EECS-LAB IC LAB – Summer 2023

Lab01 Exercise

Design: Code Calculator

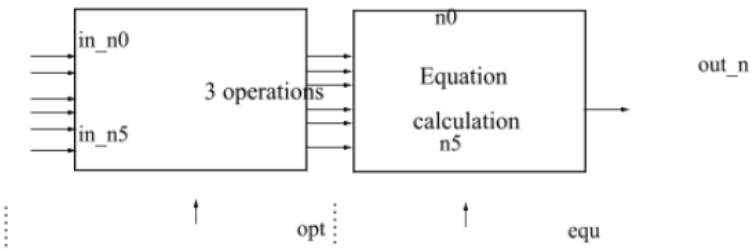
Data Preparation

1. Extract files from TA's directory: **% tar xvf ~train_ta/lab01.tar**

Design Description and Examples

At the final stage of NCTU Millionaire, you are asked to answer a question based on a series of simple mathematic operations. The only challenge is the remaining time, 20 ns. If you answer it in time correctly, you will win a prize of a million dollar. "Ready… Start…".

You will receive a sequence with 6 numbers {**in_n0, in_n1, in_n2, in_n3, in_n4, in_n5**} a **3-bit opt** signal and a **1-bit equ** signal. Then you should calculate the result in the following order:



First, please do the 3 possible operations indicated by **opt** signal in the following order:

1. Signed/Unsigned	If **opt[0] is 1**, the 6 numbers will be regarded as **2's complement signed** value, which means that there MSB is signed bit. For example, in_n0=4'b1010, then its value is -6. in_n0=4'b0010, then its value is 2 If **opt[0] is 0**, the 6 numbers will be regarded as **unsigned** value. For example, in_n0=4'b1010, then its value is 10 in_n0=4'b0010, then its value is 2
2. Sort	If **opt[1] is 1**, sort the sequence from the **largest to the smallest** For example, {2, -1, 3, 5, 6, 4} becomes {6, 5, 4, 3, 2, -1}. If **opt[1] is 0**, sort the sequence from the **smallest to the largest**. For example, {2, -1, 3, 5, 6, 4} remains {-1, 2, 3, 4, 5, 6}.

3. Cumulate/ Normalization	If **opt[2] is 1,** indicates to do accumulation with the rule like moving average. The ratio of old value to new value is 2:1. (round-down the answer if it is not integer) e.g. original series: 5, 3, -7, 0, 3, 6 1st number of new series: (5 * 2 + 5 * 1) / 3 = 5 2nd number of new series: (5 * 2 + 3 * 1) / 3 = 4 (round down) 3rd number of new series: (4 * 2 + -7 * 1) / 3 = 0 (round down) 4th number of new series: (0 * 2 + 0 * 1) / 3 = 0 5th number of new series: (0 * 2 + 3 * 1) / 3 = 1 (round down) 6th number of new series: (1 * 2 + 6 * 1) / 3 = 2 (round down) → After moving you'll get : 5, 4, 0, 0, 1, 2 If **opt[2] is 0,** make the first number to 0 by shifting the whole sequence. For example, the original sequence is {2, 1, 8, 9, 3, 4}. It will shift two intervals left for each number, and the sequence will be {0, -1, 6, 7, 1, 2}.

After these three operations, you will get a sequence {**n0, n1, n2, n3, n4, n5**}. Finally, the output answer can be obtained by one of the following equations

1. Eq0 : ((n3 + n4*4) * n5) / 3 (round-down the answer if it is not integer) For example, if result is -3.75, round down to -3, if result is 5.5, round down to 5
2. Eq1 : | (n5 * n1) –(n5*n0) |

The summary of the description and specifications are as followings:

Input Signal	**Bit Width**	**Description**
in_n0	4	The first number of code. If opt[0] is 0, ranged from **0~15** If opt[0] is 1, ranged from **-8~7**.
in_n1	4	The second number of code. If opt[0] is 0, ranged from **0~15** If opt[0] is 1, ranged from **-8~7**.
in_n2	4	The third number of code.

