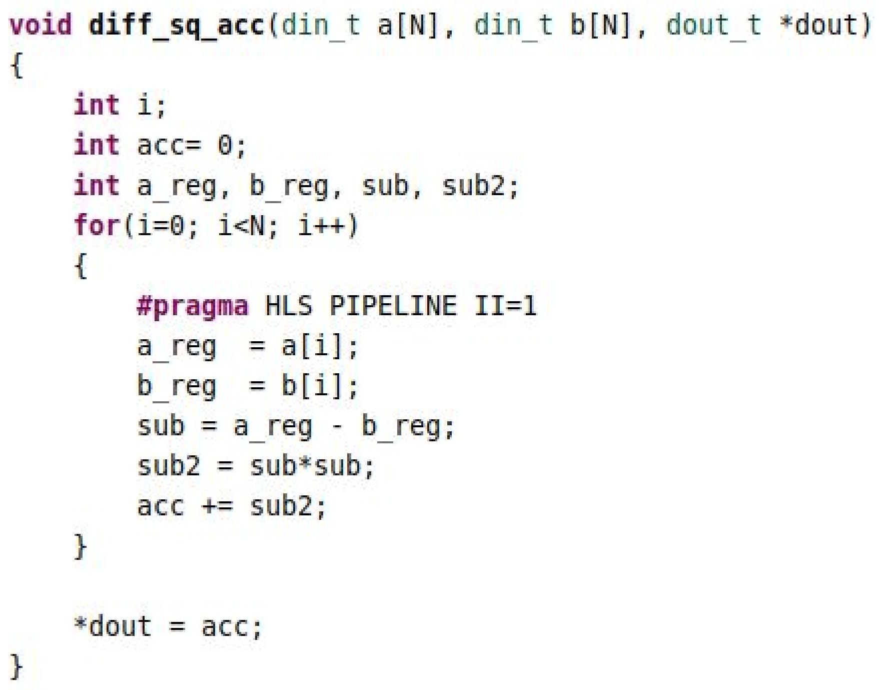
MSoC self-paced learning project Squared Difference Accumulator

R09943017 林奕廷

1. Introduction

這項專案的目的是藉由Vivado HLS來實作一個均方誤差累加器。Top function在diff_sq_acc.h和diff_sq_acc.cpp檔中定義。原本的code如下：



