Floating-point

Van Duc NGUYEN

Viettel IC Design

Over View

1. Introduction
2. Floating-point Representation
3. IEEE 754 floating-point standard for binary
4. Floating-point Arithmetic
5. Implementation in FPGA

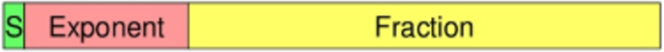
1.Introduction

Why Floating-point?

1. Advantages
   - Dynamic range
     with fixed-point $DR_{fxpt} = r^n - 1.$
     with floating-point $DR_{flpt} = \frac{M_{max} * b^{E_{max}}}{M_{min} * b^{E_{min}}}$
2. Disadvantages
   - Precision
   - Roundoff error
   - Complex implementation

2.Floating-point Representation

Form represent a floating-point number:



With three fields:

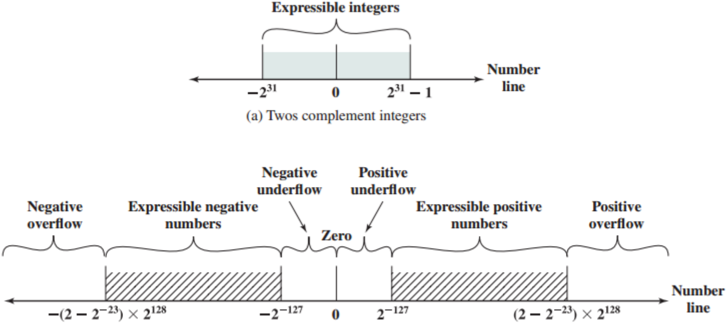
- Sign bit:S (0 is positive and 1 is negative)
- Fraction:F (Significand or mantissa)
- Exponent:E

Value of floating-point number:$(-1)^S * F * B^{\pm E}$
with B [1]

[1] The base B is implicit.(base is 2,10..)

Normalized and Denormalized representation

- Normalized:$\pm 1.mmm...m * B^{\pm E}$
- Denormalized:$\pm 0.mmm...m * B^{E_{min}}$

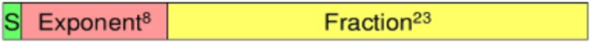


(a) Twos complement integers

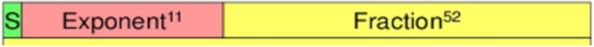
(b) Floating-point numbers

3.IEEE 754 Standard for Binary Floating-point

The three basic format have bit lengts of 32,64 and 128 bits:

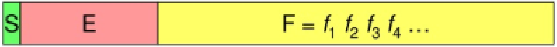


a) Binary32 format



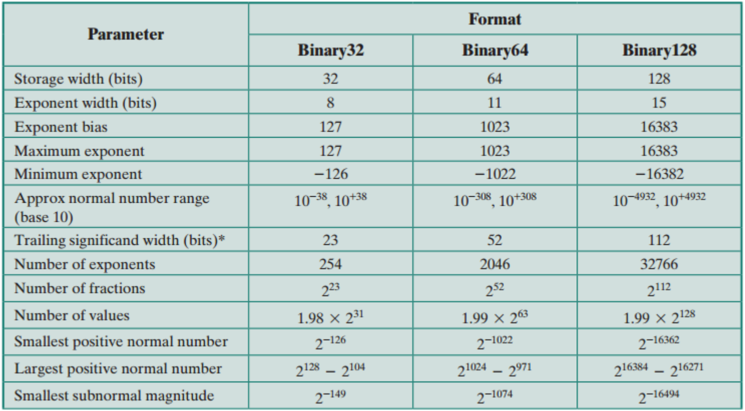
b) Binary64 format

For a normalized floating-point number:



