

人工智慧晶片設計與應用

AI-ON-CHIP FOR MACHINE LEARNING AND INFERENCE

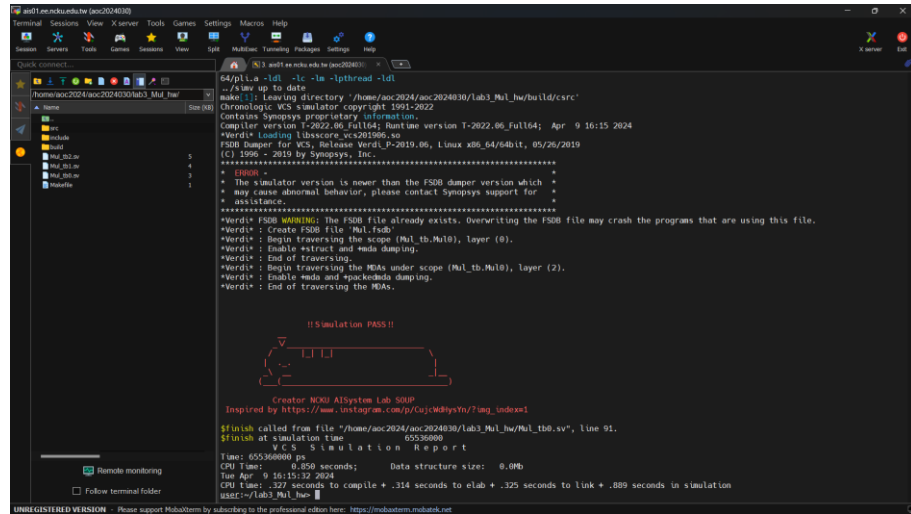
Lab 3

E14096724

鄭喆嚴

1. (10% / 10% / 15%) Choose 1 from 3 algorithms to implement. Pass TB0/TB1/TB2.Screenshot all pass screens.

TB0 pass screens:



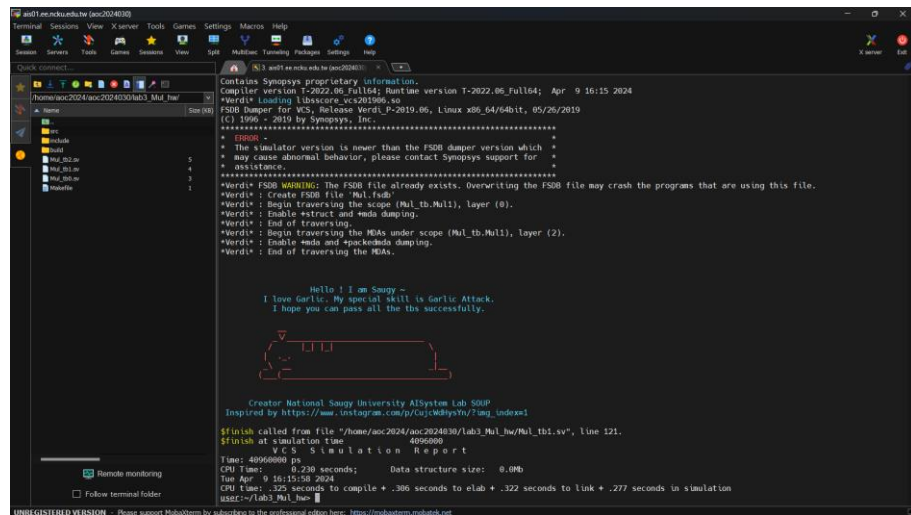
```
64bit -ld -lc -lm -lpthread -ldt
./view up to date
make[1]: Leaving directory '/home/asc2024/asc2024030/lab3_Mul_hw/build/csrc'
Chronologic WCS simulator copyright 1991-2022
Contains Synopsys proprietary information.
Compiler version T-2022.06_Full164; Runtime version T-2022.06_Full164; Apr 9 16:15 2024
*Verdi* Loading libscore_vc201906.so
FSOB Dumper for WCS, Release Verdi_P-2019.06, Linux x86_64/64bit, 05/26/2019
(C) 1996 - 2019 by Synopsys, Inc.
*****
+ ERROR +
+ The simulator version is newer than the FSOB dumper version which
+ may cause abnormal behavior, please contact Synopsys support for
+ assistance.
*****
*Verdi* FSOB WARNING: The FSOB file already exists. Overwriting the FSOB file may crash the programs that are using this file.
*Verdi* Create FSOB file 'Mul.fsob'
*Verdi* Begin traversing the scope (Mul.tb.Mul0), layer (0).
*Verdi* Enable *struct and *mda dumping.
*Verdi* End of traversing.
*Verdi* Begin traversing the MDAs under scope (Mul.tb.Mul0), layer (2).
*Verdi* Enable *mda and *packedmda dumping.
*Verdi* End of traversing the MDAs.

!! Simulation PASS !!

Creator NCOU ASystem Lab SGP
Inspired by https://www.instagram.com/p/CyJcWdYsYn/7uq_index01

$finish called from file "/home/asc2024/asc2024030/lab3_Mul_hw/Mul_tb0.v", line 91.
$finish at simulation time 65536000
*****
VCS Simulation Report
*****
Time: 65536000 ps
CPU Time: 0.850 seconds; Data structure size: 0.0Mb
Tue Apr 9 16:15:32 2024
CPU time: .327 seconds to compile + .314 seconds to elab + .325 seconds to link + .889 seconds in simulation
user~/lab3_Mul_hw
```