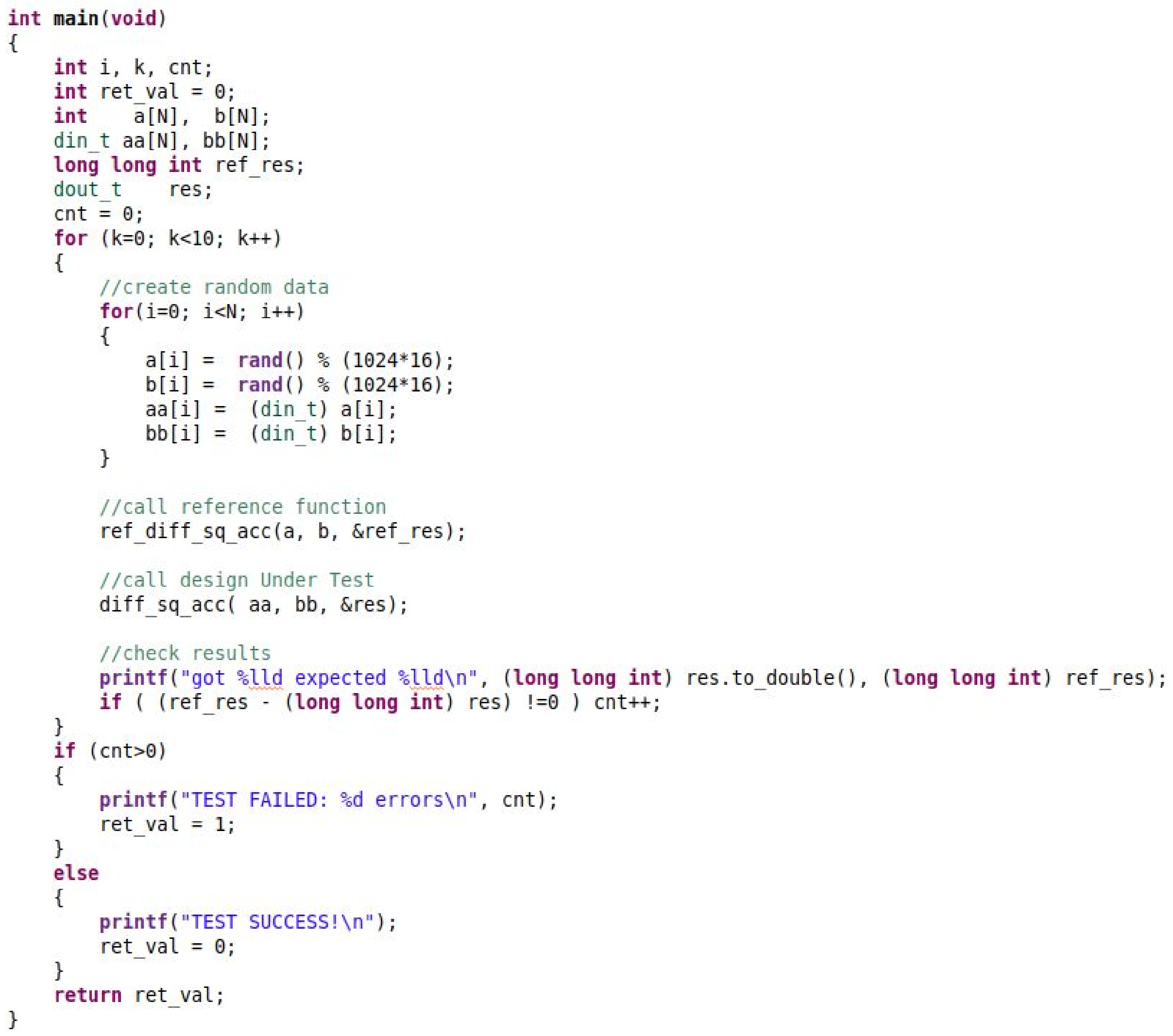
```
void diff_sq_acc(din_t a[N], din_t b[N], dout_t *dout)
{
    int i;
    int acc= 0;
    int a_reg, b_reg, sub, sub2;
    for(i=0; i<N; i++)
    {
        #pragma HLS PIPELINE II=1
        a_reg  = a[i];
        b_reg  = b[i];
        sub = a_reg - b_reg;
        sub2 = sub*sub;
        acc += sub2;
    }

    *dout = acc;
}
```

kernel function (top function)在迴圈中會依序從array a 和 array b取值，相減平方後暫存到acc中。所有iterations執行完畢後再將acc輸出。

2. HLS C-simulation

透過C-simulation執行資料夾中提供的testbench



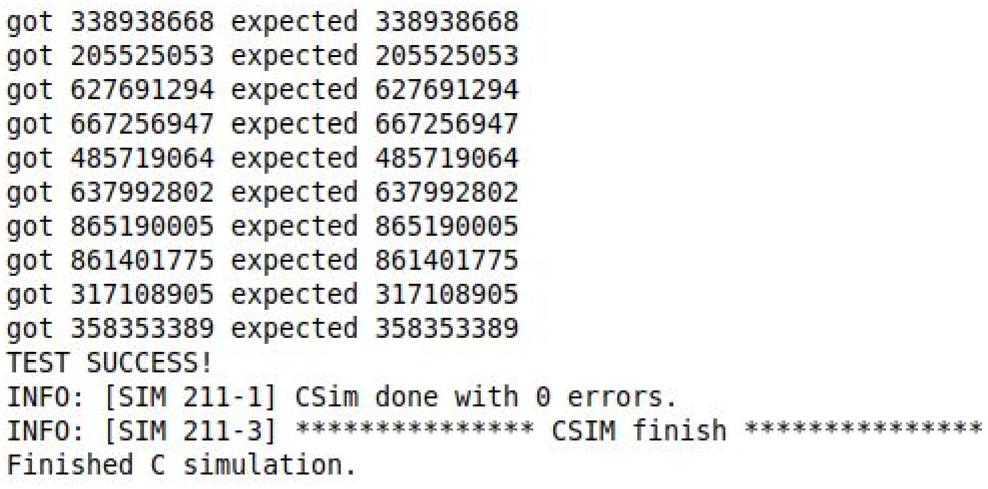
```
int main(void)
{
    int i, k, cnt;
    int ret_val = 0;
    int    a[N],  b[N];
    din_t aa[N], bb[N];
    long long int ref_res;
    dout_t    res;
    cnt = 0;
    for (k=0; k<10; k++)
    {
        //create random data
        for(i=0; i<N; i++)
        {
            a[i] =  rand() % (1024*16);
            b[i] =  rand() % (1024*16);
            aa[i] =  (din_t) a[i];
            bb[i] =  (din_t) b[i];
        }

        //call reference function
        ref_diff_sq_acc(a, b, &ref_res);

        //call design Under Test
        diff_sq_acc( aa, bb, &res);

        //check results
        printf("got %lld expected %lld\n", (long long int) res.to_double(), (long long int) ref_res);
        if ( (ref_res - (long long int) res) !=0 ) cnt++;
    }
    if (cnt>0)
    {
        printf("TEST FAILED: %d errors\n", cnt);
        ret_val = 1;
    }
    else
    {
        printf("TEST SUCCESS!\n");
        ret_val = 0;
    }
    return ret_val;
}
```

