Mr. Phot Koseekrainiramon z5387411

23T2 COMP4601 Chapter 5 Lab/Week 8 Hand-in exercises

FFT implementation exercises

Performance Estimates

Timing

Clock		solution1	solution2
ap_clk	Target	10.00 ns	10.00 ns
	Estimated	8.635 ns	8.635 ns

Latency

		solution1	solution2
Latency (cycles)	min	?	2600805
	max	?	2602549
Latency (absolute)	min	?	26.008 ms
	max	?	26.025 ms
Interval (cycles)	min	?	2600805
	max	?	2602549

Utilization Estimates

	solution1	solution2
BRAM_18K	16	16
DSP48E	204	204
FF	9119	9119
LUT	17576	17576
URAM	0	0

Exercise 1:

The latency/interval is not determined due to having a number of iterations depending on variables numBF and DFTpts. However, they are due to nested loops. The inner loop called dft_loop: has a latency of 21 clock cycles. Should be having up to 512 iterations. The middle loop called butterfly_loop: should be having up to 512 iterations depending on stages and it takes 5 clock cycles before entering dft_loop. And the outer loop: stage_loop: has 10 iterations (number of stages) and a latency of 28 clock cycles before entering butterfly_loop.

In terms of utilization, this design uses 16 BRAMs (5%), 204 DSPs (16%), 9119 FFs (3%), and 17576 LUTs (15%). We have 10 stages (for a 1024-point FFT, with 512 butterfly operations per stage).

Exercise 2:

The estimated latency should be around ((((21*512) + 5)*24) + 28)*10 = 2581960 total clock cycles from having 512 trip counts for dft_loop, 24 trip counts for butterfly_loop, and 10 trip counts for stage_loop. This is because butterfly_loop has 2^{n-1} iteration in n^{th} stage. With 10

stages we achieve a median of 24, and in every stage, the body of dft_loop is executed the same number of times in total which is 512.

What we achieved from the table above, is a minimum latency of 2600805 cycles and a maximum latency of 2602549 cycles, giving the difference between them only 0.067% difference. The resource utilization hasn't changed since adding the directives. They are performing using the same logic, just by having an estimated number of iterations.

Exercise 3:

	origin	Unrolling reverse_bits_loop	Pipelining reverse_bits_loop	Pipelining reverse_bits
Clock Target (ns)	10	10	10	10
Clock Estimated (ns)	2.704	2.704	2.704	2.704
Uncertainty (ns)	1.25	1.25	1.25	1.25
Latency min (cycles)	13313	2049	14337	2049
Latency max (cycles)	14337	3073	15361	3073
Latency min (ms)	0.133	0.020	0.143	0.020
Latency max (ms)	0.143	0.030	0.154	0.030
Interval min (cycles)	13313	2049	14337	2049
Interval max (cycles)	14337	3073	15361	3073
BRAMs	0	0	0	0
DSPs	0	0	0	0
FFs	105	47	106	47
LUTs	183	122	188	122
URAMs	0	0	0	0

Unrolling the reverse_bits_loop is the best in terms of both performance and utilization. This is due to the fact that the function itself doesn't need logic resources and could be done with only wires, and Vidavo HLS could notify that when trying to unroll the loop (normally it would require more resources when unrolling, however, this is not the case). The same goes with pipelining the function which would automatically unroll the loop.

By pipelining the reverse_bits_loop, even achieving the initiation interval of 1, this makes it even worse than the original design since the bit_reverse_loop needs to waste 1 clock cycle entering the pipeline without parallel loading input arrays X_R and X_I and even requires extra logic resources.

If we are focusing on adding directives to reverse_bits_loop then there're no better ways than unrolling as mentioned that Vivado HLS modifies it such there's no need for logic resources and could be done in no time. However, there might be a way to unroll or pipeline the for loop in the bit_reverse function if we could make sure that there're no dependencies.

Exercise 4:

Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- stage_loop	2587730	2588450	258773 ~ 258845	-	-	10	no
+ butterfly_loop	258744	258816	10781 ~ 10784	-	-	24	no
++ dft_loop	10752	10752	21	21	1	512	yes

For arrays X_I and X_R, they need to be written at the end of the iteration (21 clock cycles) before getting read at the start of the next iteration, this is clearly a false dependency (Read-After-Write). When indexes i and i_lower could be intersecting. Currently achieving an initiation interval of 21 due to this issue.

