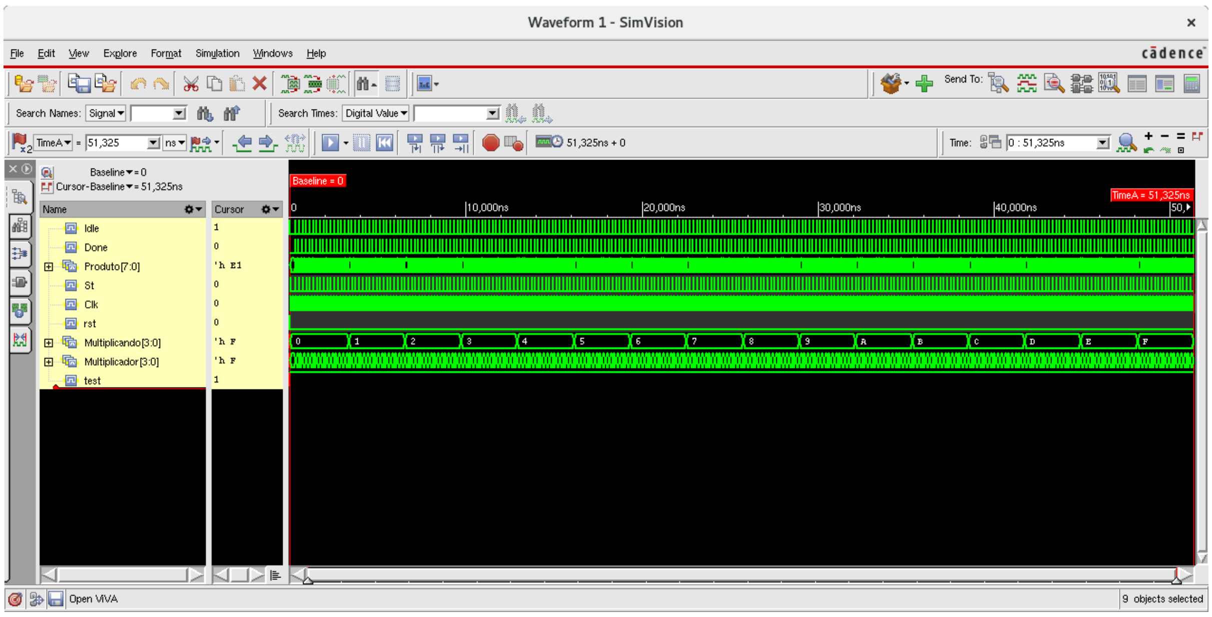
```
module Multiplicador #(parameter DATA_LENGTH = 4) (
        input   [DATA_LENGTH-1:0] Multiplicando, Multiplicador,
        input   St, Clk, rst,
        output [(DATA_LENGTH*2)-1:0] Produto,
        output Idle, Done
```

```verilog
);

wire Load, Sh, Ad, K, M;
wire [DATA_LENGTH:0] soma;
wire [DATA_LENGTH-1:0] operandoB;
wire [(DATA_LENGTH*2):0] resultado;

assign M = resultado[0];
assign operandoB = resultado[(DATA_LENGTH*2)-1 : DATA_LENGTH];

CONTROL control (
      .Idle(Idle),
      .Done(Done),
      .St(St),
      .Load(Load),
      .Sh(Sh),
      .Ad(Ad),
      .Clk(Clk),
      .k(K),
      .M(M),
      .rst(rst)
);

ACC #(DATA_LENGTH) acc (
      .Saidas(resultado),
      .Entradas({soma, Multiplicador}),
      .Load(Load),
      .Sh(Sh),
      .Ad(Ad),
      .Clk(Clk),
      .rst(rst)
);

Adder #(DATA_LENGTH) adder (
      .OperandoA(Multiplicando),
      .OperandoB(operandoB),
      .Soma(soma)
);

Counter #(DATA_LENGTH) counter (
      .Load(Load),
      .rst(rst),
      .Clk(Clk),
      .K(K)
);
```

```verilog
assign Produto = resultado[(DATA_LENGTH*2)-1 : 0];

endmodule
```

```verilog
module Multiplicador_TB();
parameter DATA_LENGTH = 4;

reg St, Clk, rst;
reg [DATA_LENGTH-1:0] Multiplicando, Multiplicador;
wire Idle, Done;
wire [(DATA_LENGTH*2)-1:0] Produto;

Multiplicador #(DATA_LENGTH) dut (
      .St(St),
      .Clk(Clk),
      .rst(rst),
      .Multiplicando(Multiplicando),
      .Multiplicador(Multiplicador),
      .Idle(Idle),
      .Done(Done),
      .Produto(Produto)
);

always #10 Clk = ~Clk;

reg test;
integer i, j;

initial begin
      Clk = 0;
      St = 0;
      rst = 1;
      #10 rst = 0;
      test = 1;

      for (i = 0; i < 2**DATA_LENGTH; i = i + 1) begin
      for (j = 0; j < 2**DATA_LENGTH; j = j + 1) begin
            Multiplicando = i;
            Multiplicador = j;
            St = 1;
            #20;
            St = 0;
            wait (Done == 1);
            #35;
            if (Produto !== (i * j)) begin
```

