# Reducing Complexity of SCMA Decoding Algorithm Using Hardware Optimization

Presented By:-

Prateek Kumar - 2021181

Sagar Keim - 2019196

# 5G Challenges

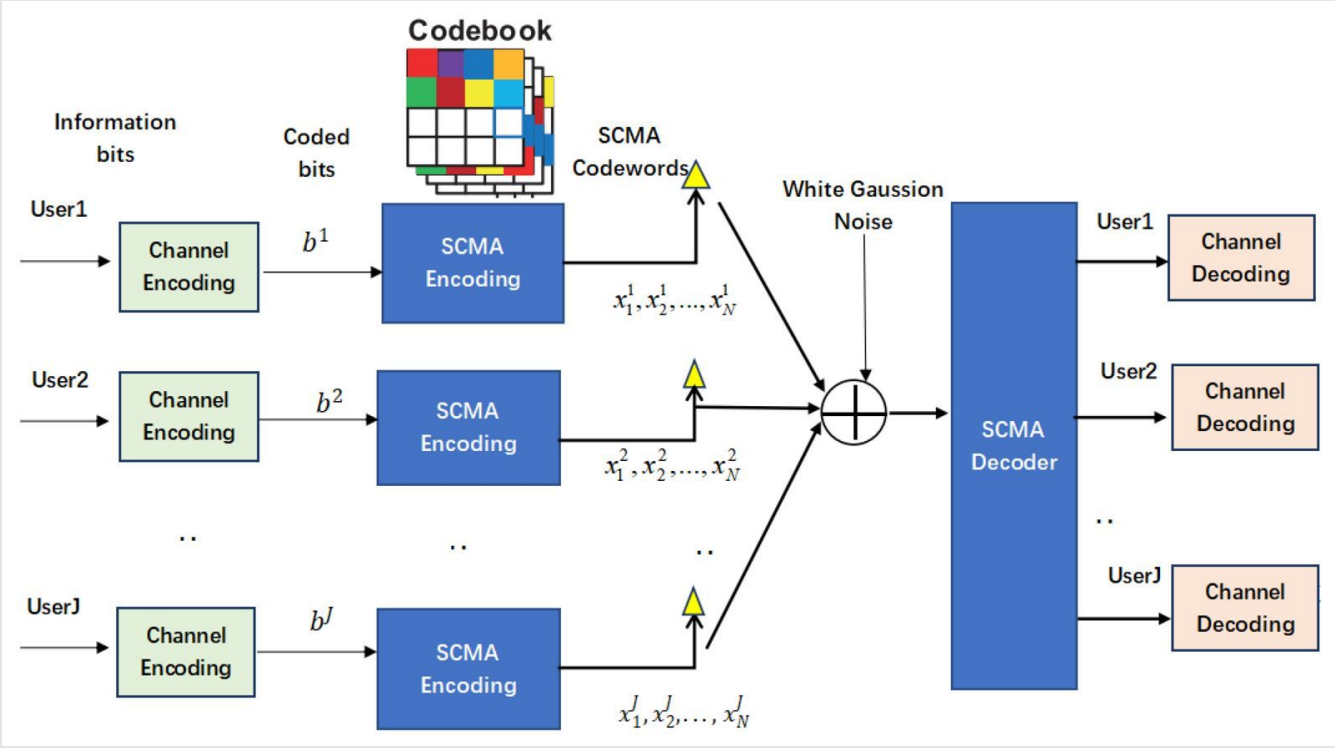
- Increasing no. of connected users and devices
- **Target connection density: 1 million devices per km** [2] [1]
- Difficulty in satisfying this with flexible - orthogonal multiple access techniques

# SCMA

*(Sparse Code Multiple Access)*

- A CD-NOMA technique.

- Bit <---> Constellation Mapping + Low Density Spreading = Bit <---------> Different Sparse Codewords



# Literature Review

*Sparse Code Multiple Access*

- Hardware Design and Implementation of Sparse Code Multiple Access
  - Focus on optimizing the SCMA decoding algorithms, specifically Log-MPA.
  - Explored SIMD-based parallelization techniques, Gaussian approximations for noise modeling.
  - Achieved up to **21x performance boost** using hardware-level parallelization.
  - Showed trade-offs between throughput and power efficiency using multi-threaded SIMD architectures.
- Performance Characterization of an SCMA Decoder
  - Characterized **hardware complexity and latency** of SCMA decoding.
  - Implemented parallel processing using high-level synthesis (HLS) tools for hardware acceleration.
  - Reduced latency and energy consumption while maintaining error performance.
  - Demonstrated how **Vivado HLS optimizations** improve throughput on FPGA implementations.

# Proposed Project

- Provide a hardware implementation of SCMA decoder.
- Offload bottleneck tasks to FPGA (Field-Programmable Gate Array)
- Reduce time complexity and resource utilization with hardware optimizations on the FPGA.

