Project Portfolio

KARTHIK MUKUND

GitHub: karthik-m-11 | LinkedIn: Karthik Mukund | Email: karthik.mukund2002@gmail.com | Phone: (+91) 9867174316

“Passionate about VLSI and designing performance-oriented ICs”

About Me

My Top Skills

- Functional Verification (UVM)
- Formal Verification using Assertions (SV)
- RTL Design (Verilog)
- Analog Circuit Design (SPICE)
- Programming for Automation (Python)

My Qualifications

- Industry Trained Design Verification Fresher, ChipEdge (2025)
- Research Internship in Satellite Control Systems, IIST (2024)
- B.Tech ECE, Vellore Institute of Technology - Vellore (2021 - 2025)

TABLE OF CONTENTS

Internships, Research Papers, and Patents

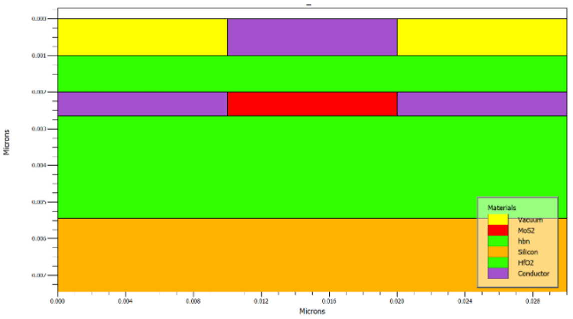
1. DC Performance Analysis of MOS2FETs (Research Paper)

Tool: Silvaco TCAD

Objective

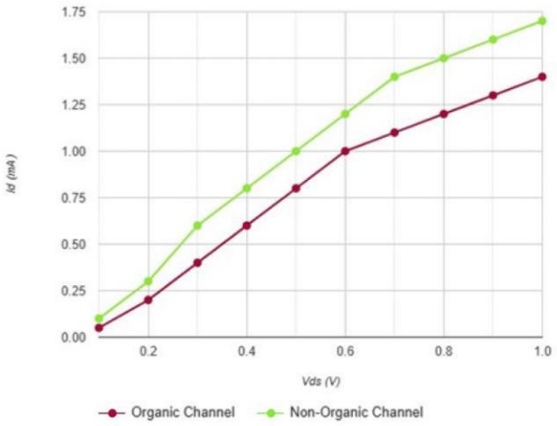
To analyze the impact of channel length, oxide height, and gate electrode work function on the I-V characteristics of MOS2FETs, and find the optimal specifications for high-speed and low-power switching applications such as wearable electronics and flexible displays.

Approach



A 2-D prototype for MOS2FET was designed in Silvaco TCAD, and each parameter was modified maintaining the others constant. The I-V curve was generated for each scenario, and the cut-off values for each noted and graphed for comparison.

Outcome



It was found that while keeping channel material constant, any changes in dimension or work function had negligible impact. However, when the channel material was changed from non-organic to organic, there was a marked increase in the voltage required to achieve the same current.

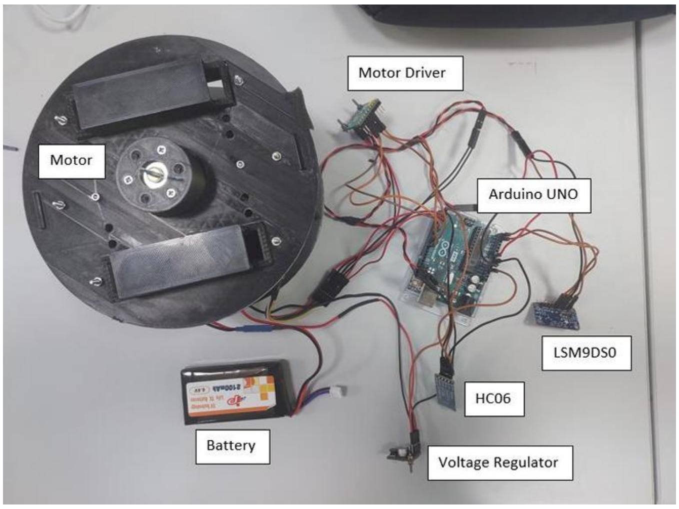
2. Reaction Wheel Attitude Control System in CubeSats (Internship)

Language: C++ | Tool: Arduino IDE | Project was implemented in hardware

Objective

To design a single-axis reaction wheel control system, that will later be extended to 4-axes, for rapid recovery of attitude from disorientation during payload deployment.

Approach



A custom algorithm was developed and uploaded into an Arduino Uno microcontroller. The Arduino was then connected to a power source, sensors, and the reaction wheel motor actuator. The response times for correcting displacements in attitude were observed and noted.

Outcome

The designed prototype was tested on an air bearing floating bed test setup, and it was observed that it was successfully able to self-correct to its original attitude when externally disrupted.

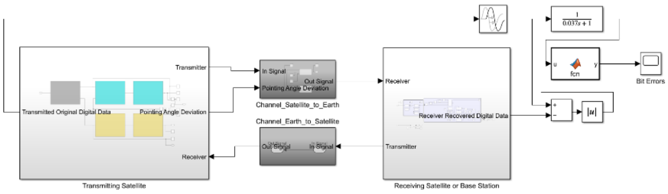
3. Mono-Frequency Transmission for LEO Satellites (Patent)

Tool: MATLAB Simulink

Objective

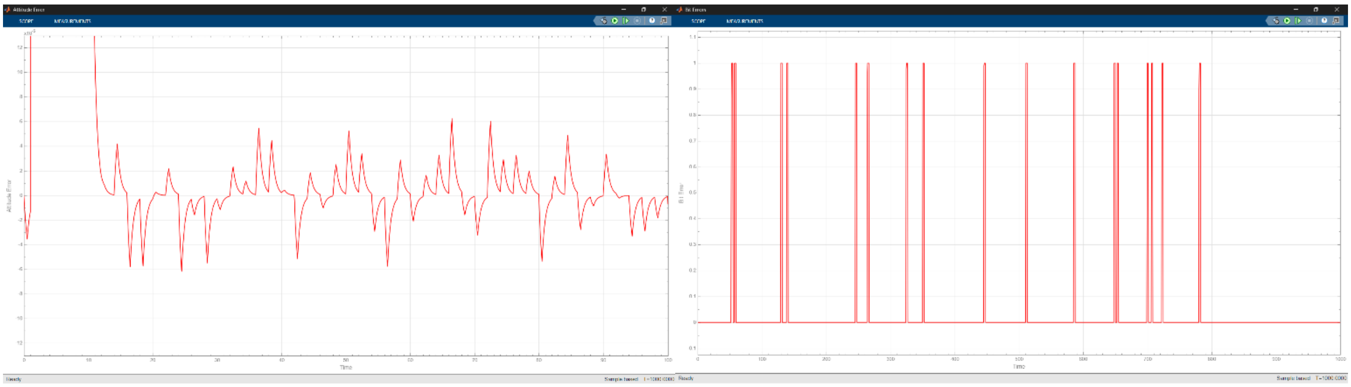
To explore mono-frequency transmission aided by a Lyapunov stability control system for power efficient and secure communication in autonomous satellites.

