Project Diary: Unveiling the Heart of Timer2

The academic year prior held its usual rhythm of coursework and assessments, a familiar cycle of learning just enough to navigate each hurdle. Yet, a subtle shift occurred this year, a quiet awakening of genuine interest in the realms of electronics and engineering. Gone is the sole focus on grades; in its place, a burgeoning delight in understanding the intricate mechanisms that underpin the technology around us. This newfound drive fuels a deeper engagement with my studies, transforming learning from a task into a joyful exploration. Currently immersed in a microcontroller course, programming in C, and a parallel journey into the world of Verilog, a question sparked within me during our exploration of the ADuC841's Timer2: what lies beneath the surface? How is this crucial peripheral actually built? This curiosity ignited a personal project – to construct a Verilog model of the Timer2. This endeavor began with a step back to the fundamentals, tracing the architecture from its control registers down to the basic building blocks of D flip-flops. My aim is not just to replicate functionality, but to truly understand the underlying hardware and how it translates into a hardware description language. Through this project, I hope to bridge the gap between the abstract world of code and the tangible reality of electronic circuits, deepening my knowledge of both hardware architecture and the art of hardware description

**Project Diary: Getting Curious About What’s Under the Hood**

In the earlier years of my degree, I was mostly focused on doing what I had to do — finishing assignments, passing exams, and moving on. But this year something changed. Last year I took a course on microcontrollers and another on VHDL, and for the first time, I found myself genuinely enjoying the material. I started to really *like* electronics, engineering, and most of all — understanding how things work. Learning stopped being just a task and became something I actually look forward to. I’ve been more curious, more focused, and more motivated to dig deeper. Now, in a microcontroller course where we program in C, and another course focused on Verilog, I started asking more questions. One of them hit me while working with Timer2 on the ADuC841 microcontroller: “How is this built internally?” That single thought kicked off a personal project — to build Timer2 myself, in Verilog. I began by reading up on the basics, trying to understand what registers actually are and how they’re built. That led me into D flip-flops, and from there into a deeper dive into digital architecture and hardware modeling. Along the way, I’ve been learning how to write better code, how to design good testbenches, and how to think both like a programmer *and* like a hardware engineer.

**register**

Stores data – a type of memory.

It serves as a temporary storage area and can be accessed and manipulated quickly.

Composed of multiple flip-flops – capable of storing a single bit.

Registers also contain control logic circuitry such as decoding control signals, performing data manipulation, and using mulitplexers to route data to a specific location within the register.

Size of register: 8-bit = 1-byte = 256, 16-bit = 2-byte ≈ 65K, 32-bit ≈ 4.3B, 64-bit ≈ 18.5Q, is the amount of data the can be handled in a single operation.

**D Flip Flop**

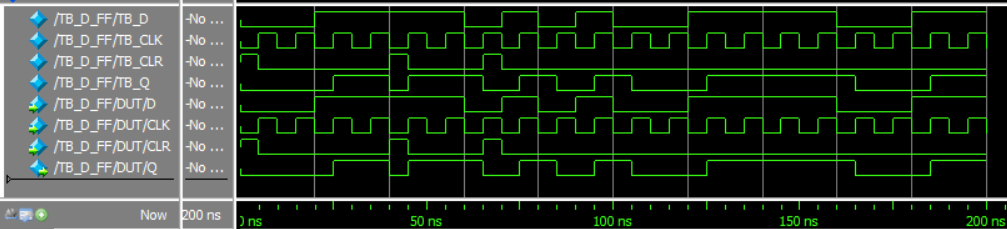
**Assign**: for combinational logic – Continuously driven → wire.

**always** block: for sequential logic – Holds value between clock edges → reg.

posedge clk → Trigger the block **on the rising edge** of the clock signal.

posedge rst → Also trigger the block **on the rising edge** of the reset signal.

Waveform ModelSim:



The test bench "ticks" the clock (CLK) every 10ns, while the data (D) gets a new random value every 10ns. The flip-flop resets (CLR) randomly every 20ns and stays high for 5ns.

We can see that when CLR is high, the output Q is always low. Also, we see that when CLR is low during each rising edge of CLK, if D is high, then Q becomes high as well.