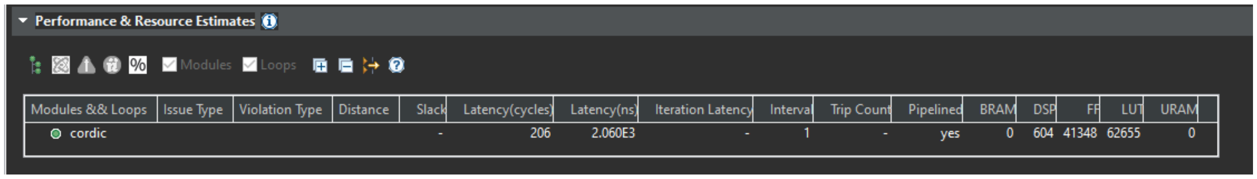
Report

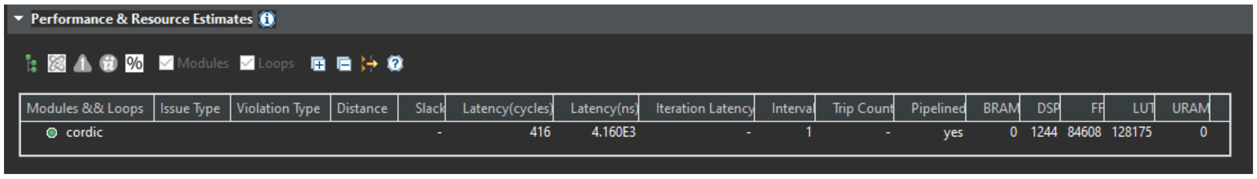
Project: CORDIC

Question 1: *One important design parameter is the number of rotations. Change that number to numbers between 10 and 20 and describe the trends. What happens to performance? Resource usage? Accuracy of the results? Why does the accuracy stop improving after some number of iterations? Can you precisely state when that occurs?*



Modules && Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
cordic				-	206	2.060E3	-	1	-	yes	0	604	41348	62655	0

Figure 1 Iterations = 10



Modules && Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
cordic				-	416	4.160E3	-	1	-	yes	0	1244	84608	128175	0

Figure 2 Iterations = 20

Accuracy is another common design evaluation metric alongside performance and resource usage. The difference between 10 and 20 iterations is approx. double. The trend is that the latency time and resources increase that makes performance decrease. We find that double CORDIC iterations are self-correcting and compensate possible wrong rotations in subsequent iterations. Additional iterations produce a more precise output result.

The increase of error should be related to the simplified conversion between floating-point and fix-point, which was used for the preparation of the input data. The mantissa of 32 bit floating-point numbers holds only 23 bits. So sometimes the last four iterations yield no improvement. Therefore, some trailing bits are advisable to buffer accumulating truncation errors

Question 2: *Another important design parameter is the data type of the variables. Is one data type sufficient for every variable or is it better for each variable to have a different type? Does the best data type depend on the input data? What is the best technique for the designer to determine the data type(s)?*

digital representation of numbers has a huge impact on the complexity of the logic to compute with those numbers. In many cases, HLS tools are able to optimize the representation of each value to simplify the generated hardware. Today code and the amount of data has grown exponentially it's become even more critical to optimize data and datatypes. The size of data doesn't just impact storage size and costs, it also affects code performance.

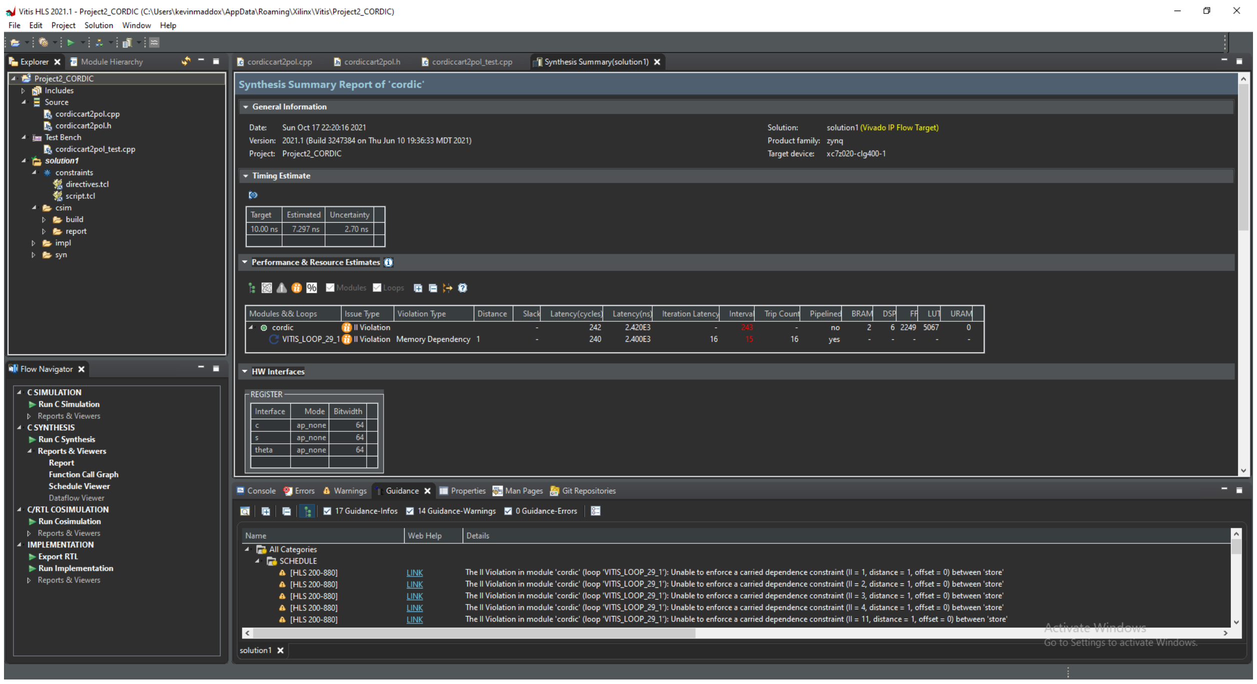
Yes, the best data type depends on the input data because we need to gather clean and consistent data. Optimizing schema and datatypes good logical and physical design is the cornerstone of high performance.

