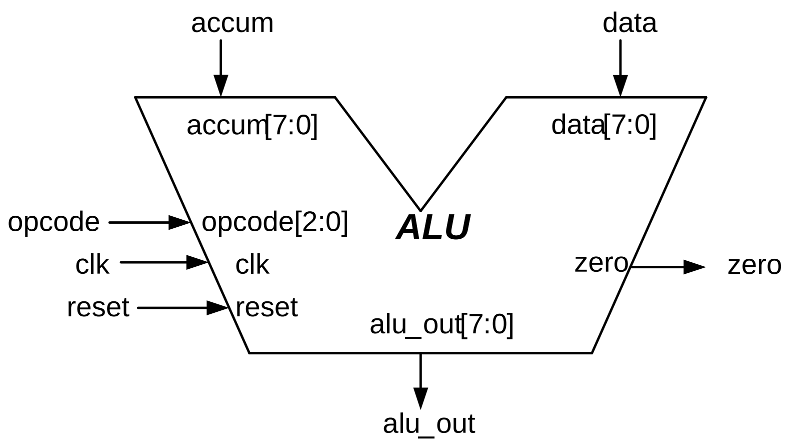
Lab2-1: ALU

- Please modeling an Arithmetic Logic Unit (ALU)
- Specifications
  - ✓ Module name : alu
  - ✓ Input pins : accum[7:0], data[7:0], opcode, clk, reset
  - ✓ Output pins : zero, alu_out[7:0]
  - ✓ Symbol view :



1. 在家目錄底下創建 **學號/Lab2-1** 資料夾，此資料夾內包含測試檔 **alu_test.v** 及設計檔 **alu.v** 。

   unix% cd 學號/Lab2-1

   Create the new "student **ID / Lab2-1**" folder under the home directory. This folder contains the test file, i.e. alu_test.v and the design file, i.e. alu.v.

   unix% cd student ID /Lab2-1

2. 依模組需求撰寫 **alu.v** 檔案。您必須撰寫依照上述的 symbol view 撰寫 port interface，並依照下列的規範撰寫此 alu 的功能描述。

   Write the "alu.v" file according to the requirements of the module. You must write the port interface based on the above symbol view, and please write the functional descriptions of this ALU by using Verilog HDL according to the following specifications.

   I. 所有輸入及輸出（除了「zero」訊號以外）需使用 clock 的正緣（rising edge）來觸發動作。

   All inputs and outputs (except the "zero" signal) must use the rising edge of the clock for trigger.

II. 同步 reset 架構。當 reset 為 1 時表示 reset 啟動，此時 alu_out 訊號輸出為 0。
It`s a synchronous reset architecture. When the "reset" is 1, it means the reset is started, and the "alu_out" output signal is 0 at this time.

III. 當 accum 輸入為 0 時，zero 訊號輸出為 1；而當 accum 輸入不為 0 時，zero 訊號輸出為 0。並且 zero 訊號不需理會 reset 訊號的動作，也不須跟 clock 同步。
When the "accum" input signal is 0, the "zero" output signal is 1. When the "accum" input signal is not 0, the "zero" output signal is 0. The "zero" signal does not need to pay attention to the action of the "reset" signal, and it does not need to be synchronized with the clock.

3. 使用以下 3-bit 的 opcode 訊號值來定義其 alu 的操作模式。當 opcode 輸入為其他任意值(包含 unknow)時，其 alu_out 訊號輸出為 0。
Use the following 3-bit opcodes to define the operation modes of the ALU. When the opcode is any other value (including unknow), the "alu_out" output signal will be 0.

opcode	ALU operation	
000	Pass accum	
001	accum + data	(add)
010	accum - data	(subtraction)
011	accum AND data	(bit-wise AND)
100	accum XOR data	(bit-wise XOR)
101	accum 取二補數	(2's complement)
110	accum*5 + accum/8	
111	假如(accum >= 32)，則 alu_out=data 否則 alu_out=data 取 1 補數 If (accum> = 32), then alu_out = data Otherwise alu_out = data take 1's complement	