# Expected outcome

- FPGA implementation of SCMA
- Enhanced overall performance of CD-NOMA decoding for 6G

# Simulation Flow Overview

**INITIALIZATION:** Set up parameters and codebooks.

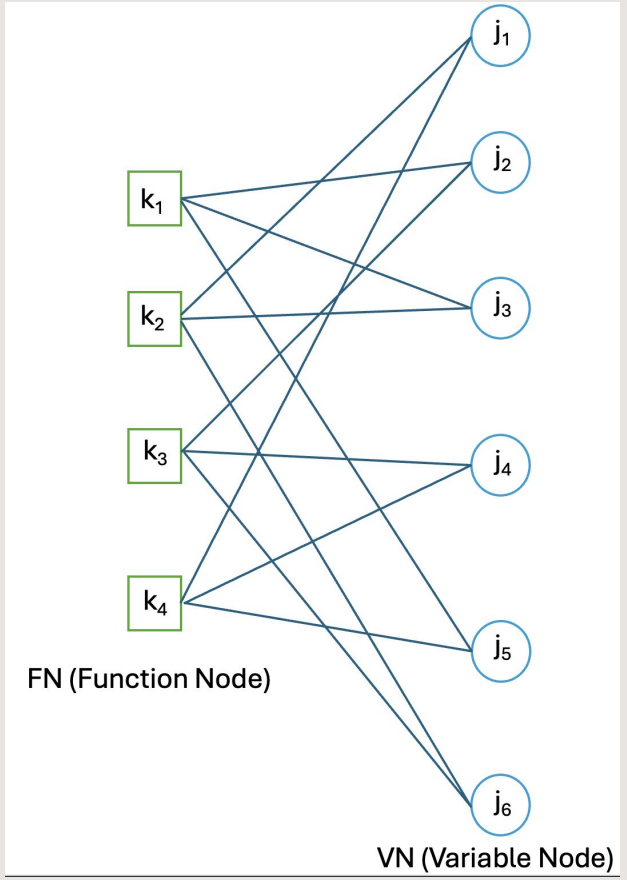
**DATA CREATION:** Randomly generate data to be transmitted

**ENCODING:** Map user data to codewords using predefined SCMA codebooks.

**TRANSMISSION:** Adds noise according to SNR

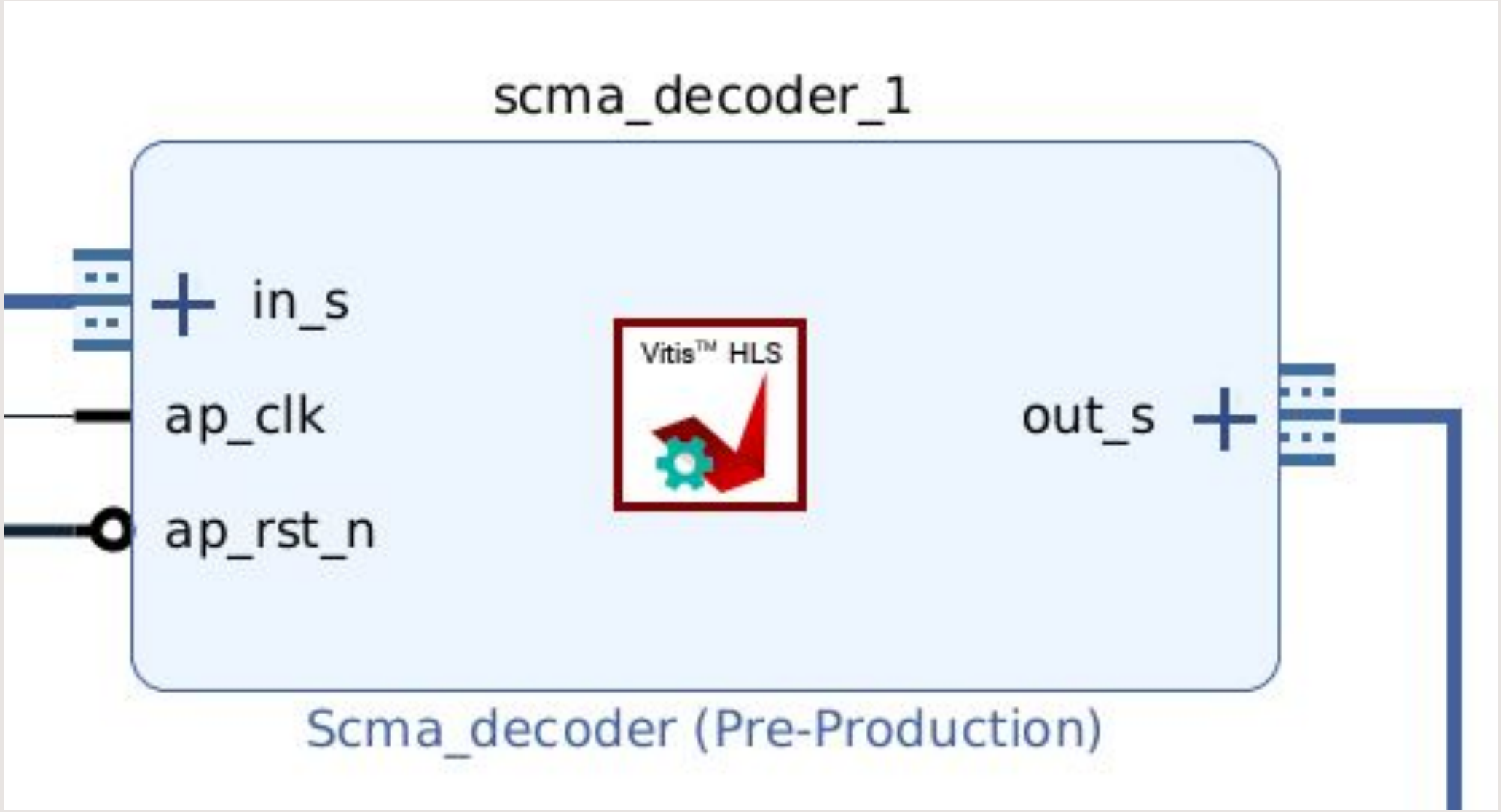
**DECODING:** Apply message passing algorithm to recover user data.