Value of floating-point:$\pm 1.f_1 f_2 f_3 f_4 ... f_l * 2^{\pm E}$

IEEE 754 Format Paremeter



Parameter	**Format**		
	Binary32	**Binary64**	**Binary128**
Storage width (bits)	32	64	128
Exponent width (bits)	8	11	15
Exponent bias	127	1023	16383
Maximum exponent	127	1023	16383
Minimum exponent	−126	−1022	−16382
Approx normal number range (base 10)	$10^{-38}, 10^{+38}$	$10^{-308}, 10^{+308}$	$10^{-4932}, 10^{+4932}$
Trailing significand width (bits)*	23	52	112
Number of exponents	254	2046	32766
Number of fractions	2^{23}	2^{52}	2^{112}
Number of values	1.98×2^{31}	1.99×2^{63}	1.99×2^{128}
Smallest positive normal number	2^{-126}	2^{-1022}	2^{-16362}
Largest positive normal number	$2^{128} - 2^{104}$	$2^{1024} - 2^{971}$	$2^{16384} - 2^{16271}$
Smallest subnormal magnitude	2^{-149}	2^{-1074}	2^{-16494}

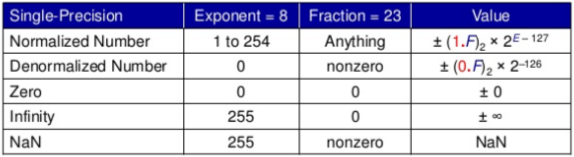
Biased Exponent Representation

IEEE 754 use biased representation for the exponent:

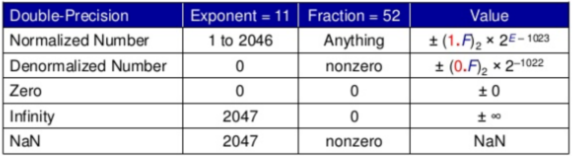
- Value of exponet= val(E)=E - Bias(Bias is a constant)
- Bias is computed base on $bias = 2^{k-1} - 1$ (with k is lengths of bit)
- For signle precision,k=8 and bias=127,value of E(biased)=val(E)+127.

Special value

IEEE 754 define some special value as NaN,Infinity to represent for underflow,overflow and not a number...



Single-Precision	Exponent = 8	Fraction = 23	Value
Normalized Number	1 to 254	Anything	$\pm (1.F)_2 \times 2^{E-127}$
Denormalized Number	0	nonzero	$\pm (0.F)_2 \times 2^{-126}$
Zero	0	0	± 0
Infinity	255	0	$\pm \infty$
NaN	255	nonzero	NaN



Double-Precision	Exponent = 11	Fraction = 52	Value
Normalized Number	1 to 2046	Anything	$\pm (1.F)_2 \times 2^{E-1023}$
Denormalized Number	0	nonzero	$\pm (0.F)_2 \times 2^{-1022}$
Zero	0	0	± 0
Infinity	2047	0	$\pm \infty$
NaN	2047	nonzero	NaN

Rounding Mode

IEEE 754 standard specifies four rounding modes

1. Round to nearest even
2. Round toward plus infinity:result is rounded up
3. Round toward minus infinity:result is rounded down
4. Round toward zero:always truncate result

Round to Nearest Even(default rouding mode)

Normalized result has the form:$\mathbf{1.}f_1 f_2 f_3 ... f_l$**RS**

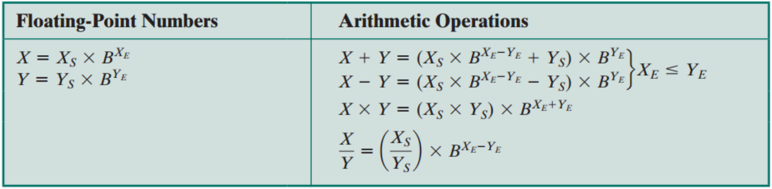
- f_l:last fration bit.
- round bit**R**:appears after the last fraction bit f_l
- sticky bit **S**:is the OR of all remaining addtional bits.

