Logisim 4-bit Register

James Riley Dorough

Liberty University

February 15th, 2021

This paper outlines the construction and operation of a simple 4 bit register with a temporary button to mimic the clock system. The register’s inputs, operation, functionality, and outputs will be demonstrated through a grouping of images and external research.

Diagram, schematic

Description automatically generatedDiagram, schematic

Description automatically generated

First are screen caps of the initial environment of the 4-bit register. The first includes the registers without input or initialization from the clock input. The second demonstrates the functionality of the registers not accepting input without the activation of the clock. At this point of the program, the number five is activated on the register’s data inputs. This is accomplished through mimicking base two notation with binary inputs and outputs. As previously noted, since the clock has not yet been activated, the circuit is still uninitialized, and the data will not be written to the registers until the circuit is closed with the clock’s activation. This allows any variation in power to the inputs to not affect the output in any way without the clock being activated.

Diagram, schematic

Description automatically generatedDiagram, schematic

Description automatically generated

Once the clock activates the data streams pass through the NAND gates and any binary one inputs are output through the Q’s path of output. All other inputs with a binary zero value as input is output though the Not Q’s path. At this point of understanding, Not Q can be overlooked as an unimportant value as it has no effect on the result of the designated input. Once the clock activates the inputs are stored in the now initialized registers. Here the binary value of five is written over the uninitialized or zero value output. As seen in the last screenshot, after the clock input deactivates, the binary inputs can be altered without affecting the outputs recorded in the registers. The inputs are depowered back to the binary value of zero and the binary five still resides in the register’s output.

This functionality means little without understanding the purpose and development of registers. Back in the early 80s we were introduced to some of the first improved register inclusive processors such as the 4004 for simple arithmetic intended for calculators and the 8008 microprocessors as the predecessor for our modern PCs (Morse, Ravenel, Mazor, & Pohlman, 1980). By including registers along with the CPUs, we were able to design specific chips, optimized for various tasks. The main benefit of this being that registers can now be addressed by name to feed instructions and data to the CPU as opposed to solely relying on cache memory (Stallings, 2019). While registers bring more complexity to the computer’s design, their versatility is highly important as they can be optimized to decrease memory bandwidth requirements (Goodman & Hsu, 1986). Through this research and experimentation, I have gathered that registers act as a form of variables to store data or instructions to be passed to the CPU. The CPU then followed the provided instructions to forward data and instructions to their respected chips to be further processed and altered. They are then returned to the CPU to the stored back in main memory or reside in short tern storage to be altered yet again.

**References**

Goodman, J. R., & Hsu, W.-C. (1986). ON THE USE OF REGISTERS VS. CACHE TO MINIMIZE MEMORY TRAFFIC. *Computer Sciences Department The University of Wisconsin-Madison*, 376.

Morse, S. P., Ravenel, B. W., Mazor, S., & Pohlman, W. B. (1980). Intel Microprocessors: 8008 to 8086. *IEEE Computer, Vol 13, No. 10*, 42-60.

Stallings, W. (2019). *Computer Organization and Architecture.* Hoboken: Pearson Education, Inc.