CORDIC-SQRT

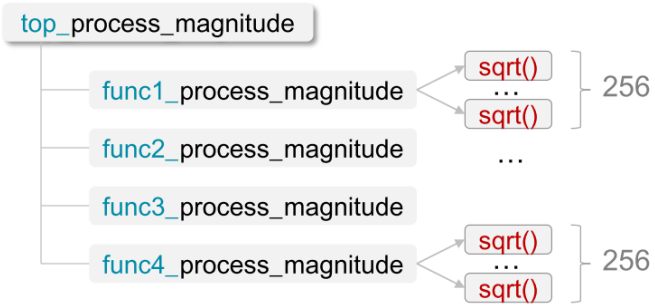
➢ Introduction

Design Under Test

$$\sqrt{a^2 + b^2}$$

The design goals are small and fast (and possibly accurate) – I/O signals are integer numbers – The real design contains 4 cores like this

Top-level design contains 4 functions, each one contains a sqrt() operation called 256 times within a loop.



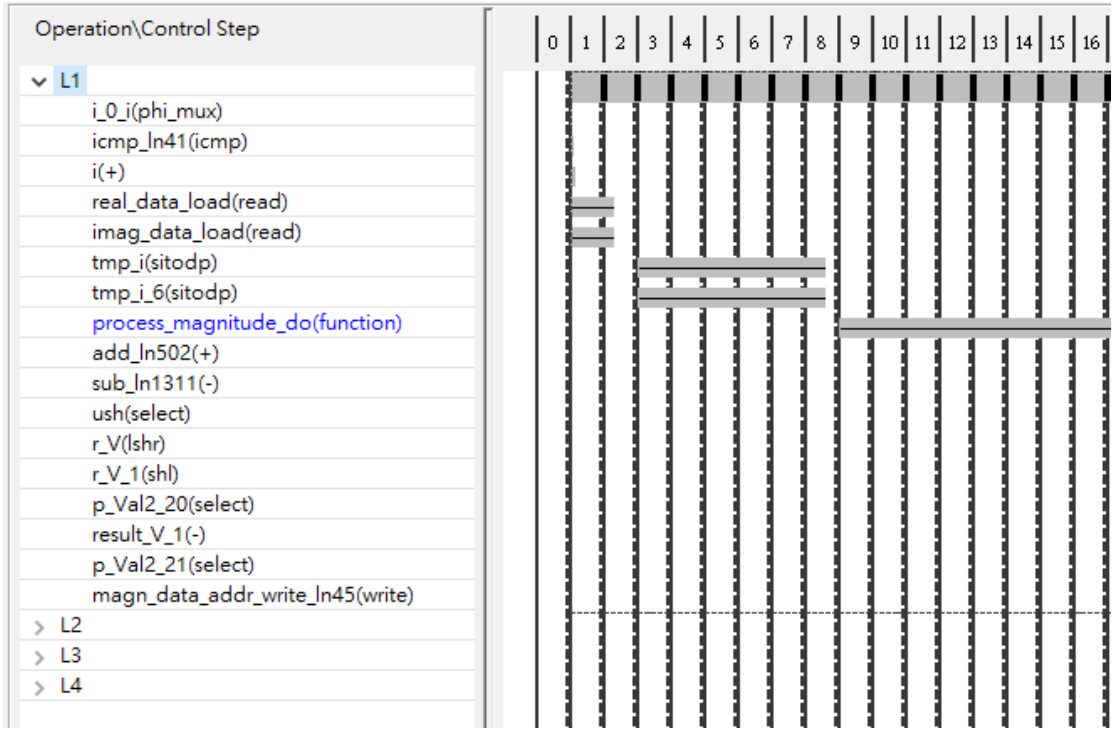
➢ Solution 1 - everything in 64-bit Floating Point

The variable passed in is in double type

```
63  double process_magnitude_double(double real_data, double imag_data)
64  {
65  #pragma HLS INLINE OFF
66  #pragma HLS PIPELINE
67
68      double mag_data, accu_plus, temp_datar, temp_datai;
69
70      temp_datar = real_data * real_data;
71      temp_datai = imag_data * imag_data;
72      accu_plus = temp_datar + temp_datai;
73
74      mag_data =       sqrt(accu_plus);
75
76      return mag_data;
77  }
```

