

CS577 : Project Report

Project Number :	Project 16
Group Number :	23
Name of the Top Module :	crypto_sign_keypair
Link for the Git Repo :	https://github.com/ExplosionArt/picnic_keypair_16

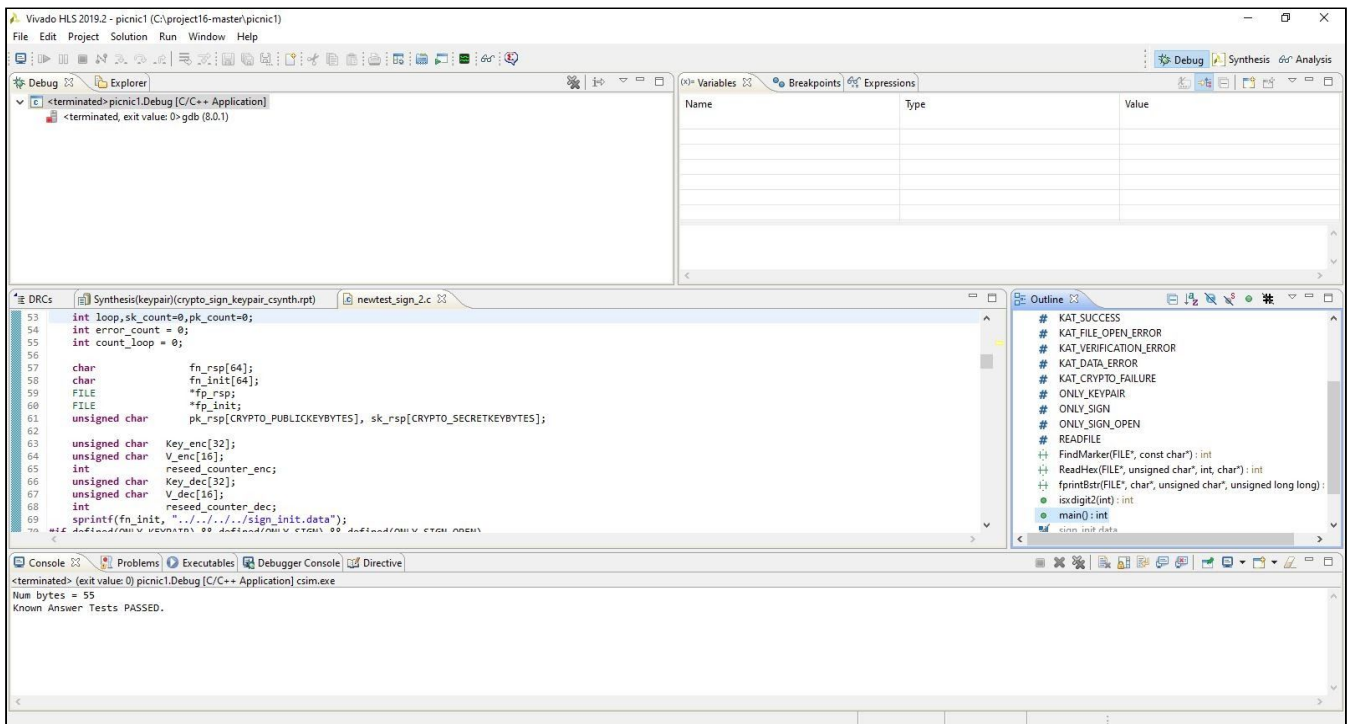
Group Members	Roll Numbers
Ajitem Joshi	194101021
Dhananjay Shukla	194101018
Abhay Chandra	194101002

