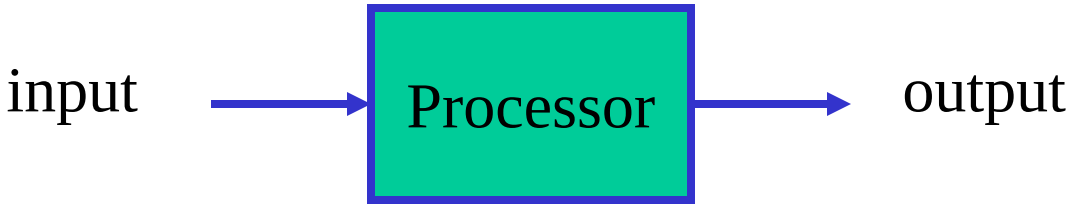
Final Exam

- Friday, December 17, 1:30 – 3:30 pm
- Bring
  - Pencil(s) and eraser
  - One page note sheet (both sides)
- Cumulative but with an emphasis on material since the midterm
- Covers material from
  - Assigned readings
  - Lectures
  - Homework
  - Labs

What Is A *Processor*?

- Something that processes!



Example applications:

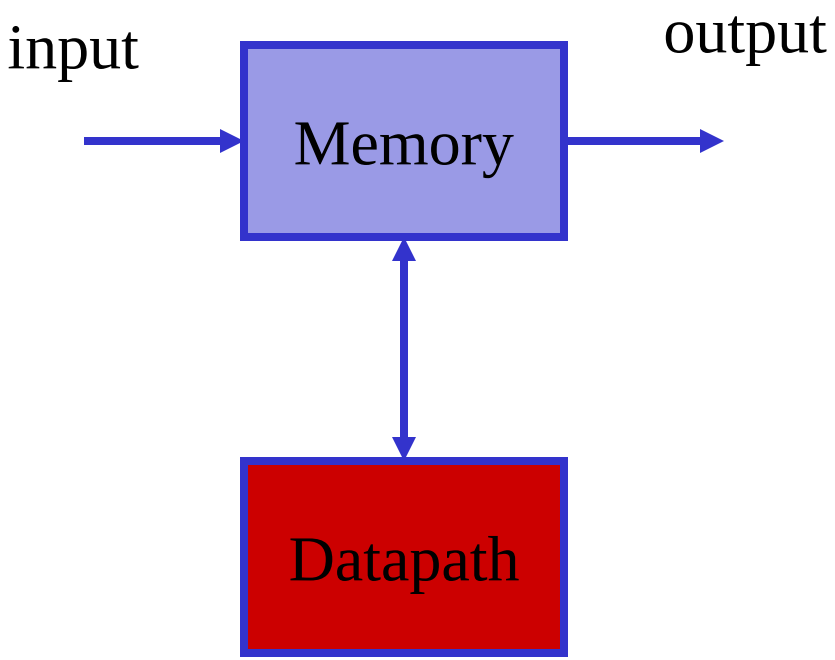
- Microwave oven controller
- CD or DVD player
- Hearing aid
- Personal computer
- Digital watch
- Cell phone
- Radar for an airplane
- Wireless network processor

What Is Inside One?