An “integer-to-double” converter is required, and thus increases much latency

Sitodp: Signed-integer to double-precision floating point conversion


Operation\Control Step
L1
i_0_i(phi_mux)
icmp_ln41(icmp)
i(+)
real_data_load(read)
imag_data_load(read)
tmp_i(sitodp)
tmp_i_6(sitodp)
process_magnitude_do(function)
add_ln502(+)
sub_ln1311(-)
ush(select)
r_V(lshr)
r_V_1(shl)
p_Val2_20(select)
result_V_1(-)
p_Val2_21(select)
magn_data_addr_write_ln45(write)
L2
L3
L4
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.750	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
468	468	468	468	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	3228	-
FIFO	-	-	-	-	-
Instance	-	25	4120	5784	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	263	-
Register	0	-	1189	512	-
Total	0	25	5309	9787	0
Available	280	220	106400	53200	0
Utilization (%)	0	11	4	18	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_process_magnitude_do_fu_251	process_magnitude_do	0	25	3296	4494	0
top_process_magnieOg_U10	top_process_magnieOg	0	0	412	645	0
top_process_magnieOg_U11	top_process_magnieOg	0	0	412	645	0
Total		3	0	25	4120	5784	0

```
running Design Under Test
now running 64-bit floating point method
check results
Test successful!

RELATIVE ERROR = 0.0002
MAX ERROR = 1 in sqrt(  9832^2 +  30932^2): EXPECTED= 32457, EFFECTIVE= 32456
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] *************** CSIM finish ***************
```

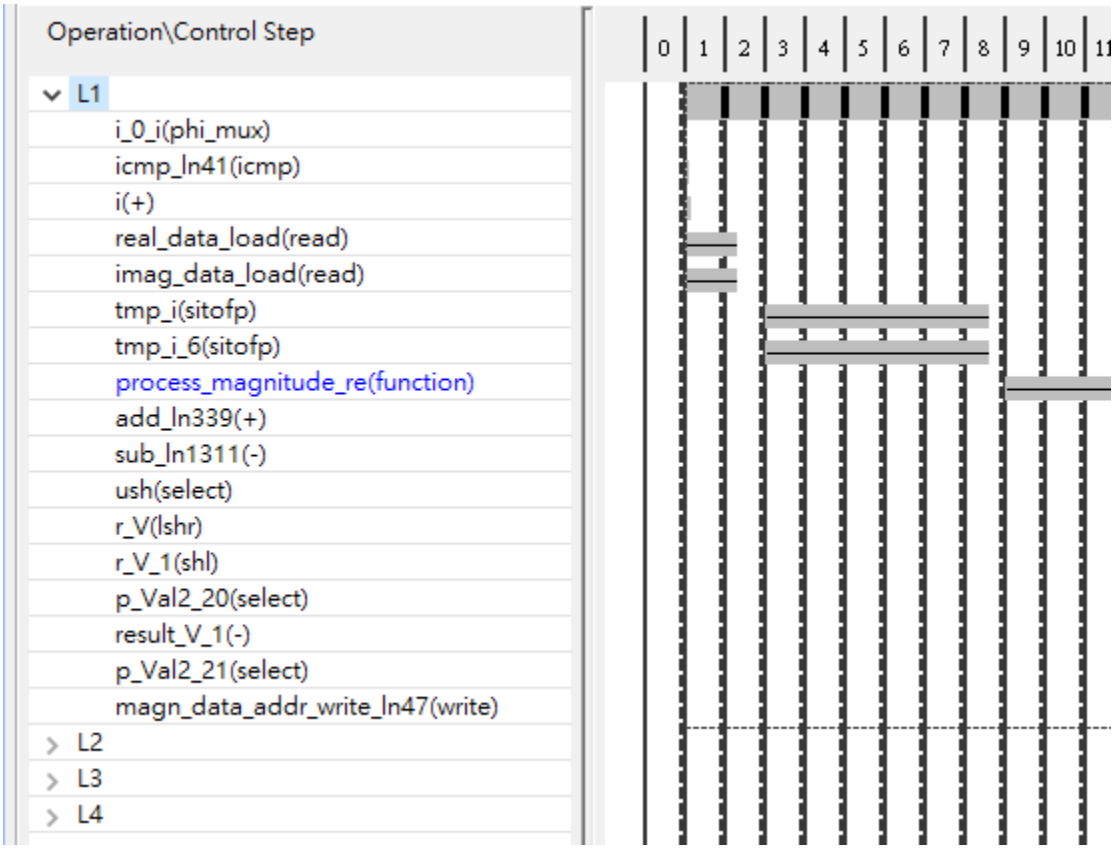
- **Solution 2 - everything in 32-bit Floating Point**