The basic strategy for selecting the best data type is to select the smallest data type that matches the kind of data you have and that allows for all the feasible values of your data.

Question 3: *What is the effect of using simple operations (add and shift) in the CORDIC as opposed to multiply and divide? How does the resource usage change? Performance? Accuracy?*

these operations are not easy to implement in hardware. For example, sine, cosine, square root, and arctan are not simple operations and they require significant number of resources. But we can use the CORDIC to perform these operations using a series of simple iterative rotation operations.

Having a CORDIC, which eliminates the need for costly multiplication. Additions and shifts are very easy to perform for computers, while multiplications and divisions are usually more expensive. shift-and-add allows the possibility to solve items with cheap and simple hardware components.



Vitis HLS 2021.1 - Project2_CORDIC (C:\Users\kevinmaddox\AppData\Roaming\Xilinx\Vitis\Project2_CORDIC)

Synthesis Summary Report of 'cordic'

General Information

Date: Sun Oct 17 22:20:16 2021
Version: 2021.1 (Build 3247384 on Thu Jun 10 19:36:33 MDT 2021)
Project: Project2_CORDIC
Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg400-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.297 ns	2.70 ns

Performance & Resource Estimates

Modules && Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
cordic	II Violation			-	242	2.420E3	-	243	-	no	2	6	2249	5067	0
VITIS_LOOP_29_1	II Violation	Memory Dependency	1	-	240	2.400E3	16	15	16	yes	-	-	-	-	-