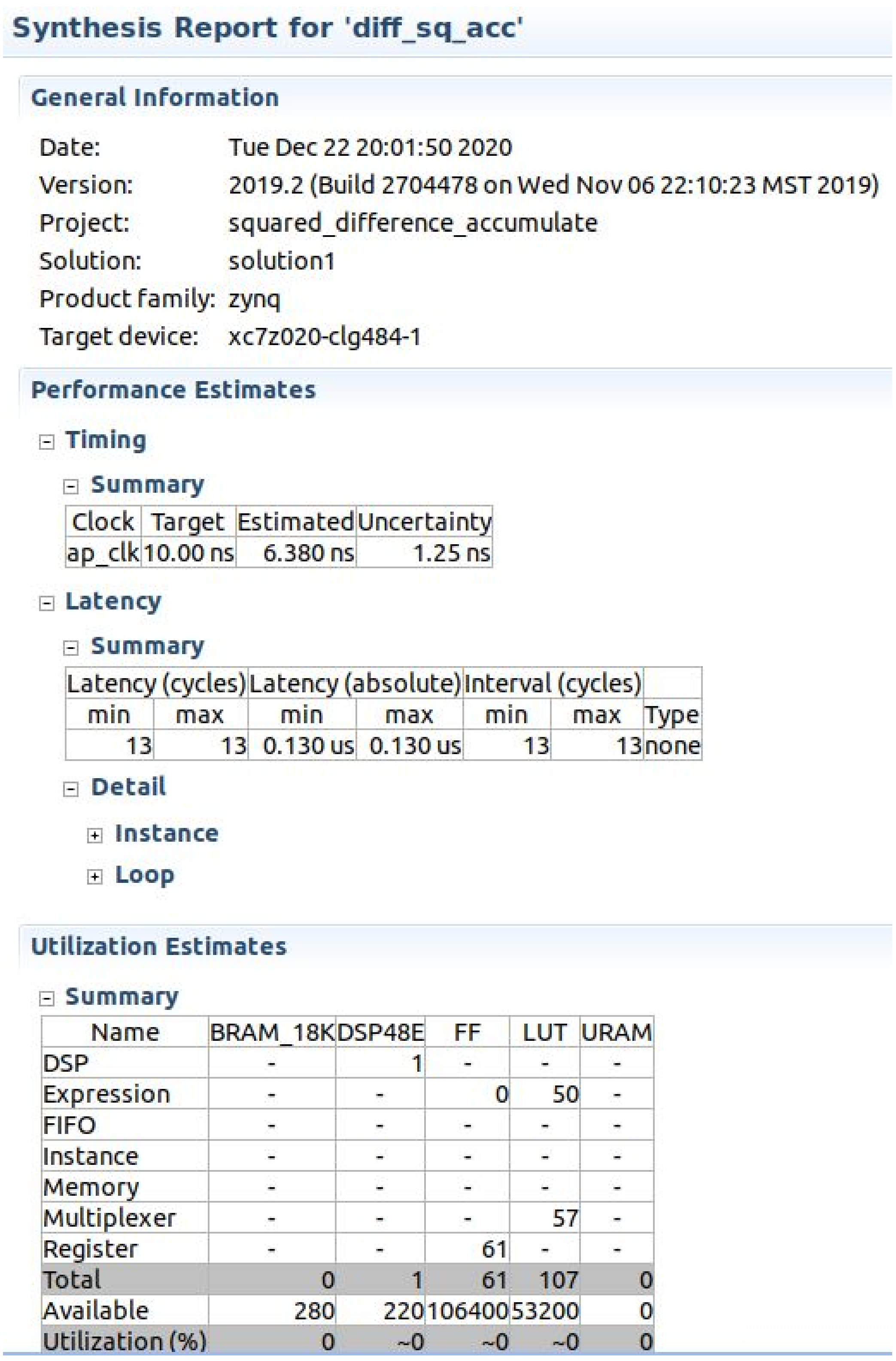
得到的結果如下所示：



```
got 338938668 expected 338938668
got 205525053 expected 205525053
got 627691294 expected 627691294
got 667256947 expected 667256947
got 485719064 expected 485719064
got 637992802 expected 637992802
got 865190005 expected 865190005
got 861401775 expected 861401775
got 317108905 expected 317108905
got 358353389 expected 358353389
TEST SUCCESS!
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] *************** CSIM finish ***************
Finished C simulation.
```