Four cases for **RS**:

- **RS=00**:Result is exact,no need for rounding.
- **RS=01**:Truncate result by discarding **RS**
- **RS=11**:Increment result by add **1** to last fraction bit
- **RS=10**:Increment or Truncate depend on f_l
  - $f_l = 0$:truncate result
  - $f_l = 1$:increment result

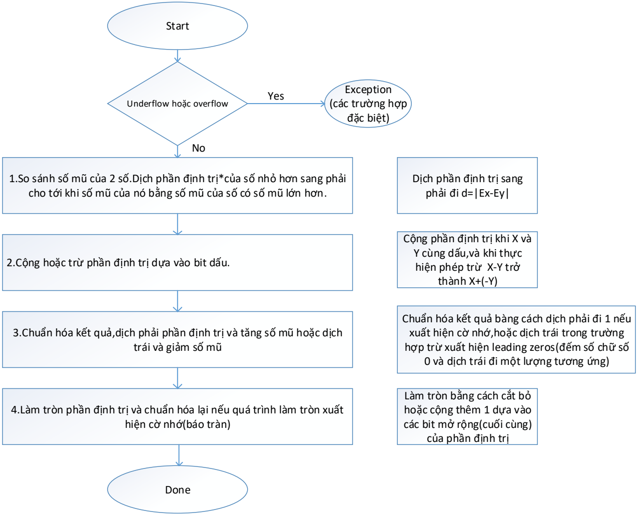
3.Floating-point Arithmetic

Basic opreations for floating-point arithmetic:



Floating-Point Numbers	Arithmetic Operations
$X = X_S \times B^{X_E}$ $Y = Y_S \times B^{Y_E}$	$\left.\begin{aligned} X + Y &= (X_S \times B^{X_E - Y_E} + Y_S) \times B^{Y_E} \\ X - Y &= (X_S \times B^{X_E - Y_E} - Y_S) \times B^{Y_E} \end{aligned}\right\} X_E \leq Y_E$ $X \times Y = (X_S \times Y_S) \times B^{X_E + Y_E}$ $\frac{X}{Y} = \left(\frac{X_S}{Y_S}\right) \times B^{X_E - Y_E}$

Addition and Subtration(1):Pseudocode



Start

Underflow hoặc overflow

Yes

Exception (các trường hợp đặc biệt)

No

1.So sánh số mũ của 2 số.Dịch phần định trị*của số nhỏ hơn sang phải cho tới khi số mũ của nó bằng số mũ của số có số mũ lớn hơn.

Dịch phần định trị sang phải đi d=|Ex-Ey|

2.Cộng hoặc trừ phần định trị dựa vào bit dấu.

Cộng phần định trị khi X và Y cùng dấu,và khi thực hiện phép trừ X-Y trở thành X+(-Y)

3.Chuẩn hóa kết quả,dịch phải phần định trị và tăng số mũ hoặc dịch trái và giảm số mũ

Chuẩn hóa kết quả bàng cách dịch phải đi 1 nếu xuất hiện cờ nhớ,hoặc dịch trái trong trường hợp trừ xuất hiện leading zeros(đếm số chữ số 0 và dịch trái đi một lượng tương ứng)

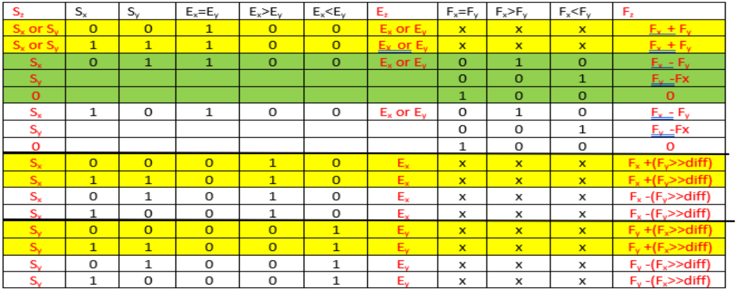
4.Làm tròn phần định trị và chuẩn hóa lại nếu quá trình làm tròn xuất hiện cờ nhớ(báo tràn)

Làm tròn bằng cách cắt bỏ hoặc cộng thêm 1 dựa vào các bit mở rộng(cuối cùng) của phần định trị

Done

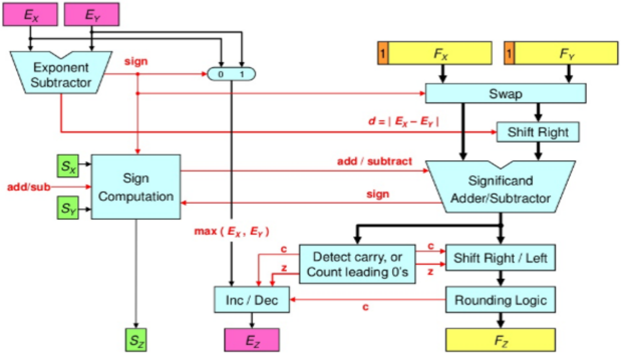
Addition and Subtraction(2):Table of result