  The variable passed in is in float type

```
79  float process_magnitude_ref(float real_data, float imag_data)
80  {
81  #pragma HLS INLINE OFF
82  #pragma HLS PIPELINE
83
84      float mag_data, accu_plus, temp_datar, temp_datai;
85
86      temp_datar = real_data * real_data;
87      temp_datai = imag_data * imag_data;
88      accu_plus = temp_datar + temp_datai;
89
90      mag_data =       sqrtf(accu_plus);
91
92      return mag_data;
93  }
```

An “integer-to-float” converter is required, and thus increases much latency

Sitodp: Signed-integer to floating point conversion



Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.750	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
384	384	384	384	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	1956	-
FIFO	-	-	-	-	-
Instance	-	8	1769	2764	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	263	-
Register	0	-	1009	256	-
Total	0	8	2778	5239	0
Available	280	220	106400	53200	0
Utilization (%)	0	3	2	9	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_process_magnitude_re_fu_251	process_magnitude_re	0	8	1089	1656	0
top_process_magnieOg_U10	top_process_magnieOg	0	0	340	554	0
top_process_magnieOg_U11	top_process_magnieOg	0	0	340	554	0
Total		3	0	8	1769	2764	0

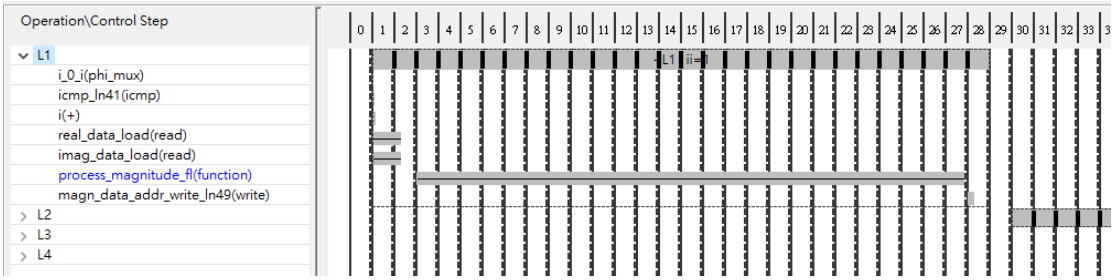
```
running Design Under Test
now running 32-bit floating point method
check results
Test successful!

RELATIVE ERROR = 0.0000
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] *************** CSIM finish ***************
```

- **Solution 3 - mixing 32-bit Fixed and Floating Point**

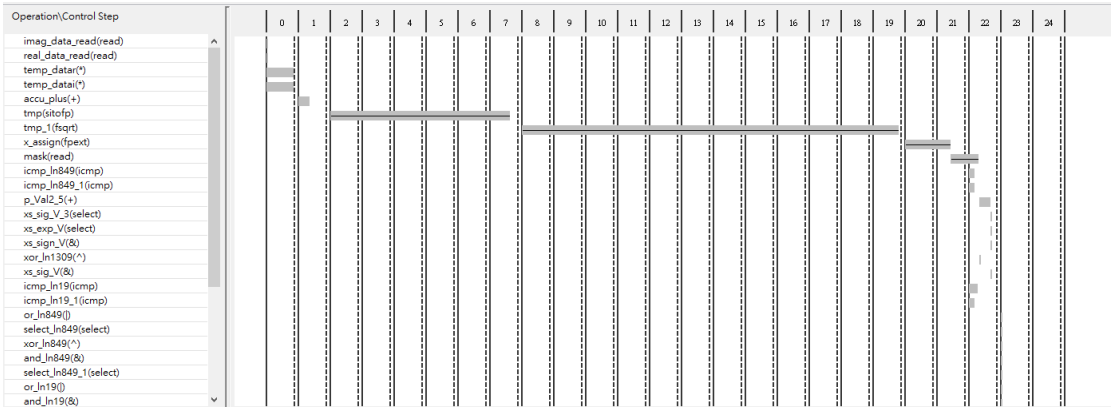
  The variable passed in is in **mixing** type

```
95  #include <math.h> typedef long long int acc_t; // 64-bit data type
96  int process_magnitude_float(int real_data, int imag_data) {
97      #pragma HLS INLINE OFF
98      #pragma HLS PIPELINE
99  int mag_data; // 32-bit data type
100 acc_t accu_plus, temp_datar, temp_datai;
101 // 32x32 to 64-bit integer multiplications
102 temp_datar = (acc_t)real_data * (acc_t)real_data;
103 temp_datai = (acc_t)imag_data * (acc_t)imag_data;
104 accu_plus = temp_datar + temp_datai;
105 mag_data = (int) floor(sqrtf( (float)accu_plus ));
106 return mag_data;
107 }
```