- Three main components
  - Memory

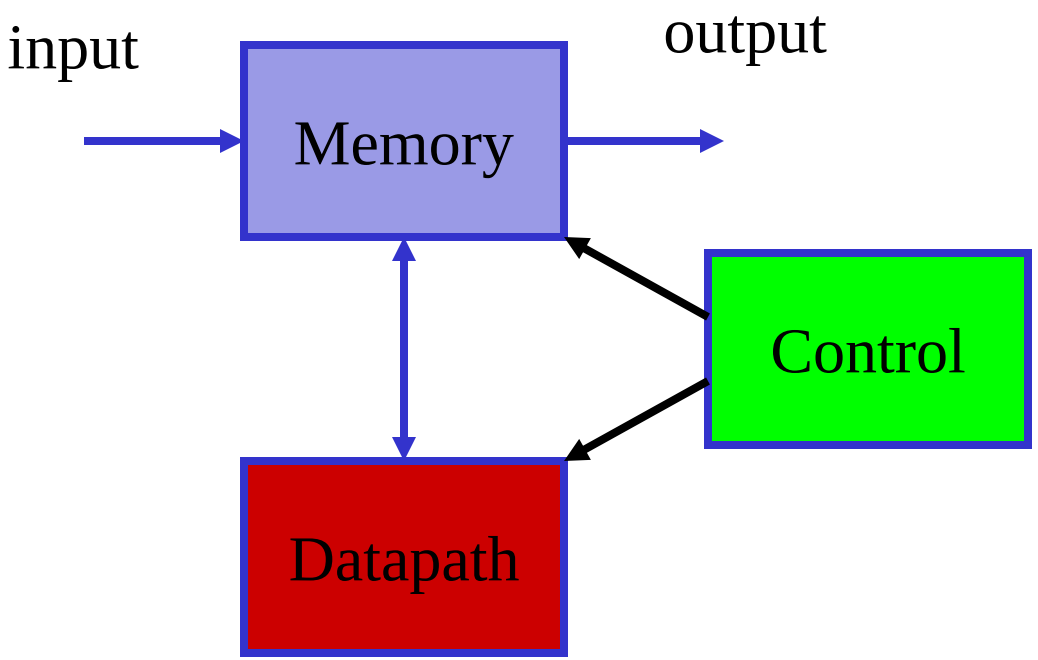
What Is Inside One?

- Three main components
  - Memory
  - Datapath



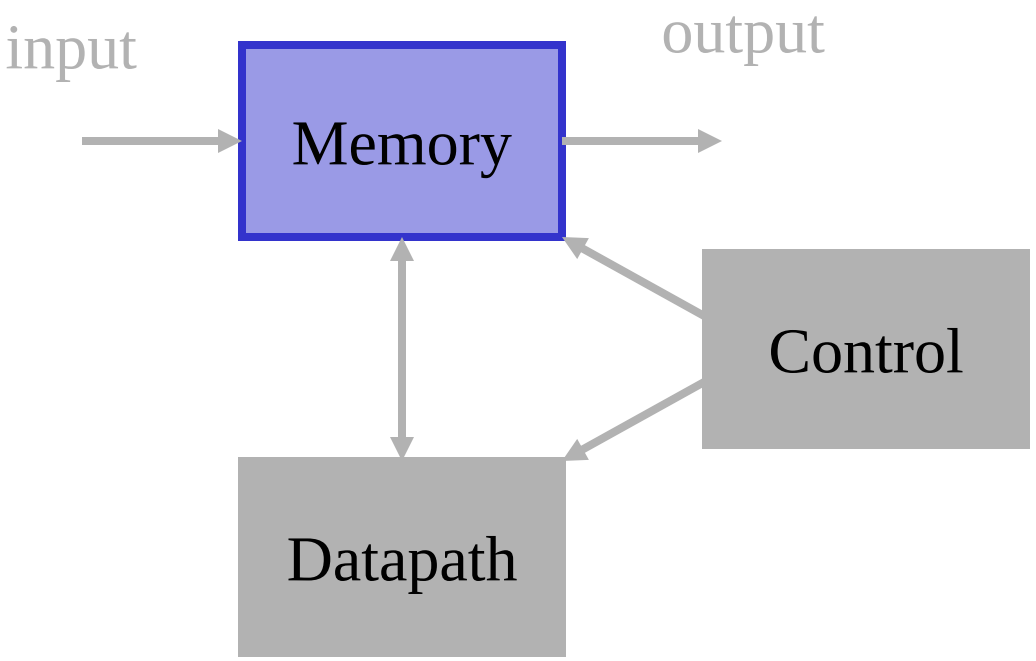
What Is Inside One?

