Chapter 3: Processor Implementation

What is involved in Processor Implementation?

There are mainly 2 aspects

1. organization of the electrical components (ALUs, buses, registers, etc.) commensurate with the expected price/performance characteristic of the processor
2. thermal and mechanical issues including cooling and physical geometry for placement of the processor in a printed circuit board (often referred to as motherboard)

Key Hardware Concepts

Circuits

Combinational Logic: There is no concept of a state (i.e. there is no feedback from the output of the circuit to input)

Sequential Logic: output of such a circuit is a Boolean combination of the current inputs to the circuit and the current state of the circuit

Hardware Resources of the Datapath

- **Memory** stores the instructions and operands
- **ALU** to perform arithmetic and logic operations
- **Register File** to store values (many instructions use them)
- **Program Counter** or PC used to point to current instruction and for implementing branch/jump
- **Instruction Register** to hold the instruction when it is brought up from memory

Edge-triggered logic - change happens on the *rising* or *falling* edge of the clock
- Rising edge = positive edge triggered
- Falling edge = negative edge triggered

Level-triggered logic - change happens when the clock is on high

Memory is special, and for the purpose of this discussion, we shall say memory is not edge-triggered

To read

- Supply address and the read signal to memory
- After a finite amount of time (called the read access time of the memory) the contents of the particular address is available on the "Data out" lines

To write

- Supply address, data in, and write signal
- After a finite amount of time (the write access time) the particular memory location would have the value supplied via “Data in”

Connecting Datapath Elements

1. We need to use **PC** to specify to the **memory** what location contains the instruction
2. Once the instruction is read from the **memory** then it has to be stored in the **IR**
3. Once the instruction is available in **IR**, then we can use register numbers specified in the instruction to read from the appropriate registers in the **register file**, perform the operation using the **ALU**, and write into the appropriate register in the register file

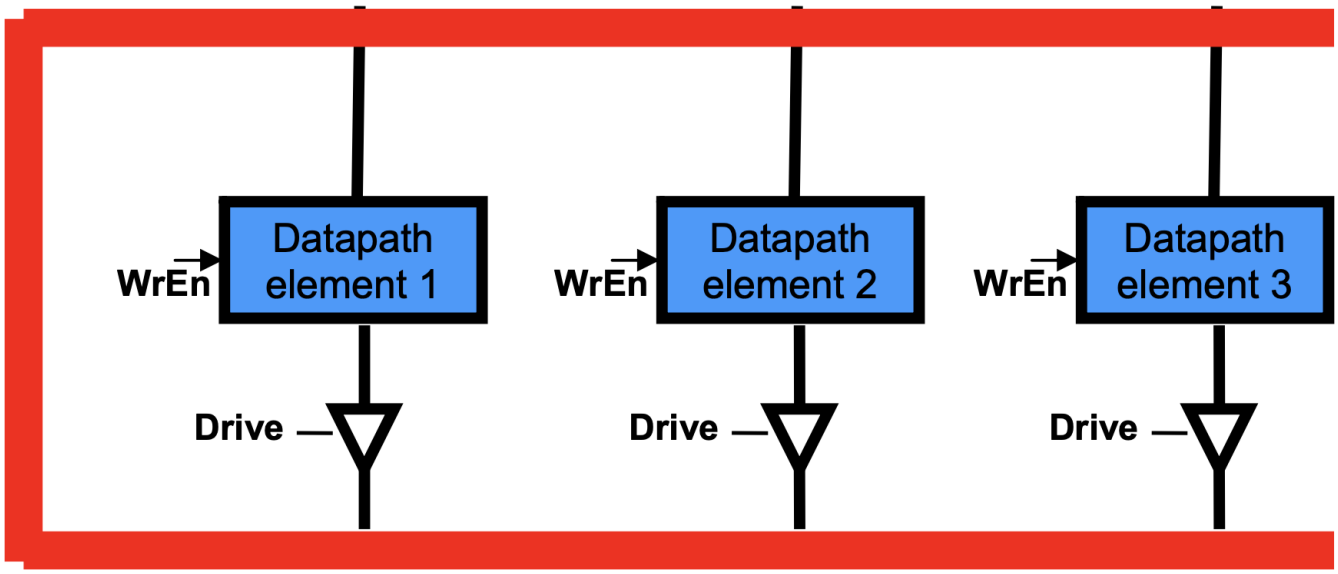
Determining the clock cycle time

- Time that has to elapse for the output of PC to be stable for reading
- Wire delay for the address to propagate from the output of PC to the Addr input of the memory
- Access time of the memory to be read
- Wire delay for the value read from the memory to propagate to the input of the IR
- Time that has to elapse for the input of IR to be stable before the second rising edge, usually called the setup time
- time that the input of IR has to be stable after the second rising edge, usually called the hold time