Exercise 5:

Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- stage_loop	381170	381890	38117 ~ 38189	-	-	10	no
+ butterfly_loop	38088	38160	1587 ~ 1590	-	-	24	no
++ dft_loop	1558	1558	26	3	1	512	yes

The minimum II I achieved is 3 with an iteration latency of 26 clock cycles by directing inter (meaning the dependency is between different iterations as mentioned in Exercise 4) for arrays X_I and X_R with FALSE specified (allowing Vivado HLS to operate in parallel when unrolled).

Unrolling the outer loop wouldn't do any good. Furthermore, it is impossible to pipeline stage_loop as long as we couldn't unroll the inner loops. In order to do that, we would need to modify many things with no guarantee of a better result. So, I would say it is not worthwhile trying to do so.

Exercise 6:

directives.tcl:

```
set_directive_loop_tripcount -min 512 -max 512 -avg 512 "fft/dft_loop"
set_directive_loop_tripcount -min 24 -max 24 -avg 24 "fft/butterfly_loop"
set_directive_unroll "reverse_bits/reverse_bits_loop"
set_directive_pipeline "fft/dft_loop"
set_directive_dependence -variable X_R -type inter -dependent false "fft/dft_loop"
set_directive_dependence -variable X_I -type inter -dependent false "fft/dft_loop"
```

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.635 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
378171	379915	3.782 ms	3.799 ms	378171	379915	none

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
grp_sin_or_cos_double_s_fu_262	sin_or_cos_double_s	22	25	0.220 us	0.250 us	22	25	none
grp_sin_or_cos_double_s_fu_281	sin_or_cos_double_s	22	25	0.220 us	0.250 us	22	25	none
grp_bit_reverse_fu_300	bit_reverse	2049	3073	20.490 us	30.730 us	2049	3073	none

Loop

	Latency (cycles)			Initiation Interval			
Loop Name	min	max	Iteration Latency	achieved	target	Trip Count	Pipelined
- stage_loop	376120	376840	37612 ~ 37684	-	-	10	no
+ butterfly_loop	37584	37656	1566 ~ 1569	-	-	24	no
++ dft_loop	1538	1538	6	3	1	512	yes

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	8	40	3316	-
FIFO	-	-	-	-	-
Instance	16	52	6711	15129	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	391	-
Register	-	-	906	-	-
Total	16	60	7657	18836	0
Available	288	1248	234240	117120	64
Utilization (%)	5	4	3	16	0

Performance Estimates

Timing

Clock		solution1	solution2	unroll_reverse_bits_loop	solution4	solution5	solution6
ap_clk	Target	10.00 ns	10.00 ns	10.00 ns	10.00 ns	10.00 ns	10.00 ns
	Estimated	8.635 ns	8.635 ns	8.635 ns	8.635 ns	8.635 ns	8.635 ns

Latency

		solution1	solution2	unroll_reverse_bits_loop	solution4	solution5	solution6
Latency (cycles)	min	?	2600805	2589541	2589781	383221	378171
	max	?	2602549	2591285	2591525	384965	379915
Latency (absolute)	min	?	26.008 ms	25.895 ms	25.898 ms	3.832 ms	3.782 ms
	max	?	26.025 ms	25.913 ms	25.915 ms	3.850 ms	3.799 ms
Interval (cycles)	min	?	2600805	2589541	2589781	383221	378171
	max	?	2602549	2591285	2591525	384965	379915

Utilization Estimates

	solution1	solution2	unroll_reverse_bits_loop	solution4	solution5	solution6
BRAM_18K	16	16	16	16	16	16
DSP48E	204	204	204	204	198	60
FF	9119	9119	9061	9116	9396	7657
LUT	17576	17576	17515	17579	17511	18836
URAM	0	0	0	0	0	0

Solution3 is better in terms of both performance and utilization compared to solution1 as explained in Exercise 3. Even though solution4 requires fewer resources, it is no faster than solution3 in not achieving a fewer II due to false dependency but overall, still better than the original design. Solution5 is the best among all designs using floating type after fixing the dependency, with slightly more FFs than solution3 but fewer DSP from achieving II of 3.

It is clear that solution6 performs the best with fewer II (3) and iteration latency of inner loops (only 6 for dft_loop) compared to solution5 (using float). This greatly reduces the usage of DSPs by more than twice compared to other designs and fewer FFs, however, with an increased number of LUTs as a trade-off.

