Introduction to Structured VLSI Design

- Digital systems and recap

Joachim Rodrigues

Overview

- **Why digital?**
- **Some Applications**
- **Device Technologies**
- **Digital Components**
- **Timing**
- **DFF, Latches**
- **Registers**

Advantages

- Advantage of digital devices
  - Reproducibility of information
  - Flexibility and functionality: easier to store, transmit and manipulate information
  - Economy: cheaper device and easier to design
- Moore's law
  - Transistor geometry
  - Chips double density (number of transistor) every 18 months
  - Devices become smaller, faster and cheaper
  - Now a chip consists of hundreds of million gates
  - And we can have a "wireless-PDA-MP3-player-camera-GPS-cell-phone" gadget very soon (statement after 2005)

Applications of digital systems

- "Digitization" has spread to a wide range of applications, including information (computers), telecommunications, control systems etc.
- Digital circuitry replaces many analog systems:
  - Audio recording: from tape to music CD to MP3 (MPEG Layer 3) player
  - Image processing: from silver-halide film to digital camera
  - Telephone switching networks
  - Combustion control in car engines

Challenge

Implement the best HW realization. Best??

Different applications, different demands...
Thus, "just good enough" is the best in engineering.

Try to find a balance between effort and cost!

Cost?

Flexibilty
Complexity

- **Processors**
- **FPGAs**

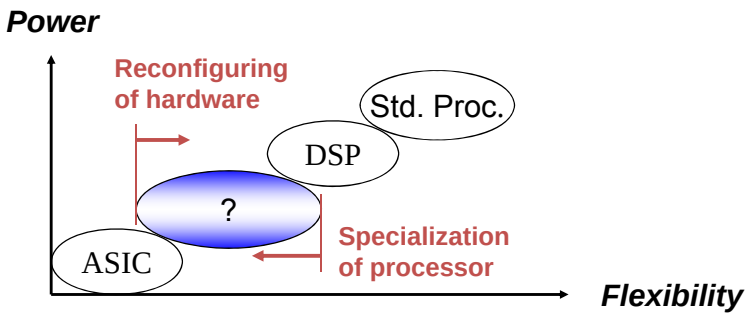
Low power
Low cost
Flexibilty

- **Processors**
- **Dedicated HW**

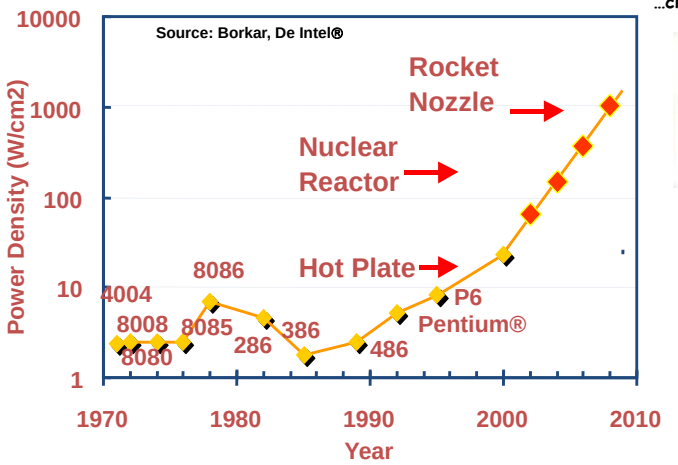
Lower power
Lower cost

- **Dedicated HW**
- **Processors**

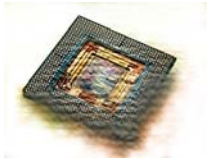
Application Domain Specific Computing



Chip power density



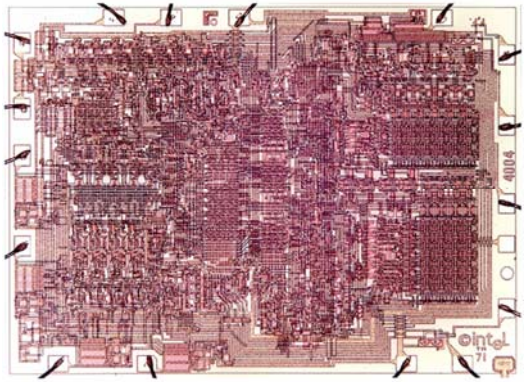
...chips might become hot...



