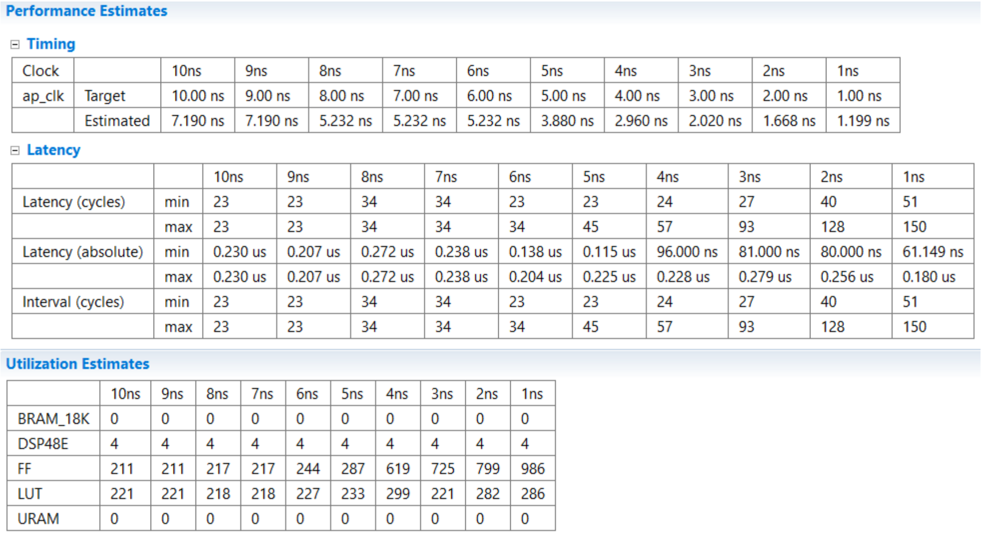
Mr. Phot Koseekrainiramon z5387411

23T2 COMP4601 Week 2 Lab Hand-in exercises

Exercise 1:

i.



Performance Estimates

⊟ Timing

Clock		10ns	9ns	8ns	7ns	6ns	5ns	4ns	3ns	2ns	1ns
ap_clk	Target	10.00 ns	9.00 ns	8.00 ns	7.00 ns	6.00 ns	5.00 ns	4.00 ns	3.00 ns	2.00 ns	1.00 ns
	Estimated	7.190 ns	7.190 ns	5.232 ns	5.232 ns	5.232 ns	3.880 ns	2.960 ns	2.020 ns	1.668 ns	1.199 ns

⊟ Latency

		10ns	9ns	8ns	7ns	6ns	5ns	4ns	3ns	2ns	1ns
Latency (cycles)	min	23	23	34	34	23	23	24	27	40	51
	max	23	23	34	34	34	45	57	93	128	150
Latency (absolute)	min	0.230 us	0.207 us	0.272 us	0.238 us	0.138 us	0.115 us	96.000 ns	81.000 ns	80.000 ns	61.149 ns
	max	0.230 us	0.207 us	0.272 us	0.238 us	0.204 us	0.225 us	0.228 us	0.279 us	0.256 us	0.180 us
Interval (cycles)	min	23	23	34	34	23	23	24	27	40	51
	max	23	23	34	34	34	45	57	93	128	150

Utilization Estimates

	10ns	9ns	8ns	7ns	6ns	5ns	4ns	3ns	2ns	1ns
BRAM_18K	0	0	0	0	0	0	0	0	0	0
DSP48E	4	4	4	4	4	4	4	4	4	4
FF	211	211	217	217	244	287	619	725	799	986
LUT	221	221	218	218	227	233	299	221	282	286
URAM	0	0	0	0	0	0	0	0	0	0

ii. More clock period provides smaller area usage, with fewer intervals, fewer cycles, and fewer gaps between min and max, but more absolute latency.

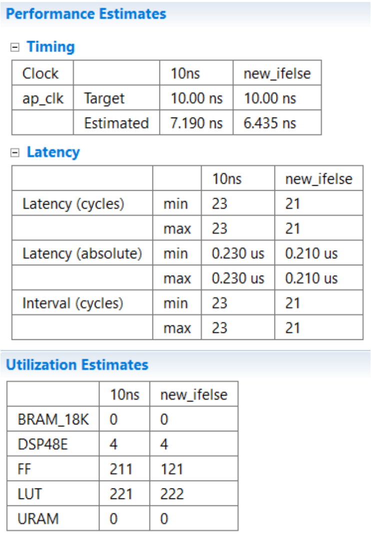
iii. In terms of latency, a clock period of 1ns performs the best with an average of 120ns absolute latency even though it takes the highest number of cycles. In terms of utilization, a clock period of 9ns performs the best with the least number of both flip-flops and look-up-table usages (with the same number of usages as a clock period of 10ns but having less absolute latency).

iv. The FPGA has 234240 flip-flops available and 117120 look-up tables available. By comparing 10ns and 1ns solutions, the usage of flip-flops is reduced by 0.003% while the usage of look-up tables is reduced by 0.0005%. This means the reduction of the usage of flip-flops is more significant than the usage of look-up tables even though the number of look-up tables available is less than the flip-flops.

v. The factor influencing the estimated clock period (worst-case delay) is the resources used since it needs to be optimized to meet the timing requirement.

vi. In the if-else statement, the first branch requires only 4 cycles but the other requires 13 cycles. That’s why the minimum iteration latency is 4 and the maximum is 13.