TB1 pass screens:



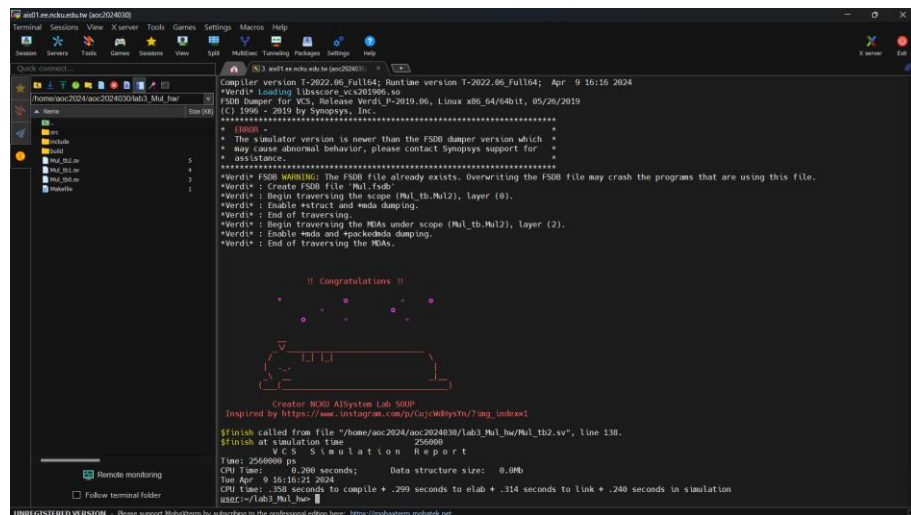
```
*****
+ ERROR +
+ The simulator version is newer than the FSOB dumper version which
+ may cause abnormal behavior, please contact Synopsys support for
+ assistance.
*****
*Verdi* FSOB WARNING: The FSOB file already exists. Overwriting the FSOB file may crash the programs that are using this file.
*Verdi* Create FSOB file 'Mul.fsob'
*Verdi* Begin traversing the scope (Mul.tb.Mul1), layer (0).
*Verdi* Enable *struct and *mda dumping.
*Verdi* End of traversing.
*Verdi* Begin traversing the MDAs under scope (Mul.tb.Mul1), layer (2).
*Verdi* Enable *mda and *packedmda dumping.
*Verdi* End of traversing the MDAs.

Hello! I am Smgy --
I love Garlic. My special skill is Garlic Attack.
I hope you can pass all the ths successfully.

Creator National Smgy University ASystem Lab SGP
Inspired by https://www.instagram.com/p/CyJcWdYsYn/7uq_index01

$finish called from file "/home/asc2024/asc2024030/lab3_Mul_hw/Mul_tb1.v", line 121.
$finish at simulation time 4096000
*****
VCS Simulation Report
*****
Time: 40960000 ps
CPU Time: 0.230 seconds; Data structure size: 0.0Mb
Tue Apr 9 16:15:58 2024
CPU time: .325 seconds to compile + .306 seconds to elab + .322 seconds to link + .277 seconds in simulation
user~/lab3_Mul_hw
```