- Three main components
  - Memory
  - Datapath
  - Control



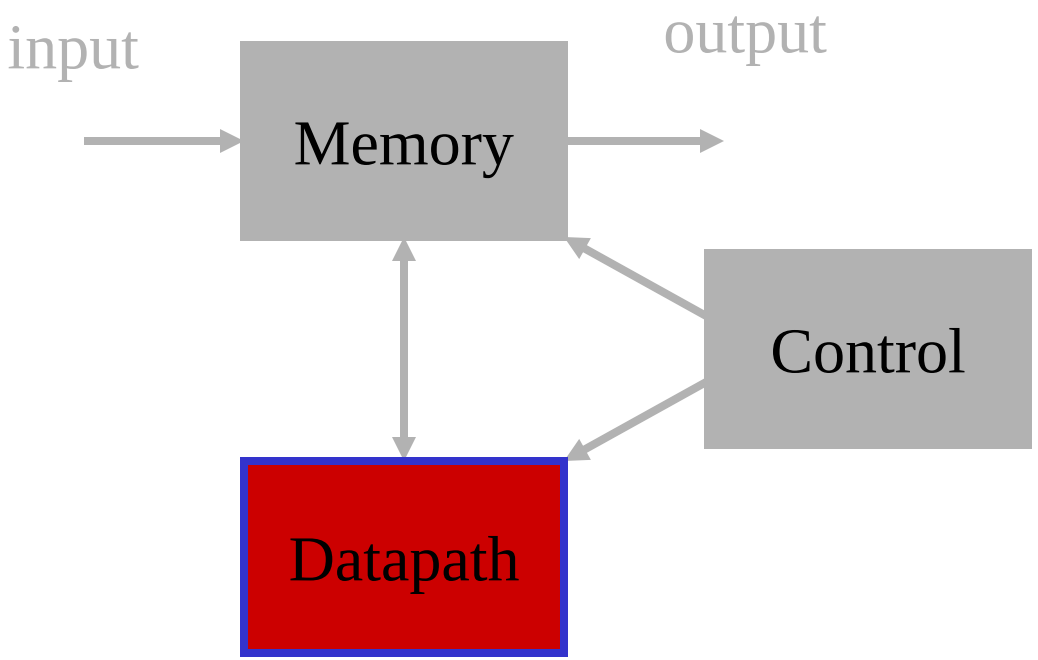
Memory

- Stores information
- Examples
  - Temporary chip memory
    - "256 MB RAM"
  - Permanent chip memory
    - "Flash non-volatile memory"
  - Hard disk
    - "80 GB disk"