INTRODUCTION

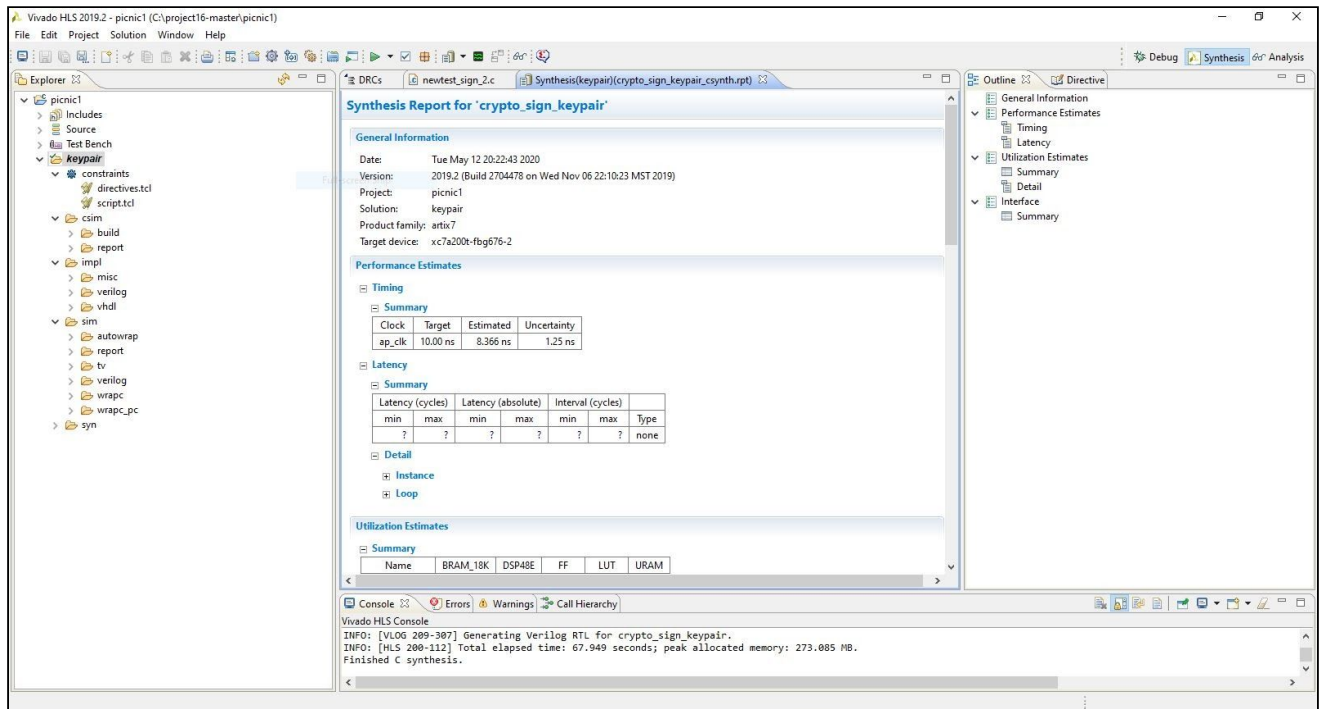
PHASE-1

1. Running the algorithm

1.1 Simulation screenshot



1.2 Synthesis screenshot



The screenshot shows the Vivado HLS 2019.2 interface with the Synthesis Report for 'crypto_sign_keypair' open. The report is divided into several sections: General Information, Performance Estimates, and Utilization Estimates.

General Information

- Date: Tue May 12 20:22:43 2020
- Version: 2019.2 (Build 2704478 on Wed Nov 06 22:10:23 MST 2019)
- Project: picnic1
- Solution: keypair
- Product family: artix7
- Target device: xc7a200t-fbg676-2

Performance Estimates

Timing

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.366 ns	1.25 ns

Latency

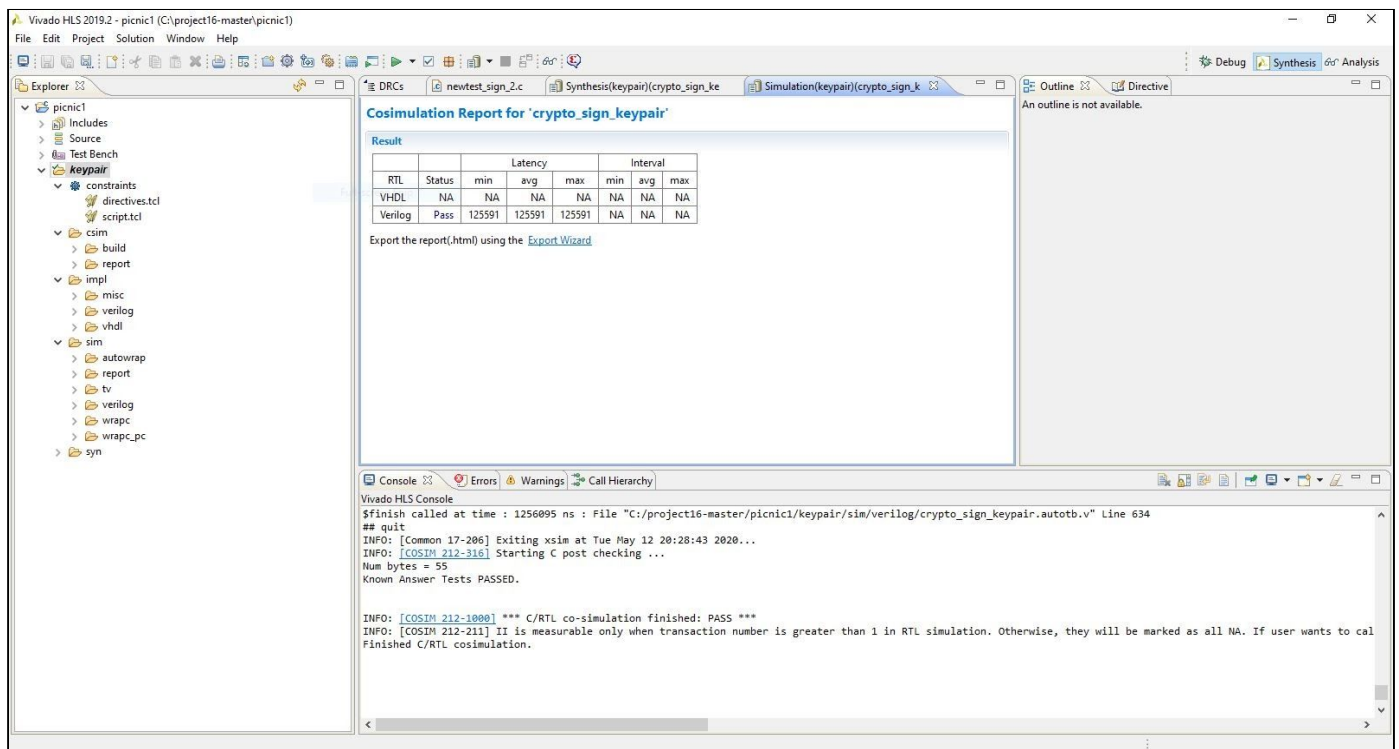
Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
?	?	?	?	?	?	none