HW Interfaces

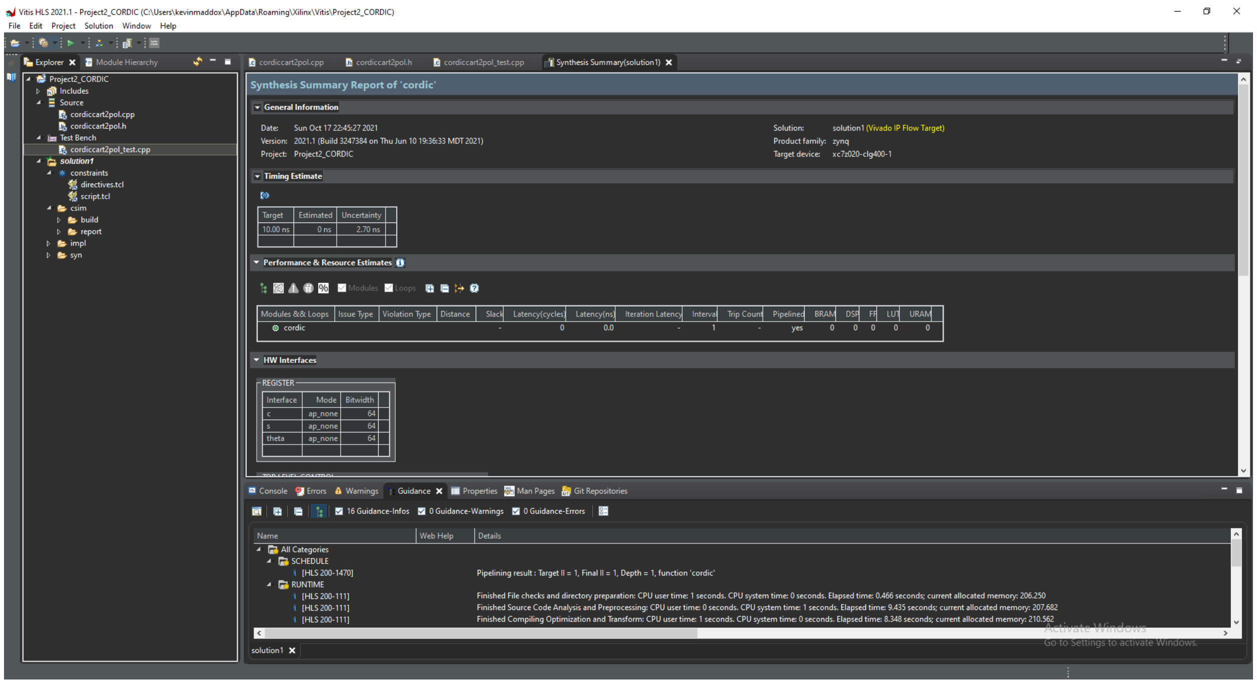
REGISTER

Interface	Mode	Bitwidth
c	ap_none	64
s	ap_none	64
theta	ap_none	64

Guidance: 17 Guidance-Infos, 14 Guidance-Warnings, 0 Guidance-Errors

Name	Web Help	Details
All Categories		
SCHEDULE		
[HLS 200-880]	LINK	The II Violation in module 'cordic' (loop 'VITIS_LOOP_29_1'): Unable to enforce a carried dependence constraint (II = 1, distance = 1, offset = 0) between 'store'
[HLS 200-880]	LINK	The II Violation in module 'cordic' (loop 'VITIS_LOOP_29_1'): Unable to enforce a carried dependence constraint (II = 2, distance = 1, offset = 0) between 'store'
[HLS 200-880]	LINK	The II Violation in module 'cordic' (loop 'VITIS_LOOP_29_1'): Unable to enforce a carried dependence constraint (II = 3, distance = 1, offset = 0) between 'store'
[HLS 200-880]	LINK	The II Violation in module 'cordic' (loop 'VITIS_LOOP_29_1'): Unable to enforce a carried dependence constraint (II = 4, distance = 1, offset = 0) between 'store'
[HLS 200-880]	LINK	The II Violation in module 'cordic' (loop 'VITIS_LOOP_29_1'): Unable to enforce a carried dependence constraint (II = 11, distance = 1, offset = 0) between 'store'

Figure 3 Baseline 1



Vitis HLS 2021.1 - Project2_CORDIC (C:\Users\kevinmaddox\AppData\Roaming\Xilinx\Vitis\Project2_CORDIC)

Synthesis Summary Report of 'cordic'

General Information

Date: Sun Oct 17 22:45:27 2021
Version: 2021.1 (Build 3247384 on Thu Jun 10 19:36:33 MDT 2021)
Project: Project2_CORDIC
Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg400-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	0 ns	2.70 ns

Performance & Resource Estimates

Modules && Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
cordic				-	0	0.0	-	1	-	yes	0	0	0	0	0

HW Interfaces

REGISTER

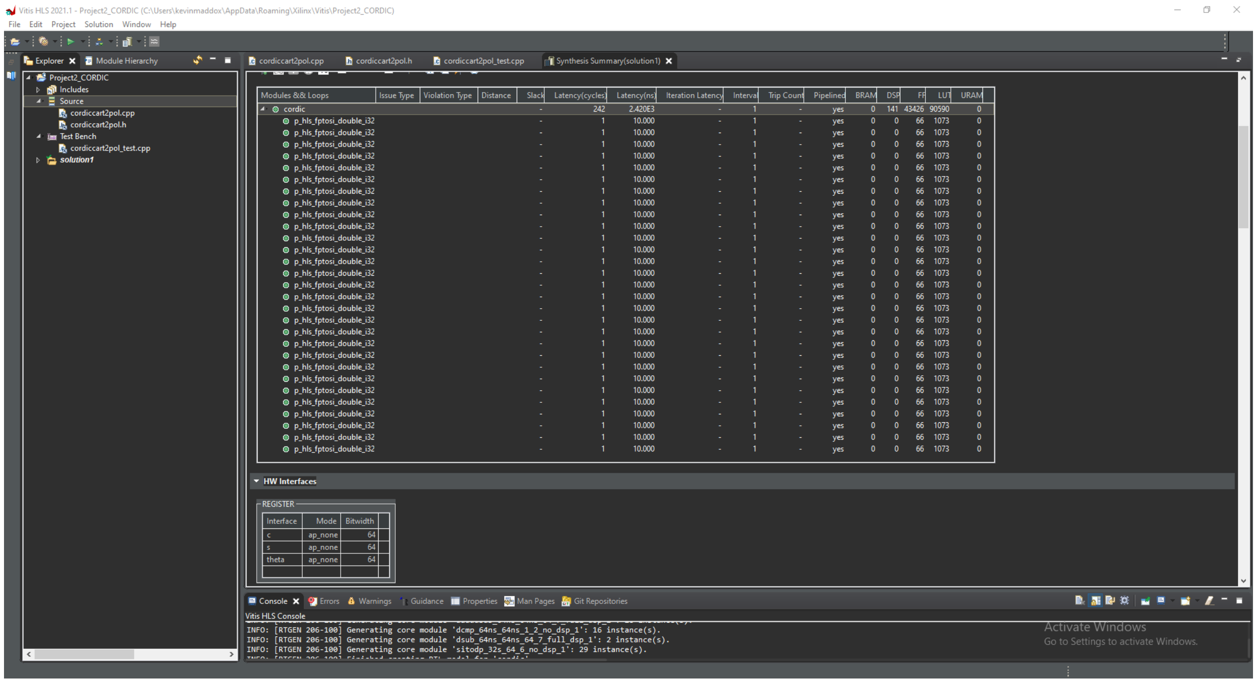
Interface	Mode	Bitwidth
c	ap_none	64
s	ap_none	64
theta	ap_none	64

Guidance: 16 Guidance-Infos, 0 Guidance-Warnings, 0 Guidance-Errors

Name	Web Help	Details
All Categories		
SCHEDULE		
[HLS 200-1470]		Pipelining result : Target II = 1, Final II = 1, Depth = 1, function 'cordic'
RUNTIME		
[HLS 200-111]		Finished File checks and directory preparation: CPU user time: 1 seconds. CPU system time: 0 seconds. Elapsed time: 0.466 seconds; current allocated memory: 206.250
[HLS 200-111]		Finished Source Code Analysis and Preprocessing: CPU user time: 0 seconds. CPU system time: 1 seconds. Elapsed time: 9.435 seconds; current allocated memory: 207.682
[HLS 200-111]		Finished Compiling Optimization and Transform: CPU user time: 1 seconds. CPU system time: 0 seconds. Elapsed time: 8.348 seconds; current allocated memory: 210.562

Figure 4 Baseline 2

Finished Command csynth_design CPU user time: 5 seconds. CPU system time: 1 seconds. Elapsed time: 22.153 seconds; current allocated memory: 295.963 MB.
Total CPU user time: 8 seconds. Total CPU system time: 3 seconds. Total elapsed time: 27.443 seconds; peak allocated memory: 295.977 MB.



