



# DCS Lab 6

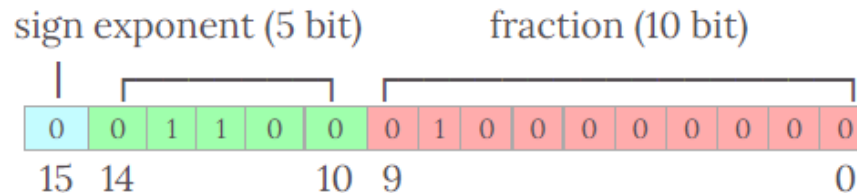
## Floating Point Computation

---

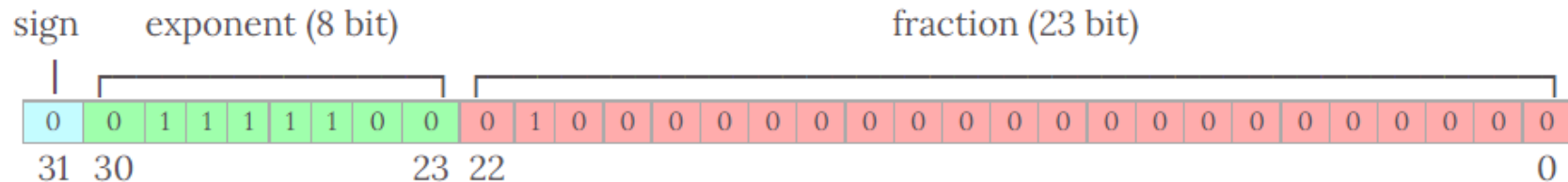
Arithmetic  
徐志嘉

# Floating Point – bfloat16

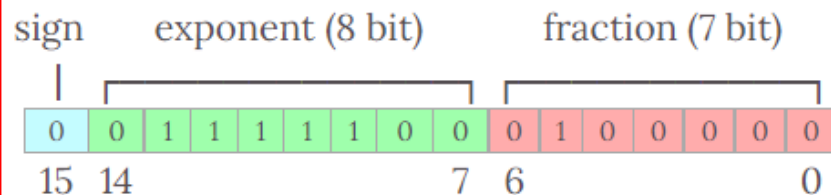
## IEEE half-precision 16-bit float



## IEEE 754 single-precision 32-bit float

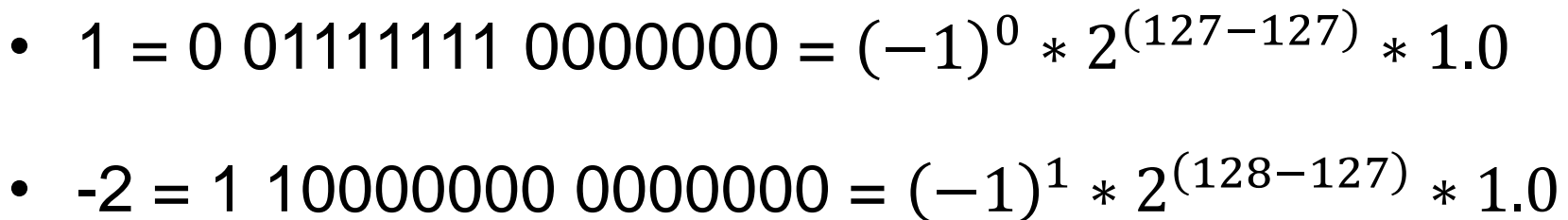


## bfloat16



Use this format

- Sign: 0代表正，1代表負
- Bias Exponent: 2的幾次方
- Fraction: 小數點後的值 (1.
- 公式:  $(-1)^{sign} \times 2^{(exponent)}$
- Ex:



# Floating Point – 加法

1. 還原成1.Fraction (8-bit)
2. 對齊小數點 (8-bit if 右移小的數對齊大的數)
3. 如果是負的( $\text{Sign}==1$ )話取補數( $\sim a+1$ ) (signed 9-bit)
4.  $\text{sum} = a + b$  (signed 10-bit)
5. 轉成bfloat16格式
  - 判斷正負( $\text{sum}[9]$ )，可得Sign
  - 找到第一個1的位置，四捨五入(optional)，可得Fraction
  - 根據小數點移動多少，就把大的數的Exponent加減多少