- Input:X(S_X, E_X, F_X)và Y(S_Y, E_Y, F_Y)
- Output:Z(S_Z, E_Z, F_Z)

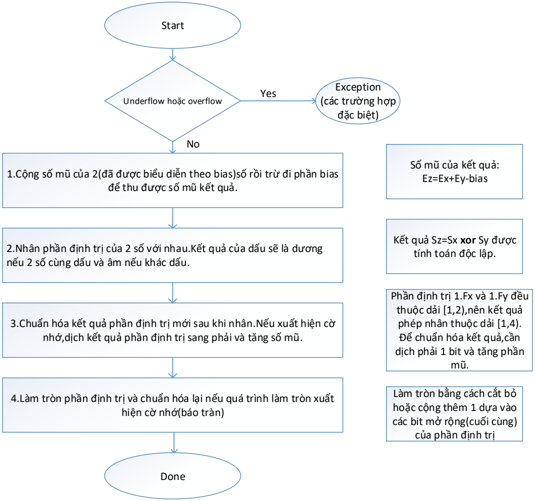


S_z	S_x	S_y	E_x=E_y	E_x>E_y	E_x<E_y	E_z	F_x=F_y	F_x>F_y	F_x<F_y	F_z
S_x or S_y	0	0	1	0	0	E_x or E_y	x	x	x	F_x + F_y
S_x or S_y	1	1	1	0	0	E_x or E_y	x	x	x	F_x + F_y
S_x	0	1	1	0	0	E_x or E_y	0	1	0	F_x - F_y
S_y							0	0	1	F_y -Fx
0							1	0	0	0
S_x	1	0	1	0	0	E_x or E_y	0	1	0	F_x - F_y
S_y							0	0	1	F_y -Fx
0							1	0	0	0
S_x	0	0	0	1	0	E_x	x	x	x	F_x +(F_y>>diff)
S_x	1	1	0	1	0	E_x	x	x	x	F_x +(F_y>>diff)
S_x	0	1	0	1	0	E_x	x	x	x	F_x -(F_y>>diff)
S_x	1	0	0	1	0	E_x	x	x	x	F_x -(F_y>>diff)
S_y	0	0	0	0	1	E_y	x	x	x	F_y +(F_x>>diff)
S_y	1	1	0	0	1	E_y	x	x	x	F_y +(F_x>>diff)
S_y	0	1	0	0	1	E_y	x	x	x	F_y -(F_x>>diff)
S_y	1	0	0	0	1	E_y	x	x	x	F_y -(F_x>>diff)

Addition and Subtraction(3):Adder Block Diagram



Multiplication(1):Pseudocode



Start

Underflow hoặc overflow

Yes

Exception (các trường hợp đặc biệt)

No

1.Cộng số mũ của 2(đã được biểu diễn theo bias)số rồi trừ đi phần bias để thu được số mũ kết quả.

Số mũ của kết quả: Ez=Ex+Ey-bias

2.Nhân phần định trị của 2 số với nhau.Kết quả của dấu sẽ là dương nếu 2 số cùng dấu và âm nếu khác dấu.

Kết quả Sz=Sx **xor** Sy được tính toán độc lập.

3.Chuẩn hóa kết quả phần định trị mới sau khi nhân.Nếu xuất hiện cờ nhớ,dịch kết quả phần định trị sang phải và tăng số mũ.

