NCKUES

DIGITAL INTEGRATED CIRCUIT DESIGN
2021 fall

Final project
Exponential-Golomb Decoder

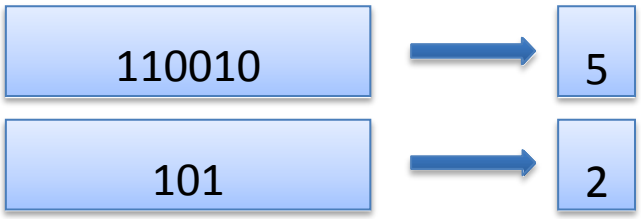
Professor : Wen-Long Chin
TA : Pin-Wei Chen

VSP LAB

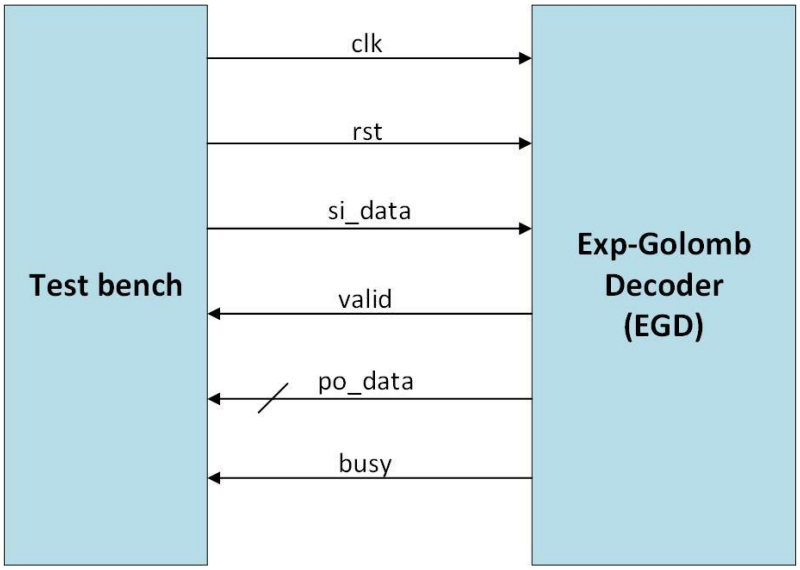
Exponential-Golomb Decoder

Design Description

Golomb coding is a lossless data compression method, invented by mathematician Solomon W. Golomb in the 1960s. Exp-Golomb coding is a variant of Golomb coding. Exp-Golomb coding is more suitable for small number encoding, because it uses a shorter code length to encode smaller numbers, and a longer code length encodes larger numbers. This question is to design a circuit which can decode an Exp-Golomb bit-stream.



Block Diagram

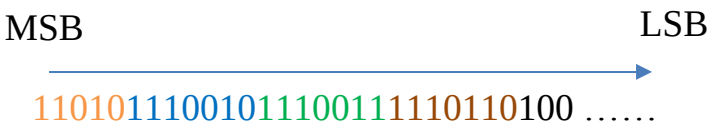


Specifications

Top module name : EGD (File name: EGD.v).
Input ports: si_data, clk, rst .
Output ports: po_data [3:0], valid.
All the outputs are synchronized at clock positive edge.
It is asynchronous-reset architecture.

- Functionality

There are total 512 numbers need to be decoded. After the reset is completed, the testbench will input the bit-stream sequentially (si_data), that is, one bit will be input in each clock cycle,and from MSB to LSB.



All you have to do is to decode the serial bit-stream into 512 integers between 0 and 14 and output them sequentially.

Bit-string (si_data)	Number (po_data)
0	0
100	1
101	2
11000	3
11001	4
11010	5
11011	6
1110000	7
1110001	8
1110010	9
1110011	10
1110100	11
1110101	12
1110110	13
1110111	14

An Exp-Golomb number (bit-string) can be divided into two parts. The first part is prefix (the red part in the above table), the second part is offset (the black part in the above table).