Implementing the dataflow code in hardware

Exercise 1:

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.635 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
?	?	?	?	?	?	dataflow

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	122	-
FIFO	-	-	-	-	-
Instance	160	600	46526	140458	0
Memory	40	-	0	0	0
Multiplexer	-	-	-	180	-
Register	-	-	20	-	-
Total	200	600	46546	140760	0
Available	288	1248	234240	117120	64
Utilization (%)	69	48	19	120	0

The behavior of iterations is similar to the first exercise of FFT but without stage_loop. The inner loop called dft_loop: has a latency of 2 clock cycles. The outer loop called butterfly_loop: takes 4 clock cycles before entering dft_loop. Those are called as a function, inside fft_streaming after bit_reverse is called, by a number of stages instead of looping with an outer loop with the number of stages as the number of iterations. Now that the number of stages isn't fixed, each time this function is called would have a different number of iterations according to the input stage (when being called in data flow).

In terms of utilization, this design uses 200 BRAMs (69%), 600 DSPs (48%), 46546 FFs (19%), and 140760 LUTs (120%). Which is exceeding the number of LUTs available (impossible to be implemented).

Exercise 2:

derective.tcl:

```
set_directive_loop_tripcount -min 512 -max 512 -avg 512 "fft_stage/dft_loop"
set_directive_pipeline "fft_stage/dft_loop"
set_directive_dependence -variable X_I -type inter -dependent false
"fft_stage/dft_loop"
set_directive_dependence -variable X_R -type inter -dependent false
"fft_stage/dft_loop"
set_directive_loop_tripcount -min 24 -max 24 -avg 24 "fft_stage/butterfly_loop"
set_directive_unroll "reverse_bits/reverse_bits_loop"
```

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.635 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
559604	568812	5.596 ms	5.688 ms	279042	283138	dataflow

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
fft_stage90_U0	fft_stage90	279041	283137	2.790 ms	2.831 ms	279041	283137	none
fft_stage89_U0	fft_stage89	139521	141569	1.395 ms	1.416 ms	139521	141569	none
fft_stage88_U0	fft_stage88	69761	70785	0.698 ms	0.708 ms	69761	70785	none
fft_stage87_U0	fft_stage87	34881	35393	0.349 ms	0.354 ms	34881	35393	none
fft_stage86_U0	fft_stage86	17441	17697	0.174 ms	0.177 ms	17441	17697	none
fft_stage85_U0	fft_stage85	8721	8849	87.210 us	88.490 us	8721	8849	none
fft_stage84_U0	fft_stage84	4361	4425	43.610 us	44.250 us	4361	4425	none
fft_stage83_U0	fft_stage83	2181	2213	21.810 us	22.130 us	2181	2213	none
fft_stage82_U0	fft_stage82	1091	1107	10.910 us	11.070 us	1091	1107	none
fft_stage81_U0	fft_stage81	546	554	5.460 us	5.540 us	546	554	none
bit_reverse_U0	bit_reverse	2049	3073	20.490 us	30.730 us	2049	3073	none

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	122	-
FIFO	-	-	-	-	-
Instance	160	600	45972	140517	0
Memory	40	-	0	0	0
Multiplexer	-	-	-	180	-
Register	-	-	20	-	-
Total	200	600	45992	140819	0
Available	288	1248	234240	117120	64
Utilization (%)	69	48	19	120	0

	fft_sw/solution6	fft_stages/solution2
Clock Target (ns)	10	10
Clock Estimated (ns)	8.635	8.635
Uncertainty (ns)	1.25	1.25
Latency min (cycles)	378171	559604
Latency max (cycles)	379915	568812
Latency min (ms)	3.782	5.596
Latency max (ms)	3.799	5.688
Interval min (cycles)	378171	279042
Interval max (cycles)	379915	283138
BRAMs	16	200
DSPs	60	600
FFs	7656	45992
LUTs	18836	140819
URAMs	0	0

It is clear that the new design is no better than the old one in any terms, moreover, it is so demanding in resources that the number of LUTs used is still exceeding those available. With a greatly increased total latency and around 10 times more resources compared to the old design (due to the fact that each function stage needs to be implemented independently). But now dft_loop of the new design achieved an initiation interval of 1 which is fewer (than 3) but the gap between minimum and maximum latency is also increased.

Implementing the streaming code in hardware

Exercise 1:

	fft_stages/solution2	fft_stages_loop/solution1
Clock Target (ns)	10	10
Clock Estimated (ns)	8.635	7.492
Uncertainty (ns)	1.25	1.25
Latency min (cycles)	559604	7208
Latency max (cycles)	568812	9232
Latency min (ms)	5.596	0.072
Latency max (ms)	5.688	0.082
Interval min (cycles)	279042	2050
Interval max (cycles)	283138	3074
BRAMs	200	58
DSPs	600	36
FFs	45992	1179
LUTs	140819	3376
URAMs	0	0

The latency for the new design becomes 7208-9232 clock cycles with 2050-3075 clock cycle intervals. The new design is better in both terms of performance and utilization with even fewer estimated clock periods. And the utilization is not exceeding that available on the board anymore.