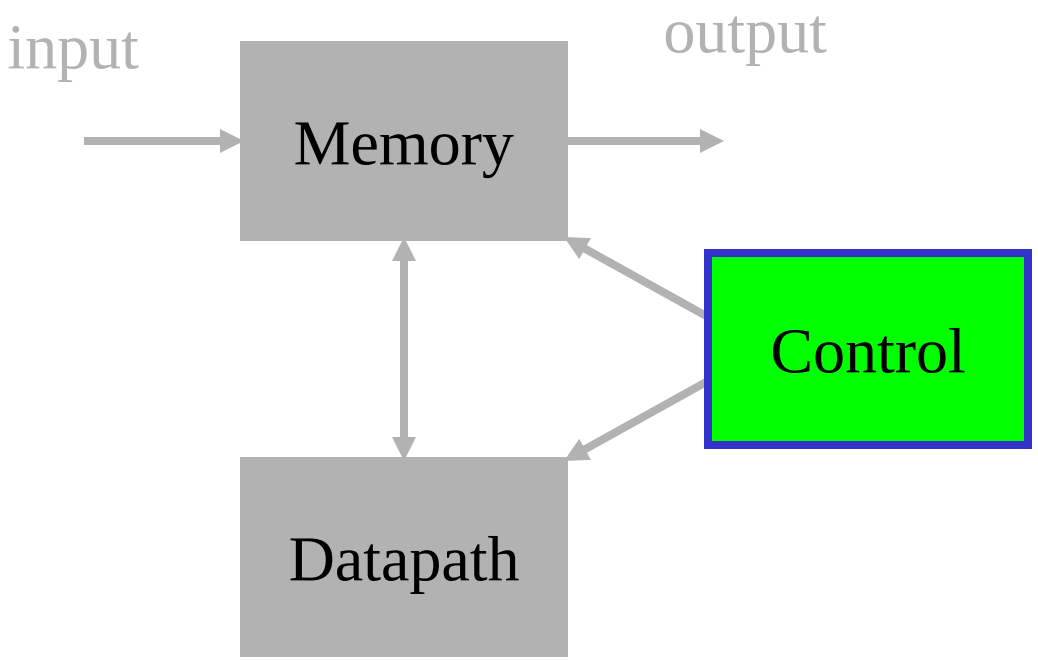
Datapath

- Processes information
- Example tasks
  - Add
  - Multiply
  - Move
  - Compare



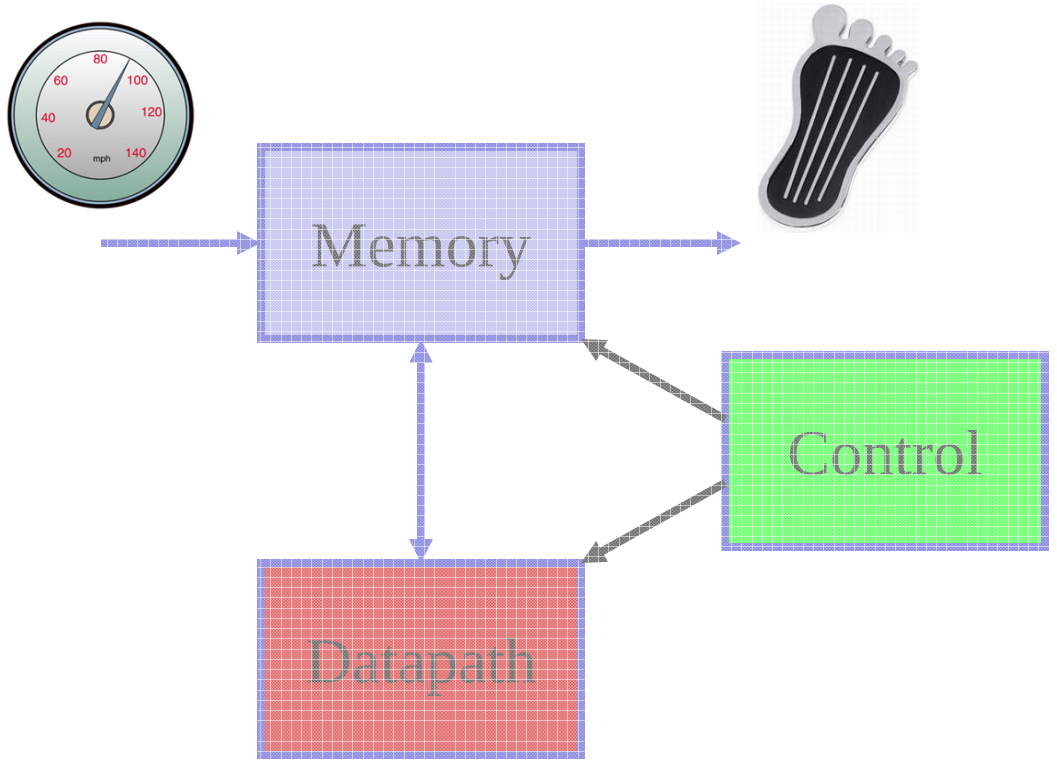
Control

- Runs the show
- Examples
  - Software
    - Word, Excel
    - Large database
  - Hardware (best for simpler, fixed, high-speed tasks)
    - MP3 player
    - Signal processing