Approach



A model was designed in Simulink for the satellite's attitude stability submodule, satellite's signal transmission submodule, and the receiving station's signal recovery module. All the modules were connected and a channel was modeled to simulate noise and attenuation in the communication link. Artificial jitter was introduced to observe the satellite's attitude correction ability. The noise level was varied to observe the maximum noise the system could handle and still recover the original data.

Outcome



The designed system was observed to be able to recover the original data from a signal as noisy as -6dB, which is a 40-fold decrease from traditional optical communication systems. This allowed the transmitting laser at the satellite to use significantly less power (0.6816 W) than usually required.

VLSI – RTL Design

1. PWM Signal Generator

Language: Verilog | Tool: Linux, Icarus Verilog + GTKWave

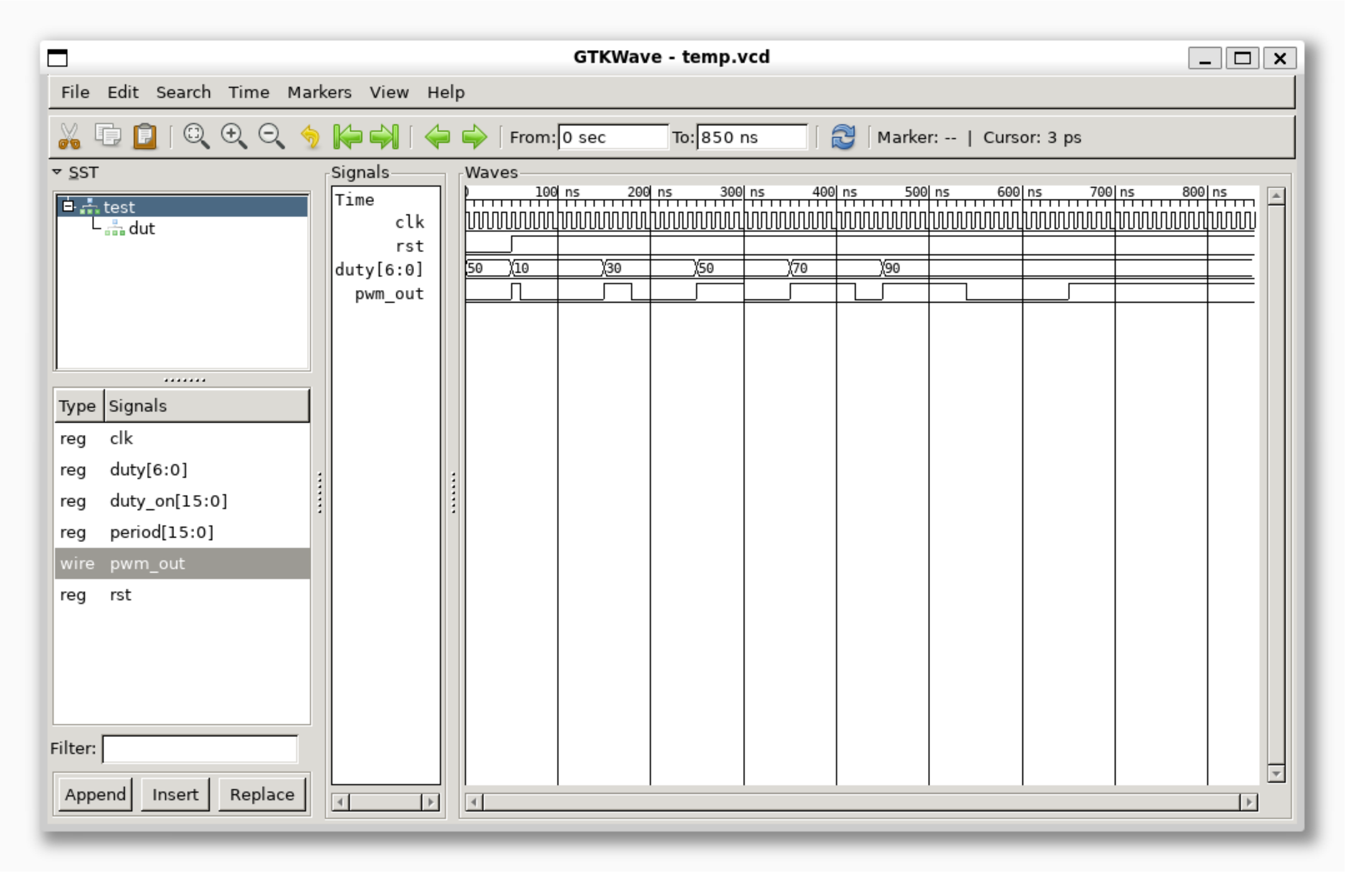
Objective

To design a PWM signal generator in Verilog.

Approach

A Verilog program was written to design a PWM signal generator modeled on an up-counter. A testbench was written to verify the design's functionality. The files were then compiled and simulated using Icarus Verilog, and the waveform was generated using GTKWave for observation and analysis.