datatype converter is not required in the top level design



However, the sqrt calculation is still done by floating point, so actually the floating-and-int converter is hidden in the “process_magnitude” submodule.



Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.750	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
384	384	384	384	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	1956	-
FIFO	-	-	-	-	-
Instance	-	8	1769	2764	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	263	-
Register	0	-	1009	256	-
Total	0	8	2778	5239	0
Available	280	220	106400	53200	0
Utilization (%)	0	3	2	9	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_process_magnitude_re_fu_251	process_magnitude_re	0	8	1089	1656	0
top_process_magnieOg_U10	top_process_magnieOg	0	0	340	554	0
top_process_magnieOg_U11	top_process_magnieOg	0	0	340	554	0
Total		3	0	8	1769	2764	0

```
running Design Under Test
now running 32-bit floating point sqrt on 32-bit integers
check results
Test successful!

RELATIVE ERROR = 0.0000
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] *************** CSIM finish ***************
```

- **Solution 4 - CORDIC SQRT 32-bit approximation**

$$\text{CORDIC generates } \frac{746}{453}\sqrt{a^2+b^2} \text{ output}$$

The gain has to be compensated by a multiplication per 453/746

For a very good accuracy we selected 32 bits integer and 8 further bits fractional (32+8=40) – this will cost 2 DSP48 slices per one 40x15 bits multiplication

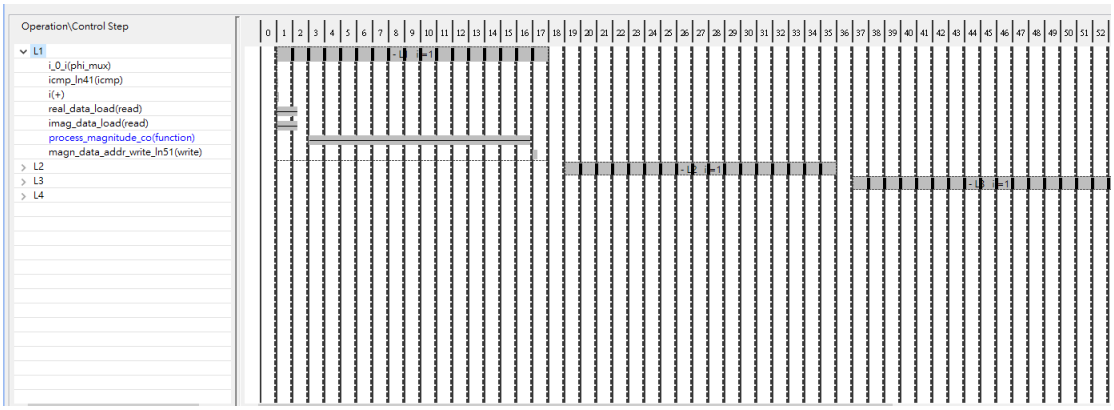
No limits on the range of I/O

```
#define ROT 11


template <int TOT, int INT>
ap_fixed<TOT, INT> cordic_sqrt(dinp_t x0, dinp_t y0)
{

    unsigned char i;


  ap_fixed<TOT, INT> x, y, xp, yp, x2;

  // to compensate the cordic gain of G = 1.0/0.6072
  const ap_fixed<16,1, AP_RND_MIN_INF, AP_SAT> inv_G = 0.607253031529134; // that is 0.6072 == 453 / 746;


  xp=x0;
  yp=y0;
```

```
  for (i=0;i<ROT;i++)
    {
  #pragma HLS PIPELINE II=1
      if (yp<0) {
        x = xp - (yp>>i);
        y = yp + (xp>>i);
      } else {
        x = xp + (yp>>i);
        y = yp - (xp>>i);
      }
    xp=x;
    yp=y;
    }

  // compensating the cordic gain
  #pragma HLS RESOURCE variable=x2 core=MUL6S


    x2     =  xp * inv_G; //xn = ((x * 453) / 746) ;

  //x =   (xp *311)>>9; // much less accurate


    return x2;
```

```
int process_magnitude_cordic(int real_data, int imag_data) {
  int mag_data; // 32-bit data type
  mag_data = (int) cordic_sqrt<40,32>(real_data, imag_data);
  return mag_data;
  }
```

