MSoC Self-paced 1: Xdesign dataflow_stable_content

R08943011 黃文璁

- **[v] HLS C-sim/synthesis/cosim**
- **[v] System bring-up (zedboard)**
- **[v] Improvement (latency, area)**
- **[v] Github: https://github.com/b04901060/MSoC-Application-Acceleration-with-High-Level-Synthesis**

1. Introduction

本 example 主要用來展示如何使用 #pragma HLS stable variable=X 指令來消除使用 #pragma HLS dataflow 時產生的額外同步操作。

這個 example 中實作功能如下：

給定陣列 A 和 B，計算 A[B[i]] 且 B[i]!=0 的總和。

1. C Synthesis and cosim

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.888 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
39	39	0.390 us	0.390 us	39	39	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	6	-
FIFO	-	-	-	-	-
Instance	0	-	533	1708	-
Memory	2	-	0	0	0
Multiplexer	-	-	-	104	-
Register	-	-	10	-	-
Total	2	0	543	1818	0
Available	280	220	106400	53200	0
Utilization (%)	~0	0	~0	3	0

另外可以注意到合成結果出現 II violation：

Name	Details
[SCHED 204-68]	The II Violation in module 'readmemA' (Function: readmemA): Unable to enforce a carried dependence constraint (II = 2, distance = 1, offset = 1)
[SCHED 204-68]	The II Violation in module 'readmemA' (Function: readmemA): Unable to enforce a carried dependence constraint (II = 3, distance = 1, offset = 1)
[SCHED 204-68]	The II Violation in module 'readmemA' (Function: readmemA): Unable to enforce a carried dependence constraint (II = 4, distance = 1, offset = 1)
[SCHED 204-68]	The II Violation in module 'readmemA' (Function: readmemA): Unable to enforce a carried dependence constraint (II = 7, distance = 1, offset = 1)
[SCHED 204-68]	The II Violation in module 'readmemA' (Function: readmemA): Unable to enforce a carried dependence constraint (II = 9, distance = 1, offset = 1)
[SCHED 204-68]	The II Violation in module 'readmemB' (Function: readmemB): Unable to enforce a carried dependence constraint (II = 1, distance = 1, offset = 1)
[SCHED 204-68]	The II Violation in module 'readmemB' (Function: readmemB): Unable to enforce a carried dependence constraint (II = 2, distance = 1, offset = 1)
[SCHED 204-68]	The II Violation in module 'readmemB' (Function: readmemB): Unable to enforce a carried dependence constraint (II = 3, distance = 1, offset = 1)

Cosim:

Cosimulation Report for 'example'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	46	46	46	NA	NA	NA

-> Verilog pass

2. Optimization

注意到原本的 code 出現 II violation 使得 II 無法為 1。將原本放在 function top 的 #pragma HLS PIPELINE 移動到 for loop 內部後，合成後便可達到 II=1。但實際上整個函數只有一個 for loop，在 function top 和 for loop top 使用 PIPELINE 應該要有相同的結果，具體原因還不太確定。

- **Optimized synthesis result**

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	6.888 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
32	32	0.320 us	0.320 us	32	32	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	6	-
FIFO	-	-	-	-	-
Instance	0	-	225	803	-
Memory	2	-	0	0	0
Multiplexer	-	-	-	104	-
Register	-	-	10	-	-
Total	2	0	235	913	0
Available	280	220	106400	53200	0
Utilization (%)	~0	0	~0	1	0

- **Latency**: 39 -> 32 **(-18%)**
- **FF**: 543 -> 235 **(-57%)**
- **LUT**: 1818 -> 913 **(-50%)**

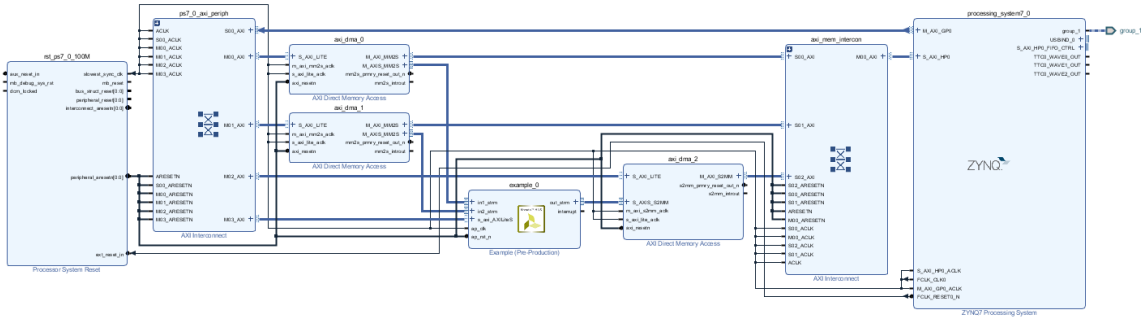
Before

```
void readmemA(stream_t &in_strm, stream_t &req_strm, int *tb)
{
#pragma HLS PIPELINE
A:
    for (int i = 0; i < 10; i++)
    {
        value_t in = in_strm.read();
        int b = tb[in.data];
        if (b < 10)
        {
            req_strm.write(in);
        }
```

After

```
void readmemA(stream_t &in_strm, stream_t &req_strm, int *tb)
{
A:
    for (int i = 0; i < 10; i++)
    {
#pragma HLS PIPELINE
        value_t in = in_strm.read();
        int b = tb[in.data];
        if (b < 10)
        {
            req_strm.write(in);
        }
```

3. Block Diagram



系統整合完成後，便能使用和 lab2 類似的方式，以 python 程式操作 DMA 和 kernel 間的資料搬動：

```
if __name__ == "__main__":
    ol = Overlay("./dataflow.bit")
    ipDesign = ol.example_0
    ipDMAIn0 = ol.axi_dma_0
    ipDMAIn1 = ol.axi_dma_1
    ipDMAOut0 = ol.axi_dma_2

    inBuffer0 = allocate(shape=(10,), dtype=np.int32)
    inBuffer1 = allocate(shape=(10,), dtype=np.int32)
    outBuffer0 = allocate(shape=(1,), dtype=np.int32)

    a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
    b = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

    for i in range(10):
        inBuffer0[i] = a[i]
        inBuffer1[i] = b[i]

    ipDesign.write(0x00, 0x01)
    ipDMAIn0.sendchannel.transfer(inBuffer0)
    ipDMAIn1.sendchannel.transfer(inBuffer1)
    ipDMAOut0.recvchannel.transfer(outBuffer0)
    ipDMAIn0.sendchannel.wait()
    ipDMAIn1.sendchannel.wait()
    ipDMAOut0.recvchannel.wait()
```