Vitis HLS 2021.1 - Project2_CORDIC (C:\Users\kevinmaddox\AppData\Roaming\Xilinx\Vitis\Project2_CORDIC)

Modules && Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
cordic				-	242	2.420E3	-	1	-	yes	0	141	43426	90590	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0
p_hls_fptosi_double_i32				-	1	10.000	-	1	-	yes	0	0	66	1073	0

HW Interfaces

REGISTER

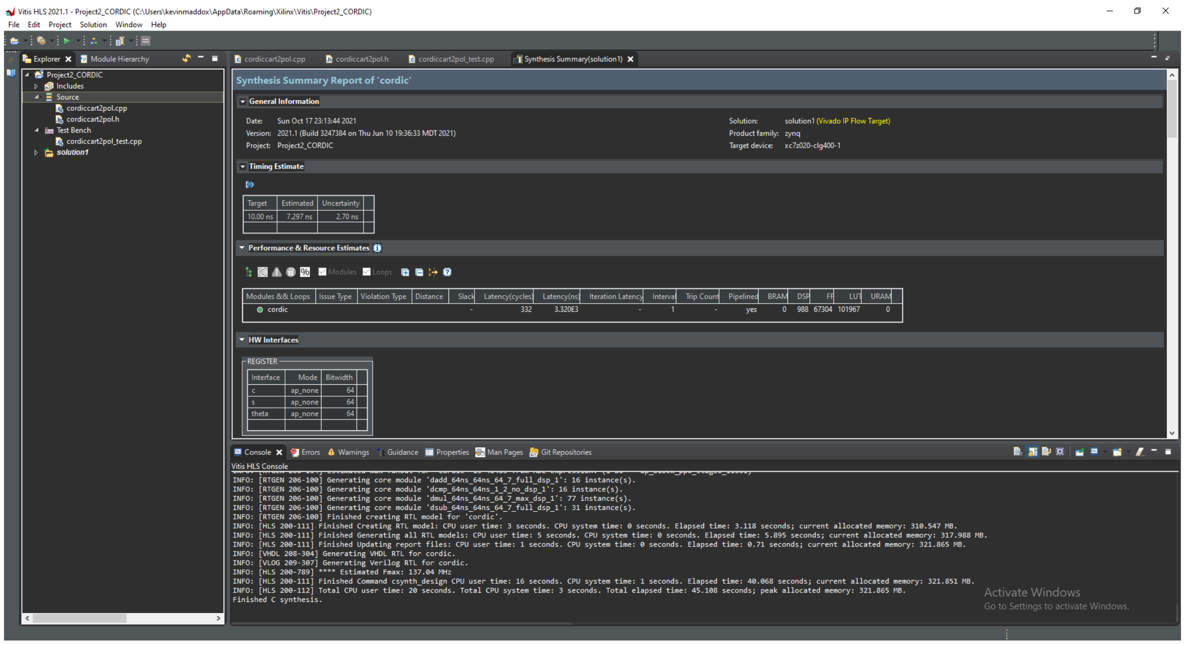
Interface	Mode	Bitwidth
c	ap_none	64
s	ap_none	64
theta	ap_none	64

Vitis HLS Console

```
INFO: [RTGEN 206-100] Generating core module 'dcmp_64ns_64ns_1_2_no_dsp_1': 16 instance(s).
INFO: [RTGEN 206-100] Generating core module 'dsub_64ns_64ns_64_7_full_dsp_1': 2 instance(s).
INFO: [RTGEN 206-100] Generating core module 'sitodp_32s_64_6_no_dsp_1': 29 instance(s).
```

Figure 5 Optimization w/unrolling

Finished Command csynth_design CPU user time: 16 seconds. CPU system time: 2 seconds. Elapsed time: 42.06 seconds; current allocated memory: 348.343 MB.
Total CPU user time: 19 seconds. Total CPU system time: 5 seconds. Total elapsed time: 47.196 seconds; peak allocated memory: 1.072 GB.



Vitis HLS 2021.1 - Project2_CORDIC (C:\Users\kevinmaddox\AppData\Roaming\Xilinx\Vitis\Project2_CORDIC)

Synthesis Summary Report of 'cordic'

General Information

Date: Sun Oct 17 23:13:44 2021
Version: 2021.1 (Build 3247384 on Thu Jun 10 19:36:33 MDT 2021)
Project: Project2_CORDIC
Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg400-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.297 ns	2.70 ns

Performance & Resource Estimates

Modules && Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
cordic				-	332	3.320E3	-	1	-	yes	0	988	67304	101967	0

