數位系統導論實驗

Lab7 RTL Modeling (Multiplier)

負責助教：王偉丞

Email: wmike851223@gmail.com

Outline

- 課程目的
- IEEE 754簡介
- 範例程式
- 作業說明
- 課程評分
- 附錄 - 浮點數乘法（Floating-Point Multiplication）

課程目的

- 本次實驗將介紹常見的浮點數表示法（IEEE 754）及以Verilog完成其乘法運算單元之實作

IEEE 754 – 簡介（1 / 4）

- 浮點數（Floating-point Numbers）是同學們熟知的科學記號表示法，利用正規化後的數值（即 Mantissa ）與對應的指數（Exponent ）同時兼顧數值之精確（Precision）與動態範圍（Dynamic Range）
- IEEE 二進位浮點數算術標準（IEEE 754）是當前最廣泛使用的浮點數運算標準，在 IEEE 754 中表示浮點數值的方式，包含半精確度（16 位元）、單精確度（32 位元）、雙精確度（64 位元）、延伸單精確度（43 位元以上）以及延伸雙精度（通常以 80 位元實作）
- 其浮點數以這樣表示：*Value* = *Sign* × *Exponent* × *Fraction*

IEEE 754 – 簡介（2 / 4）

- *Sign*為符號位，以0表示正值，1表示負值
- *Exponent*為二進位科學計數法表示下的指數值加上指數偏移值
  - 因為IEEE 754中以無號整數（Unsigned Integer）表示指數，其中一半值域在表示負數，因此將2^{e-1} – 1定為指數偏移值（Exponent Bias），其中e為儲存指數的位元長度。以8位元指數長度為例，指數偏移值為127，亦即二進位科學計數法表示下的指數值需再加上127，才會是IEEE 754中*Exponent*的值。
- *Fraction*
  - 當浮點數的指數部分編碼值在 $0 < Exponent \leq 2^e - 2$之間，則*Fraction*值為二進位科學計數法的尾數（Mantissa），亦即1.*Fraction*。
  - 如果指數部分編碼值是0，二進位科學計數法的尾數部分非零，則該實際值比前述涵蓋情況更接近0，因此其*Fraction*代表的值實際為0.*Fraction*

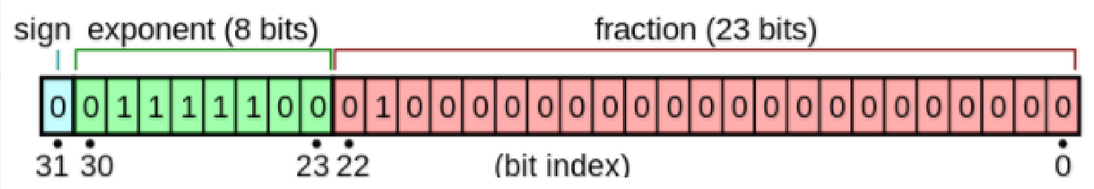
IEEE 754 – 簡介（3 / 4）

- 特殊值
  - 如果指數是0且*Fraction*亦為0，該值為正負0（視Sign Bit而定）
  - 如果指數 = 2^e – 1且*Fraction*為0，該值為正負無限大（視Sign Bit而定）
  - 如果指數 = 2^e – 1且*Fraction*不為0，表示該不為一個數（NaN）
- 總結規則如下：

形式	指數	小數部分
零	0	0
非正規形式	0	大於0小於1（0.*Fraction*）
正規形式	1到2^e – 1	大於等於1小於2（1.*Fraction*）
無窮	2^e – 1	0
NaN	2^e – 1	非0