		If opt[0] is 0, ranged from **0~15** If opt[0] is 1, ranged from **-8~7**.
in_n3	4	The forth number of code. If opt[0] is 0, ranged from **0~15** If opt[0] is 1, ranged from **-8~7**.
in_n4	4	The fifth number of code. If opt[0] is 0, ranged from **0~15** If opt[0] is 1, ranged from **-8~7**.
in_n5	4	The sixth number of code. If opt[0] is 0, ranged from **0~15** If opt[0] is 1, ranged from **-8~7**.
opt	3	Operator for different mode. The operation will be encode as following: opt[0]: 1: Signed. 0: Unsigned opt[1]: 1: Sort from L->S. 0: Sort from S->L. opt[2]: 1: Cumulate 0: Normalize
equ	1	Decide which equation to be calculated. . If equ is 0 : Eq0 If equ is 1 : Eq1

Output Signal	**Bit Width**	**Description**
out_n	10	The answer. Ranged from **-512~511**

Examples:

1. **Initial numbers {4'b0110, 4'b0001, 4'b0100, 4'b0111, 4'b1100, 4'b1111} with opt = 3'b000, equ=1'b1:**

   {4'b0110, 4'b0001, 4'b0100, 4'b0111, 4'b1100, 4'b1111} $-(unsigned)->$ {6, 1, 4, 7,12,15} $-(sort)->$ {1,4,6,7,12,15} $-(Normalize)->$ {0,3,5,6,11,14} $-(Eq1)$

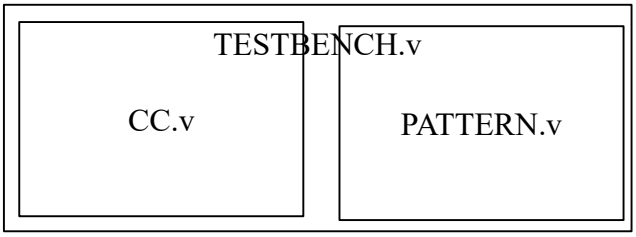
Inputs& Outputs

1. The input signals in_n0, in_n1, in_n2, in_n3, in_n4, and in_n5 are 4-bit inputs
2. The input signal **opt** is a 3-bit input indicated whether to do the operations and which equation to use to get the final result and the input signal **equ** is a 1-bit input indicated which equation to be used to calculate.
3. The output signal **out_n** is a signed number ranged from **-512~511**. This represents the correct password.

Specifications

1. Top module name : CC (File name: CC.v)
2. Input pins : in_n0, in_n1, in_n2, in_n3, in_n4, in_n5, opt, equ
3. Output pins : out_n

Block Diagram



Grading Policy

The performance is determined by the area of your design. The less area your design has, the higher grade you get. Try to reach better performance by thinking your architecture before coding.

Function Validity: 70%

Performance: area 30%

Note

1. Please upload the following file on e3 platform before **23:59** on **Mar. 6**:
   CC_iclab???.v (?? is your account no.) Ex: CC_iclab099.v
   If your file violates the naming rule, your will lose 5 point. Be careful about all detail!
2. Don't use any wire/reg/submodule/parameter name called *error*, *congratulation*, *latch* or *fail* otherwise you will fail the lab. Note: * means any char in front of or behind the word. e.g: error_note is forbidden.
3. After synthesis, check the "CC.area" and "CC.timing" in the folder "Report". **The area report is valid only when the slack in the end of "CC.timing" is "MET".**
4. The synthesis result **cannot** contain any **latch**.
   Note: You can check if there is a latch by searching the keyword "**Latch**" in 02_SYN/**syn.log**
5. Template folders and reference commands:
   In RTL simulation, the name of template folder and reference commands is: 01_RTL:
   "**./01_run"**
   02_SYN/ (Synthesis):
   ./01_run_dc
   (Check **latch** by searching the keyword "**Latch**" in 02_SYN/**syn.log**)
   (Check the design's timing in /Report/CC.timing)
   (Check the design's area in /Report/CC.area)
   03_GATE_SIM/:
   ./01_run
   You can key in **./09_clean_up** to clear all log files and dump files in each folder

Example Waveform

Input and output signal: