Course: CS 577 C Based VLSI Design

Project Phase 2 Report

Team Members:

1) Nihal M. Singh (Leader) (224101038)
2) Vaishali Chaudhari (224101055)
3) Shikha (224101066)
4) Yashvi Bipinbhai Rajvir (224101061)
5) Anuj Singh Thakur (224101008)

Solution 1: Low Area Overhead (Flip Flops, LUT, BRAM etc.)

- Initial Resource Utilization Summary:

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	253	-
FIFO	-	-	-	-	-
Instance	70	88	42281	183797	0
Memory	-	-	-	-	-
Multiplexer	-	-	-	114	-
Register	-	-	174	-	-
Total	70	88	42455	184164	0
Available	730	740	269200	134600	0
Utilization (%)	9	11	15	136	0

Here are the changes we made to optimize the code, listed as bullet points:

- Function Inlining: Replaced function calls with the actual code of the function at the call site to reduce function call overhead and improve execution speed.

- Loop unrolling: Duplicated loop bodies to reduce the number of loop iterations and eliminate loop overhead, resulting in faster execution of loop-based operations.

- Function allocation instance set to 1: Ensured that only one instance of the function is created and reused throughout the code, reducing memory overhead and improving the efficiency of function invocation.

These optimizations collectively improved the performance of our code by reducing unnecessary overhead and improving the execution speed, resulting in more efficient and optimized code.

Optimized Resource Utilization Summary:

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	3	-	-	-
Expression	-	10	0	3997	-
FIFO	-	-	-	-	-
Instance	34	75	21614	99072	0
Memory	29	-	0	0	0
Multiplexer	-	-	-	3331	-
Register	-	-	2294	-	-
Total	63	88	23908	106400	0
Available	730	740	269200	134600	0
Utilization (%)	8	11	8	79	0

Resource Utilization Comparison			
	Initial	Optimized	Percentage Reduce
LUT	184164	106400	42.22 %
FF	42455	23908	43.68%
BRAM	70	63	1.00 %
DSP	88	88	0.00 %

Solution 2: **Low Latency**

- Pipeline Implementation: Implemented a Pipeline architecture in the code to reduce latency by allowing multiple stages of processing to occur concurrently. This involved breaking down the code into smaller, independent stages and processing them in parallel, enabling efficient utilization of system resources and reducing overall latency. By leveraging the benefits of pipelining in our code, we were able to optimize the processing time and minimize the waiting time for dependent operations, resulting in reduced latency and improved performance.

Initial Co-simulation Report for 'pqcrystals_dilithium2_ref' (crypto_sign):

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	449330	573221	697113	697114	697114	697114

After Optimization:

Co-simulation Report for 'pqcrystals_dilithium2_ref' (crypto_sign):

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	376465	471544	566624	566625	566625	566625

Latency Comparison in Co-Simulation Report			
	Initial	Optimized	Percentage Reduce
Min	449330	376465	16.21%
Avg.	573221	471544	17.73%
Max	697113	566624	18.71 %