IEEE 754 – 單精度浮點數介紹（4 / 4）

- 以單精確度浮點數為例，在 32 bits 中，我們使用 1 bit 表示正值或負值，8 bits 表示指數，23 bits 表示尾數精度



Sign = +1

Exponent = $(01111100)_2$ – 127 = -3

Fraction

= 1 + $(0.01000000000000000000000)_2$

= 1 + 2^{-2}

= 1.25

Value = (+1) × 1.25 × 2^{-3} = + 0.15625

範例程式 – 32-bit浮點數乘法器（1 / 3）

- 乘法器的上層模組
  - Input: 欲計算乘法結果的輸入值input_a和input_b
  - Output: input_a與input_b的乘法計算結果
  - Multiplier: 給出對應輸入的浮點數乘法計算
  - I/O皆為遵照IEEE-754的32-bit浮點數

```
module mpy_top (
    input [31:0] input_a,        // input a
    input [31:0] input_b,        // input b
    output [31:0] mpy_output     // multiplication result
);

    wire a_sign;                 // sign of a
    wire [7:0] a_exponent;       // exponent of a
    wire [23:0] a_mantissa;      // mantissa of a

    wire b_sign;                 // sign of b
    wire [7:0] b_exponent;       // exponent of b
    wire [23:0] b_mantissa;      // mantissa of b

    reg        o_sign;           // sign of output
    reg [7:0]  o_exponent;       // exponent output
    reg [24:0] o_mantissa;       // mantissa output

    reg [31:0] multiplier_a_in; // multiplier input a
    reg [31:0] multiplier_b_in; // multiplier input b
    wire [31:0] multiplier_out; // multiplier output out

    assign  mpy_output[31] = o_sign;                  // MSB for sign
    assign  mpy_output[30:23] = o_exponent;           // 8 bits for exponent
    assign  mpy_output[22:0] = o_mantissa[22:0];      // 23 bits for mantissa

    assign a_sign = input_a[31];                      // MSB for sign
    assign a_exponent[7:0] = input_a[30:23];          // 8 bits for exponent
    assign a_mantissa[23:0] = {1'b1, input_a[22:0]};  // 23 bits for mantissa

    assign b_sign = input_b[31];                      // MSB for sign
    assign b_exponent[7:0] = input_b[30:23];          // 8 bits for exponent
    assign b_mantissa[23:0] = {1'b1, input_b[22:0]};  // 23 bits for mantissa

    // Multiplication
    multiplier mpy
    (
        .mpy_input_a(multiplier_a_in),
        .mpy_input_b(multiplier_b_in),
        .mpy_output(multiplier_out)
    );
```

範例程式 – 32-bit浮點數乘法器（2 / 3）

- 檢查輸入值是否為特殊值，例如NaN、Inf與0，給出相對應的輸出。圖中僅列出部分特殊值輸入做為示範（如右圖）。
- 如果輸入非特殊值則進行乘法運算並給出結果（如下圖）

```
always @ (*) begin
    // if input_a is NaN or input_b is NaN return NaN
    if ( ((a_exponent == 255) && (a_mantissa[22:0] != 0)) ||
        ((b_exponent == 255) && (b_mantissa[22:0] != 0) )) begin
        o_sign <= 1;
        o_exponent <= 255;

        o_mantissa[22] <= 1;
        o_mantissa[21:0] <= 0;

    // if input_a is Inf return Inf
    end else if (a_exponent == 255) begin
        o_sign <= a_sign ^ b_sign;
        o_exponent <= 255;
        o_mantissa <= 0;

        //if input_b is zero return NaN
        if ((b_exponent == 0) && (b_mantissa[22:0] == 0)) begin
            o_sign <= 1;
            o_exponent <= 255;
            o_mantissa[22] <= 1;
            o_mantissa[21:0] <= 0;
        end
```

```
        //  Multiplication Operation
        end else begin
            multiplier_a_in <= input_a;
            multiplier_b_in <= input_b;

            o_sign = multiplier_out[31];
            o_exponent = multiplier_out[30:23];
            o_mantissa = multiplier_out[22:0];
        end
    end


endmodule
```

範例程式 – 32-bit浮點數乘法器（3 / 3）

- 對特殊值外輸入進行乘法運算的浮點乘法器

```
module multiplier32(
    input  [31:0] mpy_input_a, mpy_input_b,
    output [31:0] mpy_output
);
    reg a_sign;              // sign of a
    reg [7:0] a_exponent;    // exponent of a
    reg [23:0] a_mantissa;   // mantissa of a

    reg b_sign;              // sign of b
    reg [7:0] b_exponent;    // exponent of b
    reg [23:0] b_mantissa;   // mantissa of b

    reg o_sign;              // sign of output
    reg [7:0] o_exponent;    // exponent output
    reg [22:0] o_mantissa;   // mantissa output

    reg [47:0] product;      // product of mantissa

    assign mpy_output[31] = o_sign;
    assign mpy_output[30:23] = o_exponent;
    assign mpy_output[22:0] = o_mantissa[22:0];
```

```
    always @ (*) begin
        a_sign = mpy_input_a[31];
        a_exponent = mpy_input_a[30:23];
        a_mantissa = {1'b1, mpy_input_a[22:0]};

        b_sign = mpy_input_b[31];
        b_exponent = mpy_input_b[30:23];
        b_mantissa = {1'b1, mpy_input_b[22:0]};

        o_sign = a_sign ^ b_sign;
        o_exponent = a_exponent + b_exponent - 127;
        product = a_mantissa * b_mantissa;

        // Normalization
        if(product[47] == 1) begin
            o_exponent = o_exponent + 1;
            product = product >> 1;
        end

        // Rounding
        o_mantissa = product[46:23];
        if(product[22]) begin
            o_mantissa = o_mantissa + 1;
        end


    end
endmodule
```