**PERFORMANCE METRICS:** Calculate Bit Error Rate (BER) over iterations.




**V**(# Users) = 6 **K**(# REs - Resource Elements) = 4 **M**(# Codewords) = 4 **N**(# Frames) = 1

# Hardware Decoder Architecture



# Implementation and Optimization

The SCMA decoder design was implemented using a top-down approach in Vivado HLS. The following optimizations were applied to improve performance:

**Memory Optimization:**

- Codebooks (`CB_real` and `CB_imag`) were stored as read-only constant arrays in on-chip memory to reduce access latency.
- Intermediate values, like channel-wise message contributions, were kept in local variables to minimize off-chip memory usage.

**Arithmetic Operations:**

- Hardware-optimized functions (`hls::log`, `hls::exp`, `hls::atan2`, etc.) were used for logarithmic, exponential, and trigonometric calculations, ensuring accuracy with minimal hardware overhead.

**Efficient Data Access:**

- Multidimensional arrays were designed to minimize data dependencies, enabling parallel computation of users' codewords by isolating memory regions.

# Work Done: Vitis HLS

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
   Compiling ../../../../scma_tb.cpp in debug mode
   Generating csim.exe
**********************INPUT S**********************(0.114534 + -0.809050i)
(-0.133710 + -0.089400i)
(0.877914 + -0.607059i)
(0.117609 + -0.089835i)
**********************LLR SW**********************-0.066879
-0.150131
-0.846068
-1.487909
-0.464295
-0.036847
-0.903067
-1.020223
-0.214334
0.150473
-1.244888
0.701885
**********************LLR HW**********************-0.066879
-0.150131
-0.846068
-1.487909
-0.464295
-0.036847
-0.903067
-1.020223
-0.214334
0.150473
-1.244888
0.701885
No error
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

C Simulation Result

scma.cpp | Synthesis Summary(solution1)

**Synthesis Summary Report of 'scma_decoder'**

**General Information**

Date: Sat Dec 7 04:12:54 2024
Version: 2022.2 (Build 3670227 on Oct 13 2022)
Project: scma2
Solution: solution1 (Vivado IP Flow Target)
Product family: zynquplus
Target device: xczu7ev-ffvc1156-2-e

**Timing Estimate**

| Target | Estimated | Uncertainty |
|---|---|---|
| 20.00 ns | 18.608 ns | 1.25 ns |

**Performance & Resource Estimates**

Modules Loops

| Modules & Loops | Issue Type | Violation Type | Distance | Slack | Latency(cycles) | Latency(ns) | Iteration Latency | Interval | Trip Count |
|---|---|---|---|---|---|---|---|---|---|
| scma_decoder | | | | - | 181781 | 3.636E6 | - | 181782 | |
| scma_decoder_Pipeline_1 | | | | - | 6 | 120.000 | - | 6 | |
| scma_decoder_Pipeline_2 | II Violation | | | - | 16 | 320.000 | - | 16 | |
| scma_decoder_Pipeline_VITIS_LOOP_144_4 | II Violation | | | - | 10 | 200.000 | - | 10 | |
| scma_decoder_Pipeline_VITIS_LOOP_164_6_VITIS_LOOP_165_7 | II Violation | | | - | 50 | 1.000E3 | - | 50 | |
| scma_decoder_Pipeline_5 | II Violation | | | - | 161 | 3.220E3 | - | 161 | |

Console | Errors | Warnings | Guidance | Properties | Man Pages | Git Repositories

CDT Build Console

Synthesis Result

**Cosimulation Report for 'scma_decoder'**

**General Information**

Date: Sat 07 Dec 2024 05:01:41 AM IST
Version: 2022.2 (Build 3670227 on Oct 13 2022)
Project: scma2
Status: Pass
Solution: solution1 (Vivado IP Flow Target)
Product family: zynquplus
Target device: xczu7ev-ffvc1156-2-e