TB2 pass screens:



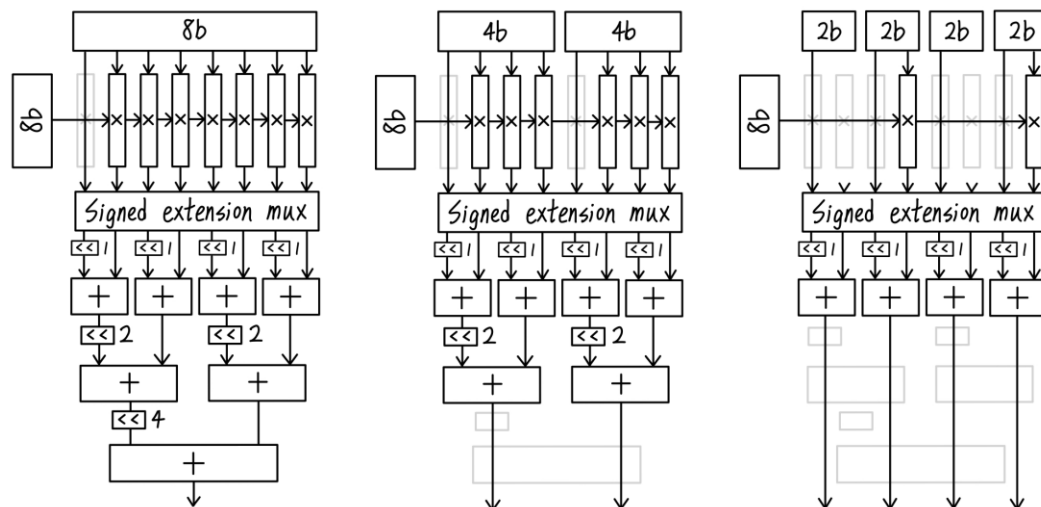
```
*****
+ ERROR +
+ The simulator version is newer than the FSOB dumper version which
+ may cause abnormal behavior, please contact Synopsys support for
+ assistance.
*****
*Verdi* FSOB WARNING: The FSOB file already exists. Overwriting the FSOB file may crash the programs that are using this file.
*Verdi* Create FSOB file 'Mul.fsob'
*Verdi* Begin traversing the scope (Mul.tb.Mul2), layer (0).
*Verdi* Enable *struct and *mda dumping.
*Verdi* End of traversing.
*Verdi* Begin traversing the MDAs under scope (Mul.tb.Mul2), layer (2).
*Verdi* Enable *mda and *packedmda dumping.
*Verdi* End of traversing the MDAs.

!! Congratulations !!

Creator NCOU ASystem Lab SGP
Inspired by https://www.instagram.com/p/CyJcWdYsYn/7uq_index01

$finish called from file "/home/asc2024/asc2024030/lab3_Mul_hw/Mul_tb2.v", line 118.
$finish at simulation time 256000
*****
VCS Simulation Report
*****
Time: 2560000 ps
CPU Time: 0.209 seconds; Data structure size: 0.0Mb
Tue Apr 9 16:16:21 2024
CPU time: .336 seconds to compile + .299 seconds to elab + .314 seconds to link + .240 seconds in simulation
user~/lab3_Mul_hw
```

2. (10%) Explain how your multiplier works, its architecture and the algorithm you apply.