Outcome



Type	Signals
reg	clk
reg	duty[6:0]
reg	duty_on[15:0]
reg	period[15:0]
wire	pwm_out
reg	rst

2. Digital Stopwatch using FSM

Language: Verilog | Tool: Linux, Icarus Verilog + GTKWave

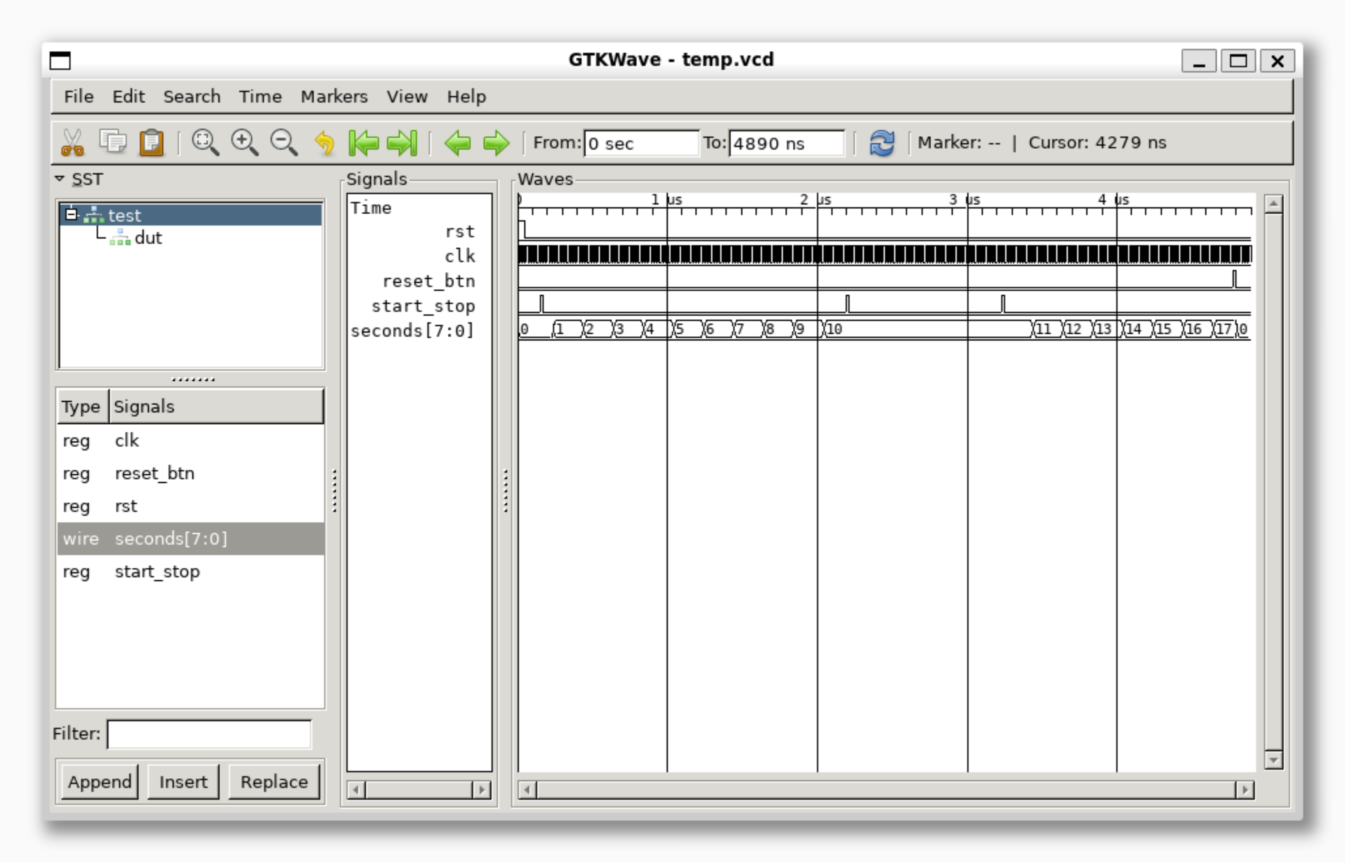
Objective

To design a digital stopwatch using FSM in Verilog.

Approach

A Verilog program was written to design a digital stopwatch modeled using an FSM. A testbench was written to verify the design's functionality. The files were then compiled and simulated using Icarus Verilog, and the waveform was generated using GTKWave for observation and analysis.

