Memory

Sunday, January 30, 2022 7:13 PM

"It's a poor sort of memory that only works backwards" - Lewis Carroll (1832-1898)

Memory

The time between a *tick* and its subsequent *tock* is called a cycle.

Cycles are used to regulate operations in a computer (The clock of a computer)

Memory Devices

Hardware supports the use of variables, arrays, and objects by the use of memory. Memory acts as a device that maintains a specific state.

Classic logic is not aware of state or time, you must create the concept and implement it to your desire

Data-flip-flop (DFF) - the fundamental building block from which all memory devices will be built on.

$DFF \rightarrow 1-\text{bit register} \rightarrow \text{n amount of } 1-\text{bit registers (n} - \text{bit register)} \rightarrow RAM\backslash ROM$

Sequential Logic

We must now develop gates that respond to time. Since we remembered x at time $t-1$ we should remember x and time t

We need to be able to create the presence of a *state*

Time Matters

Computations/arithmetic is always delayed because:

- Inputs of the chips don't appear out of thin air; rather, they have to come as outputs of other chips
- The computations take time; the more chip parts the more time (always opt for the most simple solution).

Rather than break time down into infinitesimally small increments we break it down into cycles, i.e. cycle 1, cycle 2, cycle 3, etc.

Synchronizing time in cycles serves two purposes:

1. To neutralize the randomness within communication and calculation
2. Help sync the operations within the computer

Due to this, we can basically assume that when a not gate is inputted with x we can assume that it instantaneously outputs Not(x).

When designing a computer a hardware designer will choose the length of the cycle.

He chooses based on two things:

1. Long enough to contain and neutralize any possible time delays,
2. The shorter the cycle the faster the computer.

There will be a master clock in the computer that controls the timing of all the cycles

Flip Flops

This book uses the form data Flip Flop or DFF but there are many forms of Flip Flops

$out(t) = in(t-1)$

Gates in a DFF:

```
Chip name: DFF
Input:     in
Output:    out
Function:  out(t+1)=in(t)
           inputs will be outputted the next cycle
```

Combinational and Sequential Logic

Time-independent chips - are chips that are designed to only work under the computers (circuits) present state.

Sequential or *clocked chips* - are chips that are designed to interact with the computer's (circuit's) internal clock and change throughout states.

Specification

Memory chips that are typically used within a computer's architecture

- Data flip=flops
- Register (based on the DFF)
- RAM devices (based on the registers)
- Counters (based on the registers)

Registers

```
Chip name: Bit (1-bit register)
Input:     in, load
Output:    out
Function:  if load(t) then out(t+1) =  in(t)
           else            out(t+1) = out(t)
```

```
Chip name: Register (16-bit register)
Input:     in[16], load
Output:    out[16]
Function:  if load(t) then out(t+1) =  in(t)
           else            out(t+1) = out(t)
```

Random Access Memory (direct-access memory)

An assortment of n registers. By using the indexing method of $[0, n-1]$ you can select a specific register and read from or write to it.

The important thing to realize that the time to access a register is independent of its size or register index. It is all instantaneous.

```
Chip name: Ram_n
Input:     in[16], load, address[k] (k=log_2(n))
Output:    out[16]
Function:
Out emits the value stored at the memory location
(register) specified by address. If Load==1, then the
Memory location specified by address is set to the
Value of in. The loaded value will be emitted by
Out from the next time step onward.
```

Counter

```
Chip name: PC
Input:     in[16], load, inc, reset
Output:    out[16]
Function:  16-bit counter

If reset(t)      out(t+1) = 0
Else if load(t) out(t+1) = in(t)
Else if inc(t)  out(t+1) = out(t) + 1
Else             out(t+1) = out(t)
```