各位好，接下來我會講解Final Project : FP MUL 架構，簡單帶過所有流程以及最終APR DRC結束的功耗、時間、面積

1. **RTL 設計和Testbench修改**

首先先來講解RTL設計

**宣告**

令state為5bits暫存器狀態, 存取紀錄狀態的變化

令A,B各為64bits暫存器,存取乘數和被乘數的資料

令Z為64bits暫存器,存取A\*B的最終結果

令count為7bits暫存器,記數用

令m\_a,m\_b為53 bits暫存器,存取mantissa乘法乘數和被乘數的資料

令m\_z為106 bits暫存器, 存取mantissa乘法的結果

令G,R,S,LSB為1 bit暫存器

分別儲存

G : Guard Bit

R : Round Bit

S : Sticky Bit

LSB : Least Significant Bit

**RESET**

在RESET和執行完一整輪時(state 8結束後)

把暫存器初始化為0以免亂數導致結果錯誤

**state 0 :**

由於資料是以1個CLK輸入1個8bits 共8個CLK構成一組64bits浮點數

所以將輸入資料(DATA\_IN)和A去做OR運算並利用左移將每個CLK輸入的bits都能成功存入到A之中(並且是由低位存到高位)

利用變數count控制次數

當A成功存完資料跳到state 1並把變數count初始化成0以利後續使用

**state 1 :**

與state0邏輯相同，換成存入到B

當B成功存完資料跳到state 2並把變數count初始化成0以利後續使用

**備註 : state 0, state 1皆在ENABLE為1時執行**

**state 2 :**

判斷並處理IEEE754的特殊情況

* 若A 或 B為NaN時直接輸出NaN
* 無限\*0或0\*無限 直接輸出NaN
* 若A 或 B是+0,-0 直接輸出+0,-0 (考慮signed bit)
* 若A 或 B是無限且另一邊不是0的情況 直接輸出+無限,-無限

(考慮signed bit)

若判斷無上述情況，則進入state3進行正常IEEE754流程

**state 3 :**

對A,B的signed bit進行XOR結果存到Z的signed bit

將A,B的指數部分相加並減掉bias (64bits乘法器要減掉1023)

為了m\_a,m\_b能做mantissa運算 添加1bit的1

進入state 4

**state 4 :**

利用Shift-and-Add進行mantissa乘法

如果直接用乘法器的話會讓1個CLK做太多事情，所以學生將其拆分為53個CLK進行

每1個CLK進行對m\_b[0]的檢查

若m\_b[0]為1就把A 左移count次累加到m\_z若無則不用累加

然後每次都需要將m\_b進行右移並增加count，這樣就能每一個bit都檢查且可以保證累加到m\_z的是正確位元

結束之後進入state 5

**state 5 :**

判斷並正規化

若最高位m\_z[105]為1代表乘積超出範圍 需要右移bit同時指數部分Z[62:52]也需要加1保持真實數值不變

然後根據兩種不同情況儲存

G : Guard Bit 捨去部分第一位

R : Round Bit捨去部分第二位

S : Sticky Bit 捨去部分剩餘所有位做OR判斷是否有任何1

LSB : Least Significant Bit

結束之後進入state 6

**state 6 :**

把正規化結果傳入Z

初始化count為0做後續使用

結束之後進入state 7

**state 7 :**

根據Guard Bit, Round Bit, Sticky Bit,LSB判斷並round to nearest

* Guard Bit = 1 且 (Round Bit = 1 或 Sticky Bit = 1) 就進位
* Guard Bit = 1 且 (Round Bit = 0且 Sticky Bit = 0 且LSB = 1) 也進位
* 其餘情況不進位

結束之後進入state 8

**state 8 :**

利用case來將資料由低位到高位輸出並拉起READY

輸出結束之後將所有暫存器值包括READY歸0

**再來講解Testbench修改**

由於原本的Testbench沒有判斷特殊處理的部分，所以學生加入了可以判斷IEEE754的特殊情況

* 若A 或 B為NaN時直接輸出NaN
* 無限\*0或0\*無限 直接輸出NaN
* 若A 或 B是+0,-0 直接輸出+0,-0 (考慮signed bit)
* 若A 或 B是無限且另一邊不是0的情況 直接輸出+無限,-無限

若判斷結果和預想不符則會輸出Simulation Fail並列出錯在哪和正確答案

圖1為輸出正確的情況

一張含有 文字, 螢幕擷取畫面, 字型, 黑與白 的圖片

AI 產生的內容可能不正確。

圖 1

1. **執行ncverilog -f run.f**

確認設計功能有無錯誤，成功截圖如圖2所示

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

AI 產生的內容可能不正確。

圖 2

Output latency控制在58 clocks小於規定的60 clocks

1. **執行04\_run\_low\_power\_syn**

執行前加入所需檔案如.tef,.lef,.lib,qrcTechFile,.sdc(改CLK符合1GHz),.tcl

1. **執行ncverilog -f run.f**

在TEST\_gate.v加入sdf資訊最終產出.tcf做power report

1. **執行05\_report\_power**

執行前加入.tcl(設定正確路徑)

執行結果圖3 report.power.gate

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

圖 3

1. **進行APR流程**

確認CHIP.sdc ,CHIP\_syn.v 檔案沒問題(top module名字為CHIP)

改動CHIP.io檔案

改動MMMC.view的sdc

設定完畢開始進行APR

floorplan到實體繞線後導出CHIP.v CHIP.sdf

1. **進行Post-Layout Gate-level Simulation**

執行ncverilog -f run.f

測試成功如圖4

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

AI 產生的內容可能不正確。

圖 4

成功產出CHIP.tcf檔進行power rail analysis

1. **實驗結果**

最終功耗如圖5 , 0.7196mW

一張含有 文字, 螢幕擷取畫面, 文件, 黑與白 的圖片

AI 產生的內容可能不正確。

圖 5

最終時間如圖6 , 圖7

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

AI 產生的內容可能不正確。

圖 6

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

AI 產生的內容可能不正確。

圖 7

面積如圖8所示

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

圖 8

1. **Post-Layout STA and Verifications with TSMC ADFP**

**DRC Check**

確認沒問題如圖9所示

一張含有 文字, 螢幕擷取畫面, 字型, 黑與白 的圖片

AI 產生的內容可能不正確。

圖 9

**LVS Check**

確認沒問題如圖10所示

**一張含有 文字, 螢幕擷取畫面, 字型, 文件 的圖片

AI 產生的內容可能不正確。**

圖 10

**Virtuoso and Calibre RVE**

確認沒問題如圖11, 圖12所示

一張含有 文字, 螢幕擷取畫面, 陳列, 字型 的圖片

AI 產生的內容可能不正確。

圖 11

一張含有 文字, 螢幕擷取畫面, 軟體, 陳列 的圖片

AI 產生的內容可能不正確。

圖 12