SYNOPSYS®
Silicon to Software™

Verilog Lab I

December 2023

Basic modules

- This lab will consist on creating some basic designs (modules).
- A description of the design functionality will be provided.
- The objective is to translate the description into a synthesizable Verilog code.
- It is very important to use the module interface that will be provided.
- In order to do a syntax check on the Verilog code written, **vlogan** tool can be used.

```
vlogan -sverilog -debug_access+all verilog_file.v
```

- You can compile the simulation executable with

```
vcs -sverilog -debug_access+all verilog_file.v
```

- Then start the simulation with

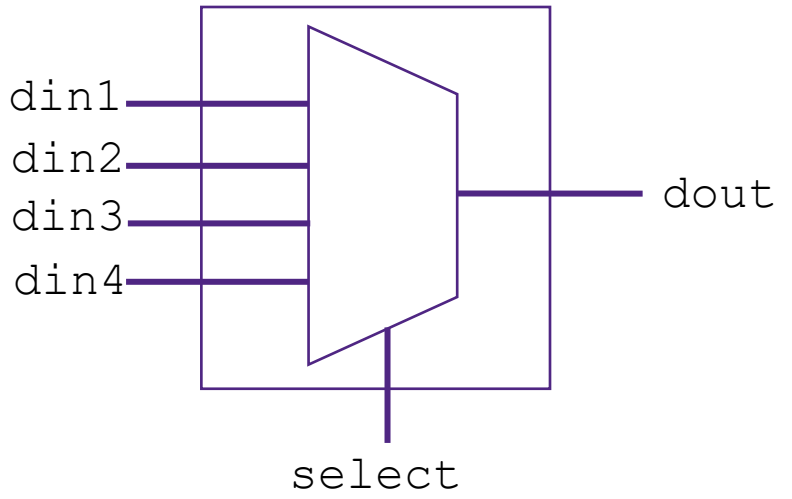
```
./simv
```

Parameterized 4-input MUX

- Write the Verilog code that implements a 4-input multiplexer (mux) with variable (parameterized) input and output bus width.

- Use the following interface

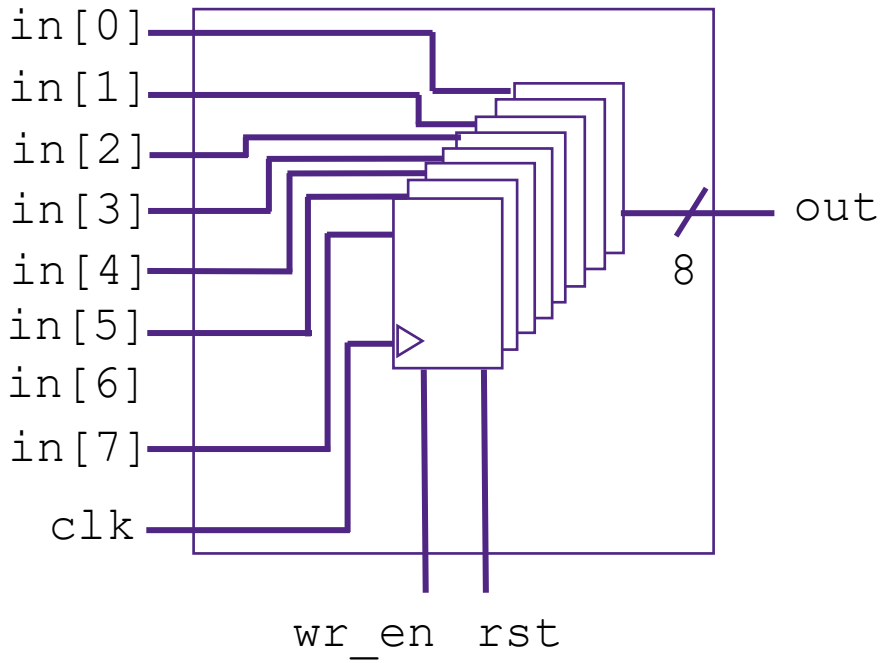
```
1 module mux4 #(
2   parameter WIDTH= 8
3   ) (
4   input  [WIDTH-1:0] din1, din2, din3, din4,
5   input  [1:0]       select,
6   output [WIDTH-1:0] dout
7 );
```



Parameterized Register Bank

- Write the Verilog code that implements a D-type Register Bank with parameterized depth. It should:
  - Operate on positive edge of the clock (`clk`)
  - Have a Synchronous reset (`rst`)
  - Have an enable signal (`wr_en`) which allows data capturing only when asserted.
- Use the following interface

```
1 module register_bank #(
2   parameter WIDTH = 8
3   ) (
4   input  clk,
5   input  rst,
6   input  wr_en,
7   input  [WIDTH-1:0] in,
8   output [WIDTH-1:0] out
9 );
```



How to instantiate a module?

Two ways of port connection (can't be mixed)

Port connection by ordered list

```
1 module myblock ( input a, b, c
2                  output x );
.
1 module top ( );
2  logic a,b,c,x;
3  myblock instancia_name(
4             a,b,c,x);
5 endmodule
```

