Project3: Discrete Fourier Transform

Cheng-Yu Chen

Question 1:

What changes would this code require if you were to use a custom CORDIC similar to what you designed for Project 2?

We could use a custom CORDIC to replace cos() and sin() in this code. The function of the custom CORDIC is opposed to the one in project 2. In this project, we have r=1 and theta so we used them to calculate x and y which are cos(theta) and sin(theta).

Compared to a baseline code with HLS math functions for cos() and sin(), would changing the accuracy of your CORDIC core make the DFT hardware resource usage change? How would it affect the performance? Note that you do not need to implement the CORDIC in your code, we are just asking you to discuss potential tradeoffs that would be possible if you used a CORDIC that you designed instead of the one from Xilinx.

Yes, changing the accuracy of the CORDIC core would change resource usage and performance. When we increase the accuracy, this means that we need to increase the iteration to get more accurate cos() and sin(), which means there will be more adders and multiplexers and these will increase the latency of the circuit.

Question 2:

Rewrite the code to eliminate these math function calls (i.e. cos() and sin()) by utilizing a table lookup. How does this change the throughput and area?

Baseline: It uses cos() and sin() to implement DFT.

Optimized2: It uses lookup table to calculate cos() and sin().

	baseline	optimized2
Throughput(no pipeline)	21.823Hz	120.79Hz
BRAM	18	4
DSP48E	203	16
FF	12945	1474
LUT	18610	2495

Using a table lookup reduce lots of resources and by virtue of reduced computation complexity, the throughput increase.

What happens to the table lookup when you change the size of your DFT?

Size of DFT	32	256
BRAM	4	4
DSP48E	16	16
FF	1461	1474
LUT	2472	2495

The table lookup will increase if we increase the size of DFT since it will need to look up more coefficients to calculate sin() or cos().

Question 3:

Modify the DFT function interface so that the input and outputs are stored in separate arrays. How does this affect the optimizations that you can perform?

With separate arrays, the circuit won't need to wait for the result every time. So, we could use more optimizations such as pipeline, dataflow.

How does it change the performance? What about the area results? Modify your testbench to accommodate this change to DFT interface.**You should use this modified interface for the remaining questions.**

Optimized2: It uses lookup table to calculate cos() and sin().

Optimized3: It uses optimized2 and separates input and output port.

	optimized2	optimized3
Throughput(no pipeline)	120.79Hz	120.846Hz
BRAM	4	2
DSP48E	16	16
FF	1474	1445
LUT	2495	2399

The throughput slightly increases and every resource reduces or stays the same but if we implement more optimization, this optimization will be a huge impact.

Question 4:

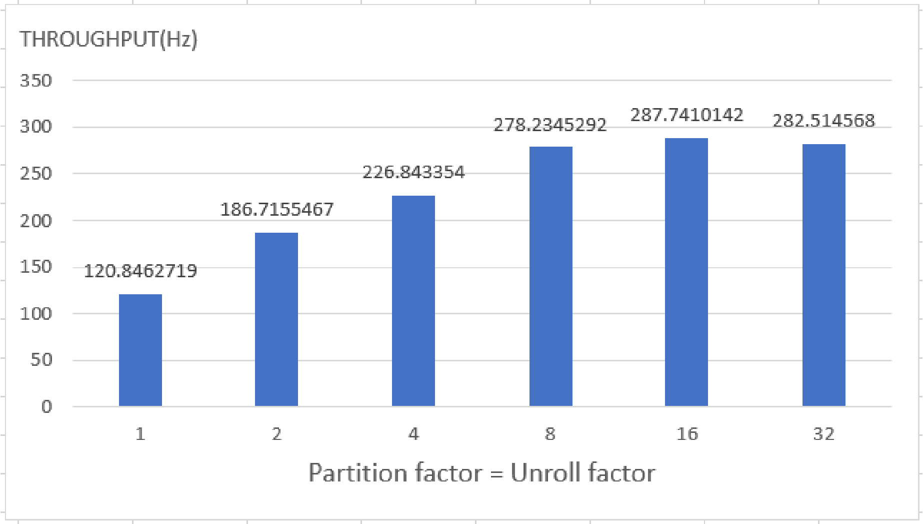
Study the effects of loop unrolling and array partitioning on the performance and area. What is the relationship between array partitioning and loop unrolling? Does it help to perform one without the other?

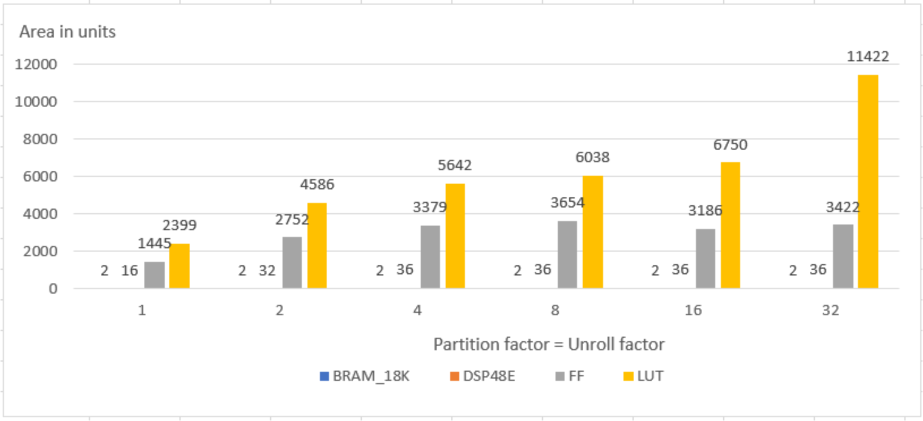
Array partitioning divide array to increase the data which the circuit could access every time and loop unrolling use more computation unit to increase the computation which the circuit could do every single time. If we just enhance array partitioning, it is useless since we don't have enough computation unit to process the data. So, the factor of array partitioning should be proportional to the factor of loop unrolling and normally, one should perform with the other to get better performance.

Plot the performance in terms of number of matrix vector multiply operations per second (throughput) versus the unroll and array partitioning factor. Plot the same trend for area (showing LUTs, FFs, DSP blocks, BRAMs).

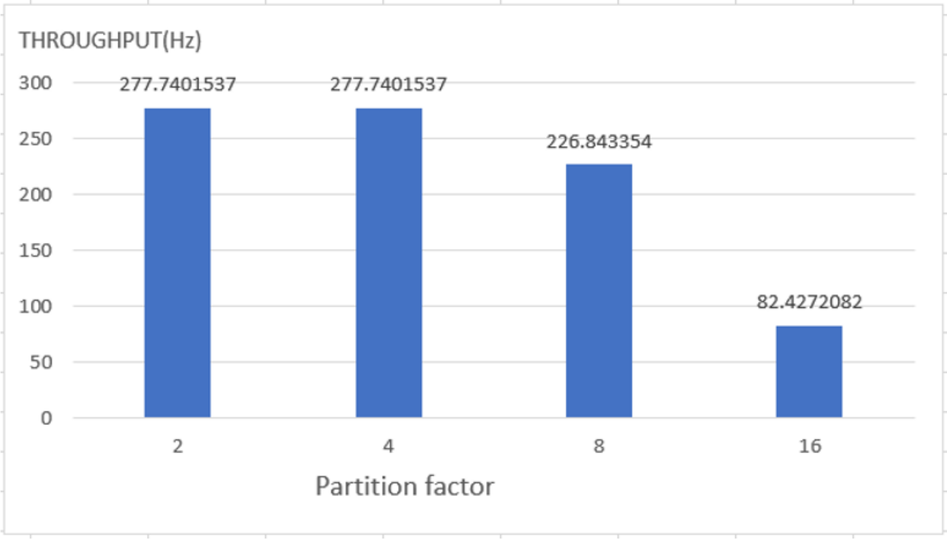
Optimized4: It uses loop unrolling(factor=16) and array partitioning(factor=2). Since the two factors are correlated, I first specify same value to both.