Utilization Estimates

Name	BRAM_18K	DSP48E	FF	LUT	URAM

The console output shows the synthesis process: "INFO: [VLOG 209-307] Generating Verilog RTL for crypto_sign_keypair. INFO: [HLS 200-112] Total elapsed time: 67.949 seconds; peak allocated memory: 273.085 MB. Finished C synthesis."

1.3 C/RTL Co-Simulation Screenshot



The screenshot shows the Vivado HLS 2019.2 interface with the Cosimulation Report for 'crypto_sign_keypair' open. The report displays the results of the C/RTL co-simulation.

Cosimulation Report for 'crypto_sign_keypair'

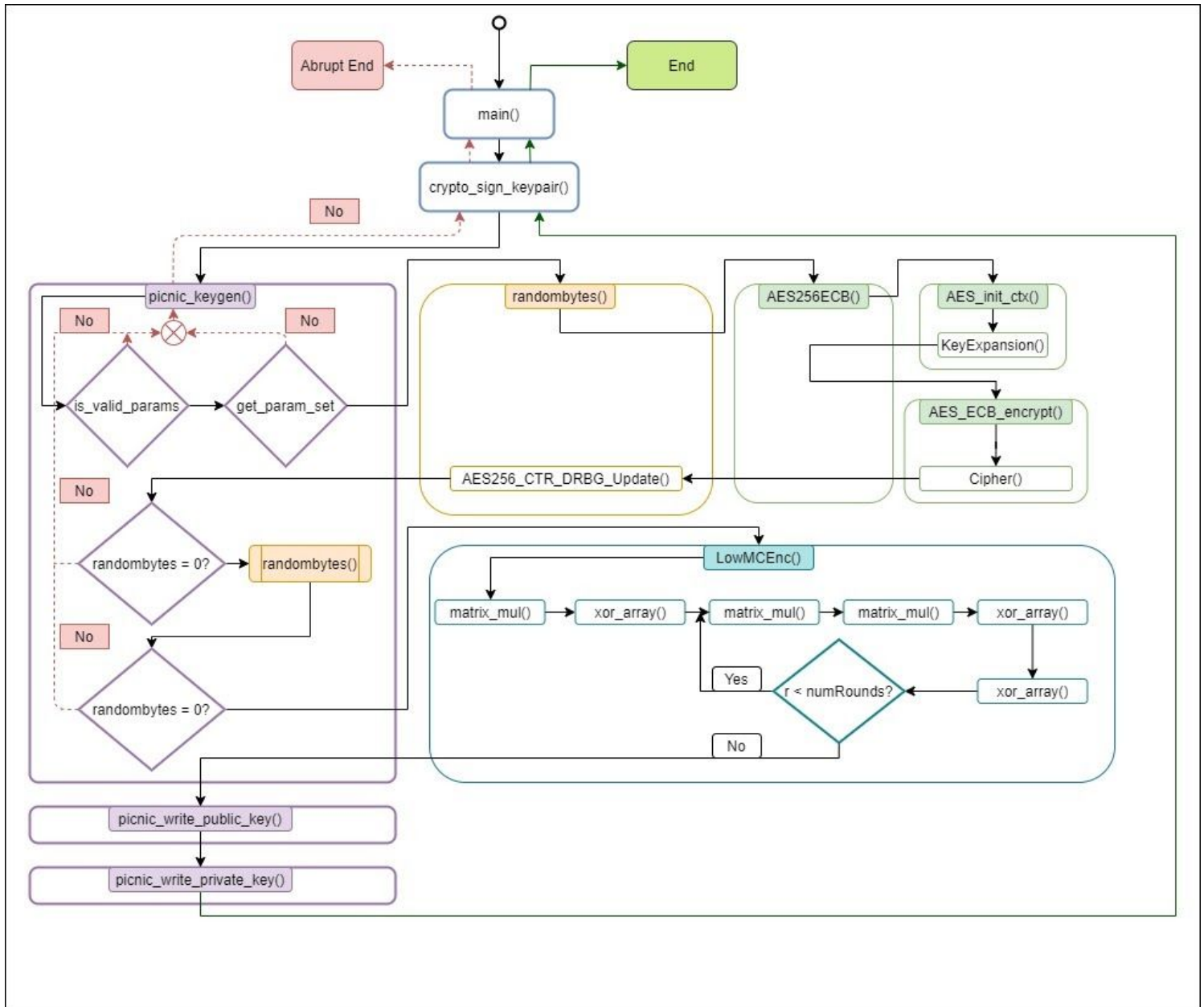
Result

RTL	Status	Latency			Interval		
		min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	125591	125591	125591	NA	NA	NA

Export the report(.html) using the [Export Wizard](#)

The console output shows the co-simulation process: "Vivado HLS Console \$finish called at time : 1256095 ns : File 'C:/project16-master/picnic1/keypair/sim/verilog/crypto_sign_keypair.autotb.v' Line 634 ## quit INFO: [Common 17-206] Exiting xsim at Tue May 12 20:28:43 2020... INFO: [COSIM 212-316] Starting C post checking ... Num bytes = 55 Known Answer Tests PASSED. INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS *** INFO: [COSIM 212-211] II is measurable only when transaction number is greater than 1 in RTL simulation. Otherwise, they will be marked as all NA. If user wants to col Finished C/RTL cosimulation."

2. FLOWCHART



Description:

- The top function **crypto_sign_keypair()** is called with two parameters pk and sk which are public key and secret key(or private key) respectively which are initially empty.
- First Task is public key and secret key generation. Paramset is initialized with pre-defined values such as number of rounds, number of S boxes, message digest bytes etc.
- A random private key is generated from the **randombytes()** function which uses AES-256 encryption. A slightly modified version of AES from Open-SSL

Library is used for encryption which performs S-box substitution and P-box permutations and Key is generated.

- A random block of plaintext is also generated similarly.
- From the given plaintext, a ciphertext is now generated which involves the **LowMCEnc** cipher. This function fundamentally performs operations such as matrix multiplication and XOR
