UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE MATEMÁTICA ASTRONOMÍA, FÍSICA Y COMPUTACIÓN.

ARQUITECTURA DE COMPUTADORAS

TEÓRICOS: PABLO A. FERREYRA

PRÁCTICOS:
DELFINA VELEZ
AGUSTÍN LAPROVITA
GONZALO VODANOVICK

T2: HARDWARE DESCRIPTION LANGUAGES (A8 Parametrized Modules – N-bit Multiplexers)

Example A.40 Parameterized *N*-bit Multiplexers

SystemVerilog

```
module mux2
  #(parameter width = 8)
   (input  logic [width-1:0] d0, d1,
    input  logic             s,
    output logic [width-1:0] y);

   assign y = s ? d1 : d0;
endmodule
```

T2: HARDWARE DESCRIPTION LANGUAGES (A8 Parametrized Modules – N-bit Multiplexers)

SystemVerilog (continued)

```
module mux4_8(input  logic [7:0] d0, d1, d2, d3,
              input  logic [1:0] s,
              output logic [7:0] y);

  logic [7:0] low, hi;

  mux2 lowmux(d0, d1, s[0], low);
  mux2 himux(d2, d3, s[0], hi);
  mux2 outmux(low, hi, s[1], y);
endmodule
```

T2: HARDWARE DESCRIPTION LANGUAGES (A8 Parametrized Modules – N-bit Multiplexers)

```
module mux4_12(input logic [11:0] d0, d1, d2, d3,
               input logic [1:0]  s,
               output logic [11:0] y);

  logic [11:0] low, hi;

  mux2 #(12) lowmux(d0, d1, s[0], low);
  mux2 #(12) himux(d2, d3, s[0], hi);
  mux2 #(12) outmux(low, hi, s[1], y);
endmodule
```

T2: HARDWARE DESCRIPTION LANGUAGES (A8 Parametrized Modules – N-bit Multiplexers)