**Cosim Options**

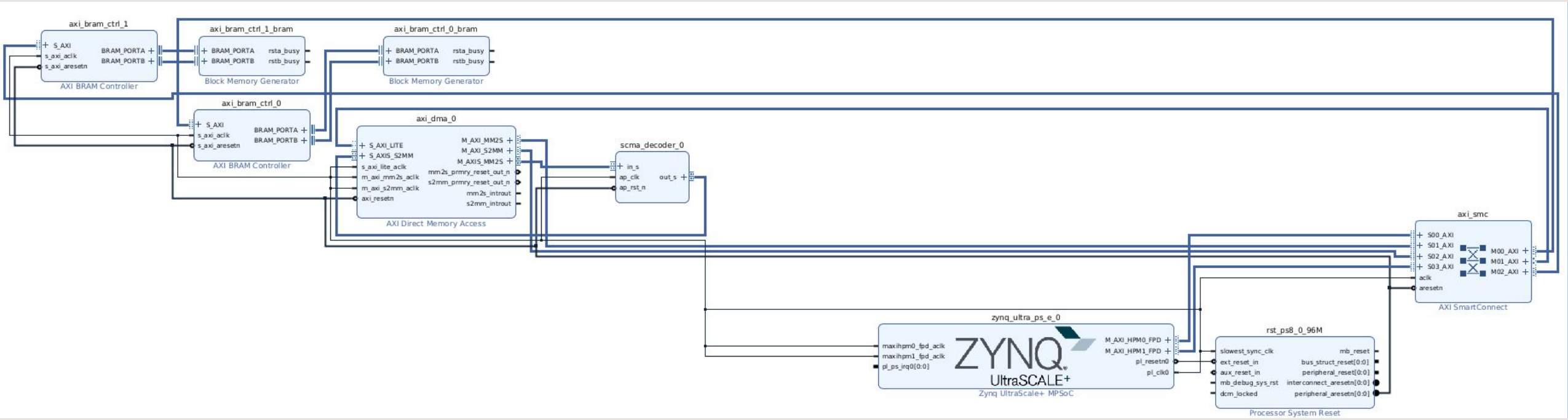
Tool: Vivado XSIM
RTL: Verilog

**Performance Estimates**

| Modules & Loops | Avg II | Max II | Min II | Avg Latency | Max Latency | Min Latency |
|---|---|---|---|---|---|---|
| scma_decoder | | | | | | |
| scma_decoder_Pipeline_1 | | | | 3 | 3 | 3 |
| scma_decoder_Pipeline_2 | | | | 11 | 11 | 11 |
| scma_decoder_Pipeline_VITIS_LOOP_144_4 | | | | 8 | 8 | 8 |
| scma_decoder_Pipeline_VITIS_LOOP_164_6_VITIS_LOOP_165_7 | | | | 48 | 48 | 48 |
| scma_decoder_Pipeline_5 | | | | 127 | 127 | 127 |
| scma_decoder_Pipeline_8 | | | | 47 | 47 | 47 |
| scma_decoder_Pipeline_9 | | | | 47 | 47 | 47 |
| scma_decoder_Pipeline_VITIS_LOOP_229_2_VITIS_LOOP_231_4_VITIS_LOOP_232_5 | | | | 257 | 257 | 257 |
| scma_decoder_Pipeline_VITIS_LOOP_274_11_VITIS_LOOP_275_12_VITIS_LOOP_276_13 | | | | 96 | 96 | 96 |

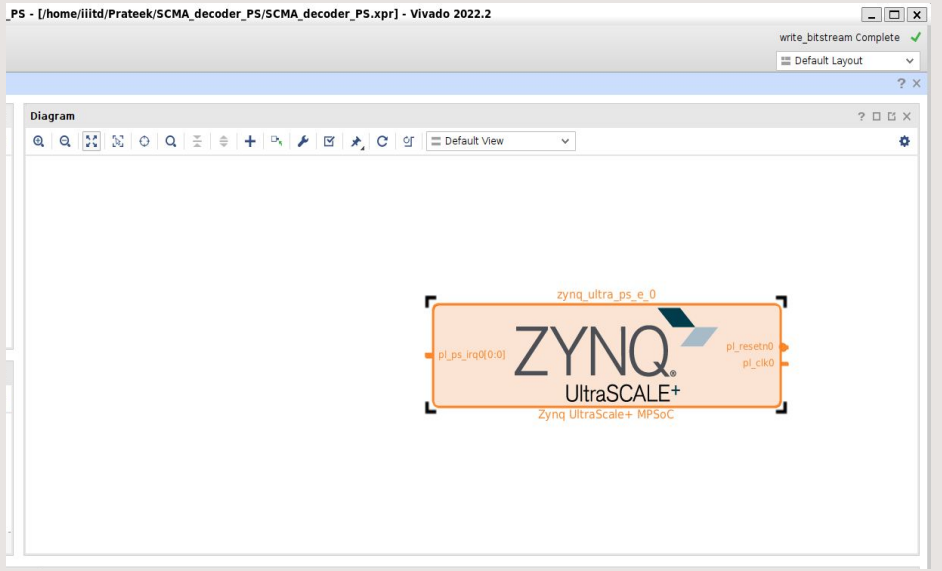
C RTL Co simulation Result

# Vivado Block Design




Implemented the baseline software code on ZCU106 PS.