# Floating Point – example 1

• Ex:  $a + b$

$$= 0\ 01111111\ 0101101 + 0\ 10000000\ 1010101$$

$$= 1.0101101 * 2^0 + 1.1010101 * 2^1$$



$$\cong 0.1010111 * 2^1 + 1.1010101 * 2^1$$



$$= 10.0101100 * 2^1$$

$$\cong 1.0010110 * 2^2$$

$$= 0\ 10000001\ 0010110$$

$$\begin{array}{r}
 00.1010111 * 2^1 \\
 + 01.1010101 * 2^1 \\
 \hline
 010.0101100 * 2^1
 \end{array}$$

Fraction   
 First 1 

 小數點左移一位  
 四捨五入 (0捨1入)

$$\text{out exponent} = \text{Max}(a \text{ exponent}, b \text{ exponent}) + 1 = 2$$

# Floating Point – example 2

• Ex:  $a + b$

$$= 1\ 01111100\ 0011001 + 0\ 01111001\ 1010000$$

$$= -1 * 1.0011001 * 2^{-3} + 1.1010000 * 2^{-6}$$

$$\cong 10.1100111 * 2^{-3} + 00.0011010 * 2^{-3}$$

$$= 11.0000001 * 2^{-3} \qquad \begin{array}{l} 10.1100111 * 2^{-3} \\ + 00.0011010 * 2^{-3} \\ \hline \end{array}$$

$$= -1 * 00.1111111 * 2^{-3}$$

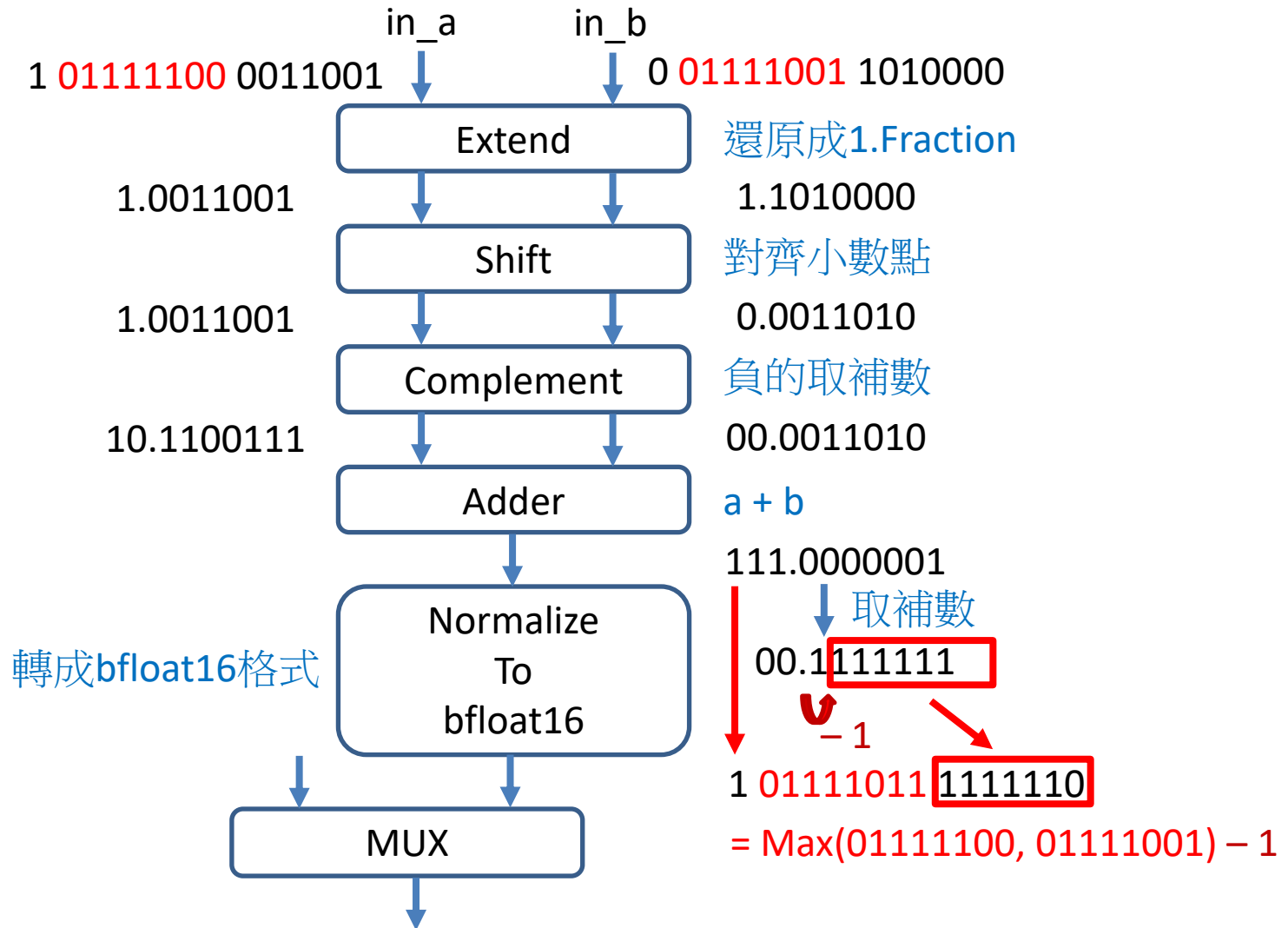
$$= -1 * 1.1111110 * 2^{-4} \qquad \begin{array}{l} \text{Fraction不足7-bit} \\ \text{補0或} \ll \end{array} \qquad \begin{array}{l} 111.0000001 * 2^{-3} \\ \rightarrow \end{array}$$