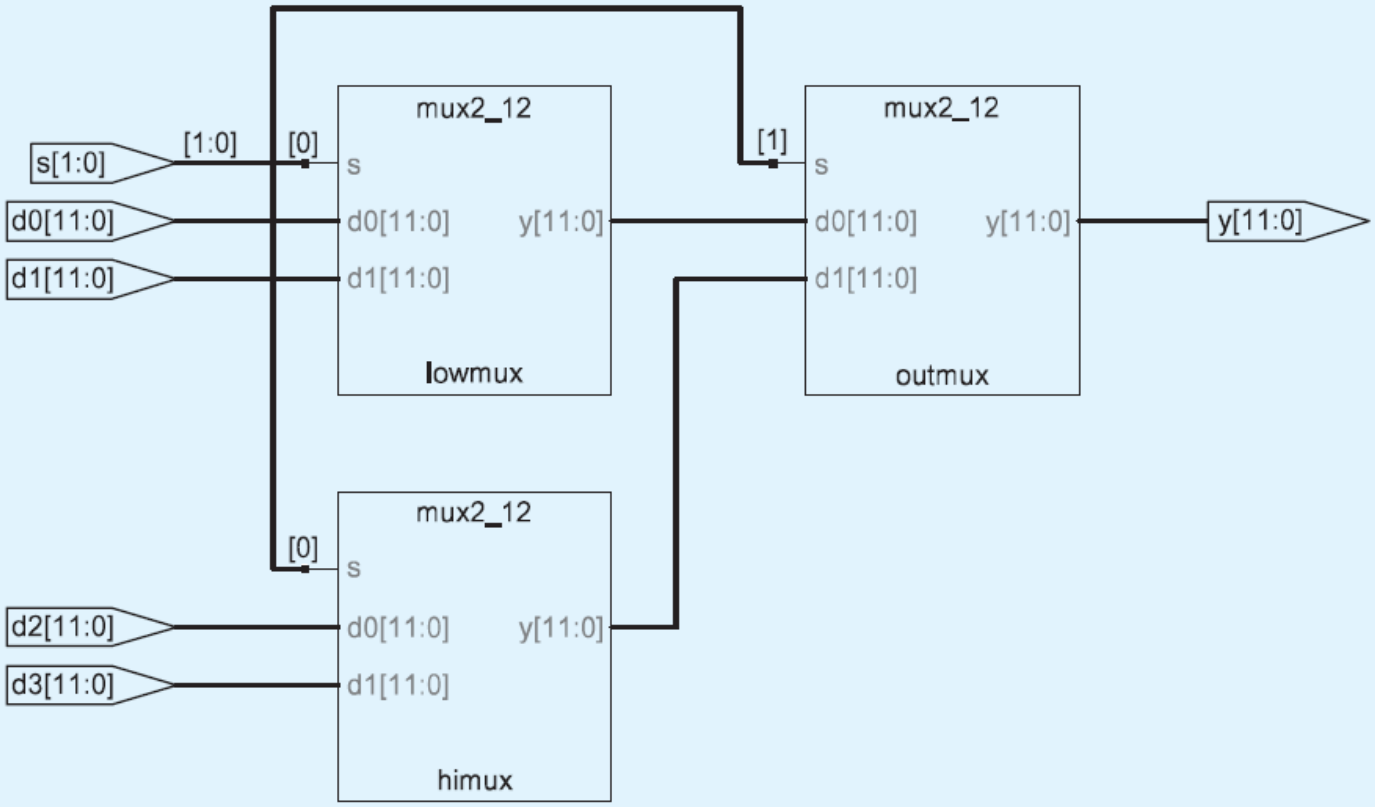

FIGURE A.39 mux4_12

T2: HARDWARE DESCRIPTION LANGUAGES (A8 Parametrized Modules – N-bit Decoder)

Example A.41 Parameterized N:2^N Decoder

SystemVerilog

```
module decoder #(parameter N = 3)
               (input  logic [N-1:0]    a,
                output logic [2**N-1:0] y);

  always_comb
    begin
      y = 0;
      y[a] = 1;
    end
endmodule
```

`2**N` indicates 2^N.

T2: HARDWARE DESCRIPTION LANGUAGES (A8 Parametrized Modules – N-bit AND remember Reductor Operators)

Example A.42 Parameterized *N*-input AND Gate

SystemVerilog

```
module andN
  #(parameter width = 8)
   (input  logic [width-1:0] a,
    output logic             y);

  genvar i;
  logic [width-1:1] x;

  generate
    for (i=1; i<width; i=i+1) begin:forloop
      if (i == 1)
        assign x[1] = a[0] & a[1];
      else
        assign x[i] = a[i] & x[i-1];
    end
  endgenerate
  assign y = x[width-1];
endmodule
```

T2: HARDWARE DESCRIPTION LANGUAGES (A8 Parametrized Modules N-bit AND remember Reductor Operators)

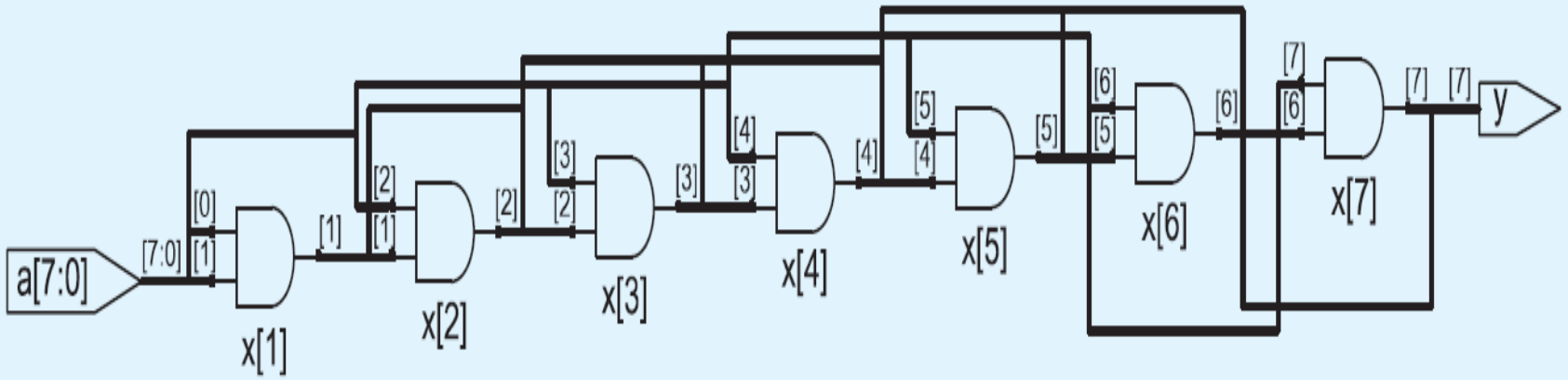

FIGURE A.40 andN

T2: HARDWARE DESCRIPTION LANGUAGES (A9 Memories – RAM With Separate Din and Dout)