Exercise 2:



Performance Estimates

⊟ Timing

Clock		10ns	new_ifelse
ap_clk	Target	10.00 ns	10.00 ns
	Estimated	7.190 ns	6.435 ns

⊟ Latency

		10ns	new_ifelse
Latency (cycles)	min	23	21
	max	23	21
Latency (absolute)	min	0.230 us	0.210 us
	max	0.230 us	0.210 us
Interval (cycles)	min	23	21
	max	23	21

Utilization Estimates

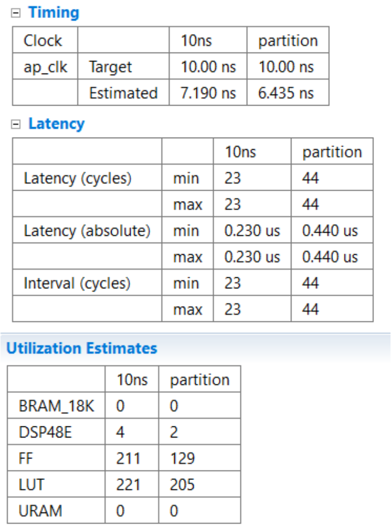
	10ns	new_ifelse
BRAM_18K	0	0
DSP48E	4	4
FF	211	121
LUT	221	222
URAM	0	0

i. The new design has less latency and interval cycles, and less estimated clock period, with 1 extra look-up table and 90 fewer flip-flops.

ii. The new solution has a smaller estimated clock period with the same number of uncertainty. In terms of latency, the new solution takes 2 fewer cycles from iterating 1 less iteration.

iii. Instead of iterating the loop 11 times, the new design only iterates 10 times and computes the last multiplication separately from the iteration.

Exercise 3:



⊟ Timing

Clock		10ns	partition
ap_clk	Target	10.00 ns	10.00 ns
	Estimated	7.190 ns	6.435 ns

⊟ Latency

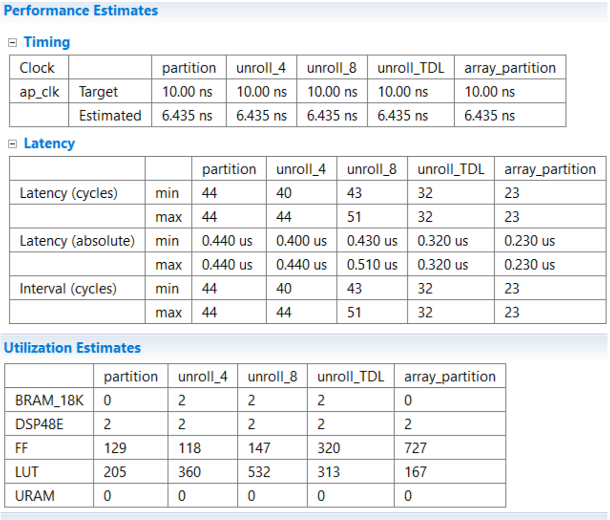
		10ns	partition
Latency (cycles)	min	23	44
	max	23	44
Latency (absolute)	min	0.230 us	0.440 us
	max	0.230 us	0.440 us
Interval (cycles)	min	23	44
	max	23	44

Utilization Estimates

	10ns	partition
BRAM_18K	0	0
DSP48E	4	2
FF	211	129
LUT	221	205
URAM	0	0

The new design has less estimated clock period. It has a higher number of interval and latency cycles since it is not pipelined and has to iterate 2 loops (TDL and MAC). It uses up fewer areas/resources (flip-flops, look-up tables, DSP) because it does fewer operations in one iteration.

Exercise 4:



Performance Estimates

⊟ Timing

Clock		partition	unroll_4	unroll_8	unroll_TDL	array_partition
ap_clk	Target	10.00 ns	10.00 ns	10.00 ns	10.00 ns	10.00 ns
	Estimated	6.435 ns	6.435 ns	6.435 ns	6.435 ns	6.435 ns

⊟ Latency

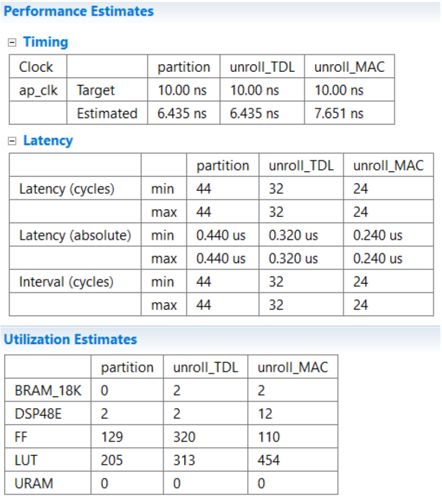
		partition	unroll_4	unroll_8	unroll_TDL	array_partition
Latency (cycles)	min	44	40	43	32	23
	max	44	44	51	32	23
Latency (absolute)	min	0.440 us	0.400 us	0.430 us	0.320 us	0.230 us
	max	0.440 us	0.440 us	0.510 us	0.320 us	0.230 us
Interval (cycles)	min	44	40	43	32	23
	max	44	44	51	32	23