- Created Block Design
- Wrote SDK code
- Executed on the HW



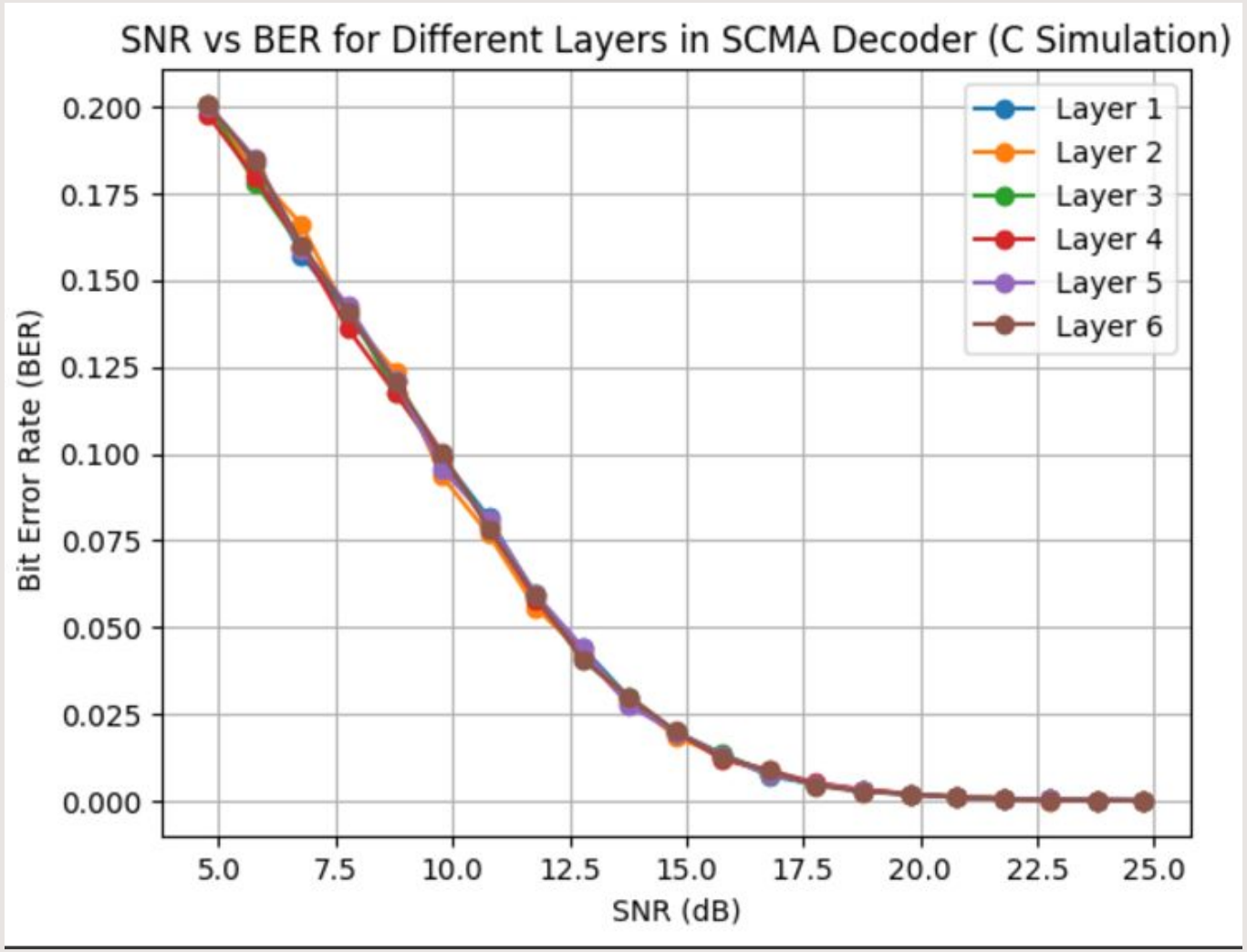
# C Simulation Results

```
(base) prateek@DESKTOP-6D
(0.114534 + -0.809050i)
(-0.133710 + -0.089400i)
(0.877914 + -0.607059i)
(0.117609 + -0.089835i)
(base) prateek@DESKTOP-6D
```