Example A.43 RAM

SystemVerilog

```
module ram #(parameter N = 6, M = 32)
            (input  logic          clk,
             input  logic          we,
             input  logic [N-1:0]  adr,
             input  logic [M-1:0]  din,
             output logic [M-1:0]  dout);

  logic [M-1:0] mem[2**N-1:0];

  always @(posedge clk)
    if (we) mem[adr] <= din;

  assign dout = mem[adr];
endmodule
```

T2: HARDWARE DESCRIPTION LANGUAGES (A9 Memories – RAM With Separate Din and Dout)

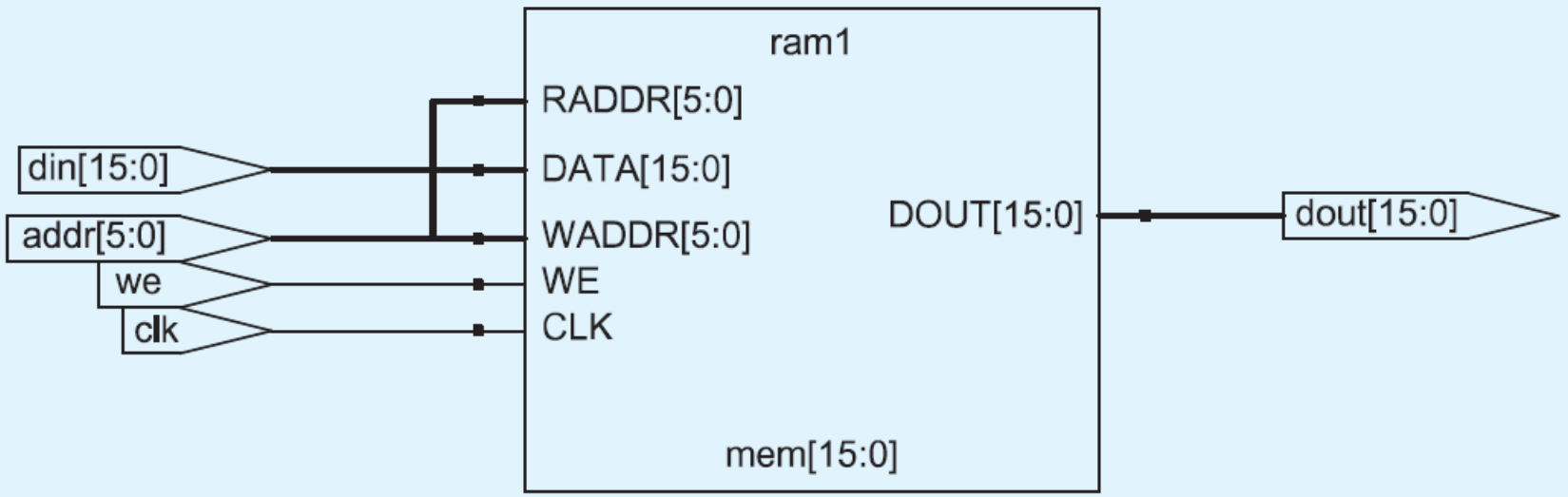

FIGURE A.41 Synthesized ram

T2: HARDWARE DESCRIPTION LANGUAGES (A9 Memories RAM With Bidirectional Data Bus)

Example A.44 RAM with Bidirectional Data Bus

SystemVerilog

```
module ram #(parameter N = 6, M = 32)
            (input  logic          clk,
             input  logic          we,
             input  logic [N-1:0]  adr,
             inout  tri   [M-1:0]  data);

  logic [M-1:0] mem[2**N-1:0];

  always @(posedge clk)
    if (we) mem[adr] <= data;

  assign data = we ? 'z : mem[adr];
endmodule
```

T2: HARDWARE DESCRIPTION LANGUAGES (A9 Memories RAM With Bidirectional Data Bus)