$$= 1\ 01111011\ 1111110 \qquad = -1 * 00.1\boxed{1111111} * 2^{-3}$$

小數點右移一位

out exponent =  $\text{Max}(a \text{ exponent}, b \text{ exponent}) - 1 = -4$

# Lab – FP + block diagram



# Floating Point – 乘法

1. 還原成1.Fraction (8-bit)
2.  $a * b$  (16-bit)
3. 轉成bfloat16格式
  - 判斷正負(a Sign和b Sign) , 可得Sign
  - 找到第一個1的位置 , 四捨五入(optional) , 可得Fraction
  - $\text{Exponent} = a \text{ Exponent} + b \text{ Exponent} - 127$  (+1 if .左移1)



# Floating Point – example

• Ex:  $a * b$

$$= 0\ 01111111\ 0101101 \times 0\ 10000000\ 1010101$$

$$= 1.0101101 * 2^0 \times 1.1010101 * 2^1$$

$$= 10.0011111110001 * 2^{(0+1)} \cong 10.010000 * 2^1$$

$$= 1.0010000 * 2^2$$

$$= 0\ 10000001\ 0010000$$

$$\begin{array}{r} 1.0101101 \\ \times 1.1010101 \\ \hline \end{array}$$

Fraction

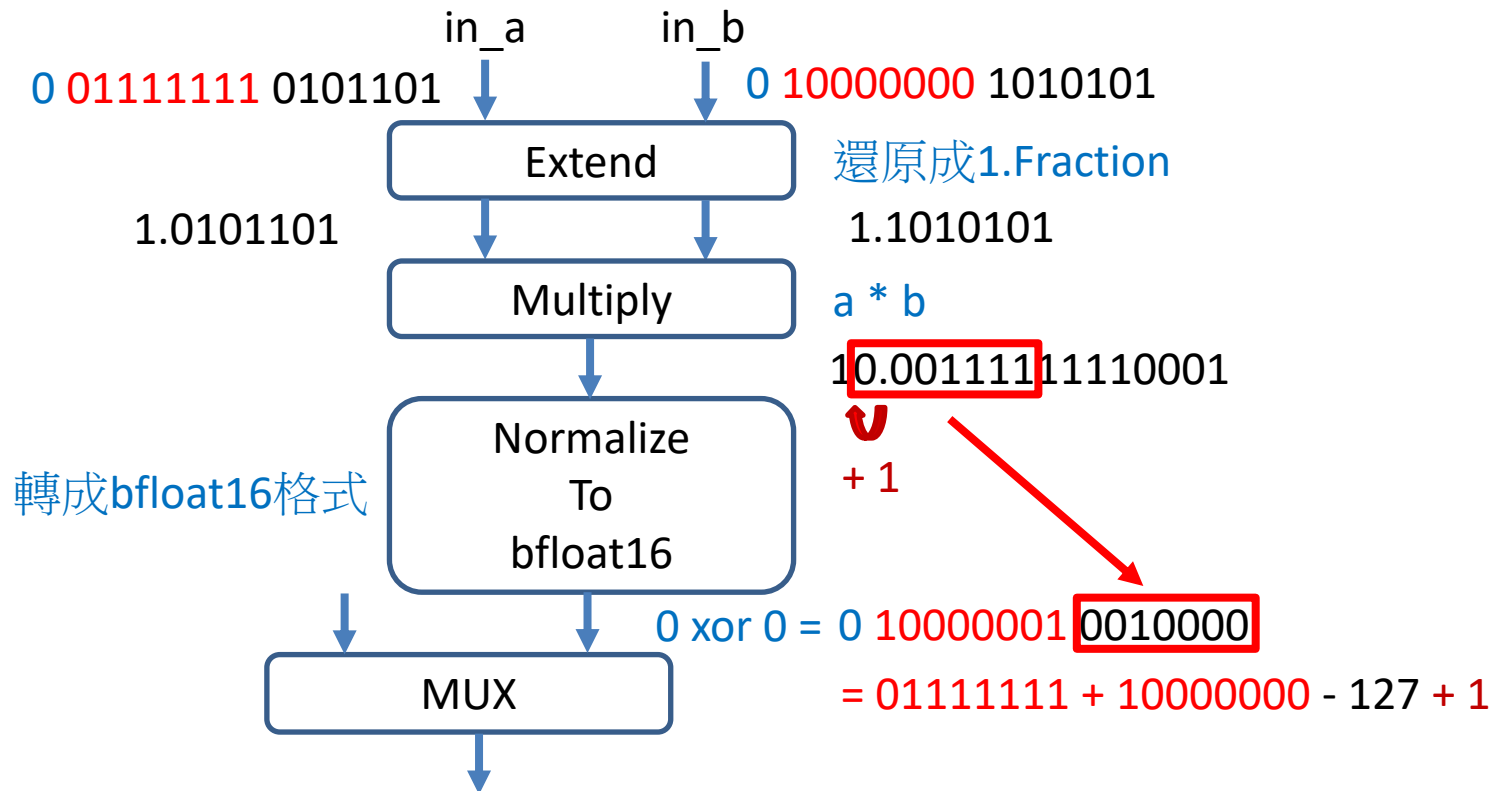
First 1

小數點左移一位

$$\text{out exponent} = 127 + 128 - 127 + 1 = 2 + 127$$

四捨五入  
(0捨1入)

# Lab – FP \* block diagram



# Lab

- $\text{mode} == 0, \text{out} = a + b$
- $\text{mode} == 1, \text{out} = a * b$
- a的Exponent範圍在135到120之間
- b的Exponent範圍在a的Exponent加減3
- 容許有  $|\text{your\_out} - \text{correct\_out}| < |\text{correct\_out}| * 0.1$   
的誤差 (所以不做四捨五入也會過)

# Fpc.sv

Input Signal	Bit width	Definition
clk	1	20ns clock signal
rst_n	1	Asynchronous negative edge reset signal
in_valid	1	Pulled high during inputs
mode	1	0 means $a+b$ , 1 means $a*b$
in_a	16	bfloat16 input
in_b	16	bfloat16 input