HW Interfaces

REGISTER

Interface	Mode	Bitwidth
c	ap_none	64
s	ap_none	64
theta	ap_none	64

Vitis HLS Console

```
INFO: [RTGEN 206-100] Generating core module 'dadd_64ns_64ns_64_7_full_dsp_1': 16 instance(s).
INFO: [RTGEN 206-100] Generating core module 'dcmp_64ns_64ns_1_2_no_dsp_1': 16 instance(s).
INFO: [RTGEN 206-100] Generating core module 'dmul_64ns_64ns_64_7_max_dsp_1': 77 instance(s).
INFO: [RTGEN 206-100] Generating core module 'dsub_64ns_64ns_64_7_full_dsp_1': 31 instance(s).
INFO: [RTGEN 206-100] Finished creating RTL model for 'cordic'.
INFO: [HLS 200-111] Finished Creating RTL model: CPU user time: 3 seconds. CPU system time: 0 seconds. Elapsed time: 3.118 seconds; current allocated memory: 310.547 MB.
INFO: [HLS 200-111] Finished Generating all RTL models: CPU user time: 5 seconds. CPU system time: 0 seconds. Elapsed time: 5.895 seconds; current allocated memory: 317.988 MB.
INFO: [HLS 200-111] Finished Updating report files: CPU user time: 1 seconds. CPU system time: 0 seconds. Elapsed time: 0.71 seconds; current allocated memory: 321.865 MB.
INFO: [VHDL 208-304] Generating VHDL RTL for cordic.
INFO: [VLOG 209-307] Generating Verilog RTL for cordic.
INFO: [HLS 200-789] **** Estimated Fmax: 137.04 MHz
INFO: [HLS 200-111] Finished Command csynth_design CPU user time: 16 seconds. CPU system time: 1 seconds. Elapsed time: 40.068 seconds; current allocated memory: 321.851 MB.
INFO: [HLS 200-112] Total CPU user time: 20 seconds. Total CPU system time: 3 seconds. Total elapsed time: 45.108 seconds; peak allocated memory: 321.865 MB.
Finished C synthesis.
```

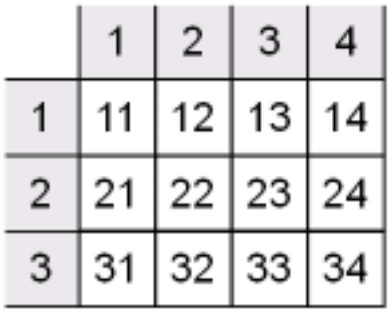
Figure 6 optimization wo if/else

Finished Command csynth_design CPU user time: 16 seconds. CPU system time: 1 seconds. Elapsed time: 40.068 seconds; current allocated memory: 321.851 MB.
Total CPU user time: 20 seconds. Total CPU system time: 3 seconds. Total elapsed time: 45.108 seconds; peak allocated memory: 321.865 MB

Question 4: *How does the input data type affect the size of the LUT? How does the output data type affect the size of the LUT? Precisely describe the relationship between input/output data types and the number of bits required for the LUT.*

The memory limitations of the system constrain the overall size of a lookup table. Lookup tables must use consistent dimensions so that the overall size of the table data reflects the size of each breakpoint data set.
The following input and output values define the 2-D lookup table that is graphically shown.

Row index input values: [1 2 3]
Column index input values: [1 2 3 4]
Table data: [11 12 13 14; 21 22 23 24; 31 32 33 34]



	1	2	3	4
1	11	12	13	14
2	21	22	23	24
3	31	32	33	34

The sizes of the vectors representing the row and column indices are 1-by-3 and 1-by-4, respectively. Consequently, the output table must be of size 3-by-4 for consistent dimensions.

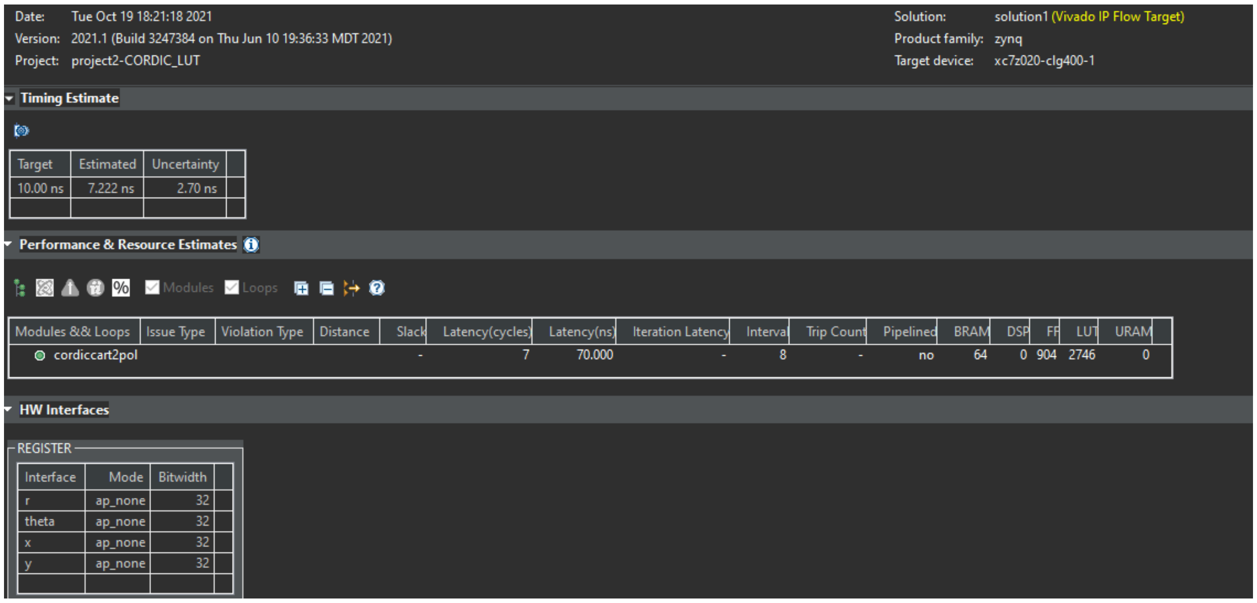
In real life, we deal with real number; numbers with fractional part. Most modern computer have native (hardware) support for floating point numbers. However, the use of floating point is not necessarily the only way to represent fractional numbers.

