SYNOPSYS®
Silicon to Software™

# Verilog Lab II

# State machines

- A description of the design functionality will be provided.
- The objective is to translate the description into a synthesizable Verilog code.
- Use the module interface that will be provided.
- 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

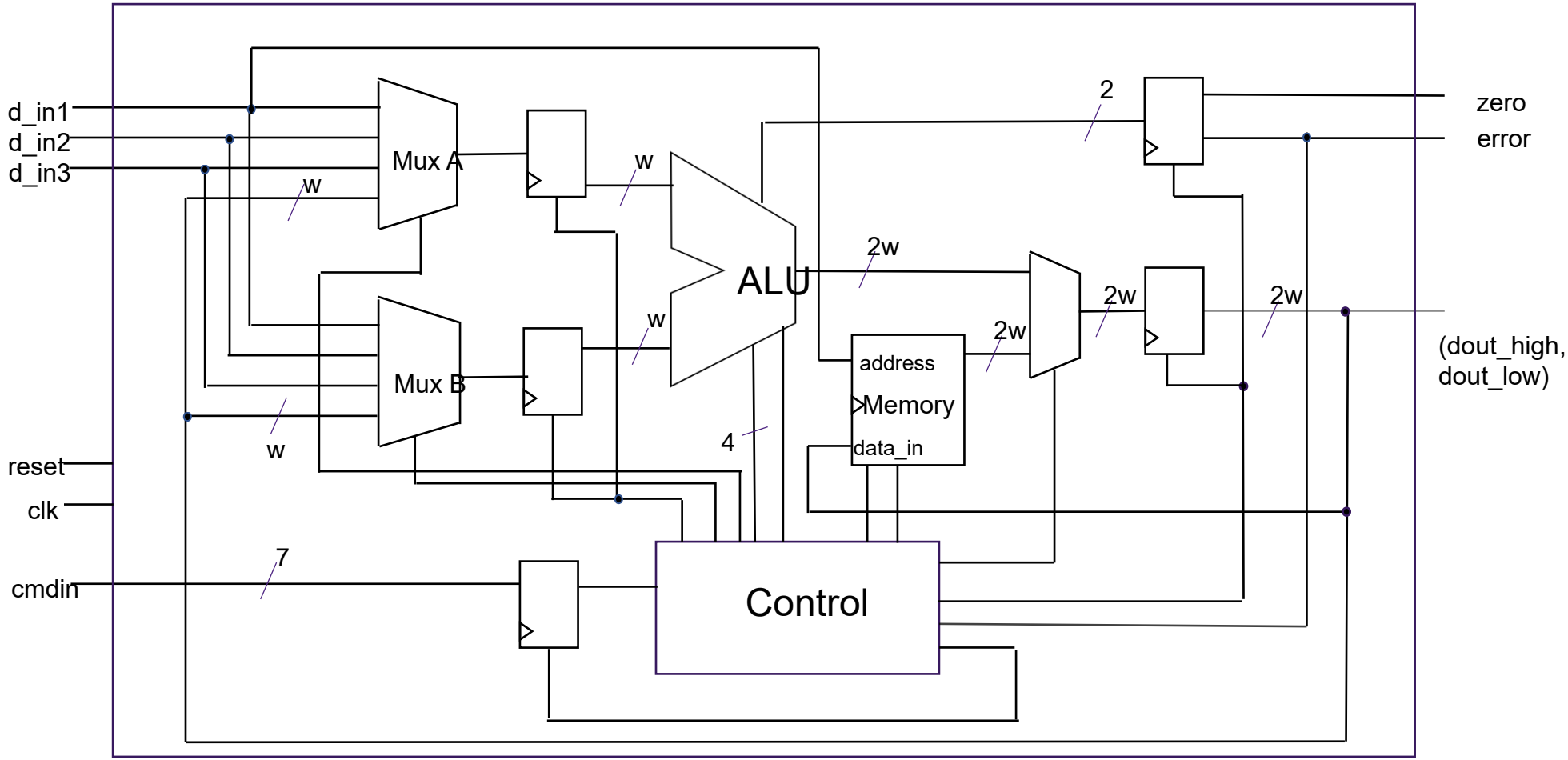
```
vcs -sverilog -debug_access+all -kdb +memcbk verilog_file.v
```