Port connection by name

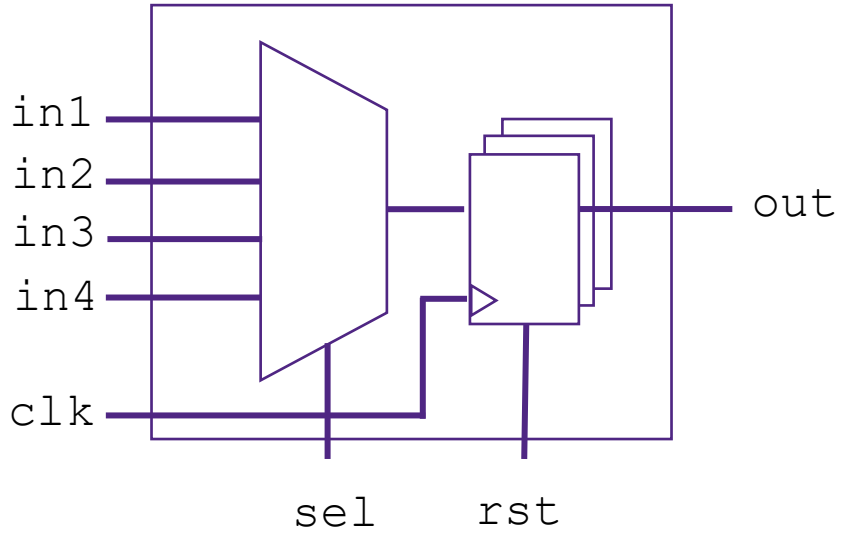
```
1 module myblock ( input a, b, c
2                  output x );
.
1 module top ( );
2  logic P,Q,R,x_top;
3  myblock instancia_name(
4             .x(x_top),
5             .a(P),
6             .c(Q),
7             .b()
8  );
9 endmodule
```

Parameterized 4-input MUX with output register

- Write the verilog code that implements a 4-input multiplexer (mux) with a variable (parameterized) input and output bus width, and synchronous output.
- Only to learn how to instantiate modules. Wil not be used later in the project.

- Use the following interface

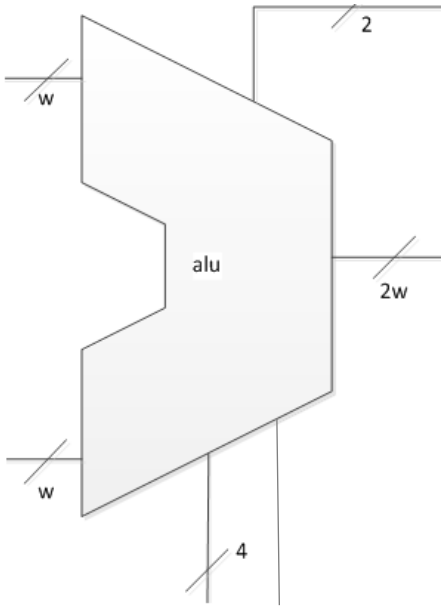
```
module mux4_registered #(
  parameter WIDTH= 8
  ) (
  input  clk,
  input  rst,
  input  [1:0] sel,
  input  [WIDTH-1:0] in1, in2, in3, in4,
  output [WIDTH-1:0] out
);
```



Basic ALU

- Write the Verilog code for a basic ALU module (Arithmetic Logic Unit).
  - This ALU has to have addition, subtraction, multiplication and division operations.
    - It should have two variable (parameterized) width inputs and output bus in accordance.
    - Signal **`zero[0]`** should be asserted when the result of the current operation is 0.
    - Signal **`error[0]`** should be asserted when dividing by 0 is attempted or when input data is **not** valid.
      - On error condition, output must be forced to be **`-1`**
  - Operations should be described using Verilog arithmetic operators.
- Use the following interface

```
module ALU #(
  parameter WIDTH = 8
) (
  input  [WIDTH-1:0] in1, in2,
  input  [3:0] op,
  input  invalid_data,
  output [2*WIDTH-1:0] out,
  output zero,
  output error
);
```

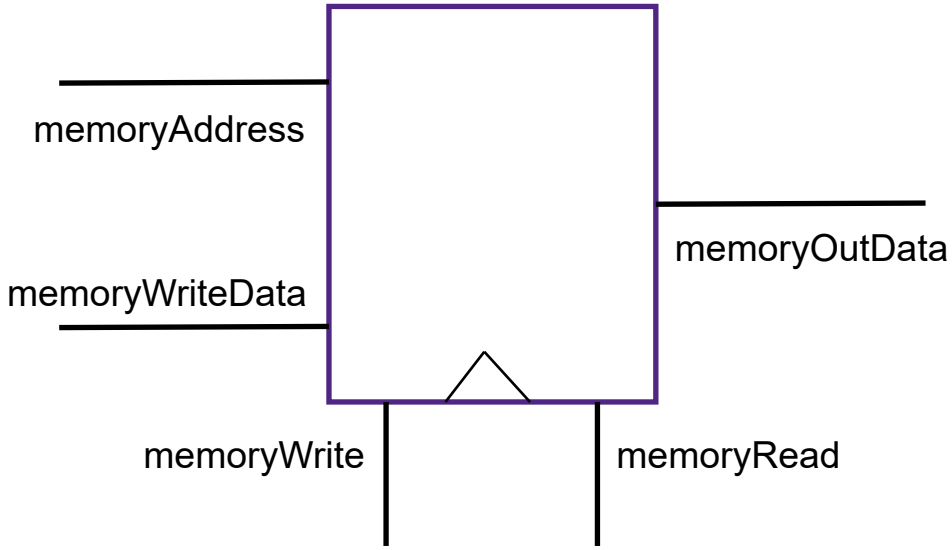


Memory

- Write the Verilog code that implements a parameterized 2*WIDTH per 256 words memory.
- The memory is synchronous for writing data and asynchronous for reading data
- The memory has a pin to enable the writing and a pin to enable the reading

```
module memory #(
    parameter WIDTH = 8
    ) (
    input clk, memoryWrite, memoryRead,
    input [2*WIDTH-1:0] memoryWriteData,
    input [WIDTH-1:0] memoryAddress,

    output [2*WIDTH-1:0] memoryOutData
);
```



SYNOPSYS®
Silicon to Software™

Thank You