MSoC Self-paced 3: PP4FPGA CORDIC

R08943011 黃文璁

- **[v] HLS C-sim/synthesis/cosim**
- **[ ] 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 實作 CORDIC 演算法，CORDIC 透過 shift 運算來計算旋轉時需要的乘、除法，搭配預先計算的參數，如此一來可以用來近似 cos, sin, cosh, sinh 和除法等複雜函數。由於本實作中預設的 fixed point 精度較低，導致計算出來的 error 較大，我將精度由 12bit 增加至 18bit 進行模擬和合成。

2. C Synthesis and cosim

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	7.467 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
97	97	0.970 us	0.970 us	97	97	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	2	-	-	-
Expression	-	-	0	301	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	1	-	0	0	-
Multiplexer	-	-	-	72	-
Register	-	-	199	-	-
Total	1	2	199	373	0
Available	280	220	106400	53200	0
Utilization (%)	~0	~0	~0	~0	0

合成結果沒有 warning：

Name	Details
All Categories	
DATAFLOW	
[XFORM 203-712]	Applying dataflow to function 'readmem', detected/extracted 2 process function(s):
THROUGHPUT	
[HLS 200-789]	**** Estimated Fmax: 145.18 MHz
SCHEDULE	
[SCHED 204-61]	Option 'relax_ii_for_timing' is enabled, will increase II to preserve clock frequency constraints.
LOOP	
[HLS 200-790]	**** Loop Constraint Status: All loop constraints were satisfied.

Cosim:

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	97	97	97	98	98	98

-> Verilog pass

3. Optimization

由於 CORDIC 內部迴圈是常數 iteration，可先採用 unroll 和 pipeline 等手段來加速，我嘗試在 CORDIC 的迴圈中加上 pragma HLS pipeline 指令來減少 latency。

- **Latency optimized synthesis result**

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	7.467 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
66	66	0.660 us	0.660 us	66	66	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	2	-	-	-
Expression	-	-	0	303	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	1	-	0	0	-
Multiplexer	-	-	-	117	-
Register	-	-	184	-	-
Total	1	2	184	420	0
Available	280	220	106400	53200	0
Utilization (%)	~0	~0	~0	~0	0

- **Latency**: 97 -> 66 (**-32%**)
- **FF**: 199 -> 184 (**+133%**)
- **LUT**: 373 -> 420 (**+152%**)

Before

```
for (int j = 0; j < NUM_ITERATIONS; j++) {
    // Determine if we are rotating by a positive or negative angle
    int sigma = (theta < 0) ? -1 : 1;

    // Multiply previous iteration by 2^(-j)
    COS_SIN_TYPE cos_shift = current_cos * sigma * factor;
    COS_SIN_TYPE sin_shift = current_sin * sigma * factor;

    // Perform the rotation
    current_cos = current_cos - sin_shift;
    current_sin = current_sin + cos_shift;

    // Determine the new theta
    theta = theta - sigma * cordic_phase[j];

    factor = factor / 2;
}
```

After

```
for (int j = 0; j < NUM_ITERATIONS; j++) {
  #pragma HLS pipeline
    // Determine if we are rotating by a positive or negative angle
    int sigma = (theta < 0) ? -1 : 1;

    // Multiply previous iteration by 2^(-j)
    COS_SIN_TYPE cos_shift = current_cos * sigma * factor;
    COS_SIN_TYPE sin_shift = current_sin * sigma * factor;

    // Perform the rotation
    current_cos = current_cos - sin_shift;
    current_sin = current_sin + cos_shift;

    // Determine the new theta
    theta = theta - sigma * cordic_phase[j];

    factor = factor / 2;
}
```