3. HLS Synthesis

合成的結果如下圖所示：



Synthesis Report for 'diff_sq_acc'

General Information

Date:	Tue Dec 22 20:01:50 2020
Version:	2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project:	squared_difference_accumulate
Solution:	solution1
Product family:	zynq
Target device:	xc7z020-clg484-1

Performance Estimates

⊟ Timing

⊟ Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.380 ns	1.25 ns

⊟ Latency

⊟ Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
13	13	0.130 us	0.130 us	13	13	none

⊟ Detail

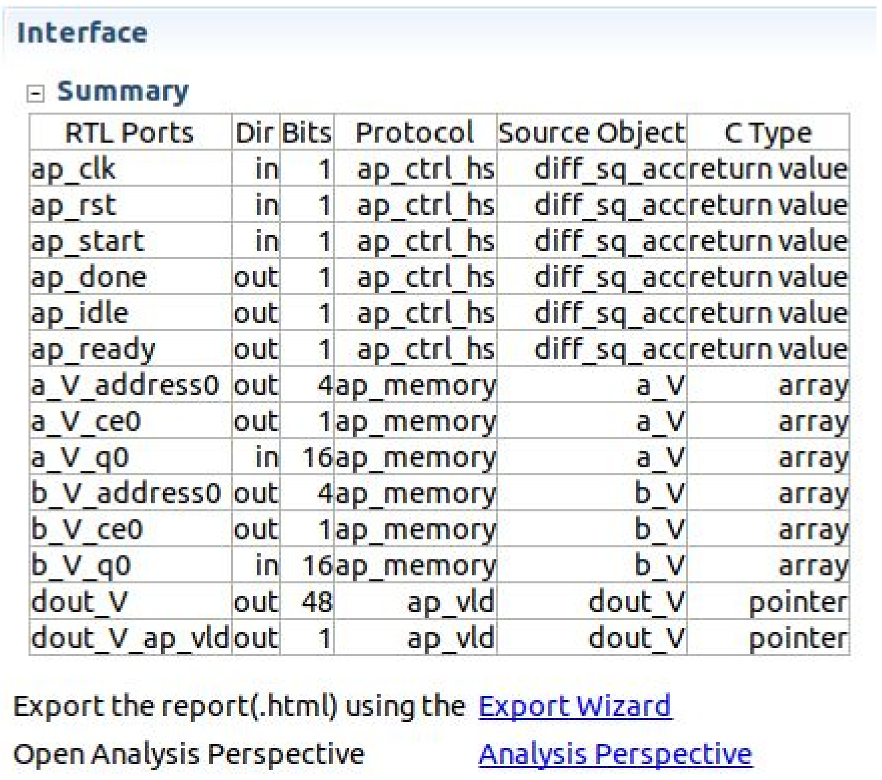
⊞ Instance

⊞ Loop

Utilization Estimates

⊟ Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	1	-	-	-
Expression	-	-	0	50	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	57	-
Register	-	-	61	-	-
Total	0	1	61	107	0
Available	280	220	106400	53200	0
Utilization (%)	0	~0	~0	~0	0