Unroll factor = Partition factor

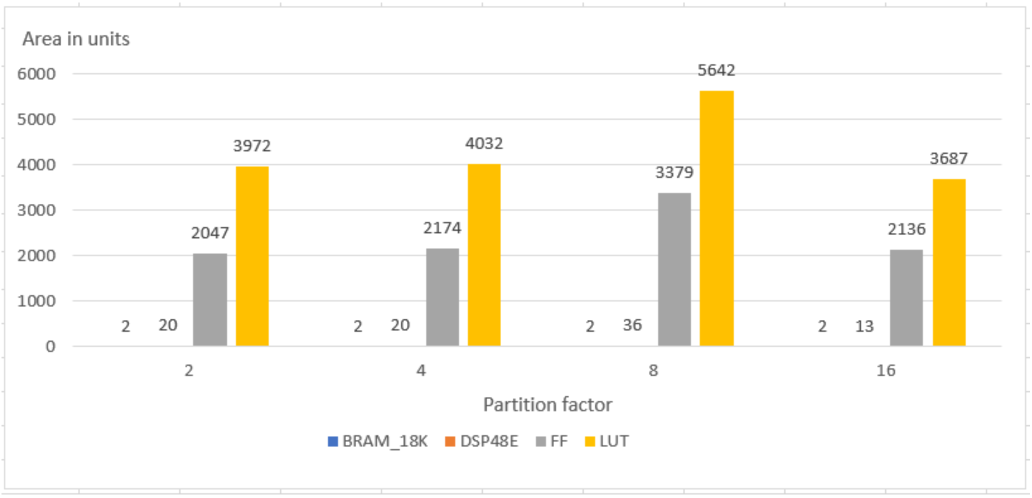




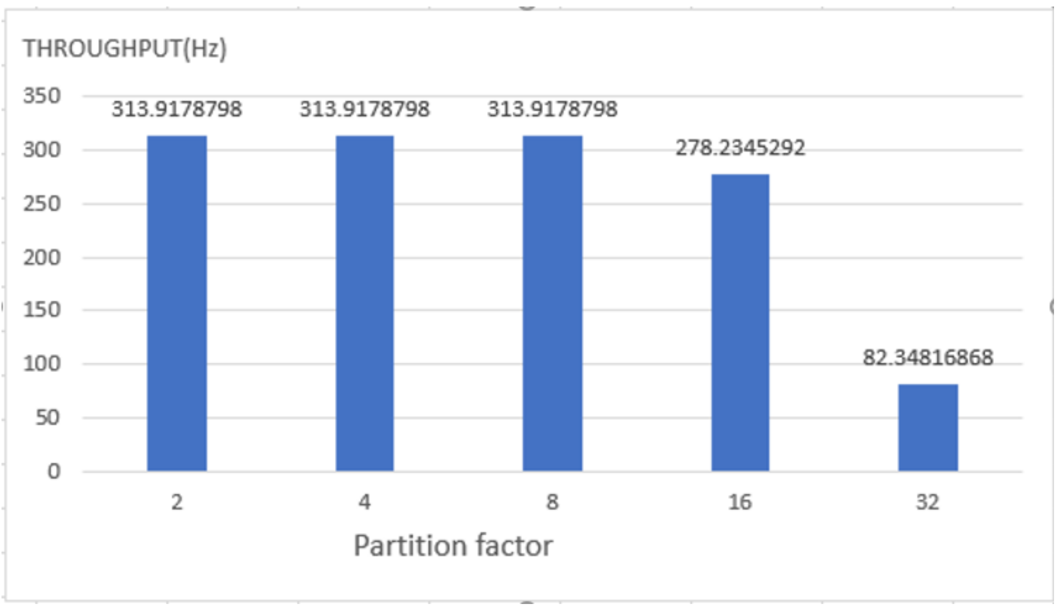
Unroll factor = 8:

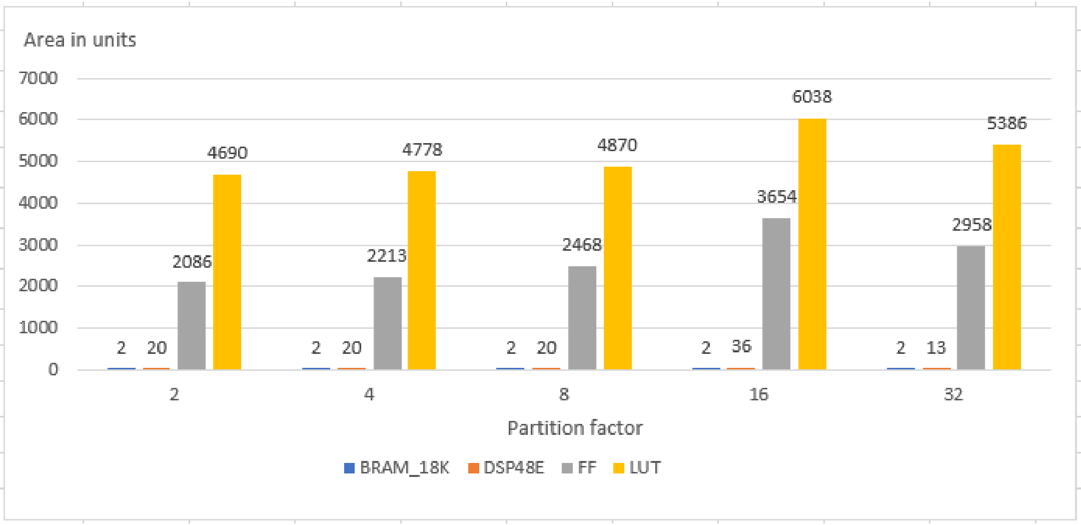


p.s. no pipeline

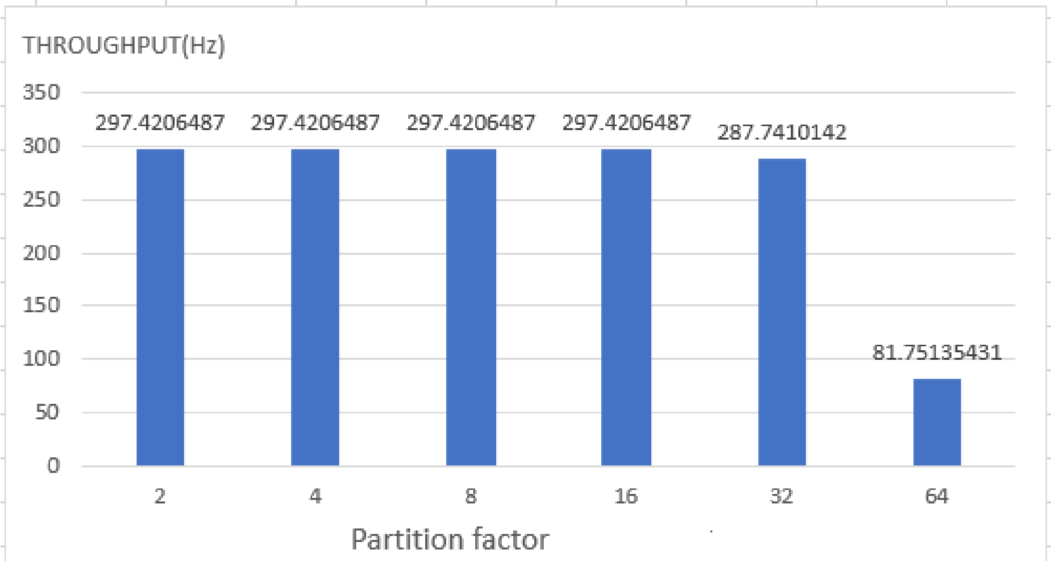


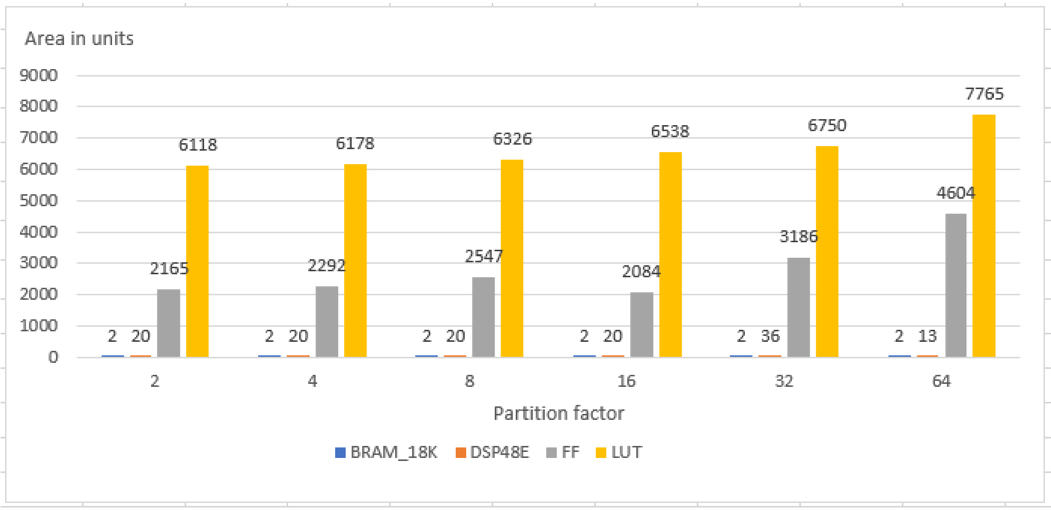
Unroll factor = 16:





Unroll factor = 32:





Although array partitioning and loop unrolling are correlated, it doesn’t work out in optimized4. I find out that the circuit need to accumulate the result at the end of inner-loop so every iteration is dependent with next iteration, which leads to limitation of loop unrolling and array partitioning. Since every iteration needs to wait for last iteration’s result, the design tool let the circuit do the instruction sequentially, which causes the result above.

<u>What is the general trend in both cases? Which design would you select? Why?</u>

Apparently, when partition factor is below unroll factor, the performance is better and resources usage is less. On the other hand, when partition factor is higher than unroll factor, the performance would decrease and resources usage would decrease, too.
I choose the circuit with unroll factor = 16 and partition factor = 2 to be my optimized4 since it has best performance among all and uses less resources compared with others with same unroll factor.

Question 5:

Please read dataflow section in the HLS user guide,and apply dataflow pragma to your design to improve throughput. You may need to change your code and make submodules. How much improvement can you make with it? How much does your design use resources? What about BRAM usage?

Optimized3: It uses lookup table and separates input and output port.

Optimized5: It uses optimized3 and dataflow optimization.

	optimized3	Optimized5
Throughput(no pipeline)	120.846Hz	186.716Hz
BRAM	2	2
DSP48E	16	20
FF	1445	2255
LUT	2399	4446