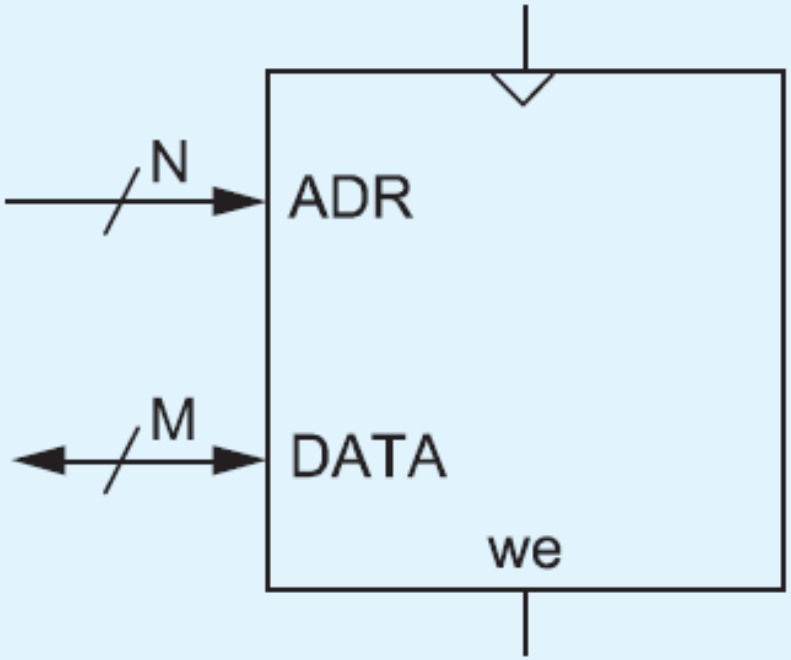

FIGURE A.42 Synthesized ram with bidirectional data bus

T2: HARDWARE DESCRIPTION LANGUAGES (A9 Memories – Three Ported Registers Files)

Example A.45 Three-Ported Register File

SystemVerilog

```
module ram3port #(parameter N = 6, M = 32)
                 (input  logic          clk,
                  input  logic          we3,
                  input  logic [N-1:0] a1, a2, a3,
                  output logic [M-1:0] d1, d2,
                  input  logic [M-1:0] d3);

  logic [M-1:0] mem[2**N-1:0];

  always @(posedge clk)
    if (we3) mem[a3] <= d3;

  assign d1 = mem[a1];
  assign d2 = mem[a2];
endmodule
```

T2: HARDWARE DESCRIPTION LANGUAGES (A9 Memories – Three Ported RegistersFiles)

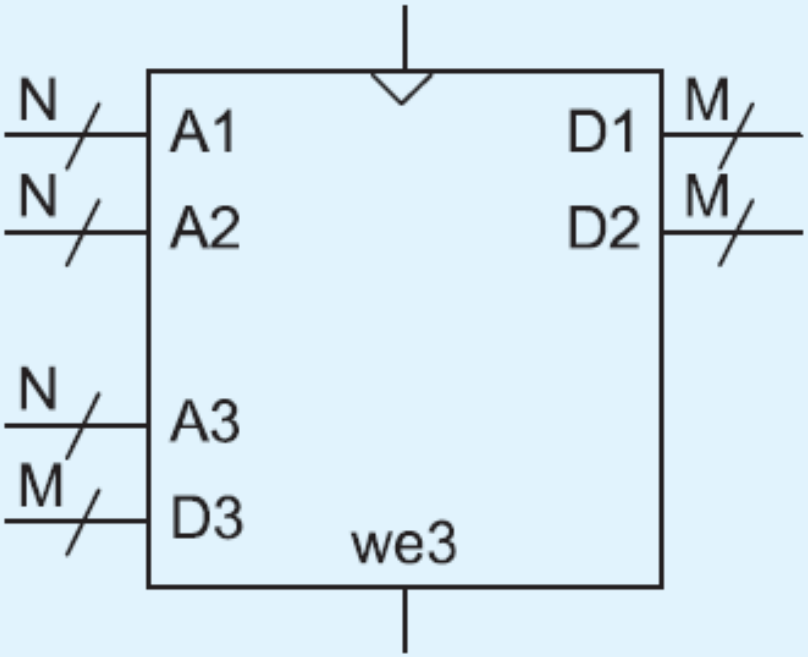

FIGURE A.43
Three-ported register file

T2: HARDWARE DESCRIPTION LANGUAGES (A9 Memories – ROMs)