Interface

⊟ Summary

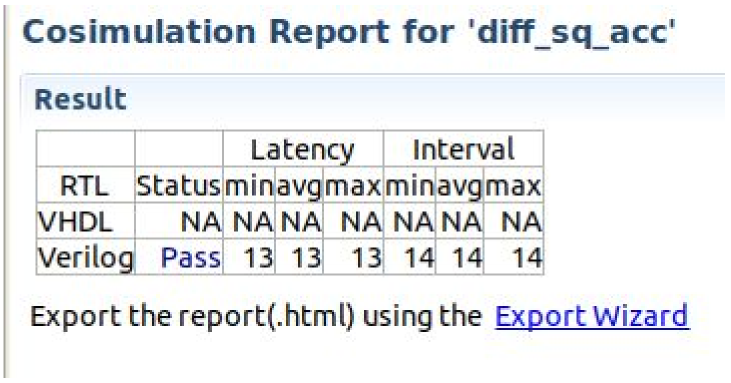
RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	diff_sq_acc	return value
ap_rst	in	1	ap_ctrl_hs	diff_sq_acc	return value
ap_start	in	1	ap_ctrl_hs	diff_sq_acc	return value
ap_done	out	1	ap_ctrl_hs	diff_sq_acc	return value
ap_idle	out	1	ap_ctrl_hs	diff_sq_acc	return value
ap_ready	out	1	ap_ctrl_hs	diff_sq_acc	return value
a_V_address0	out	4	ap_memory	a_V	array
a_V_ce0	out	1	ap_memory	a_V	array
a_V_q0	in	16	ap_memory	a_V	array
b_V_address0	out	4	ap_memory	b_V	array
b_V_ce0	out	1	ap_memory	b_V	array
b_V_q0	in	16	ap_memory	b_V	array
dout_V	out	48	ap_vld	dout_V	pointer
dout_V_ap_vld	out	1	ap_vld	dout_V	pointer