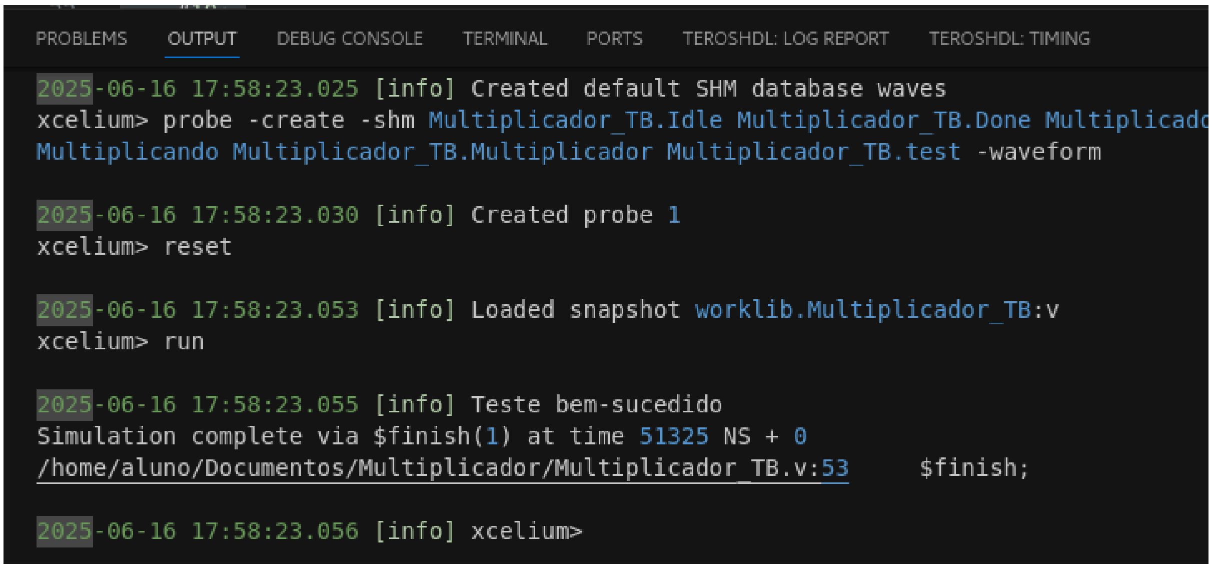
```
                  $display("Erro: A = %d, B = %d, Produto = %d",
Multiplicando, Multiplicador, Produto);
                  test = 0;
              end
          end
      end

      if (test)
      $display("Teste bem-sucedido");
      else
      $display("Teste falhou");

      $finish;
end

endmodule
```