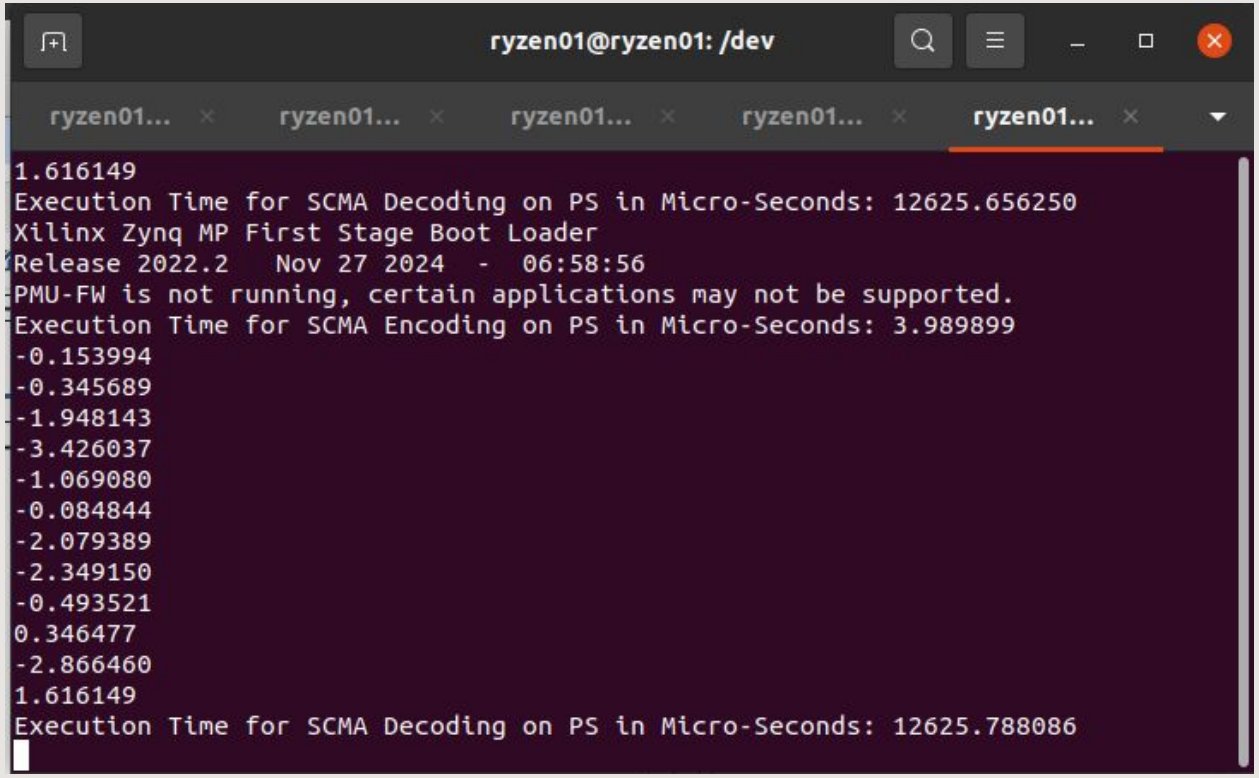
Encoded Data

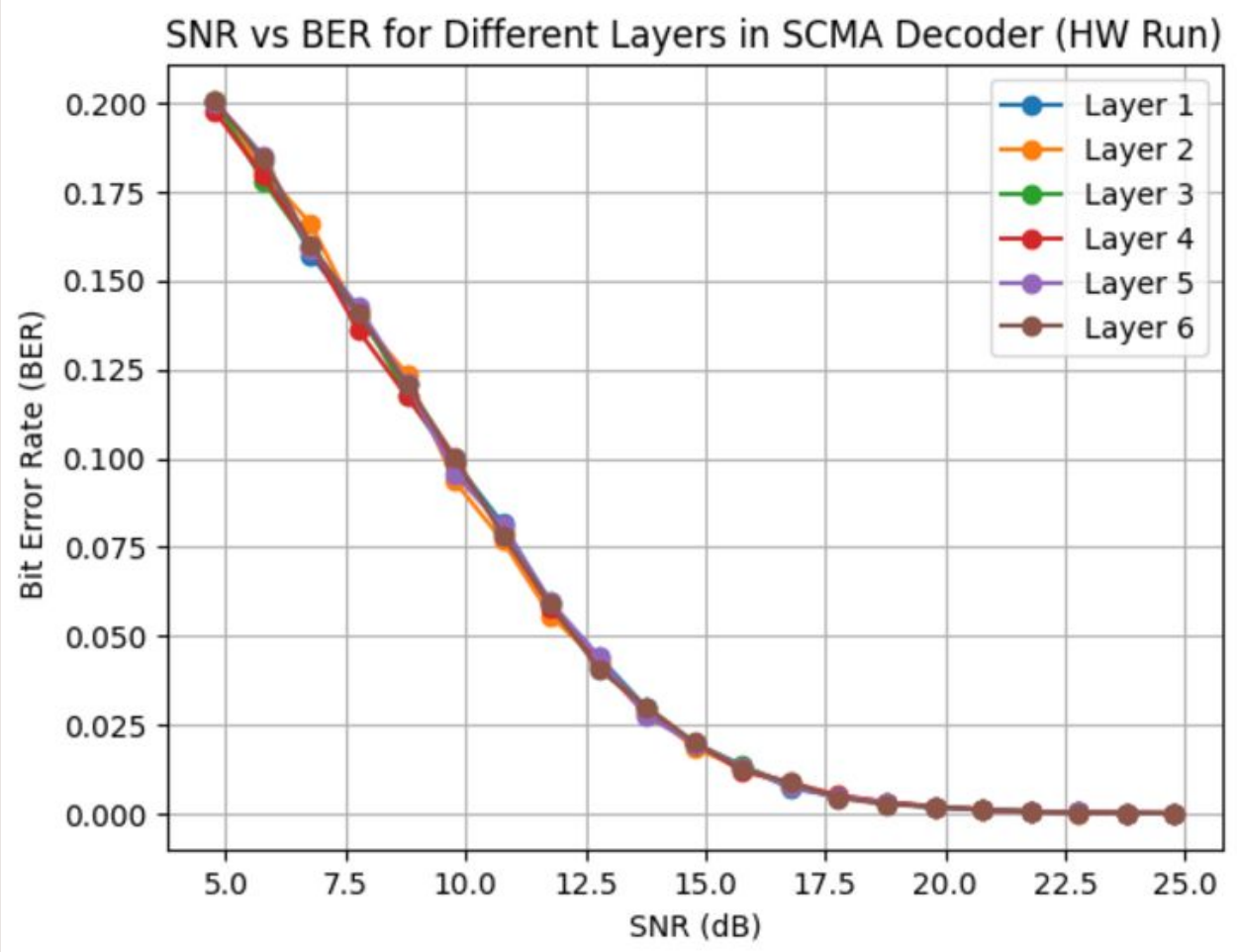
```
(base) prateek@DESKTOP-6DBV5A7
SCMA Encoding Time: 14.00 us
-0.153994
-0.345689
-1.948143
-3.426037
-1.069080
-0.084844
-2.079389
-2.349150
-0.493521
0.346477
-2.866460
1.616149
```