Export the report(.html) using the Export Wizard

Open Analysis Perspective Analysis Perspective

4. Cosimulation

執行Cosimulation得結果如下：

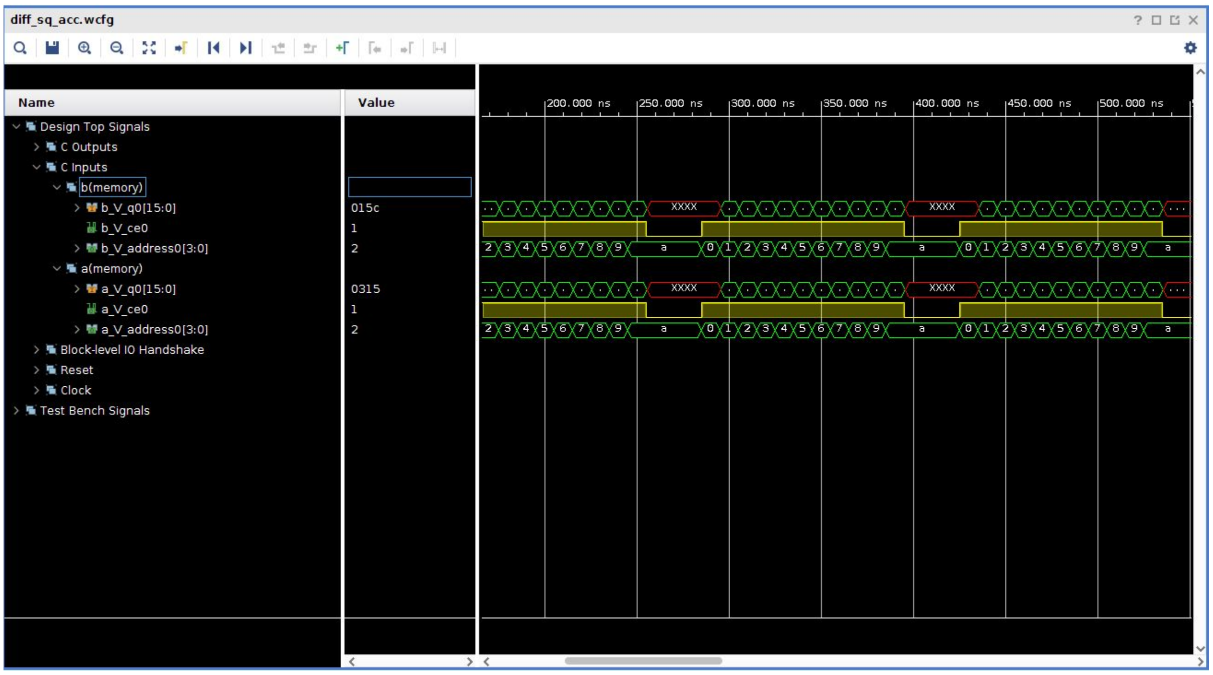

Cosimulation Report for 'diff_sq_acc'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	13	13	13	14	14	14

Export the report(.html) using the Export Wizard

波形如下圖所示(一部分截圖)：


diff_sq_acc.wcfg

```
Name                          Value
Design Top Signals
  C Outputs
  C Inputs
    b(memory)
      b_V_q0[15:0]            015c
      b_V_ce0                 1
      b_V_address0[3:0]       2
    a(memory)
      a_V_q0[15:0]            0315
      a_V_ce0                 1
      a_V_address0[3:0]       2
  Block-level IO Handshake
  Reset
  Clock
Test Bench Signals
```