since the “#pragma HLS RESOURCE variable=x2 core=MUL6S” is added, the compensation multiplier is synthesize by a 6-stage multiplier



Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.750	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
368	368	368	368	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	124	-
FIFO	-	-	-	-	-
Instance	2	8	1579	2744	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	236	-
Register	0	-	798	256	-
Total	2	8	2377	3360	0
Available	280	220	106400	53200	0
Utilization (%)	~0	3	2	6	0

Detail

Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_process_magnitude_fl_fu_221	process_magnitude_fl	2	8	1579	2744	0
Total	1	2	8	1579	2744	0

```
running Design Under Test
now running full cordic 32-bit approssimation method
check results
Test successful!

RELATIVE ERROR = 0.0369
MAX ERROR = 1 in sqrt( 21153^2 +  21520^2): EXPECTED= 30175, EFFECTIVE= 30174
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] *************** CSIM finish ***************
```

- **Solution 5 - 18-bit CORDIC**

  the word length of fixed point datatype can be further reduce to (24,18) without lossing too much accuracy.

```
123   typedef ap_int<10> teta_t;
124   typedef ap_int<18> dinp_t;
125   typedef ap_int<24> dout_t;
126   dout_t process_magnitude_cordic(dinp_t real_data, dinp_timag_data) {
127   dout_t mag_data; // 24-bit data type
128   mag_data = (int)
129   cordic_sqrt<24,18>(real_data, imag_data);
130   return mag_data;
131   }
```

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.262	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
5377	5377	5377	5377	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	28	-
FIFO	-	-	-	-	-
Instance	-	2	205	555	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	36	-
Register	-	-	96	-	-
Total	0	2	301	619	0
Available	280	220	106400	53200	0
Utilization (%)	0	~0	~0	1	0

Detail

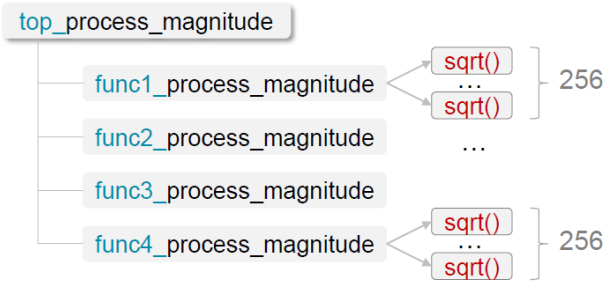
Instance

Instance	Module	BRAM_18K	DSP48E	FF	LUT	URAM
grp_process_magnitude_co_fu_80	process_magnitude_co	0	2	205	555	0
Total		1	0	2	205	555	0

we can further reduce the latency

```
running Design Under Test
now running full cordic 18-bit approssimation method
check results
Test successful!

RELATIVE ERROR = 0.0364
MAX ERROR = 1 in sqrt( 21153^2 +  21520^2): EXPECTED= 30175, EFFECTIVE= 30174
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] *************** CSIM finish ***************
```