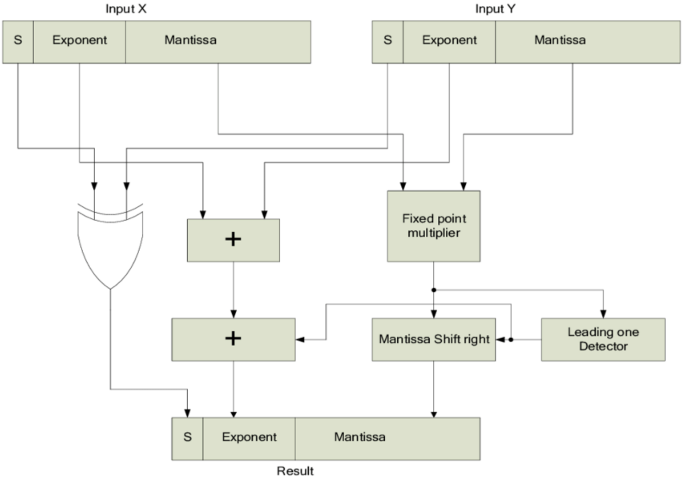
Phần định trị 1.Fx và 1.Fy đều thuộc dải [1,2),nên kết quả phép nhân thuộc dải [1,4). Để chuẩn hóa kết quả,cần dịch phải 1 bit và tăng phần mũ.

4.Làm tròn phần định trị và chuẩn hóa lại nếu quá trình làm tròn xuất hiện cờ nhớ(báo tràn)

Làm tròn bằng cách cắt bỏ hoặc cộng thêm 1 dựa vào các bit mở rộng(cuối cùng) của phần định trị

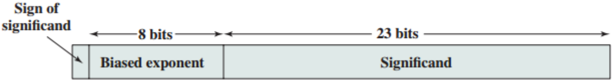
Done

Multiplication(2):Multiply Block Diagram

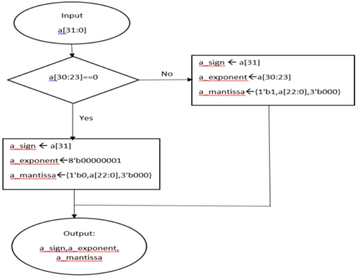


4.Implementation in FPGA

The basic operations for floating-point arithmetic performed on FPGA are of type binary32 bit format and it is implemented on the FPGA VCU118-Virtex kit



Implement Addition(1):Unpackage process



Input

a[31:0]

a[30:23]==0

No

a_sign ← a[31]

a_exponent←a[30:23]

a_mantissa←{1'b1,a[22:0],3'b000}

Yes

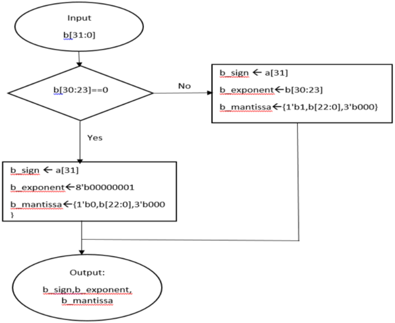
a_sign ← a[31]

a_exponent←8'b00000001

a_mantissa←{1'b0,a[22:0],3'b000}

Output:

a_sign,a_exponent,
a_mantissa



Input

b[31:0]

b[30:23]==0

No

b_sign ← a[31]

b_exponent←b[30:23]

b_mantissa←{1'b1,b[22:0],3'b000}

Yes

b_sign ← a[31]

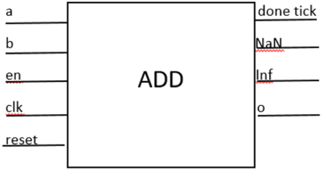
b_exponent←8'b00000001

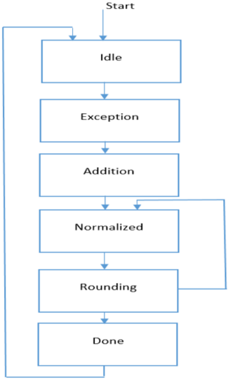
b_mantissa←{1'b0,b[22:0],3'b000
}

Output:

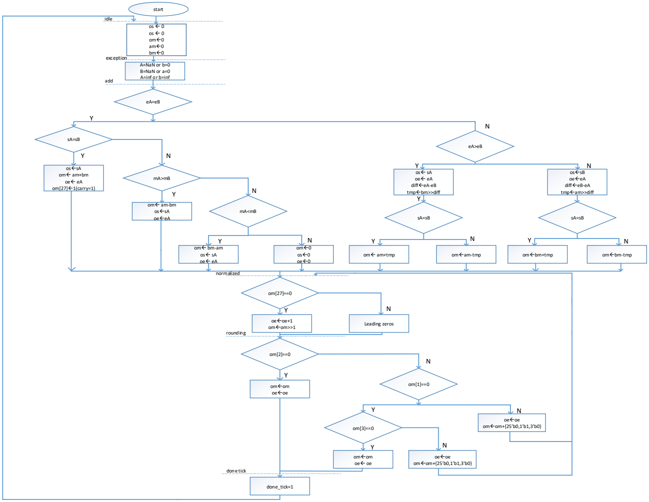
b_sign,b_exponent,
b_mantissa

Implement Addition(2):Block and Flow-chart FSMD

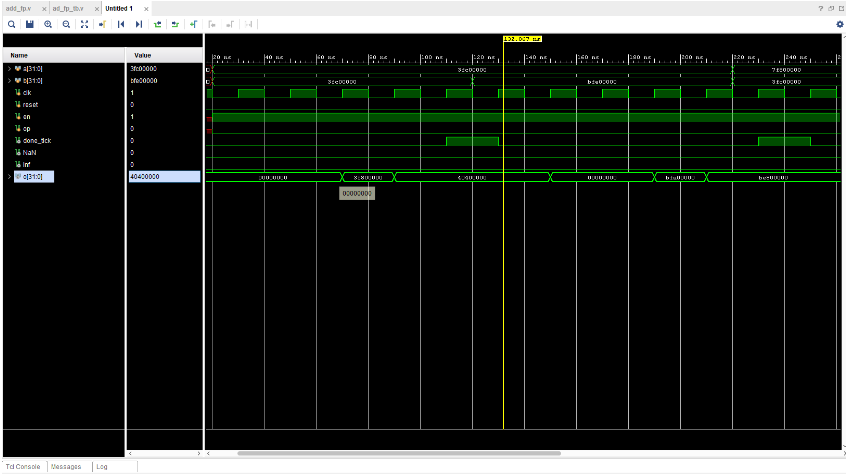




Implement Addition(3):ASMD diagram



Implement Addition(4):Simulation result

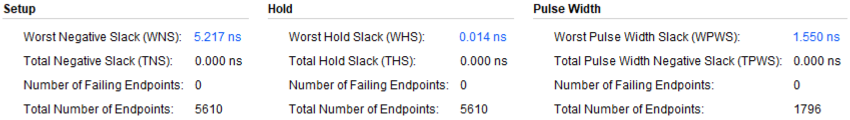


Name	Value
a[31:0]	3fc00000
b[31:0]	bfe00000
clk	1
reset	0
en	1
op	0
done_tick	0
NaN	0
inf	0
o[31:0]	40400000

Implement Addition(5):Timing analysis

With Frequency clock:100MHz,period=10ns

Critical Path :5.217ns(required time-arrival time)

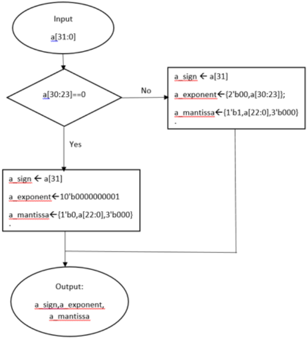


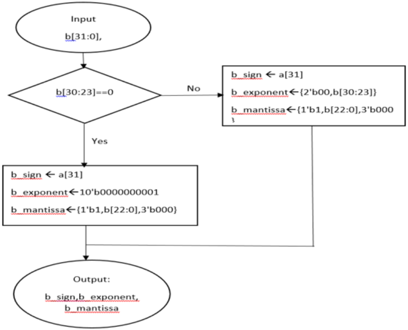
Setup	
Worst Negative Slack (WNS):	5.217 ns
Total Negative Slack (TNS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	5610

Hold	
Worst Hold Slack (WHS):	0.014 ns
Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	5610

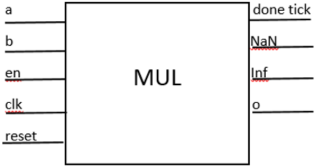
Pulse Width	
Worst Pulse Width Slack (WPWS):	1.550 ns
Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	1796

