[xhls] Squared_difference_accumulate

R08943007 黃聖竣

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

這次的實驗是運用 Vivado HLS 實作"Squared Difference Accumulate" function，其 code 如下

```
#include "../src/diff_sq_acc.h"

void diff_sq_acc(din_t a[N], din_t b[N], dout_t *dout)
{
    int i;
    int acc= 0;
    int a_reg, b_reg, sub, sub2;

    for(i=0; i<N; i++)
    {
        //#pragma HLS PIPELINE II=1

        a_reg  = a[i];
        b_reg  = b[i];
        sub = a_reg - b_reg;
        sub2 = sub*sub;
        acc += sub2;
    }

    *dout = acc;
}
```

C-sim :

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../src/diff_sq_acc.cpp in debug mode
   Generating csim.exe
got 442798871 expected 442798871
got 262947932 expected 262947932
got 314183194 expected 314183194
got 465177013 expected 465177013
got 704061072 expected 704061072
got 575273685 expected 575273685
got 544620712 expected 544620712
got 521730637 expected 521730637
got 220538590 expected 220538590
got 229031173 expected 229031173
TEST SUCCESS!
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Synthesis :

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	4.00 ns	3.127 ns	0.50 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
51	51	0.204 us	0.204 us	51	51	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	1	-	-	-
Expression	-	-	0	46	-
FIFO	-	-	-	-	-
Instance	4	-	250	276	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	56	-
Register	-	-	78	-	-
Total	4	1	328	378	0
Available	280	220	106400	53200	0
Utilization (%)	1	~0	~0	~0	0

Cosim :

Cosimulation Report for 'diff_sq_acc'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	179	182	183	180	183	184

Export the report(.html) using the Export Wizard

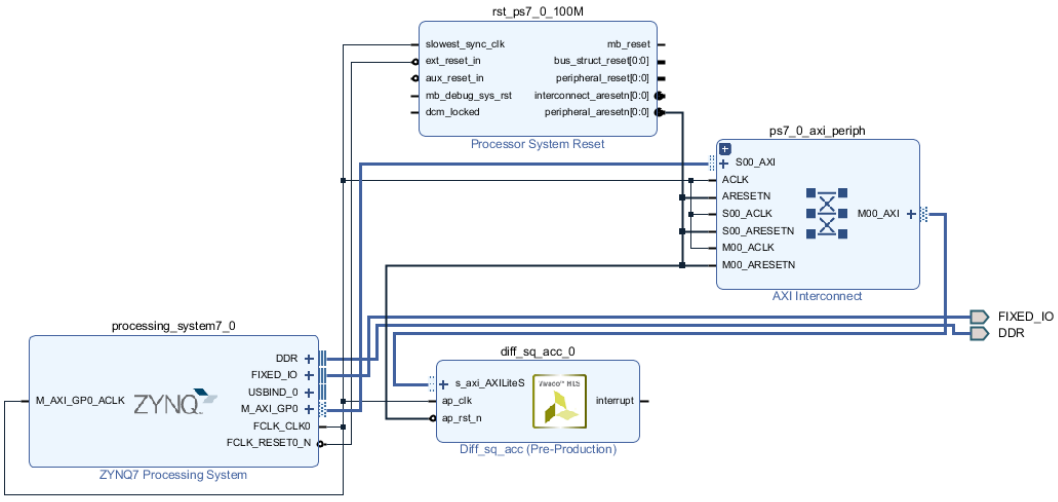
System level bring-up (Pynq or U50)

Interface 的部分皆選用 s_axilite

```
import pynq.lib.dma
import numpy as np
import sys
sys.path.append('/home/xilinx')
from pynq import allocate
import random
mmol = pynq.Overlay("/home/xilinx/IPBitFile/sd.bit")
IP = mmol.diff_sq_acc_0

A = np.zeros((3,), dtype=np.int)
B = np.zeros((3,), dtype=np.int)
acc_test = 0
for i in range(3):
    A[i] = random.randint(1,100)
    B[i] = random.randint(1,100)
print(A)
print(B)
for i in range(3):
    diff = A[i] - B[i]
    diff2 = diff*diff
    acc_test += diff2
for i in range(3):
    IP.write(0x20+i*4,int(A[i]))
    IP.write(0x40+i*4,int(B[i]))
IP.write(0x00, 0x01)
while (IP.read(0x00) & 0x4) == 0x0:
    continue
acc = IP.read(0x60)
print('acc_test = ',acc_test)
print('acc = ',acc)
if(acc_test == acc):
    print('PASS')
else:
    print('FALSE')
```

```
[42 20 45]
[13 55 21]
acc_test =  2642
acc =  2642
PASS
```



Improvement - throughput, area

因為這次的 code 部分相對簡短，因此這裡所做的優化只有將 pipeline 設成 II =1，由下面 synthesis 比較的結果可以發現，latency 減少到約原本的四分之一，但是 area 並沒有增加太多

Synthesis comparisom

Performance Estimates

Timing

Clock		baseline	Kintex_UltraScale
ap_clk	Target	4.00 ns	4.00 ns
	Estimated	3.127 ns	3.127 ns

Latency

		baseline	Kintex_UltraScale
Latency (cycles)	min	51	15
	max	51	15
Latency (absolute)	min	0.204 us	60.000 ns
	max	0.204 us	60.000 ns
Interval (cycles)	min	51	15
	max	51	15

Utilization Estimates

	baseline	Kintex_UltraScale
BRAM_18K	4	4
DSP48E	1	1
FF	328	391
LUT	378	415
URAM	0	0

Cosim :

Cosimulation Report for 'diff_sq_acc'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	140	143	144	141	144	145

Export the report(.html) using the Export Wizard

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