The decode steps are as follows:

1. Read in the bit stream in each clock cycle until 0 appears, these binary number is the prefix part. Then count the number of 1 in prefix part and record it as m;

2. Then continue to read in next m bits, which is the offset

3. The final decoded number is:

$$\text{number} = 2^m - 1 + ooooooset$$

Ex.

si_data = 1 1 0 1 0 1 1 1 0 0 1 0 1 1 1 0 0 1 1 1 1 1 0 1 1 0 …

$m_1 = 1+1 = 2$
$offset_1 = 10_2 = 2$
$po\_data_1 = 2^2 - 1 + 2 = 5$

$m_2 = 1+1+1 = 3$
$offset_2 = 010_2 = 2$
$po\_data_2 = 2^3 - 1 + 2 = 9$

$m_3 = 1+1+1 = 3$
$offset_3 = 011_2 = 3$
$po\_data_3 = 2^3 - 1 + 3 = 10$

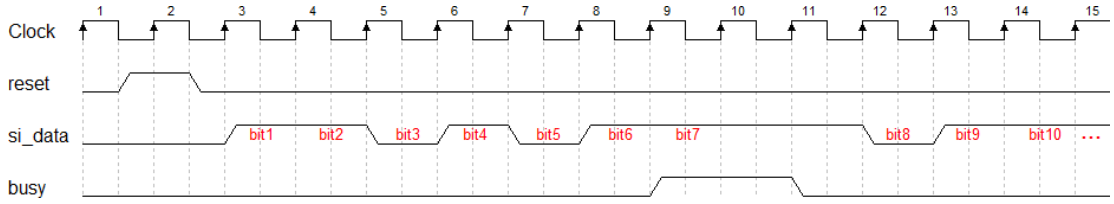
…

Note that the output order of the decoded bit-stream is also sequentially. And When you are outputting the decoded value (po_data), the valid signal also need to be pulled up (set to 1) at the same time.

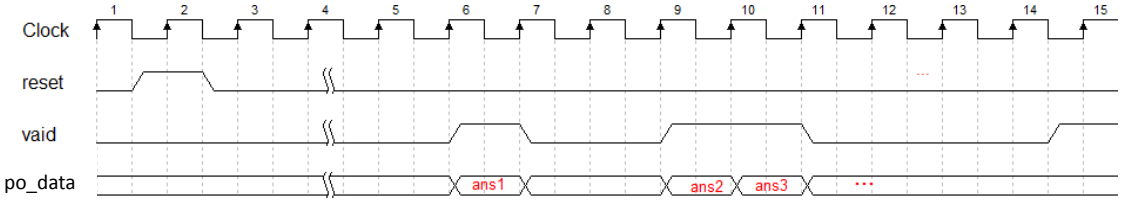
Note that the busy signal is used to control the input data flow. When the busy signal is 0, the bit-stream will be continuously input at each posedge clock. And when the busy signal is 1, the input data will be suspended until busy signal is set to 0 again.

- waveform

Input:



Output:

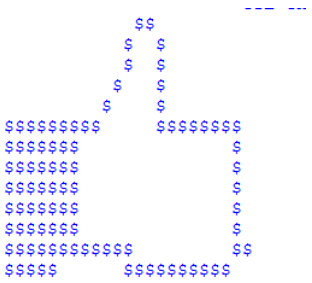


busy 訊號可以控制 testbench 輸入資料，valid 訊號用來告訴 testbench 你的電路正在輸出答案，也就是 testbench 會在每個 valid 為 1 時依序判斷你輸出的答案是否正確，所以輸出答案時 valid 務必同時拉為 1，且順序請遵照題目所要求，如何控制 busy 及 valid 訊號端看各位如何設計。

Hint: Finite State Machine、serial-in to parallel-out (SIPO)

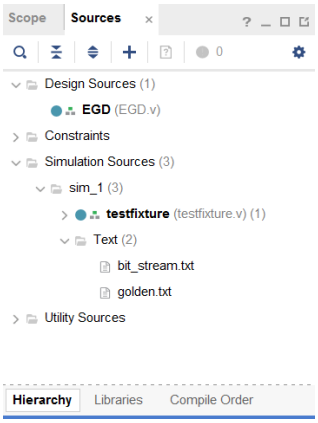
Note

如通過測試會出現下圖：



模擬

1. 請記得把 bit_stream.txt(輸入測資)、golden.txt(標準答案)加進 simulation source



2. 跑模擬時請記得按 Run all

- 評分標準依以下規定。

    1. A 等級為完成測試樣本之所有 RTL simulation 和

       Post-synthesis Functional simulation 和

       Post-synthesis Timing simulation
    2. B 等級為完成測試樣本之所有 RTL simulation
    3. C 等級為完成測試樣本 50%以上(含)之 RTL simulation
    4. D 等級為完成測試樣本 50%以下之 RTL simulation

繳交方式

- 上傳 EGD.v 至 moodle，未繳交者以 0 分計。
- 上傳 LUT 及 FF 個數截圖 (optional)
- 上傳通過 simulation 截圖 (optional)

可將檔案壓縮成.zip 再上傳至 moodle

要看自己花費多少 FF(Flip-flop)，合成完後點選左

方Open Synthesized Design -> Desgin Runs -> FF