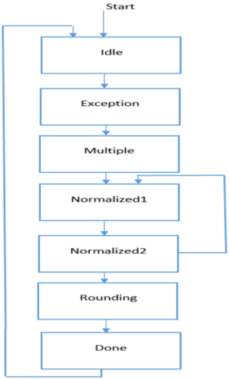
Implement Multiplication(1):Unpackage process


Input
a[31:0]
a[30:23]==0
No
a_sign ← a[31]
a_exponent←{2'b00,a[30:23]};
a_mantissa←{1'b1,a[22:0],3'b000}
Yes
a_sign ← a[31]
a_exponent←10'b0000000001
a_mantissa←{1'b0,a[22:0],3'b000}
Output:
a_sign,a_exponent,
a_mantissa

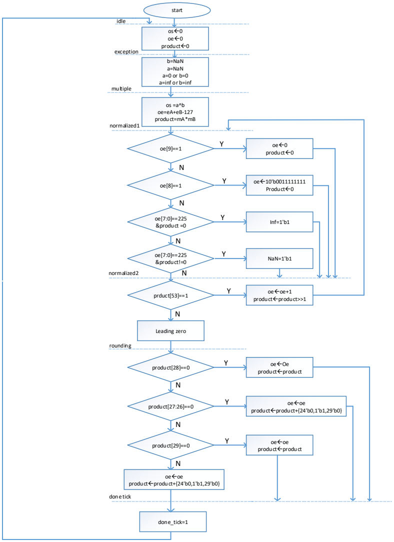

Input
b[31:0],
b[30:23]==0
No
b_sign ← a[31]
b_exponent←{2'b00,b[30:23]}
b_mantissa←{1'b1,b[22:0],3'b000
}
Yes
b_sign ← a[31]
b_exponent←10'b0000000001
b_mantissa←{1'b1,b[22:0],3'b000}
Output:
b_sign,b_exponent,
b_mantissa

Implement Multiplication(2):Block and flow-chart FSMD

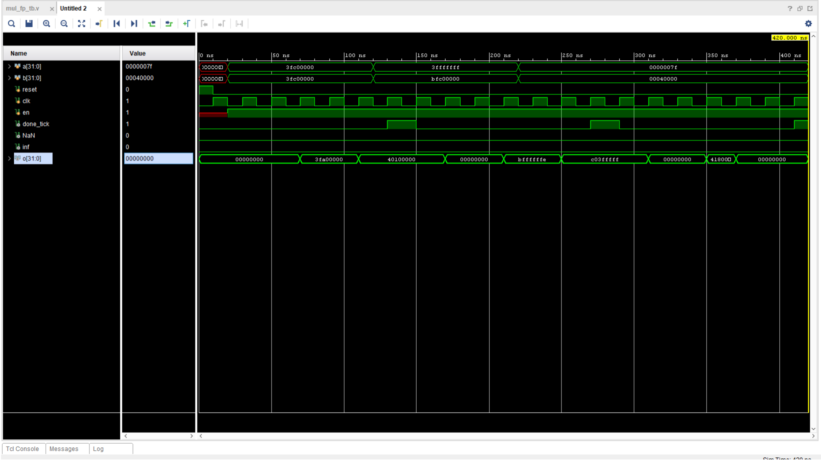




Implement Multiplication(3):ASMD diagram



Implement Multiplication(4):Simulation result

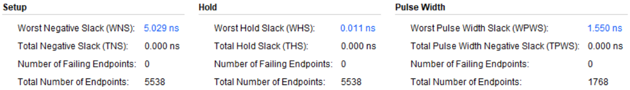


Name	Value
a[31:0]	0000007f
b[31:0]	00040000
reset	0
clk	1
en	1
done_tick	1
NaN	0
inf	0
o[31:0]	00000000

Implement Multiplication(5):Timing analysis

With Frequency clock:100MHz,period=10ns

Critical Path :5.029ns(required time-arrival time)



Setup	
Worst Negative Slack (WNS):	5.029 ns
Total Negative Slack (TNS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	5538

Hold	
Worst Hold Slack (WHS):	0.011 ns
Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	5538

Pulse Width	
Worst Pulse Width Slack (WPWS):	1.550 ns
Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	1768

Conclusion