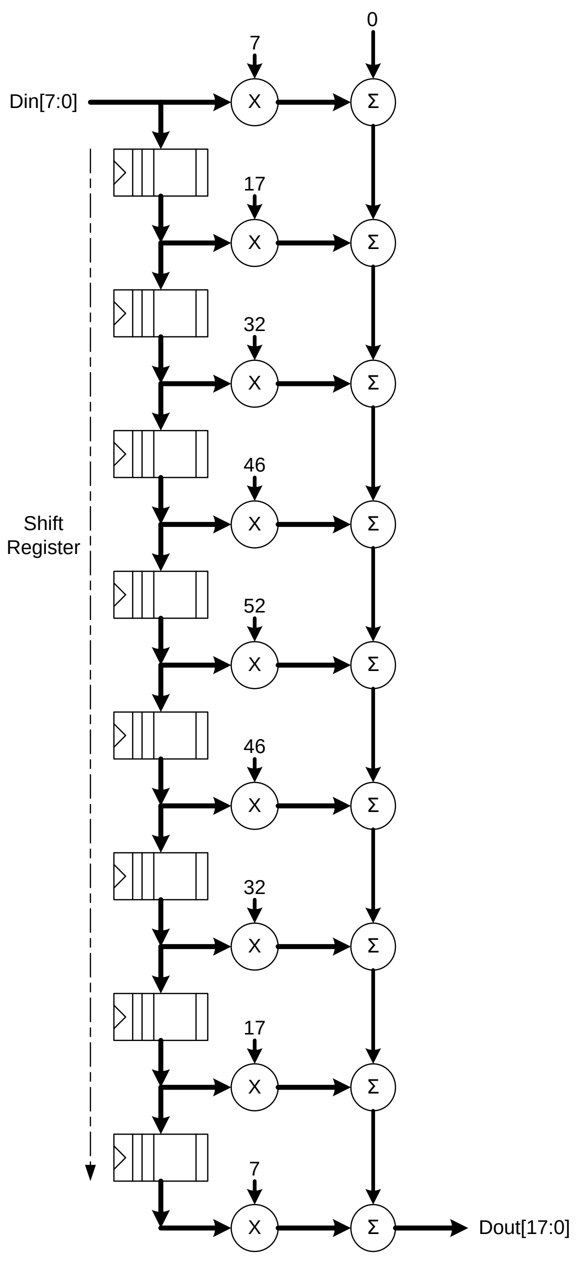
備註：本題所有訊號及運算均視為無號數運算即可，且不需考慮溢位問題
Notes: All signals and operations in this design are based on unsigned operations, and the design does not need to consider the overflow problem.

4. 當您撰寫完成這 alu 模組之後，請使用 **alu_test.v** 檔案作為測試檔來測試此 alu。
   您可使用 **NC-Verilog** 或 **Verilog-XL** 來跑 simulation。
   例如：**unix%** ncverilog alu_test.v alu.v +access+r
   After you finish the design of this alu module, please use "alu_test.v" file as a test-bench to test this alu module. You can use NC-Verilog tool or Verilog-XL tool to execute the functional simulation.
   For example：**unix%** ncverilog alu_test.v alu.v +access+r

5. 本測試檔已完成自動比對的功能，請檢查 simulation 的輸出結果。
   This test-bench file includes the function of automatic comparison for outputs, and please check the simulation results.

Lab2-2 : FIR

- Please modeling a Gaussian lowpass FIR filter
- Specifications
  - ✓ Module name : FIR
  - ✓ Input pins : Din[7:0], clk, reset
  - ✓ Output pins : Dout[17:0]
  - ✓ Symbol view :



1. 在家目錄底下創建 **學號/Lab2-2** 資料夾，此資料夾內包含測試檔 **FIR_test.v** 及設計檔 **FIR.v** 。

   unix%　　cd　　學號/Lab2-2

   Create the new "student ID / Lab2-2" folder under the home directory. This folder contains the test file, i.e. FIR_test.v and the design file, i.e. FIR.v.

   unix%　　cd　　student ID /Lab2-2

2. 依模組需求撰寫 FIR.v 檔案。您必須撰寫依照上述的 symbol view 撰寫 port interface，並依照下列的規範撰寫此 FIR 的功能描述。

   Write the "FIR.v" file according to the requirements of the module. You must write the port interface based on the above symbol view, and please write the functional descriptions of this FIR by using Verilog HDL according to the following specifications.

   I. 同步 reset 架構。當 reset 為 1 時表示 reset 啟動，此時 Shift Register 的輸入皆為 0。因此此時的輸出值為 Din[7:0]乘上 7 的結果。

      It`s a synchronous reset architecture. When the "reset" is 1, it means the "reset" is started, and the inputs of Shift Register are all 0 at this time. At the same time, the output value equals the result of multiplying Din [7: 0] by 7.

   II. 依本題 symbol view 所示可知其輸出訊號為 Shift Register、各階係數及輸入訊號分別進行相乘及累加之後的結果，因此 Dout 輸出不須與 clock 同步。

      According to the symbol view of this design, the output signal is obtained by multiplying and accumulating with Shift Register, the coefficients of each order, and the input signal, respectively. Therefore, the "Dout" output signal does not need to be synchronized with the clock.

   III. 提示 1：您可使用 for 迴圈來撰寫 Shift Register 的動作。

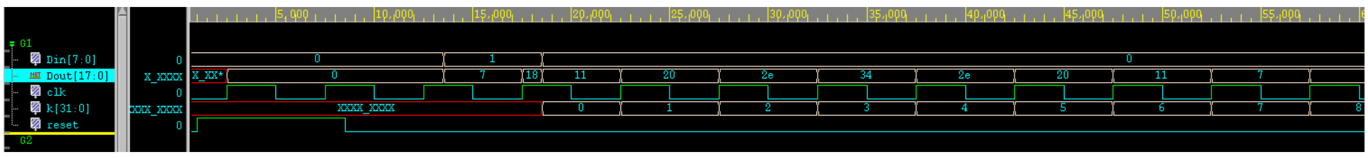
      Tip 1: You can use the "for loop" HDL description to model the actions of Shift Register.

   IV. 提示 2：您可宣告一 Register Array(Memory Array)來代表 Shift Register 架構。

      Tip 2: You can declare a Register Array (i.e. Memory Array) to model the architecture of Shift Register.

3. 當您撰寫完成這 FIR 模組之後，請使用 **FIR_test.v** 檔案作為測試檔來測試此 FIR。
   您可使用 **NC-Verilog** 或 **Verilog-XL** 來跑 simulation。
   例如：**unix%** ncverilog FIR_test.v FIR.v +access+r
   After you finish to build the FIR module, please use the "FIR_test.v" file as a test-bench file to test the FIR module. You can use NC-Verilog tool or Verilog-XL tool to execute the functional simulations.
   For example：**unix%** ncverilog FIR_test.v FIR.v +access+r

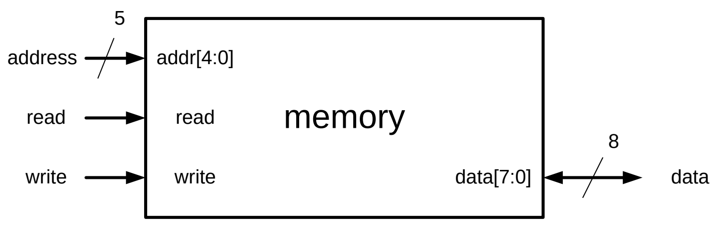
4. 您可參考以下輸出確認輸出結果是否正確。
   You can refer to the following output waveforms to confirm whether the output results are correct or not.



依序在每個 clock 負緣可以抓取到 7 → 11 → 20 → 2e → 34 → 2e → 20 → 11 → 7 → 0 ...... (16 進制)。
At the negative edge of each clock, the design can capture 7 → 11 → 20 → 2e → 34 → 2e → 20 → 11 → 7 → 0 ..... in sequence (by using Hexadecimal format).

Lab2-3 : Memory

- Modeling a Memory with a Bidirectional Data Bus.
- Specifications
  - ✓ Module name : mem
  - ✓ Input pins : addr[4:0], read, write
  - ✓ Inout pins : data[7:0]
  - ✓ Symbol view :



1. 在家目錄底下創建 **學號/Lab2-3** 資料夾，此資料夾內包含測試檔 **mem_test.v** 及設計檔 **mem.v** 。

   unix% cd 學號/Lab2-3

   Create the new “student ID / Lab2-3” folder under the home directory. This folder contains the test file, i.e. mem_test.v and the design file, i.e. mem.v.

   unix% cd student ID /Lab2-3

2. 此 lab 的設計檔 **mem.v** 內容只完成 module header 的部分，特別注意其中 data 訊號是 bidirectional 的。請依照以下規範撰寫 memory 模組的內容：

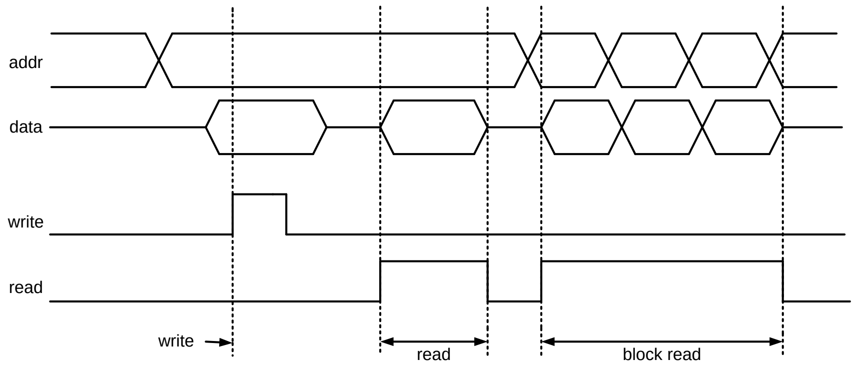
   The content of the design file “mem.v” in this lab only describes the module header part. It notes that the “data” signal is bidirectional. Please write the functional descriptions of the memory module by using Verilog HDL according to the following specifications:

   I. 使用 memory register array，並命名為 memory。此 memory register array 須符合以下規範：

      Use the description of memory register array and name it as “memory”. This memory register array need to meet the following specifications:

      A. The MSB of each word is bit **7**
      B. The LSB of each word is bit **0**
      C. The first address is address **0**
      D. The last address is address **31** (hex 1F)

II. 使用非同步存取方式此 memory model，並使用兩個控制訊號做為 memory model 讀取（read）或寫入（write）的控制。
Use the asynchronous access mode for this memory model, and use two control signals as the “read” or “write” control for the memory model.



III. 在 write 控制訊號的 positive edge 發生時，此時在 data 訊號線出現的資料將會被寫入 memory 中，其寫入位址由當下 addr 訊號線之資料所定義。您可以使用 procedural assignment 來撰寫這部分的 code。
When the positive edge of the “write” control signal occurs, the data appearing on the “data” bus at this time will be written into the memory, and its writing address is defined by the data on the current “addr” bus. You can use procedural assignment to model the function of this part.

IV. 當 read 控制訊號值為 High 時，memory 將以目前 addr 訊號線之資料做為位址，將 memory 內該位址之內容藉由 data 訊號線讀出。
When the “read” control signal value is High, the memory will use the data of the current “addr” bus as the address value, and read out the content at the address in the memory through the data bus.

V. 此 memory 須支援 block read 功能。當 read 控制訊號線維持在 High 的狀態下，若 addr 訊號線之資料改變，則被讀出資料之位址亦會隨之改變，因此可連續讀到不同位址之資料。
This memory needs to support the **block read** function. When the “read” control signal is maintained in the “High” status, if the address value on the “addr” bus changes, the address of the read data will also change. Therefore, the data can be read out continuously from different addresses.

VI. 當 read 控制訊號值為 Low 時，其 read 控制將呈現不致能的狀態，在此情況下 read 控制訊號對 data 訊號線將呈現 high-impedance 的狀態。亦即在此情形下若 write 控制訊號值為 high 的話，則寫入的動作將可正常執行。
When the “read” control signal is “Low”, the read control will be disabled. In this case, the “read” control signal will be the state of high-impedance. That is, in this case, if the “write” control signal is “high”, the write operation can be performed normally.

3. 使用 **mem_test.v** 檔來測試 **mem.v** 設計檔。執行完 simulation 後，您會發現有 error information 出現。請查看 **mem_test.v** 檔案內容，並思考以下問題：
Use the “mem_test.v” file to test the “mem.v” design file. After performing the simulation, you will find error information appears. Please check the content of the “mem_test.v” file and think about the following questions:

I. data 訊號線為一 bidirectional port，其 data type 宣告為 wire
The “data” signal (or bus) is a bidirectional port, and its data type is declared as “wire”.

II. 當在 procedural block 中，我們將資料值設給 data 訊號線時，此 data 訊號線是否應該宣告為 reg 的 data type。
When we set the data value to the “data” signal in the procedural block, whether this “data” signal should be declared as the data type of “reg” or not.

4. 為修正上述 error information 的問題，請修改 **mem_test.v** 檔案內容後再跑一次 simulation 看看，直到 error information 消失為止。
In order to correct the above error information, please modify the content of the “mem_test.v” file and execute the functional simulation again until the error information disappears.

提示：您可在 procedural assignment 中使用 shadow register 來修正。
Tip: You can use shadow register to correct the issue in procedural assignment.

i. procedural assignment 內容主要用在 write 的情況下，您可對 procedural block 內的 data 訊號線，另外宣告使用一個 register。
Procedural assignment is mainly used in the write case. You can declare a register for the “data” signal in the procedural block.

ii. 使用 continuous assignment 及 conditional operator 來決定當 read 不啟動時，可將 procedural block 內的 register 內容寫入到 memory。若 read 啟動時，表示 memory 要做讀取的動作，因此 continuous assignment 的結果須為 High-impednace 以避免互相衝突。
Use continuous assignment and conditional operator to determine that when the “read” signal is not started, you can write the contents of the register in the procedural block into the memory.
If the “read” signal is started, it means that the memory module is ready for reading. Therefore, the result of continuous assignment must be High-impedance to avoid the data conflicts.

5. 請使用 mem_test.v 檔案及 mem.v 檔案執行 simulation，並確定其功能正常。
Please use the “mem_test.v ” file and “mem.v” file to execute the functional simulations and make sure that it works functionally.

```
Setting all memory cells to zero…
Reading from one memory address…
Setting all memory cells to alternating patterns…
Doing block read from five memory addresses…

Completed Memory Tests With 0 Errors!
```