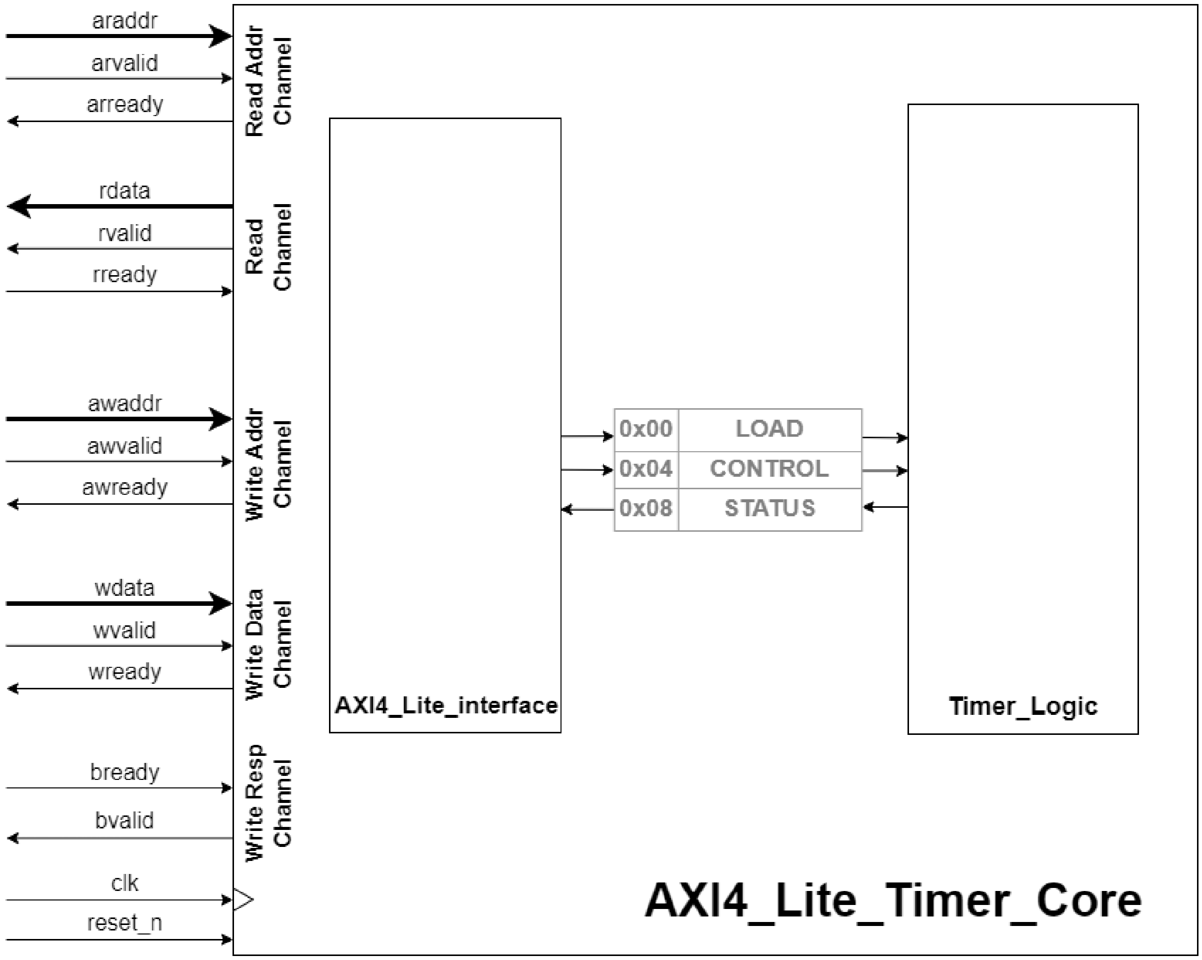
AXI4-Lite Hardware Timer: An RTL Implementation | Github

✅ Project Overview

This project presents the design and implementation of a **hardware timer using an AXI4-Lite interface**. The hardware timer is designed for FPGAbased applications requiring precise timing control and **registerbased** configuration via an **AXI4-Lite bus**.