- Then start the simulation with

```
./simv -verdi
```

# Full CPU design

- The objective is to create a basic multicycle processor as shown in the diagram below.
  - Modules previously designed should be used. (`Mux, ALU, Regbank`)

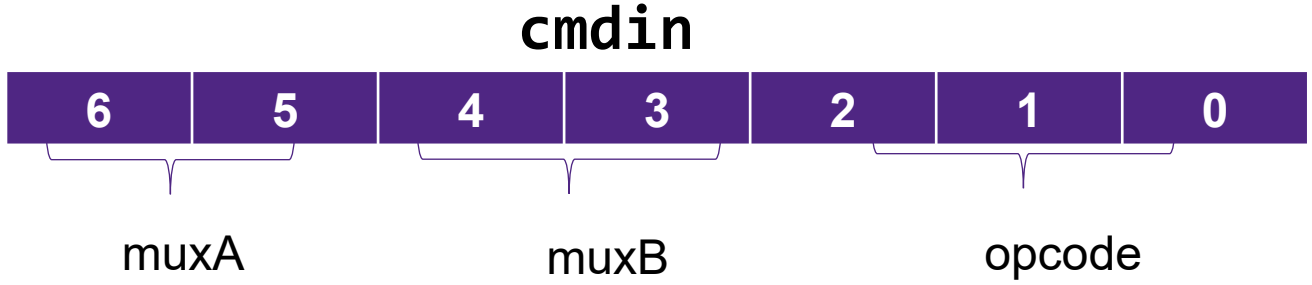


# Full CPU design

- Specifications of the design:
  - This is a multicycle processor (3 stages) with no pipelining.
  - It has 3 variable with input buses to be used as operands.
  - The **`ALU`** has 2 multiplexed inputs:
    - Each **`MUX`** is feed by the 3 input buses and the **`ALU`** output.
    - Details on how to control the **`ALU`** are available in Lab 2d.
  - Both reset (**`rst`**) and clock (**`clk`**) signals should be connected to every module that uses them.
  - It has a control unit feed by a 6 bit word (**`cmdin`**). This control unit has to:
    - Enable the register stages only when applicable
    - Select the outputs of each **`MUX`**
    - Provide the control signals for the **`ALU`**.
      - Inverted valid data signal must be asserted when data is feed from the feedback loop and the previous result was not valid (**`Error`** condition)
      - OPCODE to select the operation performed by the **`ALU`**.

# Full CPU design

- The **`control`** block shall be designed to implement the following Instruction Set Architecture (**`ISA`**):




- The **`opcode`** bus should be decoded according to the following table:

| opcode[2] | opcode[1] | opcode[0] | Operation |
|---|---|---|---|
| 0 | 0 | 0 | **`Add`** |
| 0 | 0 | 1 | **`Sub`** |
| 0 | 1 | 0 | **`Mul`** |
| 0 | 1 | 1 | **`Div`** |
| 1 | 0 | 0 | **`NOP`** |
| 1 | 0 | 1 | **`Load`** |
| 1 | 1 | 0 | **`Store`** |
| 1 | 1 | 1 | **`NOP`** |

# Full CPU design

- The control module
  - Write the Verilog code for implementing this module based on the details and ISA provided previously.
  - Write a test bench to validate this module. It should cover all possible operations, corner cases and exercise all outputs.
  - Its interface should be as follows:

```
module control (
  input clk,
  input rst,
  input [6:0] cmd_in,
  input p_error,
  output aluin_reg_en,
  output datain_reg_en,
  output memoryWrite, memoryRead, selmux2,
  output aluout_reg_en,
  output nvalid_data,
  output [1:0]  in_select_a,
  output [1:0]  in_select_b,
  output [3:0]  opcode
);
```

# Full CPU design

- Full design
  - Write the Verilog code that implements this simple microprocessor. At this point it is mostly about connecting  together all the modules created so far.
  - A test bench will be provided.
  - Its interface should be as follows

```
module top #(
  parameter WIDTH= 8
  ) (
  input clk,
  input rst,
  input [6:0] cmdin,
  input [WIDTH-1:0] din_1,
  input [WIDTH-1:0] din_2,
  input [WIDTH-1:0] din_3,
  output [WIDTH-1:0] dout_low,
  output [WIDTH-1:0] dout_high,
  output zero,
  output error
);
```

SYNOPSYS®
Silicon to Software™

# Thank You