4.2 The testbench assumes that the inputs x, y are normalized between [-1,1]. What is the minimum number of integer bits required for x and y? What is the minimal number of integer bits for the output data type R and Theta?

4 bytes

4.3 Modify the number of fractional bits for the input and output data types. How does the precision of the input and output data types affect the accuracy (RMSE) results?

RMSE is a good measure of accuracy. Highly precise data does not necessarily correlate to highly accurate data nor does highly accurate data imply high precision data. They are two separate and distinct measurements. As data is more precise overall the output data will have better accuracy, the tradeoff is latency but using fractional has a lower latency over floating-point.



Date: Tue Oct 19 18:21:18 2021
Version: 2021.1 (Build 3247384 on Thu Jun 10 19:36:33 MDT 2021)
Project: project2-CORDIC_LUT

Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg400-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.222 ns	2.70 ns

Performance & Resource Estimates

Modules && Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
● cordiccart2pol				-	7	70.000	-	8	-	no	64	0	904	2746	0

HW Interfaces

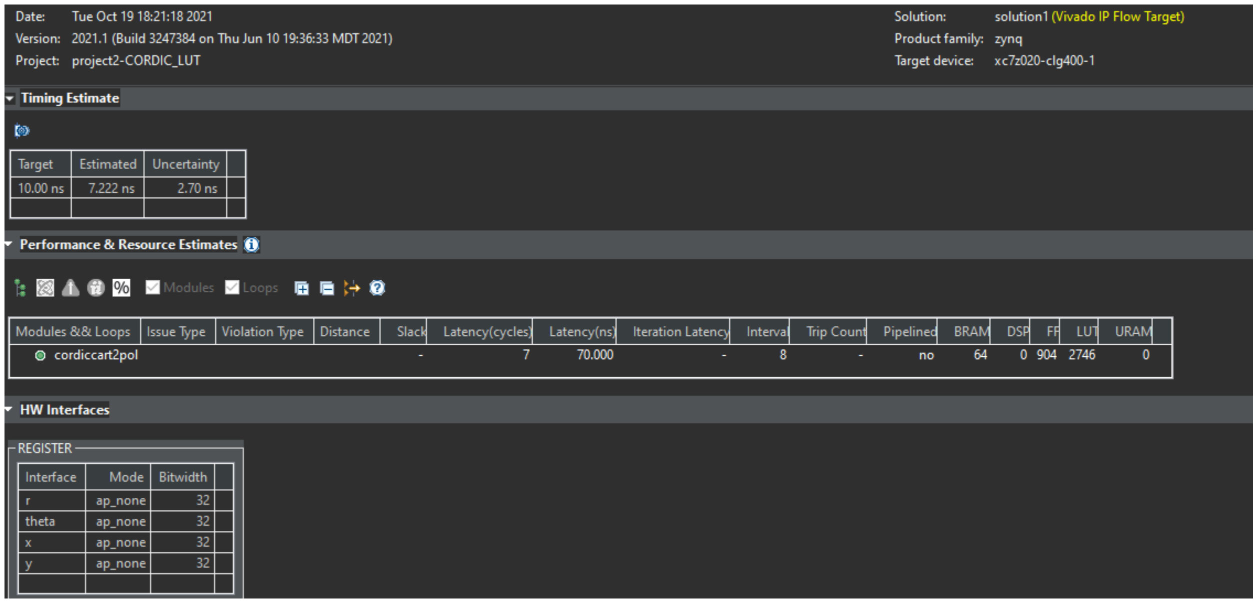
REGISTER

Interface	Mode	Bitwidth
r	ap_none	32
theta	ap_none	32
x	ap_none	32
y	ap_none	32

Figure 7 CORDIC_LUT_1

4.4 What is the performance (throughput, latency) of the LUT implementation? How does this change as the input and output data types change?

The primary benefit of using these different data types in software revolve around the amount of storage that the data type requires. The LUT is using ap_<>


Date: Tue Oct 19 18:21:18 2021
Version: 2021.1 (Build 3247384 on Thu Jun 10 19:36:33 MDT 2021)
Project: project2-CORDIC_LUT

Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg400-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.222 ns	2.70 ns

Performance & Resource Estimates

Modules && Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
cordiccart2pol				-	7	70.000	-	8	-	no	64	0	904	2746	0