0x00	LOAD
0x04	CONTROL
0x08	STATUS

✅ Key Features:

- **AXI4-Lite Interface:** Provides a simple yet effective memory-mapped communication between the timer and an external processor.
- **Configurable Timer:** Users can load a custom timer value and start/stop the timer through AXI transactions.
- **Interrupt Support:** The timer generates an expired signal when it reaches zero, enabling interruptbased processing.
- **Testbench & Verification:** Includes comprehensive testbenches to verify AXI transactions, timer logic, and overall system functionality.

✅ Implementation Details:

The **AXI4-Lite slave** interface allows **memory-mapped** access to control registers, enabling the software to configure and interact with the timer. The timer logic **decrements the loaded value and asserts an expired signal upon completion**. The design is modeled and verified using **QuestaSim**.

✅ Project Files:

- 📁 ***“axi4_lite_if.v”*** – AXI4-Lite slave interface handling register accesses.
- 📁 ***“timer_logic.v”*** – Core logic of the hardware timer.
- 📁 ***“axi_stream_data_mover.v”*** – Optional AXI Stream module for data movement.
- 📁 Multiple testbenches ***(“tb_axi4_lite_interface.v”***, ***“tb_axi_timer.v***”) to validate functionality.
- 📁 Previous versions: ***(“axi4_lite_interface.v”, “tb_ axi4_lite_interface.v”)*** more simple implementations for practice.

✅ AXI4-Lite Interface Signals (Core Set)

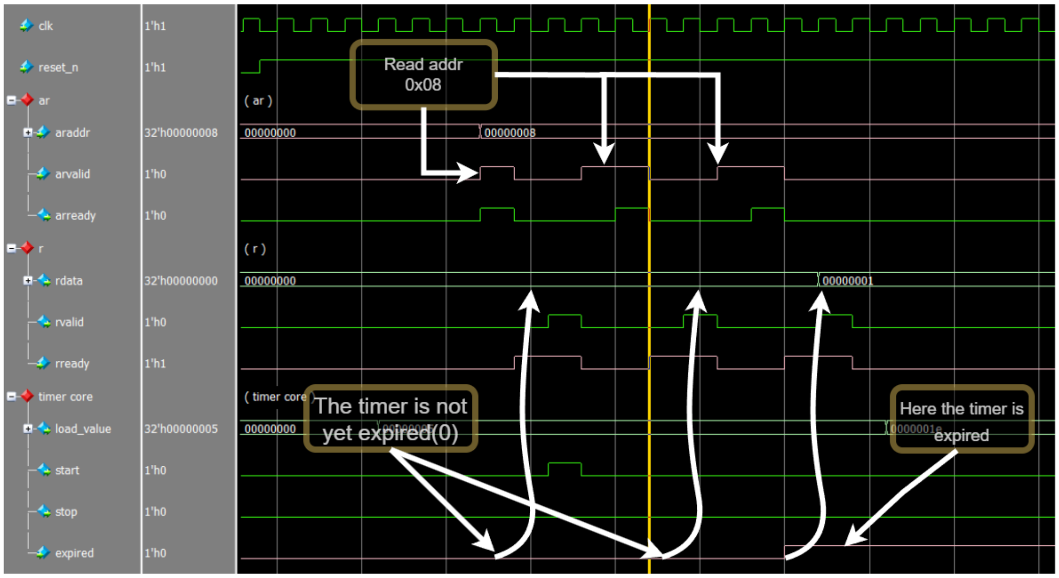
Signal	Description	Direction	Width
awaddr	Write address	Input	32
awvalid	Write address valid	Input	1
awready	Write address ready	Output	1
wdata	Write data	Input	32
wvalid	Write valid	Input	1
wready	Write ready	Output	1
bvalid	Write response valid	Output	1
bready	Write response ready	Input	1
araddr	Read address	Input	32
arvalid	Read address valid	Input	1
arready	Read address ready	Output	1
rdata	Read data	Output	32
rvalid	Read valid	Output	1
rready	Read ready	Input	1