我選擇使用的是 Robertson Algorithm，先將 input feature map 和 filter 的每個 bit 進行 8-bits 乘 1-bit 的乘法，其實是根據 filter 的每個 bit 來決定輸出輸入的是 input feature map 或是 8'd0，Signed extension mux 再根據 filter Quant size 來決定要進行多少位數的乘法。

如果 filter Quant size 是 4'd8 就進行 8-bits 乘 8-bit 的乘法，把 8-bits 乘 1-bit 的 multiplier 的 output 的第 0~6 個進行 signed extension 後為第 0~6 個 output，再根據 Robertson Algorithm 用 filter 的最高位 input feature map 的第 7 個 bit 判斷正負，並且決定第 7 個 output 應該為 24'd0 或是負的 input feature map，並將 8 個 output 根據 Robertson Algorithm 進行 shift 後相加就可以得到 8-bits 乘 8-bit 的乘法的積。

如果 filter Quant size 是 4'd4 就進行 4-bits 乘 4-bit 的乘法，把 8-bits 乘 1-bit 的 multiplier 的 output 的第 0~2 個和第 4~6 個進行 signed extension 後為第 0~2 個和第 4~6 個 output，再根據 Robertson Algorithm 用 filter 的最高位 input feature map 的第 7 個和第 3 個 bit 判斷正負，並且決定第 7 個和第 3 個 output 應該為 24'd0 或是負的 input feature map，並將 8 個 output 根據 Robertson Algorithm 進行 shift 後相加就可以得到 4-bits 乘 4-bit 的乘法的積。

如果 filter Quant size 是 4'd2 就進行 2-bits 乘 2-bit 的乘法，把 8-bits 乘 1-bit 的 multiplier 的 output 的第 0 個和第 4 個分解出兩個 input feature map 分別進行 signed extension 後為第 0 個、第 2 個、第 4 個和第 6 個 output，再根據 Robertson Algorithm 用 filter 的最高位 input feature map 的第 1 個和第 5 個 bit 判斷正負，並且決定第 1 個、第 3 個、第 5 個和第 7 個 output 應該為 24'd0 或是負的 input feature map，並將 8 個 output 根據 Robertson Algorithm 進行 shift 後相加就可以得到 2-bits 乘 2-bit 的乘法的積。

3. (10%) Introduce each module and write down how do you reuse hardware resources to accomplish 3tbs.
- Mul_hybrid.v : Hybrid multiplier , 包含所有 module 的 top module 。
 - Multiplier_8_1.v : 8-bits 乘 1-bit 的 multiplier , 其實是根據 1-bit 來決定輸出輸入的是 8-bits 或是 8'd0 的 mux 。
 - Signed_extension_mux.v : 將 8-bits 乘 1-bit 的 multiplier 輸出的 8-bits 進行 signed extension 變成 24-bits , 並且根據 Robertson Algorithm 用乘數的最高位判斷正負 , 並且決定最高位應該加零或是加負的被乘數 。
 - Shifter1.v : 將輸入向左 shift 1-bit 後輸出 。
 - Shifter2.v : 將輸入向左 shift 2-bit 後輸出 。
 - Shifter4.v : 將輸入向左 shift 4-bit 後輸出 。
 - Adder24.v : 將兩個 24-bits 的輸入相加後輸出 。

我進行重複利用的部分是 Multiplier_8_1.v 中 8-bits 乘 1-bit 的 multiplier 在每種輸入的時候都會重複利用到 , 還有在進行 shift 和相加時把 Shifter 和 Adder 重複利用 , 如果是 8-bits 乘 8-bit 的乘法會使用到 Multiplier_8_1、Shifter1、Shifter2、Shifter4 和 Adder24 ; 如果是 4-bits 乘 4-bit 的乘法會重複使用到 Multiplier_8_1、Shifter1、Shifter2 和 Adder24 ; 如果是 2-bits 乘 2-bit 的乘法會重複使用到 Multiplier_8_1、Shifter1 和 Adder24 。

4. (10%) How many (minimum) FLOPs and MACs does VGG16-Cifar10 1stlayer require? Write down your calculation process. (without batch normalization and bias)

$$\text{FLOPs} = (3 * 3 * 3 + 3 * 3 * 3 - 1) * 32 * 32 * 64 = 3473408 \text{ FLOPs}$$

$$\text{MACs} = 3 * 3 * 3 * 32 * 32 * 64 = 1769472 \text{ MACs}$$

5. (10%) How many (minimum) FLOPs and MACs does a fully connected (512-10) layer require? Write down your calculation process. (without batch normalization and bias)