Outcome



GTKWave - temp.vcd

From: 0 sec To: 4890 ns Marker: -- | Cursor: 4279 ns

Type	Signals
reg	clk
reg	reset_btn
reg	rst
wire	seconds[7:0]
reg	start_stop

3. Custom RISC-V Processor Design

Language: Verilog | Tool: Linux, Icarus Verilog + GTKWave

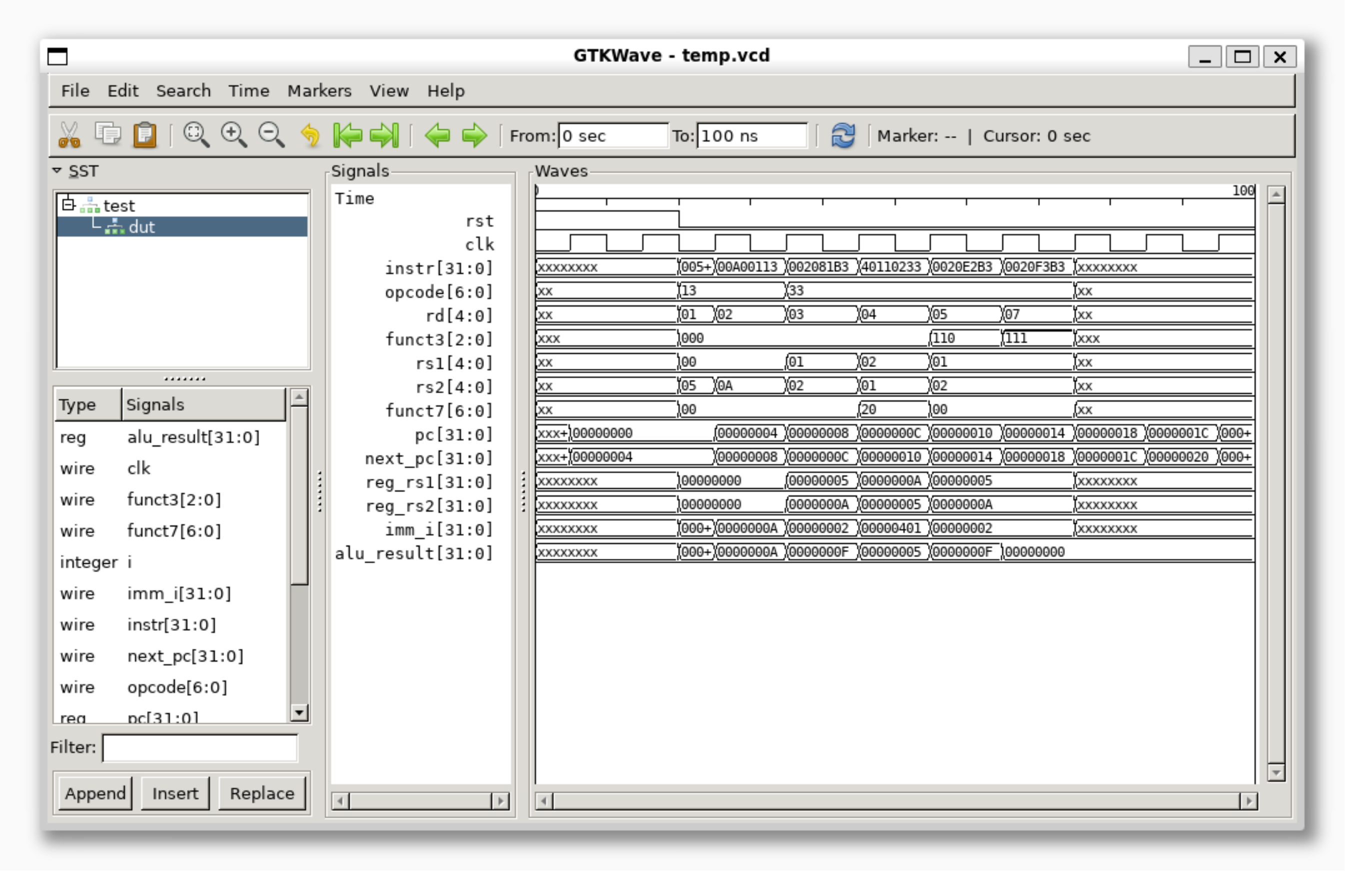
Objective

To design a basic 3 stage “decode → alu → write_back” RISC-V based 32-bit microprocessor.

Approach

A Verilog program was written to design the custom RISC-V microprocessor implementing an instruction memory, program counter, regfile, ALU, and ports. A testbench was written to verify the design’s functionality. The files were then compiled and simulated using Icarus Verilog, and the waveform was generated using GTKWave for observation and analysis.

Outcome



GTKWave - temp.vcd

From: 0 sec To: 100 ns | Marker: -- | Cursor: 0 sec

File Edit Search Time Markers View Help
SST
test
dut
Type Signals
reg alu_result[31:0]
wire clk
wire funct3[2:0]
wire funct7[6:0]
integer i
wire imm_i[31:0]
wire instr[31:0]
wire next_pc[31:0]
wire opcode[6:0]
reg pc[31:0]
Filter:
Append Insert Replace
Signals
Waves
Time 100
rst
clk
instr[31:0] xxxxxxxx 005+ 00A00113 002081B3 40110233 0020E2B3 0020F3B3 xxxxxxxx
opcode[6:0] xx 13 33 xx
rd[4:0] xx 01 02 03 04 05 07 xx
funct3[2:0] xxx 000 110 111 xxx
rs1[4:0] xx 00 01 02 01 xx
rs2[4:0] xx 05 0A 02 01 02 xx
funct7[6:0] xx 00 20 00 xx
pc[31:0] xxx+ 00000000 00000004 00000008 0000000C 00000010 00000014 00000018 0000001C 000+
next_pc[31:0] xxx+ 00000004 00000008 0000000C 00000010 00000014 00000018 0000001C 00000020 000+
reg_rs1[31:0] xxxxxxxx 00000000 00000005 0000000A 00000005 xxxxxxxx
reg_rs2[31:0] xxxxxxxx 00000000 0000000A 00000005 0000000A xxxxxxxx
imm_i[31:0] xxxxxxxx 000+ 0000000A 00000002 00000401 00000002 xxxxxxxx
alu_result[31:0] xxxxxxxx 000+ 0000000A 0000000F 00000005 0000000F 00000000

VLSI – Design Verification

1. AMBA APB Design and Verification using UVM

Language: SystemVerilog, UVM | Tool: Linux, Synopsys VCS + Verdi Waveform Viewer

Objective

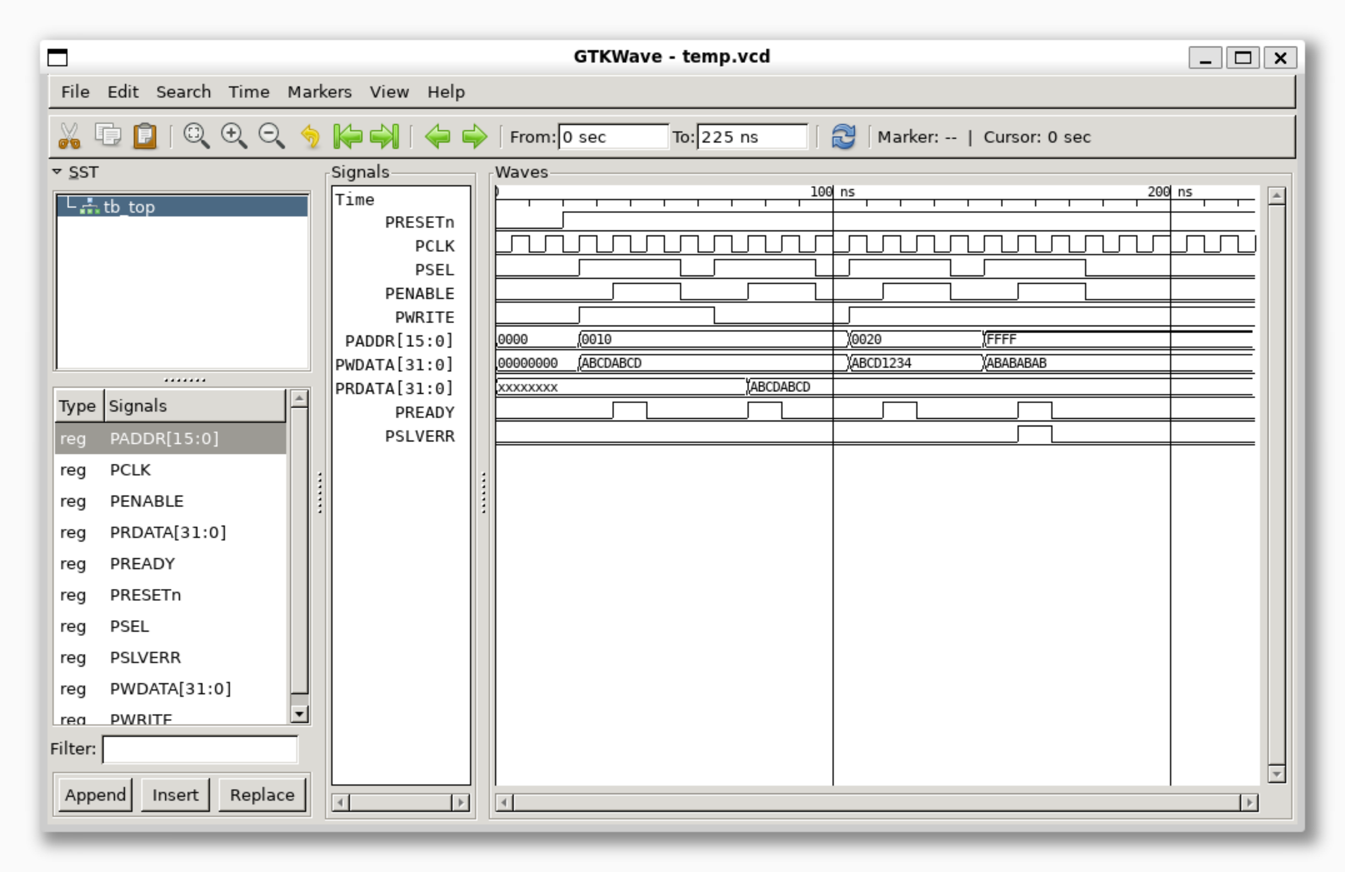
To design and verify the industry standard parallel communication protocol AMBA APB.