範例程式 – 執行

- 輸入指令執行程式，檢視設計之 Module 功能是否有錯誤：
  - iverilog –o testb testbench_32bits.v
  - vvp test

```
PS C:\Users\SOSO\Downloads\DD LAB7> iverilog -o test .\testbench_32bits.v
PS C:\Users\SOSO\Downloads\DD LAB7> vvp test
VCD info: dumpfile test.fsdb opened for output.
Test  1
///////////////////////
//// Successful  1 ////
///////////////////////
ba9dbb67 * 4148f5cb = ?
Answer = bc77a3b4

Test  2
///////////////////////
//// Successful  2 ////
///////////////////////
43e20fcc * c1ac8adb = ?
Answer = c6185d3b

Test  3
///////////////////////
//// Successful  3 ////
///////////////////////
c49a522c * 442987e6 = ?
Answer = c94c6456

Test  4
///////////////////////
//// Successful  4 ////
///////////////////////
a6ad6da0 * 2badab89 = ?
Answer = 92eb4e94
```

作業說明

- 成功執行範例程式（40%）
- 完成依循IEEE 754規範的半精度（16 bits）浮點數乘法器（40%）

課程評分

- Demo 時間：4/29(一)與5/1(三)的 19:30、19:50、 20:10 與 20:30
- Demo 地點：工一館206
- 評分方式
  - 範例成功執行（40%）
  - 16-bit Floating-point Multiplier（40%）
  - 隨堂練習：20%

附錄

Floating-Point Multiplication（1 / 3）

- 下面介紹IEEE 754單精度浮點數乘法運算（c = a×b）

1. $s_c = s_a \oplus s_b$, $|c| = |a|\times|b| = (2^{e_a-127}\times 1.f_a)\times(2^{e_b-127}\times 1.f_b) = 2^{(e_a+e_b-127)-127}\times 1.f_a\times 1.f_b$，得到$1.0 \le (1.f_a\times 1.f_b) < 4.0$
   - 如果 $1.f_a\times 1.f_b < 2.0$，得到 $e_c = e_a + e_b - 127$ 和 $1.f_c = 1.f_a\times 1.f_b$。
   - 如果 $1.f_a\times 1.f_b \ge 2.0$，$e_c = e_a+e_b-127+1$ 和 $1.f_c = (1.f_a\times 1.f_b) \gg 1$ (右移1 bit)
   - 因為正規化後的指數e滿足$1 \le e \le 254$，所以我們具有 $-125 \le (= e_a+e_b-127) \le 381$或$-124 \le (= e_a+e_b-127+1) \le 382$。即$e_c$ 可能會超出 1 – 254 的範圍。如果$1 \le e_c \le 254$，則為正規化的浮點數。如果$e_c > 254$，則結果為 ∞（$e_c = 255$，$f_c = 0$）。當$e_c <1$時，如果結果大於或等於$2^{-126}\times 0.00000000000000000000001 = 2^{-149}$，則可用非正規化浮點數表示；否則用0表示。

Floating-Point Multiplication（2 / 3）

2. 接著考慮浮點數乘法的數字規格，因為$1 \le e \le 254$符合規格，真實指數e’（即e − 127）滿足 $-126 \le e' \le 127$。假設$a = \{s_a, e_a, f_a\}$是一個符合規格的浮點數，$b = \{s_b, e_b, f_b\}$是不在規格內的浮點數（$e_b = 0$，$f_b \neq 0$），亦即非正規值（Subnormal Numbers）。
3. c = a × b 的絕對值為 $|c| = |a| \times |b| = (2^{e_a-127} \times 1.f_a) \times (2^{-126} \times 0.f_b) = 2^{e_a-253} \times (1.f_a \times 0.f_b)$。最大絕對值是$2^{254-253} \times (2 - 2^{-23}) \times (1 - 2^{-23}) = 2 \times (2 - 3 \times 2^{-23} + 2^{-46})$，這是一個規格內的浮點數。最小的絕對值是$2^{1-253} \times 1.0 \times 2^{-23} = 2^{-275}$，它超出了e’應該在的範圍內，因此，該結果可以用denormalized float number或0表示。

Floating-Point Multiplication（3 / 3）

3. 接下來，考慮兩個非規格化浮點數的浮點乘法。假設a = {s_a，e_a，f_a}和b = {s_b，e_b，f_b}都是非規格化的浮點數。c = a×b的絕對值為 |c| = |a|×|b| =（2^{-126}×0.f_a）×（2^{-126}×0.f_b）= 2^{-252}×（0.f_a×0.f_b）。它小於非規格化浮點數可以表示的最小數量；最後，考慮一些特殊的計算：
   - NaN×b = NaN
   - ∞×0 = NaN
   - 如果 b ≠ 0 且 b ≠ NaN，則 ∞×b = ∞
   - 如果 b ≠ ∞ 且 b ≠ NaN，則 0×b = 0