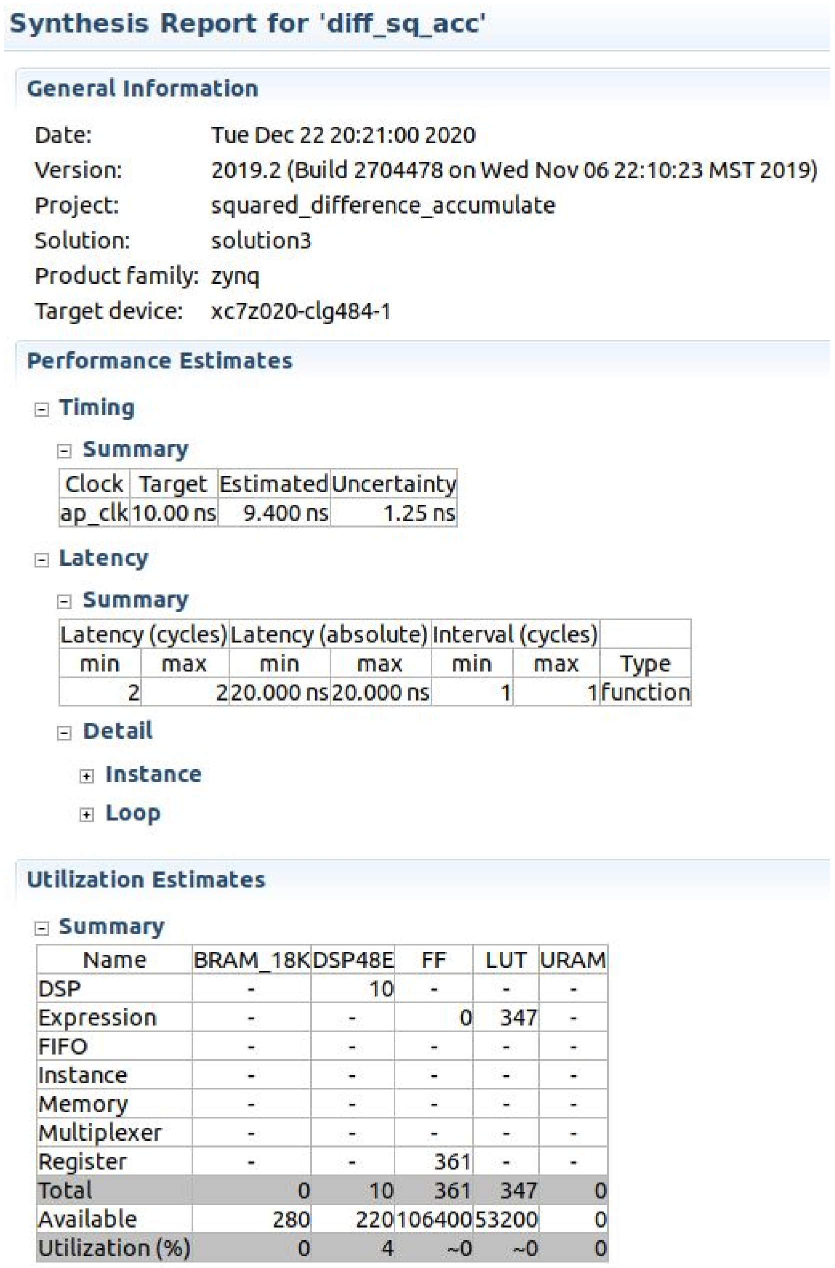
5. Improvement

我特別針對latency進行優化：

```
void diff_sq_acc(din_t a[N], din_t b[N], dout_t *dout)
{
#pragma HLS ARRAY_PARTITION variable=a complete
#pragma HLS ARRAY_PARTITION variable=b complete
    int i;
    int acc= 0;
    int a_reg, b_reg, sub, sub2;
#pragma HLS PIPELINE II=1
    for(i=0; i<N; i++)
    {
        a_reg  = a[i];
        b_reg  = b[i];
        sub = a_reg - b_reg;
        sub2 = sub*sub;
        acc += sub2;
    }

    *dout = acc;
}
```

為了使得for loop中的運算可以平行展開，我在for loop上加入了HLS_PIPELINE的pragma。這樣同時需要對於input的兩個array進行partition，在同個cycle中才有辦法得到多筆資料。
經過上述的優化後performance更改為：



Synthesis Report for 'diff_sq_acc'

General Information

Date:	Tue Dec 22 20:21:00 2020
Version:	2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
Project:	squared_difference_accumulate
Solution:	solution3
Product family:	zynq
Target device:	xc7z020-clg484-1

Performance Estimates

⊟ Timing

⊟ Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	9.400 ns	1.25 ns

⊟ Latency

⊟ Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
2	2	20.000 ns	20.000 ns	1	1	function

⊟ Detail

⊞ Instance

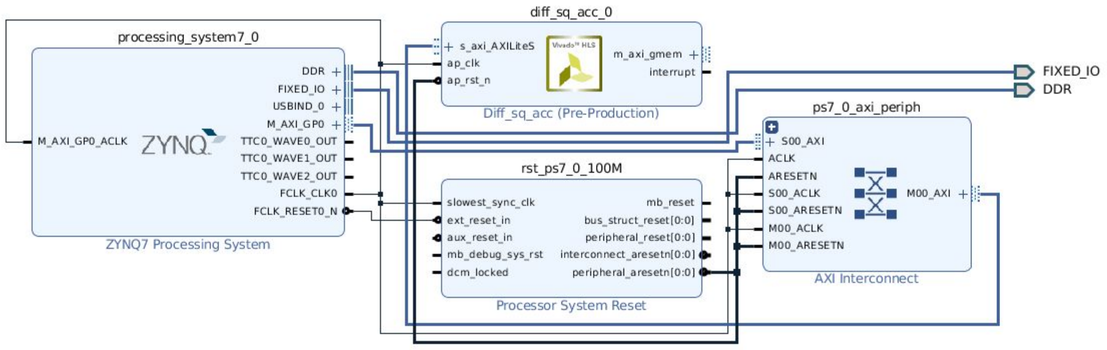
⊞ Loop

Utilization Estimates

⊟ Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	10	-	-	-
Expression	-	-	0	347	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	-	-
Register	-	-	361	-	-
Total	0	10	361	347	0
Available	280	220	106400	53200	0
Utilization (%)	0	4	~0	~0	0

6. System block diagram



此圖為系統的架構圖。IP使用AXILite的方式設定register參數。另外PL和PS端使用AXIMaster的界面進行資料傳輸。

```
#pragma HLS INTERFACE s_axilite port=dout
#pragma HLS INTERFACE m_axi depth=128 port=a offset=slave
#pragma HLS INTERFACE m_axi depth=128 port=b offset=slave
#pragma HLS INTERFACE s_axilite port=return
```

7. Github submission

project中產生的.bit, .hwh和host program皆放在github中：

https://github.com/Lin0611/MSOC_1091_self_paced/blob/main/README.md