Approach

The AMBA APB protocol was designed in Verilog and verified using UVM testbench architecture. Assertions and coverage were written to check the functionality. The result was verified by generating and analyzing the waveform.

Original project designed on licensed server with Synopsys VCS and Verdi Waveform Viewer access. Result recreated in personal computer using GTKWave.

Outcome



2. AMBA AHB Design and Verification using UVM

Language: SystemVerilog, UVM | Tool: Linux, Synopsys VCS + Verdi Waveform Viewer

Objective

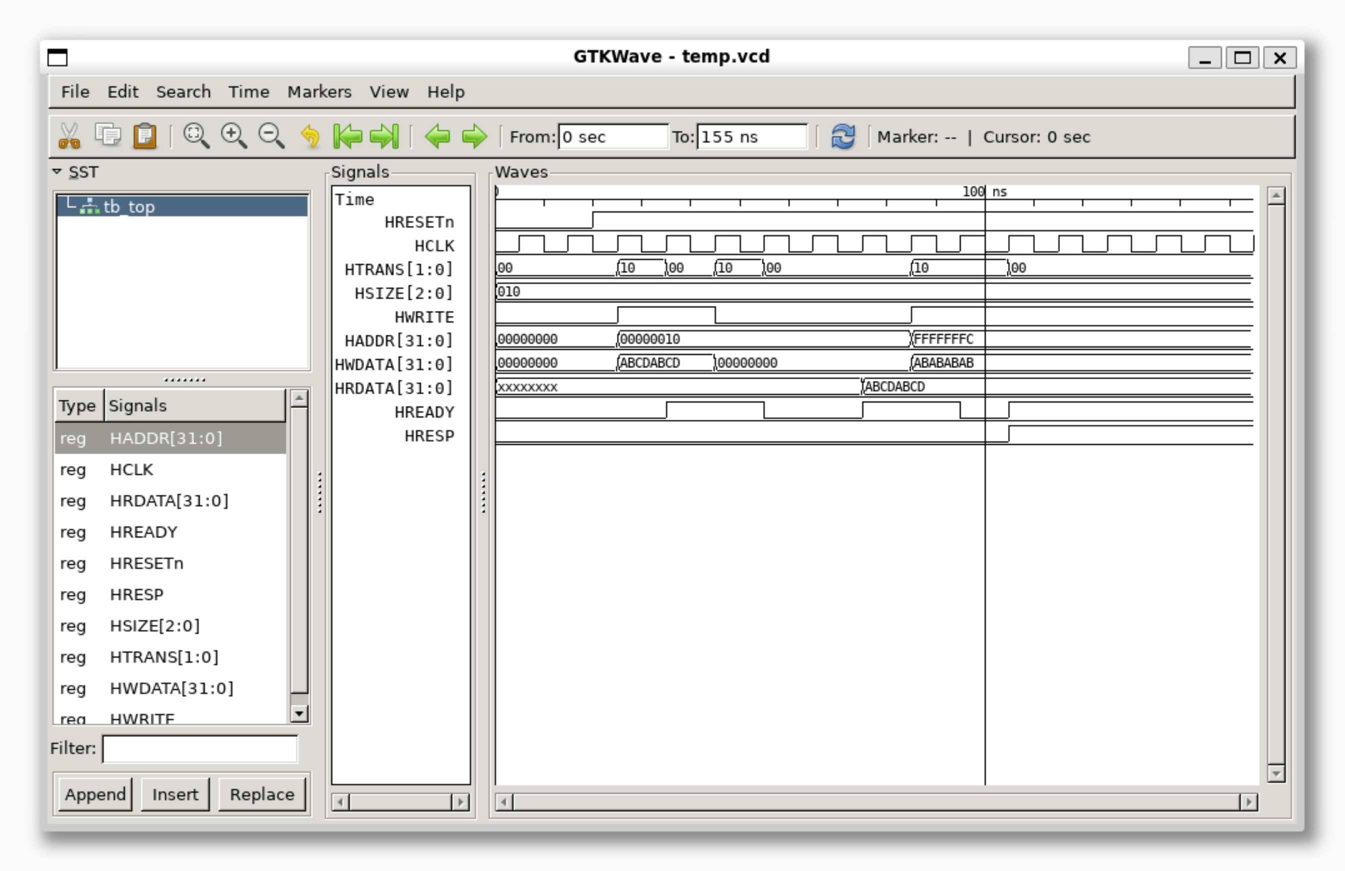
To design and verify the industry standard serial communication protocol AMBA AHB.

Approach

The AMBA AHB protocol was designed in Verilog and verified using UVM testbench architecture. Assertions and coverage were written to check the functionality. The result was verified by generating and analyzing the waveform.

Original project designed on licensed server with Synopsys VCS and Verdi Waveform Viewer access. Result recreated in personal computer using GTKWave.