Utilization Estimates

	partition	unroll_4	unroll_8	unroll_TDL	array_partition
BRAM_18K	0	2	2	2	0
DSP48E	2	2	2	2	2
FF	129	118	147	320	727
LUT	205	360	532	313	167
URAM	0	0	0	0	0

As the unroll factor is increased, the number of resources used is also increased for both flip-flips and look-up tables but not the others. Interval and latency cycles are also increased accordingly. In terms of latency, unrolling with a factor of 4 is slightly better compared to not unrolling, but with a factor of 8 is much worse with an increasing number of DSP, FF, and LUT. When unrolling completely, the number of resources used is changed (in this case, the number of flip-flops used is increased but the look-up table is decreased compared to using other factors) to gain the best performance in terms of latency (better performances but used up more resources compared to not unrolling). However, when trying to partition the arrays while unrolling completely, it achieves a better performance in terms of latency with no BRAM used, more flip-flops used but fewer look-up tables used.

Exercise 5:



Performance Estimates

⊟ Timing

Clock		partition	unroll_TDL	unroll_MAC
ap_clk	Target	10.00 ns	10.00 ns	10.00 ns
	Estimated	6.435 ns	6.435 ns	7.651 ns

⊟ Latency

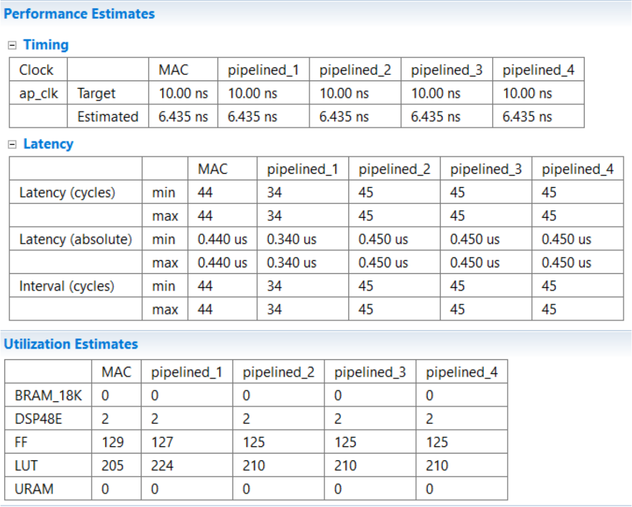
		partition	unroll_TDL	unroll_MAC
Latency (cycles)	min	44	32	24
	max	44	32	24
Latency (absolute)	min	0.440 us	0.320 us	0.240 us
	max	0.440 us	0.320 us	0.240 us
Interval (cycles)	min	44	32	24
	max	44	32	24

Utilization Estimates

	partition	unroll_TDL	unroll_MAC
BRAM_18K	0	2	2
DSP48E	2	2	12
FF	129	320	110
LUT	205	313	454
URAM	0	0	0

By unrolling the MAC loop, it has a higher estimated clock period but better performance in terms of latency (since TDL's load and write take many cycles). It uses up more BRAM, DSP, and URAM but less FF. This is because it needs to do more multiplication at a time compared to the other two designs.

Exercise 6:



Performance Estimates

⊟ Timing

Clock		MAC	pipelined_1	pipelined_2	pipelined_3	pipelined_4
ap_clk	Target	10.00 ns	10.00 ns	10.00 ns	10.00 ns	10.00 ns
	Estimated	6.435 ns	6.435 ns	6.435 ns	6.435 ns	6.435 ns

⊟ Latency

		MAC	pipelined_1	pipelined_2	pipelined_3	pipelined_4
Latency (cycles)	min	44	34	45	45	45
	max	44	34	45	45	45
Latency (absolute)	min	0.440 us	0.340 us	0.450 us	0.450 us	0.450 us
	max	0.440 us	0.340 us	0.450 us	0.450 us	0.450 us
Interval (cycles)	min	44	34	45	45	45
	max	44	34	45	45	45

Utilization Estimates

	MAC	pipelined_1	pipelined_2	pipelined_3	pipelined_4
BRAM_18K	0	0	0	0	0
DSP48E	2	2	2	2	2
FF	129	127	125	125	125
LUT	205	224	210	210	210
URAM	0	0	0	0	0

In terms of utilization, there are not many differences between them. In terms of latency, having pipelined with 1 II beats all the other solutions.

Increasing the II tends to increase latency cycles and decrease the number of resources used compared to less II, however, it stops changing after II equals to the number of the loop’s iteration latency (2 in this case).