HW Interfaces

REGISTER

Interface	Mode	Bitwidth
r	ap_none	32
theta	ap_none	32
x	ap_none	32
y	ap_none	32

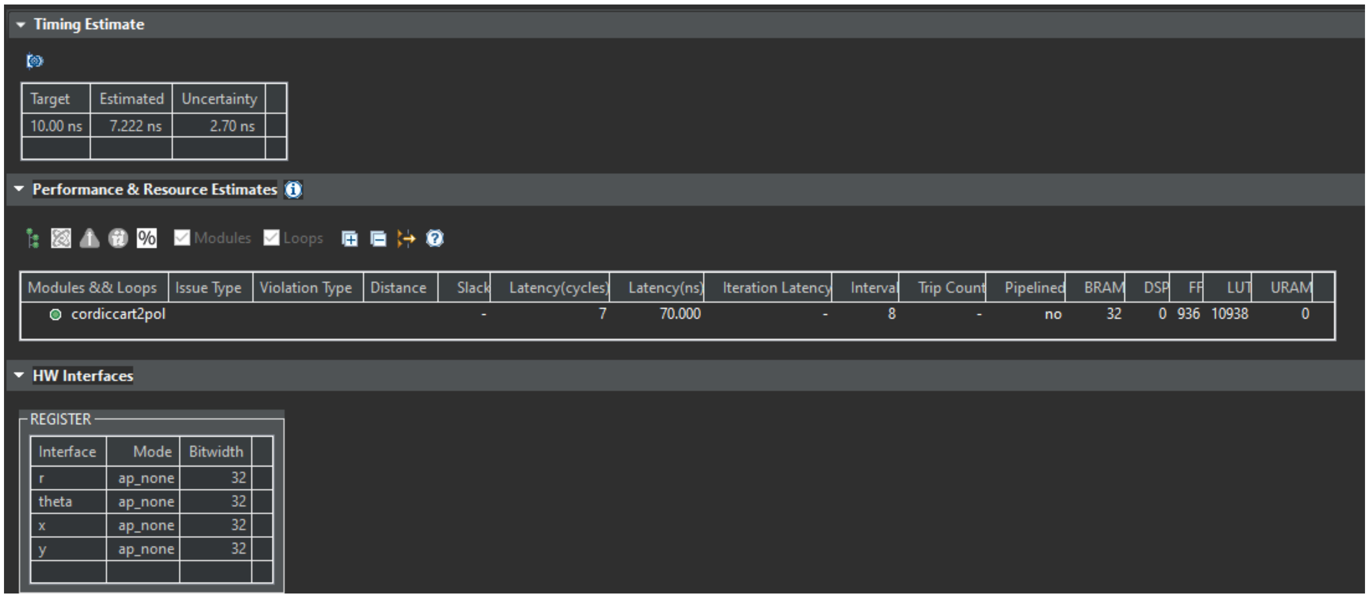
Figure 8 CORDIC_LUT_1

4.5 What advantages/disadvantages of the CORDIC implementation compared to the LUT-based implementation?

High-Speed- in a DSP microprocessor, table look-up methods are generally quicker than just CORDIC algorithm. Since FPGA technology is primarily based on referring to the look-up tables the time taken to execute is much less compared to ASIC technology.

LUT based increases that amount of memory used. LUT can increase the memory exponentially and HW complexity while the output word size increases.

```
// you have to answer to this question: What happens if you uncomment these pragma?
        #pragma HLS RESOURCE variable=my_LUT_th core=RAM_1P_LUTRAM
```


Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.222 ns	2.70 ns

Performance & Resource Estimates

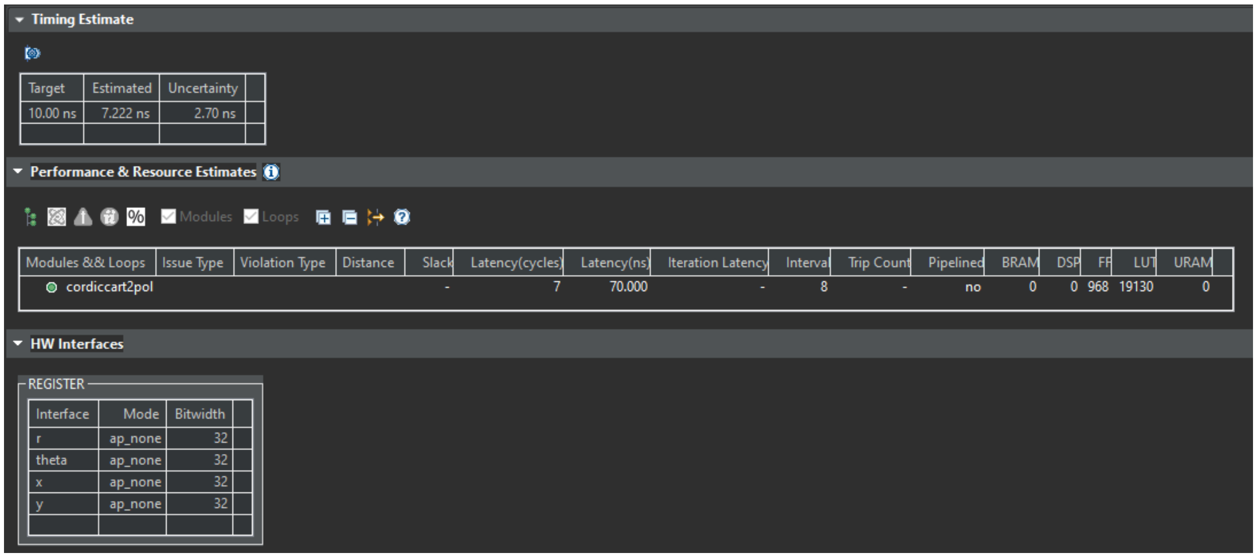
Modules && Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
● cordiccart2pol				-	7	70.000	-	8	-	no	32	0	936	10938	0

HW Interfaces

REGISTER

Interface	Mode	Bitwidth
r	ap_none	32
theta	ap_none	32
x	ap_none	32
y	ap_none	32

```
#pragma HLS RESOURCE variable=my_LUT_r core=RAM_1P_LUTRAM
```


Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.222 ns	2.70 ns

Performance & Resource Estimates

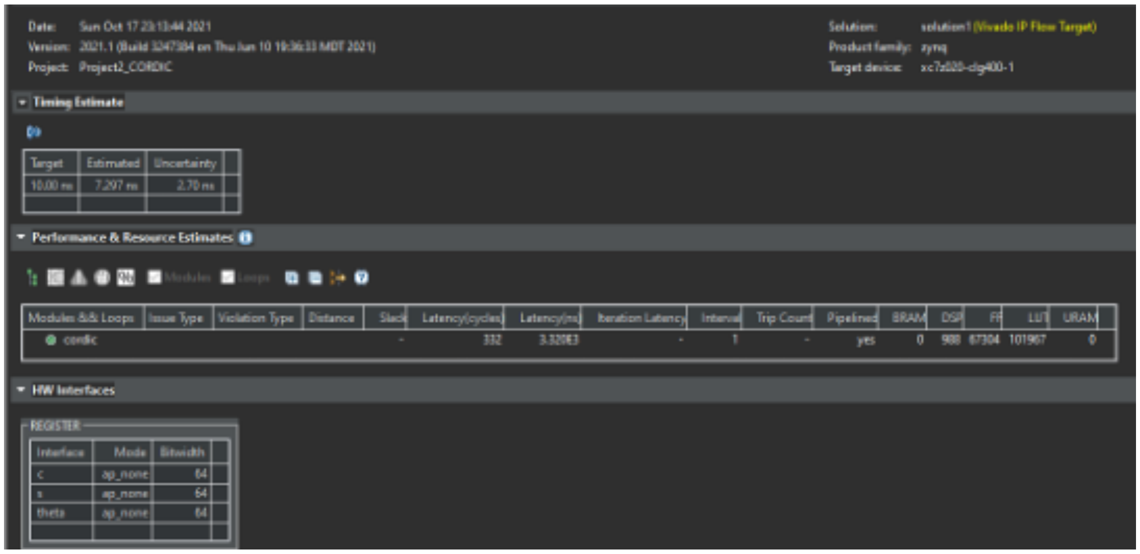
Modules && Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
● cordiccart2pol				-	7	70.000	-	8	-	no	0	0	968	19130	0

HW Interfaces

REGISTER

Interface	Mode	Bitwidth
r	ap_none	32
theta	ap_none	32
x	ap_none	32
y	ap_none	32

The RESOURCE pragma specifies that a specific library resource (core) is used to implement a variable (array, arithmetic operation, or function argument) in the register transfer level (RTL). If the RESOURCE pragma is not specified, the High-Level Synthesis (HLS) tool determines the resource to use. LUTRAM is a special LUT for implementing distrusted RAM.



Date: Sun Oct 17 23:13:44 2021
Version: 2021.1 (Build 3247384 on Thu Jun 10 19:36:33 MDT 2021)
Project: Project2_CORDIC

Solution: solution1 (Vivado IP Flow Target)
Product family: zynq
Target device: xc7z020-clg400-1

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.297 ns	2.70 ns

Performance & Resource Estimates

Modules && Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
● cordic				-	332	3.320E3	-	1	-	yes	0	988	67304	101967	0

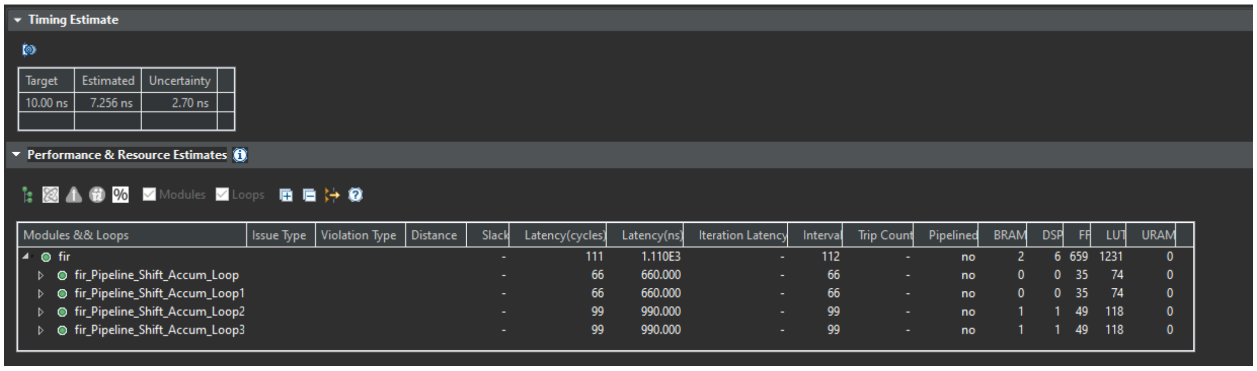
HW Interfaces

REGISTER

Interface	Mode	Bitwidth
c	ap_none	64
s	ap_none	64
theta	ap_none	64

Project: Phase Detector

Question 1: What is the throughput of your Phase Detector? How does that relate to the individual components (FIR, CORDIC, etc.)? How can you make it better?



Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.256 ns	2.70 ns

Performance & Resource Estimates

Modules && Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
● fir				-	111	1.110E3	-	112	-	no	2	6	659	1231	0
● fir_Pipeline_Shift_Accum_Loop				-	66	660.000	-	66	-	no	0	0	35	74	0
● fir_Pipeline_Shift_Accum_Loop1				-	66	660.000	-	66	-	no	0	0	35	74	0
● fir_Pipeline_Shift_Accum_Loop2				-	99	990.000	-	99	-	no	1	1	49	118	0
● fir_Pipeline_Shift_Accum_Loop3				-	99	990.000	-	99	-	no	1	1	49	118	0

Figure 9 wo CORDIC

Finished Command csynth_design CPU user time: 6 seconds. CPU system time: 2 seconds. Elapsed time: 23.389 seconds; current allocated memory: 288.616 MB.
Total CPU user time: 10 seconds. Total CPU system time: 4 seconds. Total elapsed time: 32.934 seconds; peak allocated memory: 288.630 MB.