LLR of each user



# Hardware Simulation Results



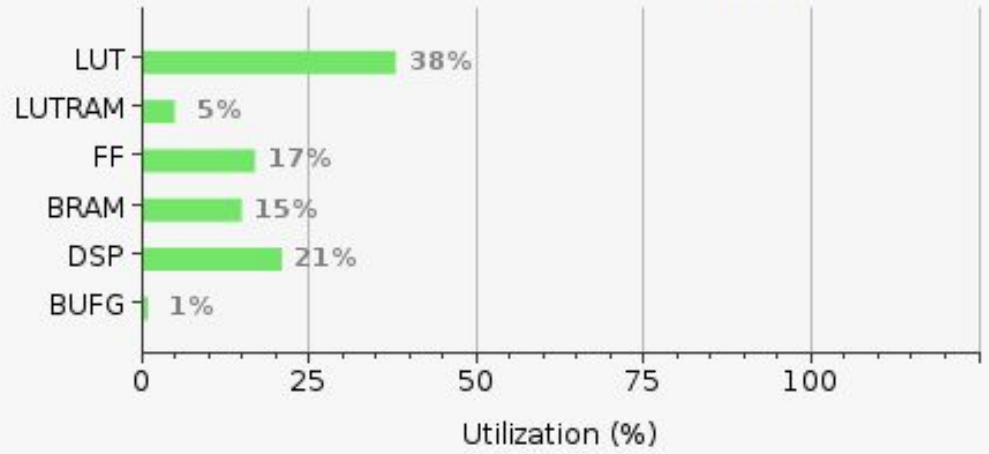




This is the execution result on the ZCU-106 FPGA board and its result was noted down.
Execution Time for encoder: **3.98 us**
Execution time for decoder: **12.625 ms**

# Hardware Power Utilization

**Utilization** Post-Synthesis | **Post-Implementation**

**Graph** | Table




**Power** **Summary** | On-Chip

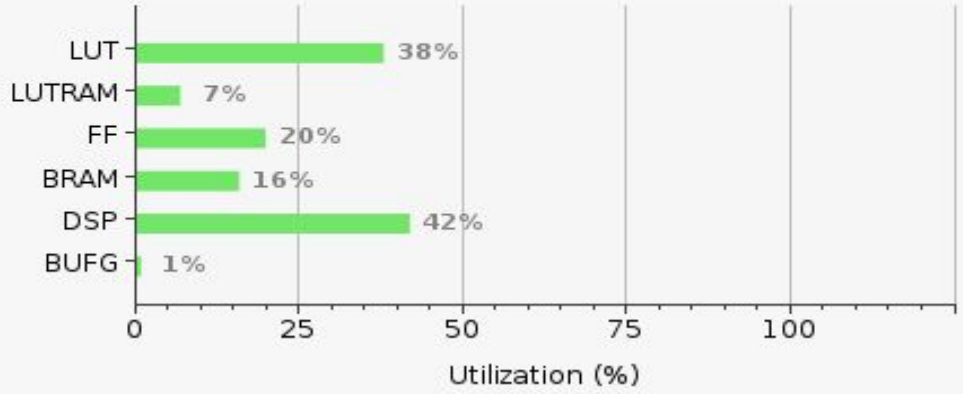
| | |
|---|---|
| **Total On-Chip Power:** | **4.996 W** |
| **Junction Temperature:** | **29.9 °C** |
| Thermal Margin: | 70.1 °C (70.3 W) |
| Effective θJA: | 1.0 °C/W |
| Power supplied to off-chip devices: | 0 W |
| Confidence level: | Medium |