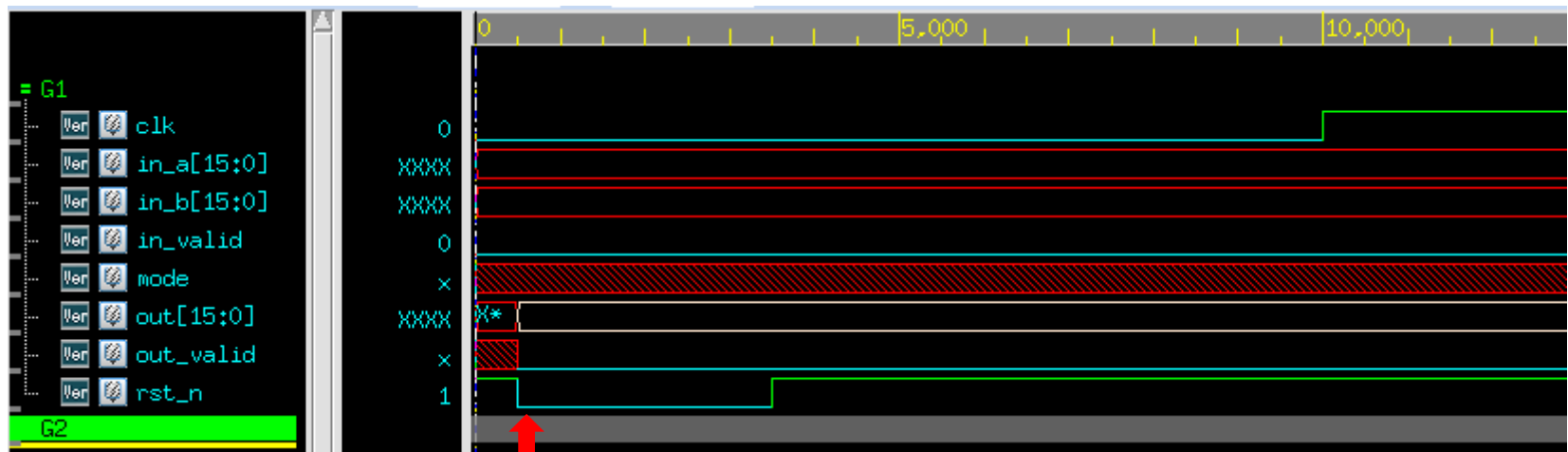
Output Signal	Bit width	Definition
out_valid	1	Pulled high 1 cycle during output
out	16	bfloat16 output

✗ All output signals should be reset to make sure it's not unknown

# Spec

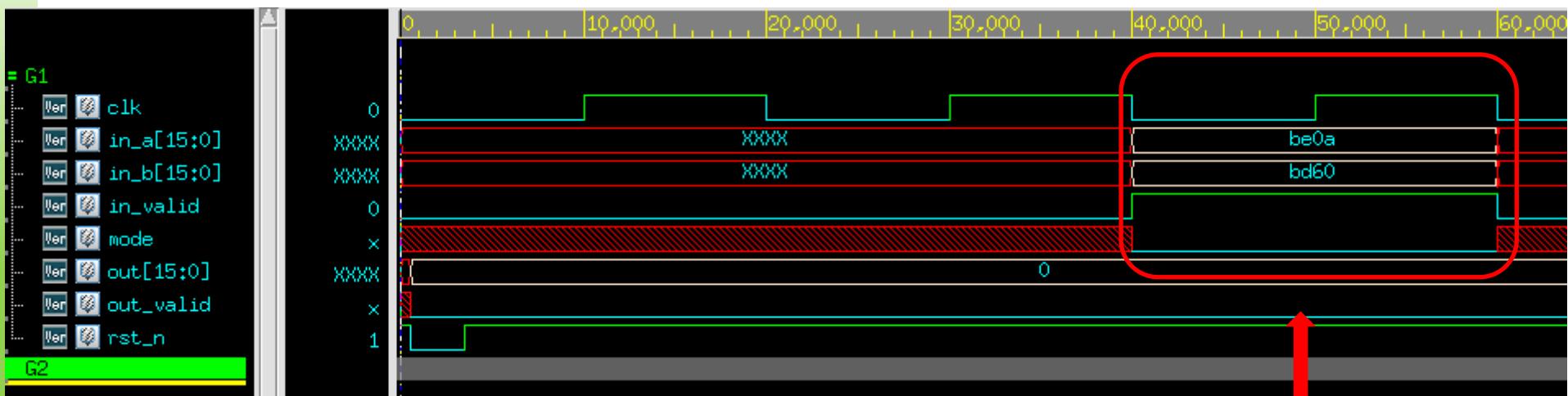
- 不可以用IP，用IP視同FAIL
- 不可以超過30個cycles沒有output(從invalid開始數)
- out\_valid只能拉起一個cycle，之後pattern會檢查out值
- out\_valid放下後，out要reset
- Next input會在out\_valid後2~5個cycle
- 所有output必須非同步負準位reset。
- 01\_RTL 需要PASS。
- 02\_SYN不能有error跟latch。
- 02\_SYN時間timing slack必須為MET。
- 03\_GATE需要PASS。

# Waveform



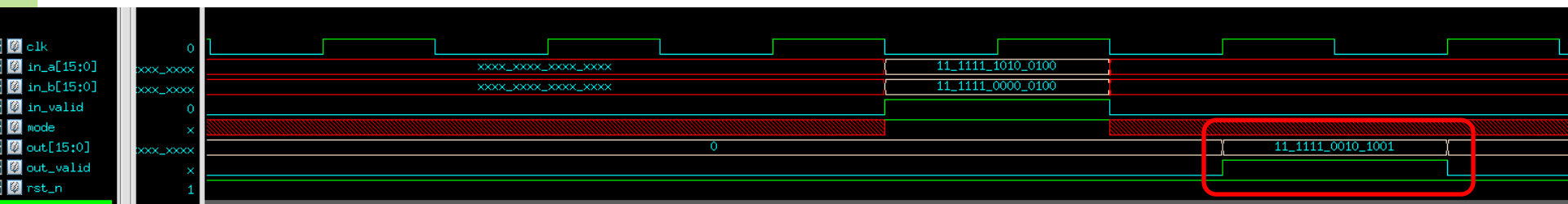
negative trigger asynchronous reset

# Waveform



Input data  
mode == 0  
out = a + b

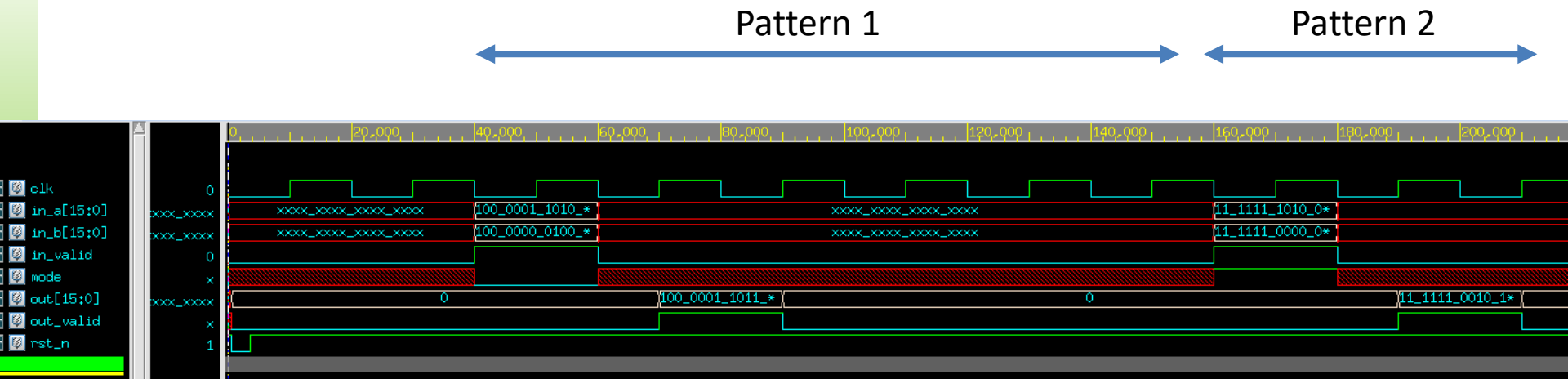
# Waveform



out\_valid & out pull high 1 cycle



# Waveform



# Command

- `tar -xvf ~dcsta01/Lab06.tar`
- `cd Lab06/01_RTL/`
- Need 02\_SYN
  - No Latch
  - No error
  - No timing violation (MET)
- Need 03\_GATE
- Separate combinational and sequential blocks

Demo1: 4/14(四), 16:25:00

Demo2: 4/14(四), 23:59:59