[pp4fpga] Matrix Multiplication

R08943007 黃聖竣

HLS C-sim/Synthesis/Cosim (Screenshot + brief intro) :

矩陣乘法是一種根據兩個矩陣得到第三個矩陣的二元運算，第三個矩陣即前兩者的乘積，矩陣可以用來表示線性映射，矩陣積則可以用來表示線性映射的複合。因此，矩陣乘法是線性代數的基礎工具，不僅在數學中有大量應用，在應用數學、物理學、工程學等領域也有廣泛使用。

這次的實驗是用 HLS 實做一個簡單的矩陣乘法，source code 如下

```
#include "matrixmultiplication.h"



void matrixmul(int A[N][M], int B[M][P], int AB[N][P]) {
  //#pragma HLS ARRAY_RESHAPE variable=A complete dim=2
  //#pragma HLS ARRAY_RESHAPE variable=B complete dim=1
  /* for each row and column of AB */
  row: for(int i = 0; i < N; ++i) {
    col: for(int j = 0; j < P; ++j) {
      //#pragma HLS PIPELINE II=1
      /* compute (AB)i,j */
      int ABij = 0;
    product: for(int k = 0; k < M; ++k) {
        ABij += A[i][k] * B[k][j];
      }
      AB[i][j] = ABij;
    }
  }
}
```

C-sim :

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../matrixmultiplication-top.cpp in debug mode
   Compiling ../../../../matrixmultiplication.cpp in debug mode
   Generating csim.exe
INPUTS
*****************************************
PASS: The output matches the golden output!
*****************************************
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Synthesis :

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	4.205 ns	0.63 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
264257	264257	1.321 ms	1.321 ms	264257	264257	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	153	-
FIFO	-	-	-	-	-
Instance	6	3	491	251	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	89	-
Register	-	-	197	-	-
Total	6	3	688	493	0
Available	280	220	106400	53200	0
Utilization (%)	2	1	~0	~0	0

Cosim :

Cosimulation Report for 'matrixmul'

Result

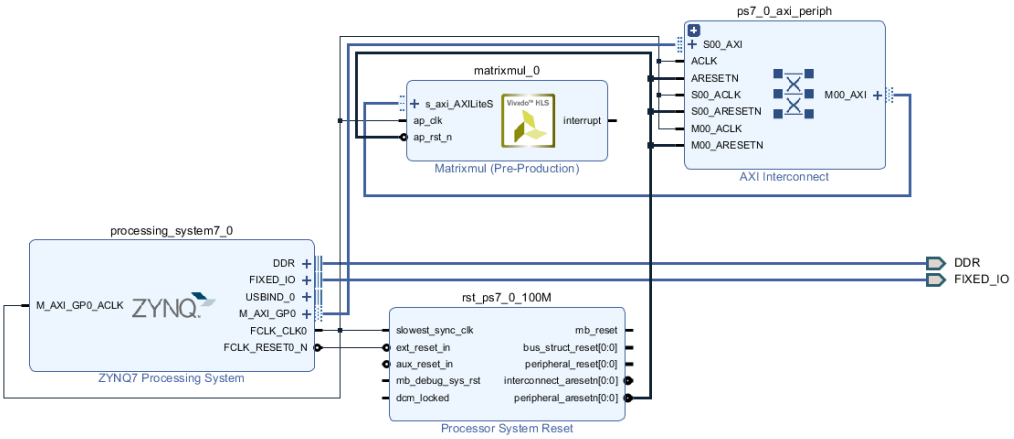
		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	299087	299087	299087	NA	NA	NA

Export the report(.html) using the Export Wizard

System level bring-up (Pynq or U50)

```
import pynq.lib.dma
import numpy as np
import sys
sys.path.append('/home/xilinx')
from pynq import allocate
mmol = pynq.Overlay("/home/xilinx/IPBitFile/mat.bit")
IP = mmol.matrixmul_0

A = np.zeros((32,32), dtype=np.int)
B = np.zeros((32,32), dtype=np.int)
#AB = np.zeros((32,32), dtype=np.int)

for i in range(32):
    for j in range(32):
        A[i][j] = 2;
        B[i][j] = 2;
for i in range(32*32):
    IP.write(0x1000+i*4,2)
    IP.write(0x2000+i*4,2)

IP.write(0x00, 0x01)

while (IP.read(0x00) & 0x4) == 0x0:
    continue
check = 0
for i in range(32*32):
    if (IP.read(0x3000+i*4) != 128):
        check =1
        print('FALSE')
        break
if(check == 0):
    print('PASS')
#print(A)
#print(B)
#print(AB)
```

```
PASS
```



Improvement - throughput, area

首先加上 pipeline II = 1，可以發現 latency 少了約一半，但相對的 resource 也多了相當多，接下來進一步地進行優化，對 input 加上 array partition，我們可以發現 latency 下降了非常多，但相對的 BRAM 的數量也多了很多。

Synthesis comparisom :

Performance Estimates

Timing

Clock		baseline	pipeline	improve
ap_clk	Target	5.00 ns	5.00 ns	5.00 ns
	Estimated	4.205 ns	4.371 ns	4.439 ns

Latency

		baseline	pipeline	improve
Latency (cycles)	min	264257	32792	1038
	max	264257	32792	1038
Latency (absolute)	min	1.321 ms	0.164 ms	5.190 us
	max	1.321 ms	0.164 ms	5.190 us
Interval (cycles)	min	264257	32792	1038
	max	264257	32792	1038

Utilization Estimates

	baseline	pipeline	improve
BRAM_18K	6	6	116
DSP48E	3	63	96
FF	688	7047	11785
LUT	493	2433	1617
URAM	0	0	0

Github: https://github.com/schuang23/MSOC.git