The width of the clock should be greater than the sum of all delays

Single Bus Design

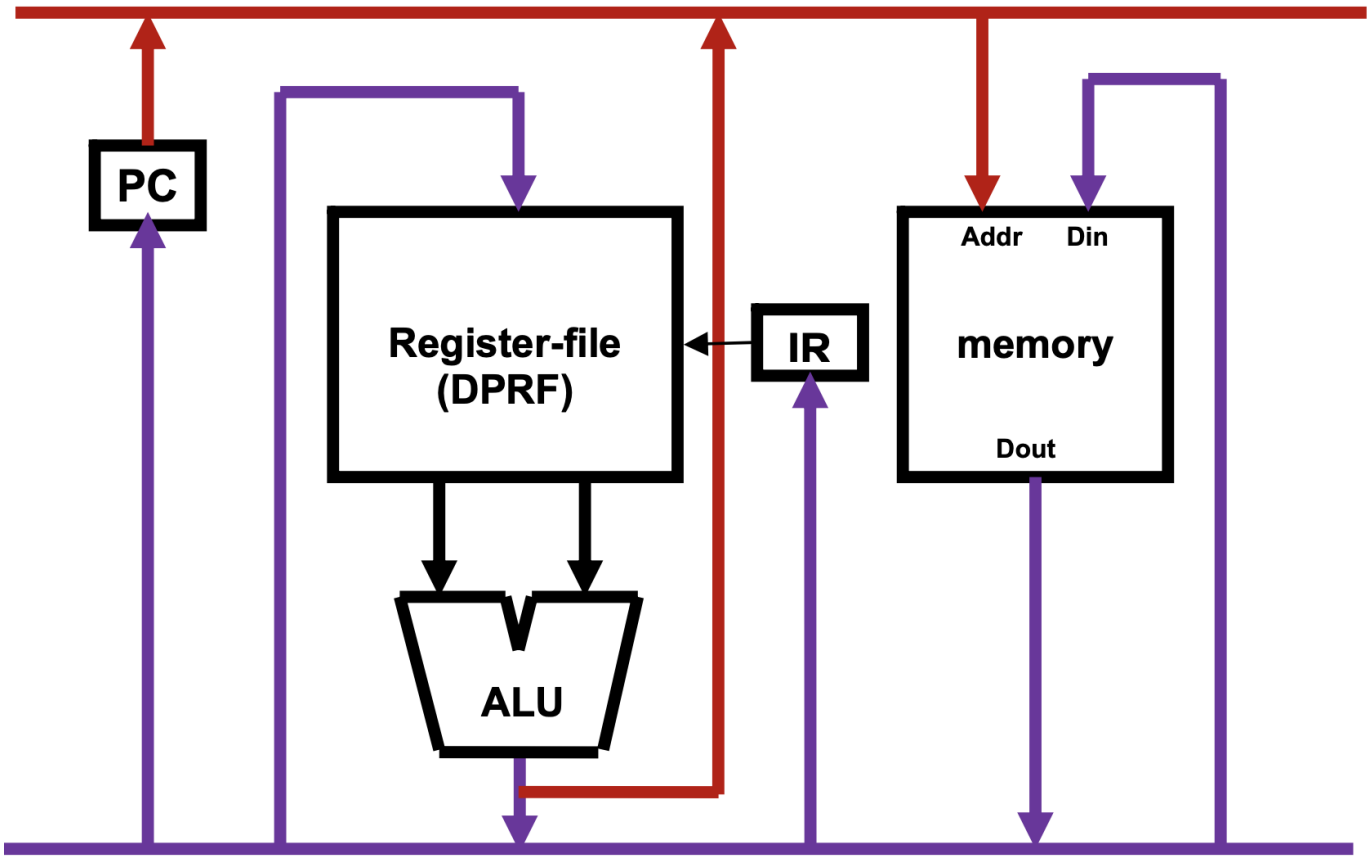
A single bus design is analogous to group meetings: one person talks and everyone else listens



The bus denotes the set of wires shared

- To connect each element to the bus, we have to toggle the drivers (Tristate Buffers)

Two Bus Design



- Generally, the red wires carries address values and the purple wires carry data values
- While not shown, it's implied that there is a bus whenever connecting to a bus

- Faster than a single bus design

Finite State Machine

A Finite State Machine (FSM) is an abstraction for any sequential logic circuit

- State corresponds to a physical state
- Transition is any external input that triggers state change in the actual circuit or any output control signals generated in the actual circuit during the state transitoin

Datapath Design

- With only one bus, we have exactly one channel of communication between any pair of datapath elements
  - Have to have A and B registers in front of the ALU
  - Have to have Memory Address Register (MAR)

ISA and Datapath Width

Since we defined the LC-2200 to be a 32-bit ISA, all instructions, addresses, and data operands are 32-bits in width

- We can conceivably create lower precision parts (e.g. 8 bit bus?)
  - Price - performance tradeoff

CPU

- CPU contains the FSM, which means we need to know what state we are in
- Macro state = FETCH, DECODE, EXECUTE
- Micro state = 1 clock cycle (ifetch1, ifetch2, ifetch3)
- CPU has to transition from one state to another

The way the CPU works is

1. State register names the state the processor is in for the clock cycle
2. ROM accessed to retrieve contents at the address pointed to by the state register
3. Output of the ROM is the current set of control signals to be passed on to the datapath
4. Datapath carries out the functions as dictated by these control signals in this clock cycle
5. The next state field of the ROM feeds the state register so that it can transition to the next state at the beginning of the next clock

Microprogrammed vs Hardwired

Control Regime	Pros	Cons	Comment	When to use	Examples
Microprogrammed	Simplicity, maintainability, flexibility Rapid prototyping	Potential for space and time inefficiency	Space inefficiency may be mitigated with vertical microcode Time inefficiency may be mitigated with prefetching	For complex instructions, and for quick non-pipelined prototyping of architectures	PDP 11 series, IBM 360 and 370 series, Motorola 68000, complex instructions in Intel x86 architecture
Hardwired	Amenable for pipelined implementation Potential for higher performance	Potentially harder to change the design Longer design time	Maintainability can be increased with the use of structured hardware such as PLAs and FPGAs	For High performance pipelined implementation of architectures	Most modern processors including Intel Xeon series, IBM PowerPC, MIPS