✅ Timer Signals (Core set)

Signal	Notes	Width
clk	input; clock signal	1
reset_n	input; negative edge triggered reset	1
start	input; start the timer control bit	1
stop	input; stop the timer control bit	1
load_value	input; the value to start countdown from	32
expired	output; the timer has reached zero	1

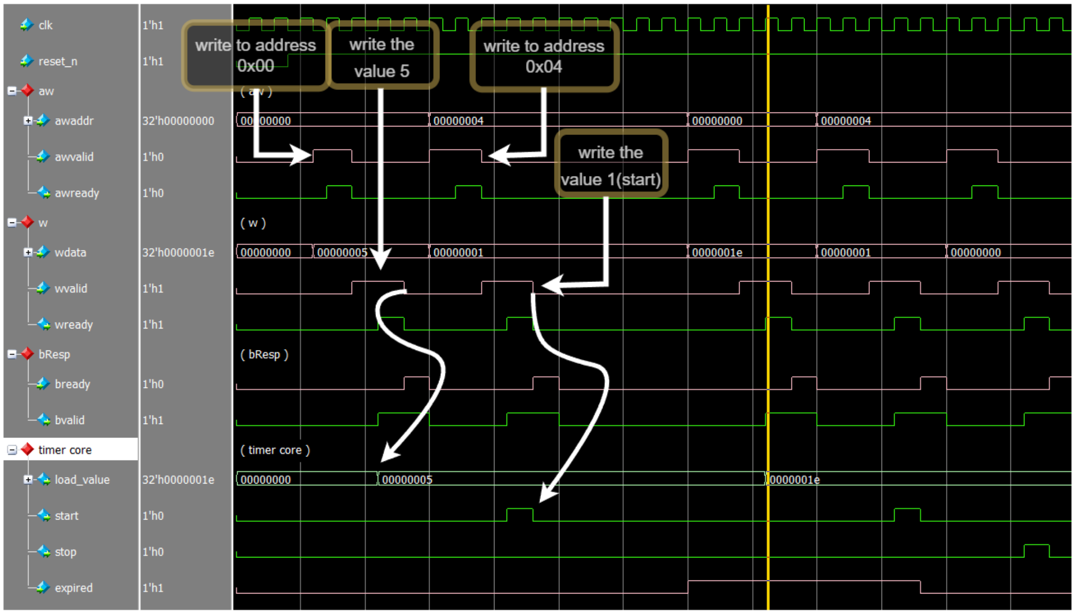
✅ Wave forms:

AXI Read Waveform:



clk 1'h1
reset_n 1'h1
ar
araddr 32'h00000008
arvalid 1'h0
arready 1'h0
r
rdata 32'h00000000
rvalid 1'h0
rready 1'h1
timer core
load_value 32'h00000005
start 1'h0
stop 1'h0
expired 1'h0

AXI Write Waveform:



clk 1'h1
reset_n 1'h1
aw
awaddr 32'h00000000
awvalid 1'h0
awready 1'h0
w
wdata 32'h0000001e
wvalid 1'h1
wready 1'h1
bResp
bready 1'h0
bvalid 1'h1
timer core
load_value 32'h0000001e
start 1'h0
stop 1'h0
expired 1'h1

✅ Future Enhancements:

- Adding an AXI4Stream interface for highspeed timer event streaming.
- Implementing lowpower optimizations for ASIC deployment.
- Extending functionality to support multiple timer instances.

✅ Conclusion:

This project showcases an efficient and modular AXI4-Lite hardware timer implementation, suitable for FPGAbased embedded systems and ASICs. The experience gained in AXI protocol handling, hardware design, and verification is a strong foundation for more advanced SoC designs.