Waveform 1 - SimVision

Baseline = 0
Cursor-Baseline = 51,325ns

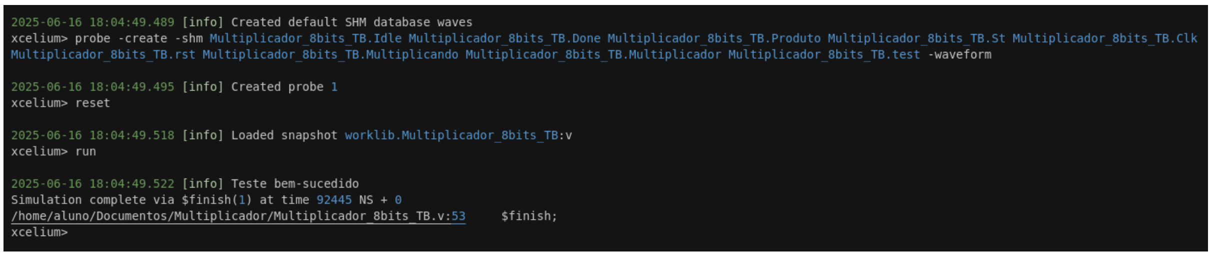
```
Name                 Cursor
Idle                 1
Done                 0
Produto[7:0]         'h E1
St                   0
Clk                  0
rst                  0
Multiplicando[3:0]   'h F
Multiplicador[3:0]   'h F
test                 1
```



```
2025-06-16 17:58:23.025 [info] Created default SHM database waves
xcelium> probe -create -shm Multiplicador_TB.Idle Multiplicador_TB.Done Multiplicado
Multiplicando Multiplicador_TB.Multiplicador Multiplicador_TB.test -waveform

2025-06-16 17:58:23.030 [info] Created probe 1
xcelium> reset

2025-06-16 17:58:23.053 [info] Loaded snapshot worklib.Multiplicador_TB:v
xcelium> run

2025-06-16 17:58:23.055 [info] Teste bem-sucedido
Simulation complete via $finish(1) at time 51325 NS + 0
/home/aluno/Documentos/Multiplicador/Multiplicador_TB.v:53     $finish;

2025-06-16 17:58:23.056 [info] xcelium>
```

- **8 Bits**

```
module Multiplicador_8bits_TB();
parameter DATA_LENGTH = 8;

reg St, Clk, rst;
reg [DATA_LENGTH-1:0] Multiplicando, Multiplicador;
wire Idle, Done;
wire [(DATA_LENGTH*2)-1:0] Produto;

Multiplicador #(DATA_LENGTH) dut (
      .St(St),
      .Clk(Clk),
      .rst(rst),
      .Multiplicando(Multiplicando),
      .Multiplicador(Multiplicador),
      .Idle(Idle),
      .Done(Done),
      .Produto(Produto)
);

always #10 Clk = ~Clk;

reg test;
integer i, j;

initial begin
      Clk = 0;
      St = 0;
      rst = 1;
      #10 rst = 0;
      test = 1;

      for (i = 0; i < 2**4; i = i + 1) begin
      for (j = 0; j < 2**4; j = j + 1) begin
            Multiplicando = i;
            Multiplicador = j;
            St = 1;
            #20;
            St = 0;
            wait (Done == 1);
            #35;
            if (Produto !== (i * j)) begin
                  $display("Erro: A = %d, B = %d, Produto = %d",
Multiplicando, Multiplicador, Produto);
                  test = 0;
            end
      end
      end

      if (test)
      $display("Teste bem-sucedido");
      else
      $display("Teste falhou");

      $finish;
end

endmodule
```



```
2025-06-16 18:04:49.489 [info] Created default SHM database waves
xcelium> probe -create -shm Multiplicador_8bits_TB.Idle Multiplicador_8bits_TB.Done Multiplicador_8bits_TB.Produto Multiplicador_8bits_TB.St Multiplicador_8bits_TB.Clk
Multiplicador_8bits_TB.rst Multiplicador_8bits_TB.Multiplicando Multiplicador_8bits_TB.Multiplicador Multiplicador_8bits_TB.test -waveform

2025-06-16 18:04:49.495 [info] Created probe 1
xcelium> reset

2025-06-16 18:04:49.518 [info] Loaded snapshot worklib.Multiplicador_8bits_TB:v
xcelium> run

2025-06-16 18:04:49.522 [info] Teste bem-sucedido
Simulation complete via $finish(1) at time 92445 NS + 0
/home/aluno/Documentos/Multiplicador/Multiplicador_8bits_TB.v:53     $finish;
xcelium>
```

Para a atividade de 8bits, foi usado o parâmetro DATA_LENGTH, assim, alterando apenas ele é possível testar outros conjuntos de bits.