Outcome



Type	Signals
reg	HADDR[31:0]
reg	HCLK
reg	HRDATA[31:0]
reg	HREADY
reg	HRESETn
reg	HRESP
reg	HSIZE[2:0]
reg	HTRANS[1:0]
reg	HWDATA[31:0]
reg	HWRITE

3. AMBA AXI Design and Verification using UVM

Language: SystemVerilog, UVM | Tool: Linux, Synopsys VCS + Verdi Waveform Viewer

Objective

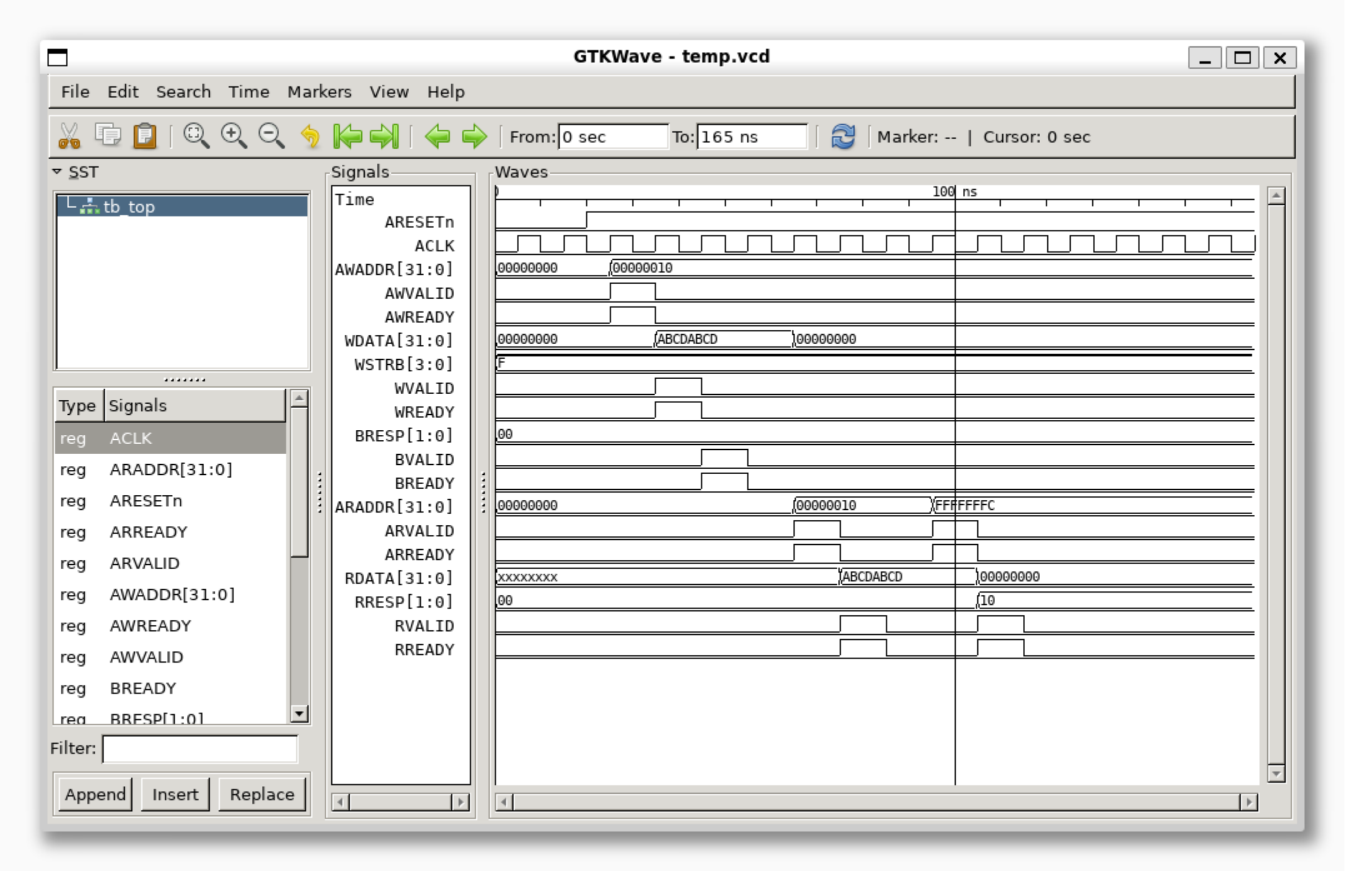
To design and verify the industry standard parallel communication protocol AMBA AXI.

Approach

The AMBA AXI protocol was designed in Verilog and verified using UVM testbench architecture. Assertions and coverage were written to check the functionality. The result was verified by generating and analyzing the waveform.

Original project designed on licensed server with Synopsys VCS and Verdi Waveform Viewer access. Result recreated in personal computer using GTKWave.