- Finally, these generated public and private keys are written into corresponding buffers for further processing.

3. RESULT

FPGA Part	Name of Top Module	FF	LUT	BRAM	DSP	Latency	II
Artix - 7	crypto_sign_keypair	1939	11120	82	0	125591	-

PHASE-2 and 3

The target FPGA board is **Artix-7 board**

Benchmark	Type (Area /Latency)		Resource Utilization				Latency		Major Optimizations
		BRAM	LUT	FF	DSP	URAM	No of Clock cycle/latency	Clock period	
Baseline	-	82	11120	1939	0	0	125591	10	-
Optimization 1	Latency	92	24648	5113	0	0	89532	10	Loop Unrolling, Loop Flattening, Dead Code Elimination, Pipelining.
Optimization 2	Area	78	10288	2371	0	0	135720	10	Loop rolling, Pipelining, Dead Code Elimination

Optimization 1:

In the case of optimization 1, the major optimizations are Loop Unrolling, Loop Flattening, Dead Code Elimination, and Pipelining.

- Vivado HLS automatically takes advantage of parallelism i.e it will schedule operations to start as soon as they can. However, in case of rolled loops, operations are executed serially. To take advantage of parallelizability , Loops are unrolled.
- There can be certain areas in the code whose removal does not affect the results of the code. Such areas are called Dead Code. Removal of Dead code shrinks program size as well prevents execution of irrelevant operations, thus saving on execution time.

- Loop flattening involves converting nested loops into a single loop with the corresponding changes made to the outer loop. This inturn reduces the number of control blocks and optimizes latency.
- Pipelining reduces the initiation interval by allowing concurrent execution of operations, thus reducing the latency of the code. However, dependencies can sometimes hamper the performance of pipelines over unrolling.

Optimization 2:

In the case of optimization 2, the major optimizations are Loop Rolling, Pipelining and Dead Code Elimination.

Loop rolling, opposite to unrolling, reduces the parallelism of the code. Parallelism comes at the cost of additional hardware utilization. Loop Rolling forces sharing of hardware thus reducing the area.

Dead Code elimination and pipelining are the same as optimization 1.