Improvement: Throughput = +0.545, BRAM = 0, DSP48E = +0.25, FF = +0.56, LUT = +0.85. BRAM usage is still 2.

Please describe your architecture with figures on your report. (Make sure to add dataflow pragma on your top function.)

Dataflow optimization uses more FF, LUT, and DSP48E to let the circuit operate more parallelly. In other words, it uses more resources to improve its throughput

Question 6:

(Best architecture) Briefly describe your "best" architecture. In what way is it the best?

At first, I use array partitioning, loop-unrolling, and pipeline. However, pipelining the inner-loop will need more LUT than the circuit have. Since I can't use pipeline and there is dependency between iterations in the inner-loop, it's unnecessary to use array partitioning. Also, data flow optimization is good when unroll factor is low but it's not helpful when unroll factor is high. Therefore, my best circuit uses loop-unrolling, look-up table, and individual input and output port. Therefore, it has really good performance(313.918Hz).

It's best because it has highest throughput with existing resources and it uses less resources than others with close throughput.

	optimized3
Throughput(no pipeline)	313.918Hz
BRAM	2
DSP48E	20
FF	2002
LUT	4546

What optimizations did you use to obtain this result? What is tradeoff you consider for the best architecture?

I use loop-unrolling, optmized2 (look-up table for cos() and sin()), and optimized3(separate input and output port). I view the throughput as the most important factor and then resource is second important factor. Therefore, I always first want to hasten the circuit such as using pipeline, dataflow, and loop unrolling and then when every methods above is used, I will try to reduce its area by comparing the circuit with others with similar throughput.