Outcome



4. Ethernet Design and Verification using UVM

Language: SystemVerilog, UVM | Tool: Linux, Synopsys VCS + Verdi Waveform Viewer

Objective

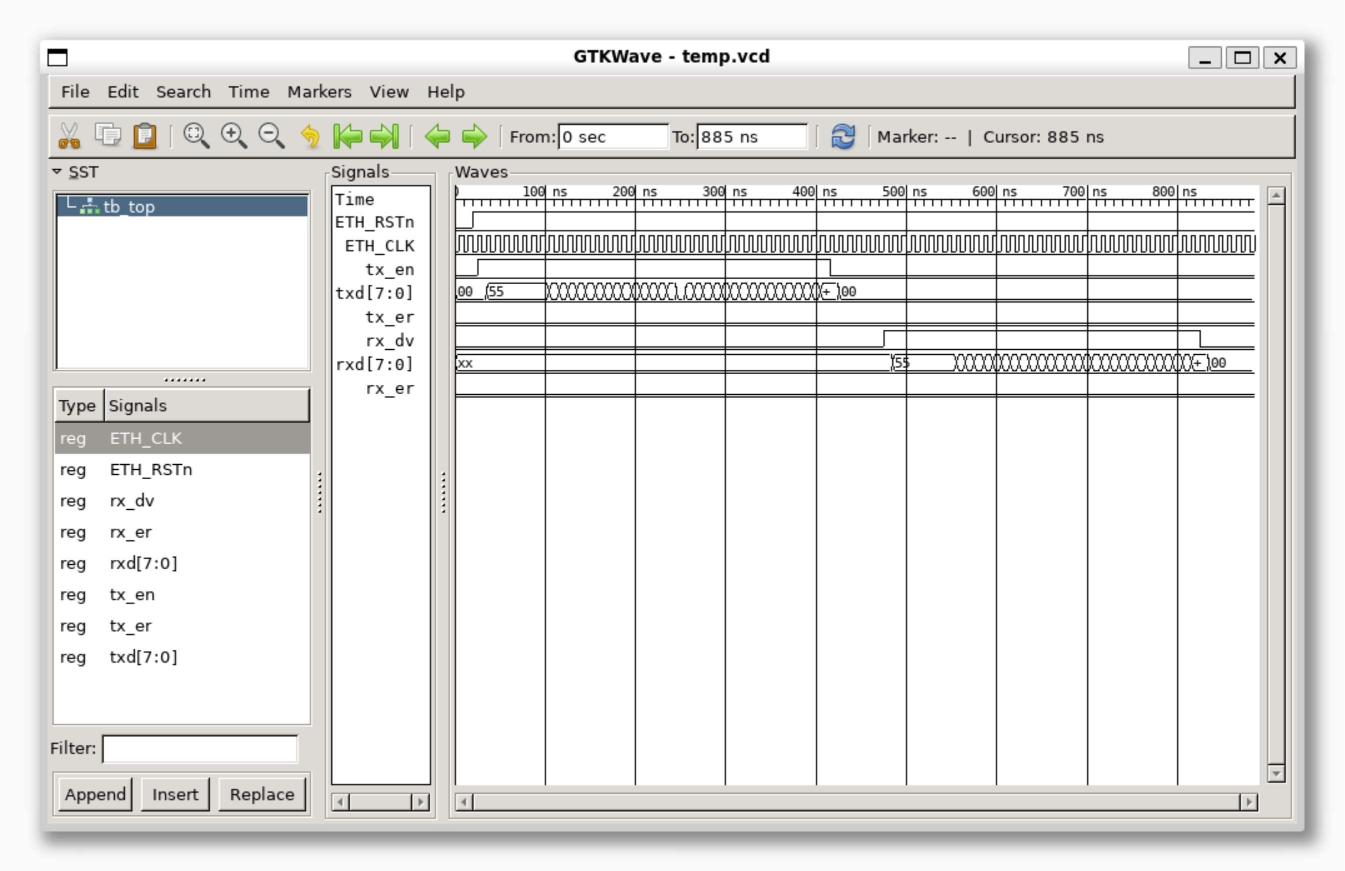
To design and verify the industry standard serial communication protocol ethernet.

Approach

The ethernet protocol was designed in Verilog and verified using UVM testbench architecture. Assertions and coverage were written to check the functionality. The result was verified by generating and analyzing the waveform.

Original project designed on licensed server with Synopsys VCS and Verdi Waveform Viewer access. Result recreated in personal computer using GTKWave.

Outcome



5. USB Design and Verification using UVM

Language: SystemVerilog, UVM | Tool: Linux, Synopsys VCS + Verdi Waveform Viewer

Objective

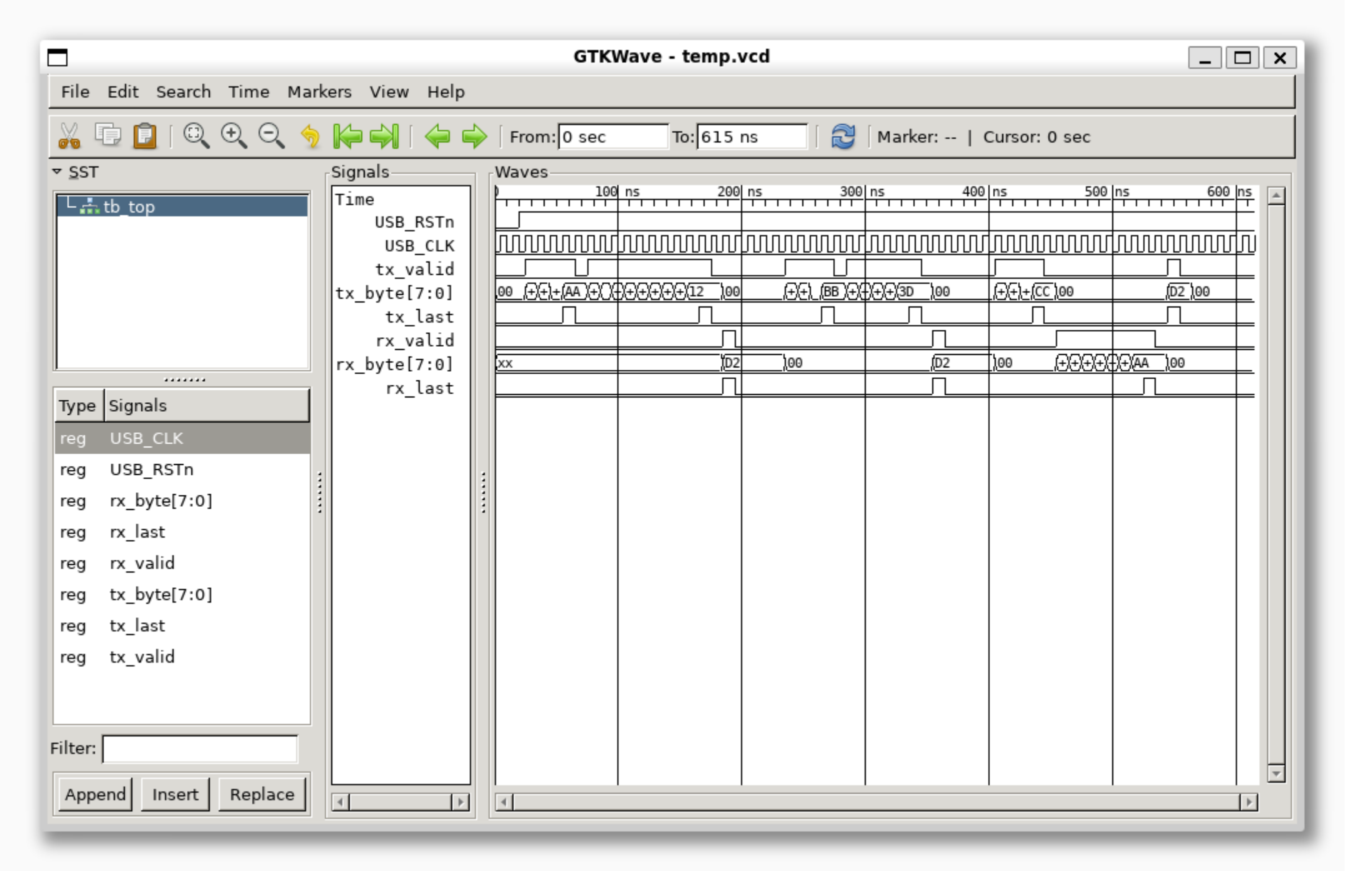
To design and verify the industry standard serial communication protocol USB.