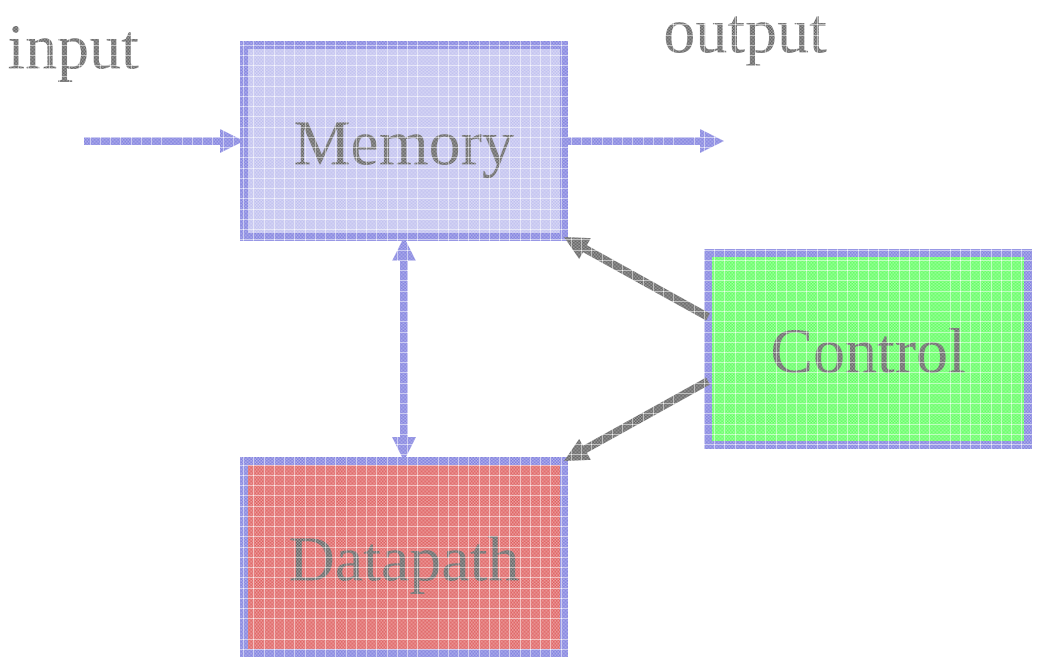
Simple Blocks Working Together Example #1

- Auto cruise control
  - Watch car's speed (*input*)
  - Compare with set speed (*datapath, compare*)
  - If adjustment is needed, add or subtract (*datapath*) from setting and send update to gas pedal (*output*)



Simple Blocks Working Together

- Key point
  - Simple operations alone can not do much, but billions of simple operations per second can do a lot!



Verilog

- Hardware Description Language
- Design process
  - Think
  - Draw diagram of hardware, figure out where logic and registers go
  - Think
  - Then…
    - Write verilog
    - Test it

Verilog

- You'll write far better verilog if you think of it differently than a standard programming language
  - a way to code an *algorithm*
- Hardware description language
  - a way to code *hardware*!

HDL Simulation Tools

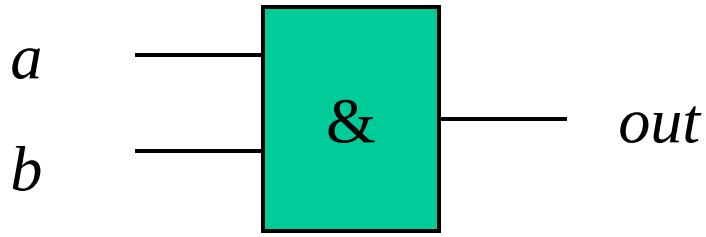
- Synopsys
  - VCS – simulator
  - Virsim – waveform viewer and environment
- Cadence
  - Verilog XL – simulator
  - NC Verilog – simulator
  - Simvision – waveform viewer and environment
- Many others…