The new design has butterfly_loop pipelined with an initiation interval of only 1.

Exercise 2:

Performance Estimates

Timing

Clock		solution1	pipeline_wo_dependence	solution2
ap_clk	Target	10.00 ns	10.00 ns	10.00 ns
	Estimated	7.492 ns	7.492 ns	7.492 ns

Latency

		solution1	pipeline_wo_dependence	solution2
Latency (cycles)	min	7208	7209	6186
	max	8232	7209	6186
Latency (absolute)	min	72.080 us	72.090 us	61.860 us
	max	82.320 us	72.090 us	61.860 us
Interval (cycles)	min	2050	2051	1028
	max	3074	2051	1028

Utilization Estimates

	solution1	pipeline_wo_dependence	solution2
BRAM_18K	58	58	58
DSP48E	36	36	36
FF	1179	1237	1193
LUT	3376	3396	3347
URAM	0	0	0

If only pipelining without determining a false dependency, the initiation interval of the bit_reverse_loop wouldn't be able to achieve 1 (only achieving 2) due to that issue (similar to Exercise 4). This causes the task latency and interval to be no better than the old design (solution1) since the new interval is equal to the minimum old interval.

After directing dependence to arrays OUT_I and OUT_R with false inters, it was able to achieve an initiation interval of 1 and improve overall performance with a slightly different number of FFs and LUTs usages.

	fft_sw/solution2	fft_stages_loop/solution2
Clock Target (ns)	10	10
Clock Estimated (ns)	8.635	7.492
Uncertainty (ns)	1.25	1.25
Latency min (cycles)	2600805	6186
Latency max (cycles)	2602549	6186
Latency min (ms)	26.008	0.061
Latency max (ms)	26.025	0.061
Interval min (cycles)	2600805	1028
Interval max (cycles)	2602549	1028
BRAMs	16	58
DSPs	204	36
FFs	9119	1193
LUTs	17576	3347
URAMs	0	0

It is clear that dataflow outperforms in terms of both performance and utilization, with no differences between min-max latency and fewer estimated clock periods. Even though it requires fewer DSPs, FFs, and LUTs, it requires a significantly increased number of BRAMs usages that could be concerned with the limited number of resources available on board (not exceeding but high demands).

With enough resources from a processor, this design should be able to be implemented. Note, it is required to know what could be done in isolation and what couldn't.

However, we are not taking accuracy into account even though we are using 2 different data types here, this would be one thing that we need to check for further improvement.

directives.tcl:

```
set_directive_pipeline "bit_reverse/bit_reverse_loop"
set_directive_dependence -variable OUT_I -type inter -dependent false
"bit_reverse/bit_reverse_loop"
set_directive_dependence -variable OUT_R -type inter -dependent false
"bit_reverse/bit_reverse_loop"
```

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	7.492 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
6186	6186	61.860 us	61.860 us	1028	1028	dataflow

Detail

Instance

		Latency (cycles)		Latency (absolute)		Interval (cycles)		
Instance	Module	min	max	min	max	min	max	Type
fft_stage_130_U0	fft_stage_130	515	515	5.150 us	5.150 us	515	515	none
fft_stage_131_U0	fft_stage_131	515	515	5.150 us	5.150 us	515	515	none
fft_stage_127_U0	fft_stage_127	515	515	5.150 us	5.150 us	515	515	none
fft_stage_129_U0	fft_stage_129	515	515	5.150 us	5.150 us	515	515	none
fft_stage_128_U0	fft_stage_128	515	515	5.150 us	5.150 us	515	515	none
fft_stage_132_U0	fft_stage_132	515	515	5.150 us	5.150 us	515	515	none
fft_stage_133_U0	fft_stage_133	515	515	5.150 us	5.150 us	515	515	none
fft_stage_134_U0	fft_stage_134	515	515	5.150 us	5.150 us	515	515	none
fft_stage35_U0	fft_stage35	515	515	5.150 us	5.150 us	515	515	none
fft_stage_126_U0	fft_stage_126	514	514	5.140 us	5.140 us	514	514	none
bit_reverse25_U0	bit_reverse25	1027	1027	10.270 us	10.270 us	1027	1027	none

Loop

N/A

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	122	-
FIFO	-	-	-	-	-
Instance	18	36	1173	3045	-
Memory	40	-	0	0	0
Multiplexer	-	-	-	180	-
Register	-	-	20	-	-
Total	58	36	1193	3347	0
Available	288	1248	234240	117120	64
Utilization (%)	20	2	~0	2	0