Lab#6
Parametrized multiplexer

Digital design principles.

Teacher:
Gloria Castro Muñoz

Work made by:
Christian Aaron Ortega Blanco

Hermosillo, Son. 6 de mayo de 2022.

Instructions

Make a variable input multiplexer module. It must have one parameter to vectorizing the number of entries, that it means that the mux module will not have a limited input number.

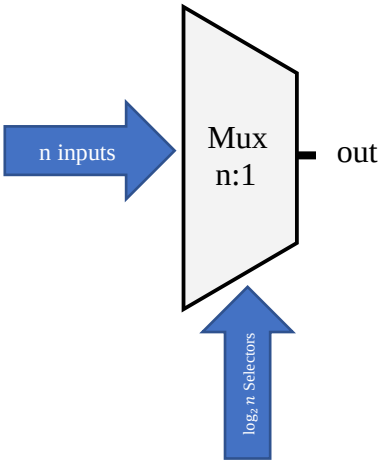

Figure 1 Module mux n:1

Module definition

```
// ------------------------------------------------------------------------------
// INAOE>lab10>parametrized_mux
// ------------------------------------------------------------------------------
// Author : christian Aaron Ortega Blanco
// File   : design.sv
// Create : 2022-05-06 13:59:43
// Revise : 2022-05-06 13:59:43
// ------------------------------------------------------------------------------

/*  ----------module------------------       */
module mux  #(parameter long )(
  input [long-1:0]in,
  input  [$clog2(long)-1:0]selection,
    output reg z
    );
/*       variables          */
  int i;
/*                  main                  */
begin
  always @(selection)
    z=in[selection];
  end


endmodule
/*  ----------module end------------------       */
```

Bench test

```
// ------------------------------------------------------------------------------
// INAOE>lab10>parametrized_mux
// ------------------------------------------------------------------------------
// Author : christian Aaron Ortega Blanco
// File   : testbench.sv
// Create : 2022-05-06 13:59:43
// Revise : 2022-05-06 13:59:43
// ------------------------------------------------------------------------------
//// time var
`timescale 1ns/100ps
/*  ----------module------------------       */
module mux_TB ();
/*         number of inputs           */
parameter long =8;
/*        inputs values               */
  wire [long-1:0]in=8'b01010100;

/*        variables          */
//// Selection
  reg [$clog2(long)-1:0] selection;
//// Outputs
wire z;
int i;
//// assing to entity

 /*  DUT(  [long]inputs , [log(long)]selection inputs , output)  */
mux
  #( long )
  DUT(in,selection,z);

/*                  main                  */
initial
begin
  $dumpvars;
  $dumpfile("dump.vcd");
  ////////////////Secuence////////////////
  for(i=0;i<long;i++)begin
    selection=i;
    #1;
    $display("%b %b %b",selection,in, z);
  end
  ////////////////////////////////////////
  $finish();
end
endmodule;
/*  ----------module end------------------       */
```

Results

To test the module, there are set already an 8bit length input array with random information registered on it, and when the scrip runs, it displays every bit of the array in the output.

```
/*        number of inputs          */
parameter long =8;
/*        inputs values             */
  wire [long-1:0]in=8'b01010100;
```

```
# KERNEL: 000 0
# KERNEL: 001 0
# KERNEL: 010 1
# KERNEL: 011 0
# KERNEL: 100 1
# KERNEL: 101 0
# KERNEL: 110 1
# KERNEL: 111 0
```

In the future this is going to be a quick tool to make big multiplexers without caring about the complexity of making manually.