Implemented Power Report

Solution 1

**Utilization** Post-Synthesis | **Post-Implementation**

**Graph** | Table



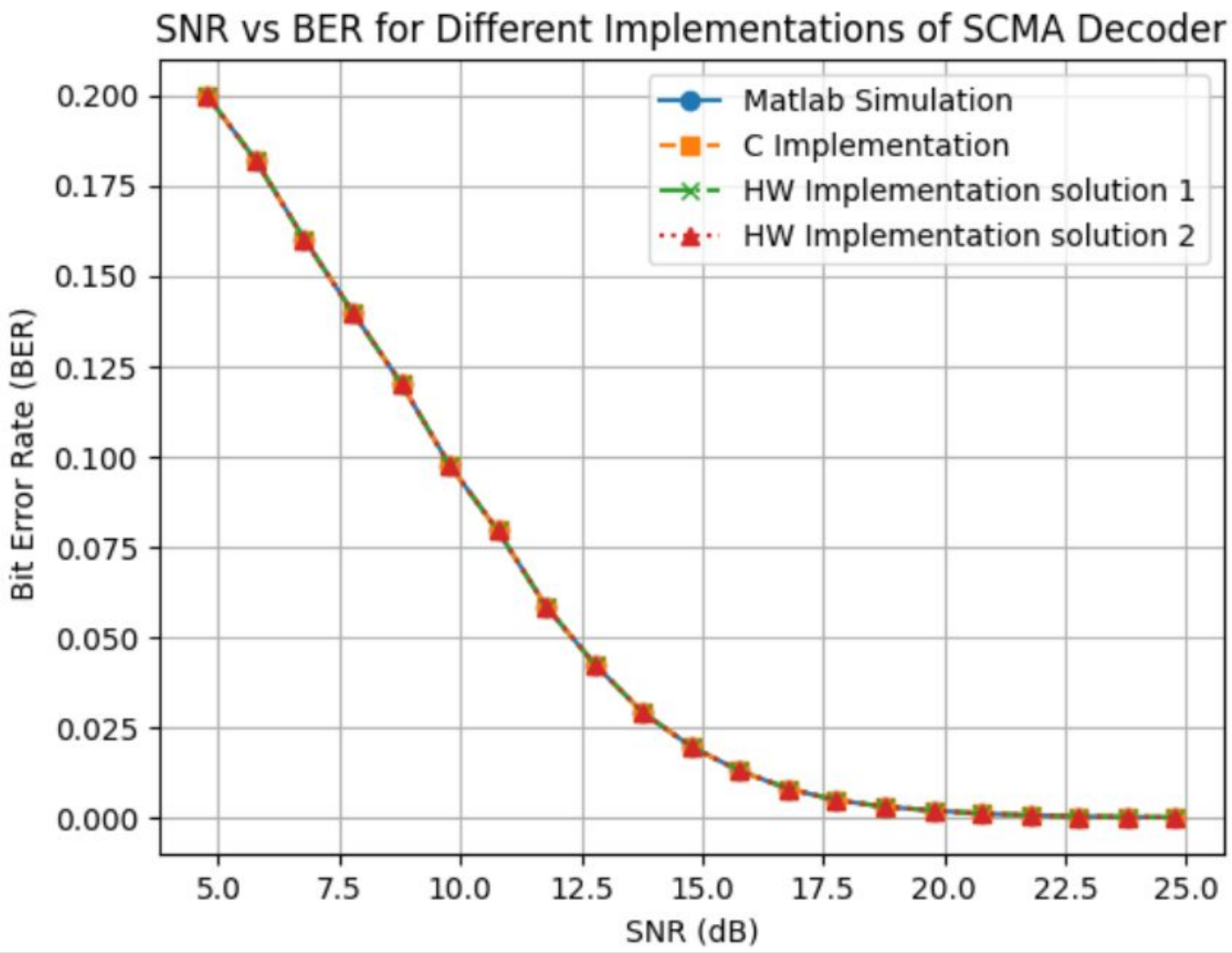

**Power** **Summary** | On-Chip

| | |
|---|---|
| **Total On-Chip Power:** | **5.169 W** |
| **Junction Temperature:** | **30.1 °C** |
| Thermal Margin: | 69.9 °C (70.1 W) |
| Effective θJA: | 1.0 °C/W |
| Power supplied to off-chip devices: | 0 W |
| Confidence level: | Medium |

Implemented Power Report

Solution 2

# SNR vs BER


SNR vs BER for Different Implementations of SCMA Decoder
Matlab Simulation
C Implementation
HW Implementation solution 1
HW Implementation solution 2
Bit Error Rate (BER)
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
SNR (dB)

# Results

| | | Algorithm evaluation using Vivado HLS | | | |
|---|---|---|---|---|---|
| | | **Solution 1** (No Directive) | **Solution 2** (pipeline,BRAM) | **C Implementation** | **Matlab Implementation** |
| **Timing Analysis** | **Estimated Clock Period [ns]** | 8.728 ns | 18.608 ns | - | - |
| | **Latency in clock cycles** | 659211 | 181781 | - | - |
| | **Latency in time [ms]** | 6.592ms | 3.383 ms | 12.626 ms | 1526 ms |
| | **Acceleration Factor** | **231x** | **451x** | **120x** | **1x** |
| **Resource utilization analysis** | **BRAM** | 56 (8%) | 64 (10%) | - | - |
| | **DSP** | 384 (22%) | 802 (46%) | - | - |
| | **FF** | 52691 (11%) | 65924 (14%) | - | - |
| | **LUT** | 50568 (21%) | 73698 (31%) | - | - |

# Future Research Directions

- Other Hardware optimizations that can be performed:
  - Word Length Optimization
  - LUT based or linear interpolation based mathematical functions
  - Loop Unrolling

- Algorithm Level Optimization:
  - Using the spatial locality and temporal locality into consideration for fading coefficients in the algorithm- reusing results from the previous iteration.

- Implementation of other SCMA algorithms on the hardware and comparing multiple decoding algorithms.


THANK YOU