Parameterized Modules

```
module my_and2(
      i_in1,
      i_in2,
      o_out
);

input [7:0] i_in1;
input [7:0] i_in2;
output [7:0] o_out;

assign o_out = i_in1 & i_in2;



endmodule //my_and2
```

```
module top_module(
      i_gated,
      i_clock,
      o_result
);

input [7:0] i_gated;
input [7:0] i_clock;
output [7:0] o_result;

my_and2 AND0 (
      .i_in1(i_gated),
      .i_in2(i_clock),
      .o_out(o_result),

endmodule //top_module
```

```
module my_and2(
      i_in1,
      i_in2,
      o_out
);
parameter WIDTH = 8;
input [WIDTH-1:0] i_in1;
input [WIDTH-1:0] i_in2;
output [WIDTH-1:0] o_out;

assign o_out = i_in1 & i_in2;



endmodule //my_and2
```

```
module top_module(
      i_gated,
      i_clock,
      o_result
);

input [15:0] i_gated;
input [15:0] i_clock;
output [15:0] o_result;

my_and2 #(16) AND0 (
      .i_in1(i_gated),
      .i_in2(i_clock),
      .o_out(o_result),

endmodule //top_module
```

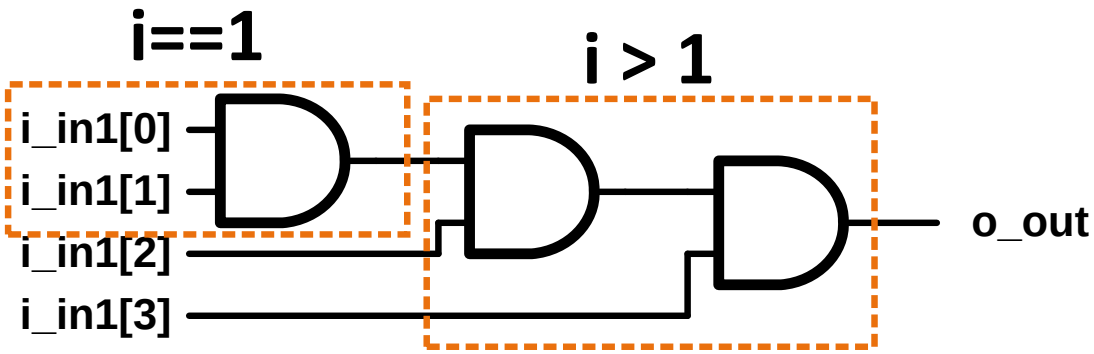
```
module my_and2(
      i_in1,
      i_in2,
      o_out
);
parameter WIDTH = 7;
parameter DOWNTO = 2;

input [WIDTH:DOWNTO] i_in1;
input [WIDTH:DOWNTO] i_in2;
output [WIDTH:DOWNTO] o_out;

assign o_out = i_in1 & i_in2;



endmodule //my_and2
```

```
module top_module(
      i_gated,
      i_clock,
      o_result
);

input [15:0] i_gated;
input [15:0] i_clock;
output [15:0] o_result;

my_and2 #(15,0) AND0 (
      .i_in1(i_gated),
      .i_in2(i_clock),
      .o_out(o_result),

endmodule //top_module
```

```
module my_and2(
      i_in1,
      i_in2,
      o_out
);
parameter WIDTH = 7;
parameter DOWNTO = 2;

input [WIDTH:DOWNTO] i_in1;
input [WIDTH:DOWNTO] i_in2;
output [WIDTH:DOWNTO] o_out;

assign o_out = i_in1 & i_in2;



endmodule //my_and2
```

```
module top_module(
      i_gated,
      i_clock,
      o_result
);

input [15:0] i_gated;
input [15:0] i_clock;
output [15:0] o_result;

my_and2 #(
      .WIDTH(15),
      .DOWNTO(0)) AND0 (
      .i_in1(i_gated),
      .i_in2(i_clock),
      .o_out(o_result),

endmodule //top_module
```

```
module multi_and(
       i_in1,
       o_out
);
  parameter WIDTH = 4;

  input [WIDTH-1:0] i_in1;
  output o_out;
  wire x;

  genvar i;
  generate
    for(i=1; I < WIDTH-1; i=i+1)
    begin:forloop
       if(I ==1)
        assign x[i] = i_in1[0] & i_in1[1];
       else
        assign x[i] = i_in1[i] & x[i-1];
    end
  endgenerate
  assign o_out= x[WIDTH-1];

endmodule //multi_and
```



Testbenches

```
module adder_testbench ();

  reg [7:0] a;
  reg [7:0] b;
  wire [8:0] y;

    my_other_module DUT0 (a,b,y);

    initial
    begin
      a = 0; #10
      b = 0; #10
    end

    always
    begin
      a = a+1; #10;
      $display(“ Adder Output: %d”, y);
      b = a; #10;
      $display(“ Adder Output: %d”, y);
    end

endmodule //adder_testbench
```