Example A.46 ROM

SystemVerilog

```
module rom(input  logic [1:0] adr,
           output logic [2:0] dout);

  always_comb
    case(adr)
      2'b00: dout = 3'b011;
      2'b01: dout = 3'b110;
      2'b10: dout = 3'b100;
      2'b11: dout = 3'b010;
    endcase
endmodule
```

T2: HARDWARE DESCRIPTION LANGUAGES (A10 Test Benches –Basic Tests)

Example A.47 Testbench

SystemVerilog

```
module testbench1();
  logic a, b, c;
  logic y;

  // instantiate device under test
  sillyfunction dut(a, b, c, y);

  // apply inputs one at a time

  initial begin
    a = 0; b = 0; c = 0; #10;
    c = 1;               #10;
    b = 1; c = 0;        #10;
    c = 1;               #10;
    a = 1; b = 0; c = 0; #10;
    c = 1;               #10;
    b = 1; c = 0;        #10;
    c = 1;               #10;
  end
endmodule
```

T2: HARDWARE DESCRIPTION LANGUAGES (A10 Test Benches – Self Ch)

Example A.48 Self-Checking Testbench

SystemVerilog

```
module testbench2();
  logic a, b, c;
  logic y;

  // instantiate device under test
  sillyfunction dut(a, b, c, y);

  // apply inputs one at a time
  // checking results

  initial begin
    a = 0; b = 0; c = 0; #10;
    assert (y === 1) else $error("000 failed.");
    c = 1;               #10;
    assert (y === 0) else $error("001 failed.");
    b = 1; c = 0;        #10;
    assert (y === 0) else $error("010 failed.");
    c = 1;               #10;
    assert (y === 0) else $error("011 failed.");
    a = 1; b = 0; c = 0; #10;
    assert (y === 1) else $error("100 failed.");
    c = 1;               #10;
    assert (y === 1) else $error("101 failed.");
    b = 1; c = 0;        #10;
    assert (y === 0) else $error("110 failed.");
```

T2: HARDWARE DESCRIPTION LANGUAGES (A10 Test Benches – Self Checking)

```
    assert (y === 0) else $error("010 failed.");
    c = 1;                  #10;
    assert (y === 0) else $error("011 failed.");
    a = 1; b = 0; c = 0; #10;
    assert (y === 1) else $error("100 failed.");
    c = 1;                  #10;
    assert (y === 1) else $error("101 failed.");
    b = 1; c = 0;           #10;
    assert (y === 0) else $error("110 failed.");
    c = 1;                  #10;
    assert (y === 0) else $error("111 failed.");
  end
endmodule
```

T2: HARDWARE DESCRIPTION LANGUAGES (A10 Test Benches)

Example A.49 Testbench with Test Vector File

SystemVerilog

```
module testbench3();
  logic        clk, reset;
  logic        a, b, c, yexpected;
  logic        y;
  logic [31:0] vectornum, errors;
  logic [3:0]  testvectors[10000:0];

  // instantiate device under test
  sillyfunction dut(a, b, c, y);

  // generate clock
  always
    begin
      clk = 1; #5; clk = 0; #5;
    end

  // at start of test, load vectors
  // and pulse reset
```

T2: HARDWARE DESCRIPTION LANGUAGES (A10 Test Benches – Automatic File Based)

```
// at start of test, load vectors
// and pulse reset
initial
  begin
    $readmemb("example.tv", testvectors);
    vectornum = 0; errors = 0;
    reset = 1; #27; reset = 0;
  end

// apply test vectors on rising edge of clk
always @(posedge clk)
  begin
    #1; {a, b, c, yexpected} =
          testvectors[vectornum];
  end
```

T2: HARDWARE DESCRIPTION LANGUAGES (A10 Test Benches – Automatic File Based)

SystemVerilog (continued)

```
// check results on falling edge of clk
   always @(negedge clk)
    if (~reset) begin // skip during reset
      if (y !== yexpected) begin
        $display("Error: inputs = %b", {a, b, c});
        $display("  outputs = %b (%b expected)",
                 y, yexpected);
        errors = errors + 1;
      end
      vectornum = vectornum + 1;
      if (testvectors[vectornum] === 'bx) begin
        $display("%d tests completed with %d
                  errors", vectornum, errors);
        $finish;
      end
    end
endmodule
```