- **Solution 6 – PIPELINE**




```
set_directive_pipeline "top_process_magnitude"
set_directive_pipeline "func1_process_magnitude"
set_directive_pipeline "func2_process_magnitude"
set_directive_pipeline "func3_process_magnitude"
set_directive_pipeline "func4_process_magnitude"
```

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	3.702 ns	0.62 ns

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
211	211	1.055 us	1.055 us	212	212	function

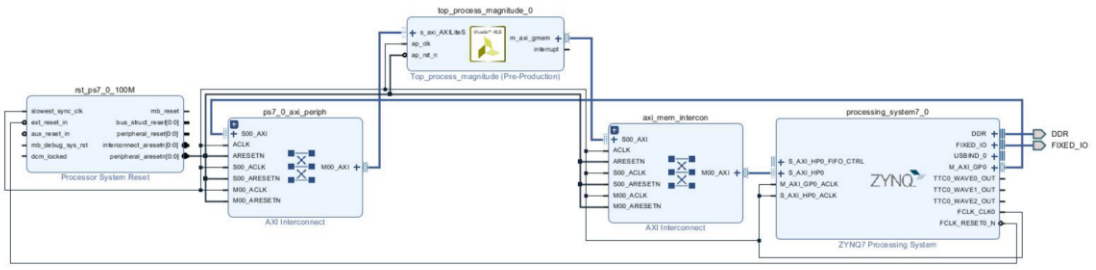
		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_func1_process_magnit_fu_30	func1_process_magnit	52	52	0.260 us	0.260 us	32	32	function
grp_func2_process_magnit_fu_40	func2_process_magnit	52	52	0.260 us	0.260 us	32	32	function
grp_func3_process_magnit_fu_50	func3_process_magnit	52	52	0.260 us	0.260 us	32	32	function
grp_func4_process_magnit_fu_60	func4_process_magnit	52	52	0.260 us	0.260 us	32	32	function

- **System level bring-up (Pynq or U50)**

  Code modification

  interface:m_axi, s_axilite

```
134   set_directive_interface -mode s_axilite "top_process_magnitude"
135   set_directive_interface -mode m_axi -depth 256 -offset slave "top_process_magnitude" real_data -bundle gmem1
136   set_directive_interface -mode m_axi -depth 256 -offset slave "top_process_magnitude" imag_data -bundle gmem2
137   set_directive_interface -mode m_axi -depth 256 -offset slave "top_process_magnitude" magn_data -bundle gmem3
```



Host program

```python
if __name__ == "__main__":
    print("Entry:", sys.argv[0])
    print("System argument(s):", len(sys.argv))

    print("Start of \"" + sys.argv[0] + "\"")

    ol = Overlay("/home/xilinx/IPBitFile/yclin/CORDIC_SQRT.bit")
    ipFIRN11 = ol.top_process_magnitude_0

    numSamples = 256
    ground_truth = np.zeros(numSamples)
    real_data_buffer = allocate(shape=(numSamples,), dtype=np.int32)
    imag_data_buffer = allocate(shape=(numSamples,), dtype=np.int32)
    output_data_buffer = allocate(shape=(numSamples,), dtype=np.int32)
    for i in range(numSamples):
        real_part = random.randint(0,32768)
        imag_part = random.randint(0,32768)
        real_data_buffer[i] = real_part
        imag_data_buffer[i] = imag_part
        ground_truth[i] = np.sqrt(imag_part*imag_part + real_part*real_part)

    ipFIRN11.write(0x10, real_data_buffer.device_address)
    ipFIRN11.write(0x18, imag_data_buffer.device_address)
    ipFIRN11.write(0x20, output_data_buffer.device_address)
    ipFIRN11.write(0x00, 0x01)
    while (ipFIRN11.read(0x00) & 0x4) == 0x0:
        continue

    error = np.abs(ground_truth-output_data_buffer)
    print("error", error)
```

```
error idx: 0
error value: 1.77659890308
error idx: 1
error value: 0.433255820706
error idx: 2
error value: 1.52337189803
error idx: 3
error value: 0.82882944111
error idx: 4
error value: 1.09952143997
error idx: 5
error value: 0.948512367142
error idx: 6
error value: 1.18465326162
error idx: 7
error value: 1.4767887931
error idx: 8
error value: 1.04657863931
error idx: 9
error value: 0.869927950705
============================
Exit process
```