Verilog

- *Modules* are basic building blocks
- Main ways to do logic
  - *wires, assign* statements
  - *registers, always* blocks

wire, assign

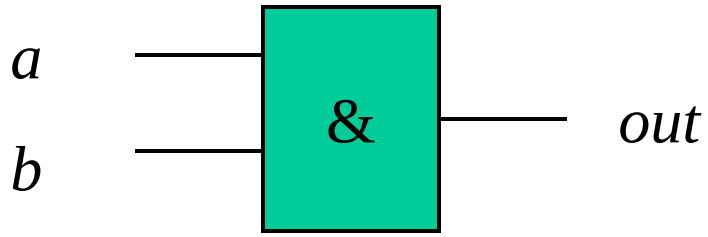
- Picture “always active” hardwired logic
- Declare all wires



wire, assign

- Example:

```
wire out;
assign out = a & b;
```



reg, always

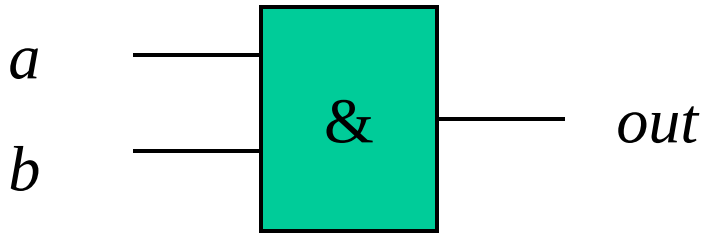
- Picture a much more general way of assigning "wires" or "nodes"
- Use "if/else" and "case" statements
- Can, but don't use "for loops" in logic blocks (ok for testing code)
- Procedural execution – statements execute in order

reg, always

- Example:

```
reg out;
always @(a or b) begin
   out = a & b;
   end
```

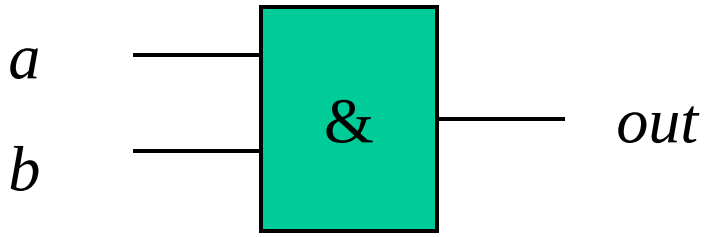
"Sensitivity list" declares when to execute the code within the *always* block.



reg, always

- Example:

```
reg out;
always @(a or b) begin
   out = a & b;
   end
```



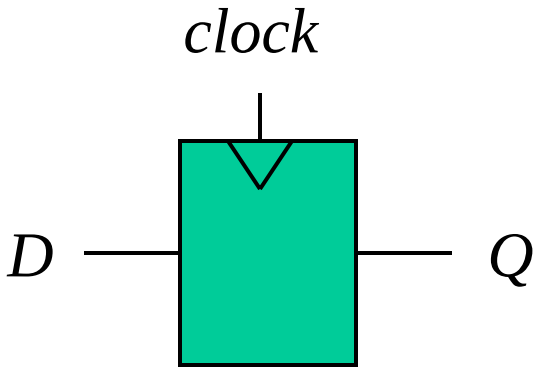
Operation goes like this:
Anytime *a* or *b* changes, re-evaluate *out = a & b;* otherwise, keep *out* the same.
Notice that there is the possibility of a problem (memory) if all inputs are not included in the sensitivity list.

reg, always

- Building a flip-flop

```
reg Q;
always @(posedge clock) begin
  Q = D;
end
```

- The only time the flip-flop output *Q* gets updated is at the positive edge of the clock signal -> exactly what we want for a flip-flop.



Concurrent Operation

- You can think of these as operating on independent circuits (remember *hardware* orientation).

```
always begin
   a = (b&c) | d;
   #5;                  // 5-unit delay
   end

always begin
   f = ~(g ^ h);
   #7;                  // 7-unit delay
   end
```

What Next?

- Computer Architecture (up the system hierarchy)
  - Uniprocessor, DSP, Multiprocessor design
  - Networking
- Circuit Design (down the system hierarchy)
  - Transistor circuits for logic gates, memories, PLAs
  - VLSI design: layout and integrated circuit fabrication
- CAD Tools
  - Automatic synthesis of logic, state machines, entire chips
  - Verification of digital designs
- Tangents
  - Quantum computing
  - Biological computing
  - Biology (logic gate models of genetics)
  - Artificial life (cellular automata)