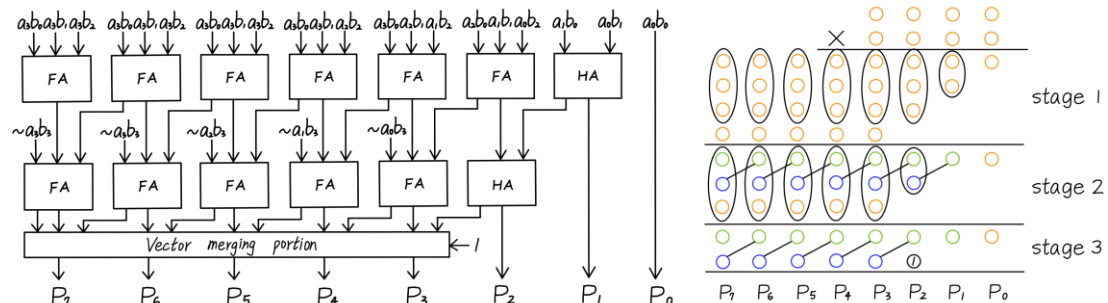
$$\text{FLOPs} = (512 + 512 - 1) * 10 = 10230 \text{ FLOPs}$$

$$\text{MACs} = 512 * 10 = 5120 \text{ MACs}$$

6. (10%) Compare the difference among using Robertson Algorithm, Booth Algorithm and Modified Booth Algorithm with Hybrid Multiplier with a table.

	缺點(增加的硬體)	優點(減少的步驟)
Robertson Algorithm	需要硬體資源較少 , 但速度和硬體使用率不好	實現較為簡單
Booth's Algorithm	需要檢測兩位數組合來決定操作的硬體 , 速度快	減少乘法器中加法減法的操作次數
Modified Booth's Algorithm	需要檢測三位數組合來決定操作的硬體 , 速度更快	減少更多乘法器中加法減法的操作次數

7. (10%) (Architecture 2) Design 4-bit & 4-bit multiplier (signed-number multiplication) with full and half adders follow architecture and point figure format in page 14. Introduce your architecture and method. Count and mark the latency/critical path of 4-bit & 4-bit multiplier. (you can use Adder as time unit).



Architecture Figure

Point Figure

我是根據 Robertson Algorithm 和 Wallace Tree 來設計這個 4-bit & 4-bit multiplier，因為 Robertson Algorithm 需要根據乘數的最高位判斷正負，並且決定最高位應該加零或是加負的被乘數，所以在 stage 2 的 FA 是加取反過後的 b_3 乘上被乘數並且在最低位數加 1 來達成 Robertson Algorithm，如果 b_3 是 1 代表乘數是負數所以要加上負的被乘數，所以是加上被乘數取反並且在最低位數加 1；如果 b_3 是 0 代表乘數是正數所以要加上 0，會因為 b_3 是 0 所以 b_3 乘上被乘數取反後會都是 1，再加上最低位數的 1 後會超出輸出的範圍，所以就會變成加上 0，同時我也使用 Wallace Tree 的方法來減少一層的運算。

latency/critical path 是 P_7 ，data 會先經過第一層的 FA 後再經過第二層的 FA，再經過 Vector merging portion，會從 P_3 一直加到 P_7 所以又會再經過 5 個 FA，所以 latency/critical path 總共會是 7 個 FA 的時間。

8. (5%) Share your thoughts on this lab. Any takeaways or advices? Or anything you want to say to the assistants. 認真表達心得一律滿分。

這次的 Lab 讓我覺得有些複雜也有點困難，但同時做完這次的 Lab 後也讓我學習到很多東西，我覺得這次的 Lab 最困難的地方是如何根據乘法的演算法和 testbench 中的要求來設計硬體的架構，這個部分也是讓我花最多時間，最後我也是有根據乘法演算法來設計適合的硬體，之後寫成 verilog 的部分就簡單許多，在通過所有 testbench 後再回答 report 的問題就完成這次的 Lab，這次的 Lab 讓我學到有關 Hybrid Multiplier 的設計及運作，讓我收穫很多。