Approach

The USB protocol was designed in Verilog and verified using UVM testbench architecture. Assertions and coverage were written to check the functionality. The result was verified by generating and analyzing the waveform.

Original project designed on licensed server with Synopsys VCS and Verdi Waveform Viewer access. Result recreated in personal computer using GTKWave.

Outcome



GTKWave - temp.vcd

From: 0 sec To: 615 ns | Marker: -- | Cursor: 0 sec

Type	Signals
reg	USB_CLK
reg	USB_RSTn
reg	rx_byte[7:0]
reg	rx_last
reg	rx_valid
reg	tx_byte[7:0]
reg	tx_last
reg	tx_valid

6. PCIe Design and Verification using UVM

Language: SystemVerilog, UVM | Tool: Linux, Synopsys VCS + Verdi Waveform Viewer

Objective

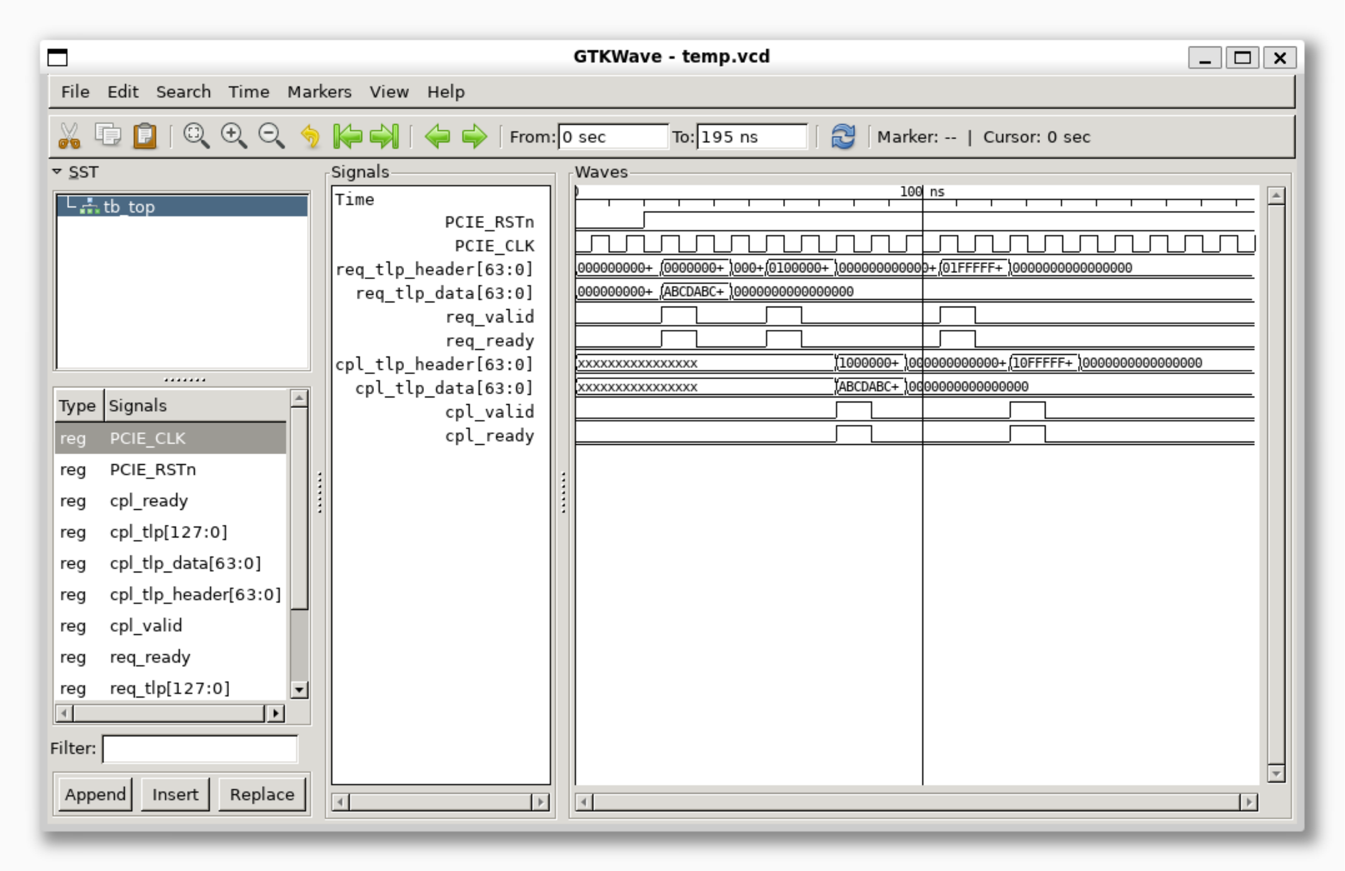
To design and verify the industry standard serial communication protocol PCIe.

Approach

The PCIe protocol was designed in Verilog and verified using UVM testbench architecture. Assertions and coverage were written to check the functionality. The result was verified by generating and analyzing the waveform.

Original project designed on licensed server with Synopsys VCS and Verdi Waveform Viewer access. Result recreated in personal computer using GTKWave.

Outcome



From: 0 sec To: 195 ns Marker: -- | Cursor: 0 sec

Type	Signals
reg	PCIE_CLK
reg	PCIE_RSTn
reg	cpl_ready
reg	cpl_tlp[127:0]
reg	cpl_tlp_data[63:0]
reg	cpl_tlp_header[63:0]
reg	cpl_valid
reg	req_ready
reg	req_tlp[127:0]

```
Time
PCIE_RSTn
PCIE_CLK
req_tlp_header[63:0]
req_tlp_data[63:0]
req_valid
req_ready
cpl_tlp_header[63:0]
cpl_tlp_data[63:0]
cpl_valid
cpl_ready
```

CONTACT ME

Phone: (+91) 9867174316

Email: karthik.mukund2002@gmail.com

LinkedIn: Karthik Mukund

GitHub: karthik-m-11