Heat sinking is a challenge in cellular devices

Iron: ca 10 W/cm^2.

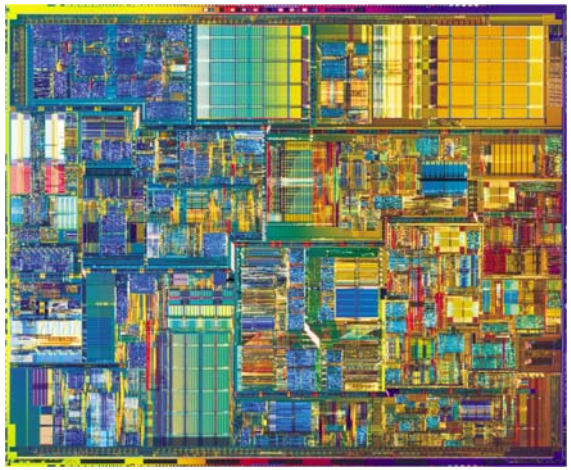
Intel 4004:1972



- First micro-processor on a single chip
- 2300 transistors
- 0.3 mm x 0.4 mm
- 4 bit words
- Clock: 0.108 MHz

You will have the possibility to design a more powerful processor in one of our courses

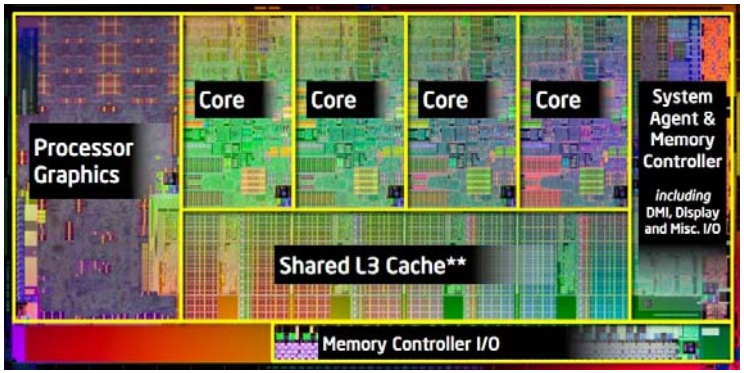
Intel Pentium 4



- 42 000 000 transistors.
- 0.18 micron CMOS
- Clock: 1.5 GHz
- Die: 20 mm²

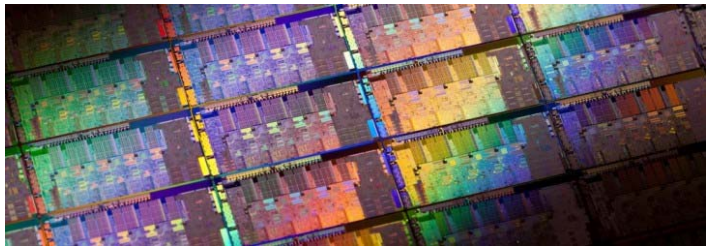
Baseband ASIC of a modern mobile phone has easily 10 times more transistors.

SandyBridge

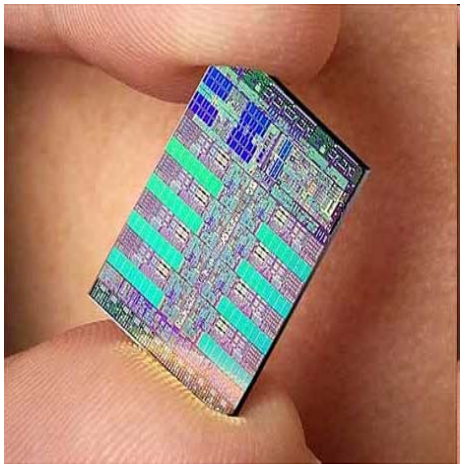


- 32 nm – 64 bit
- 4 995 000 000 Transistors
- ~3.5 GHz
- 216 mm² (10x Pentium 4)

SandyBrigge on a wafer



Typical well know processor

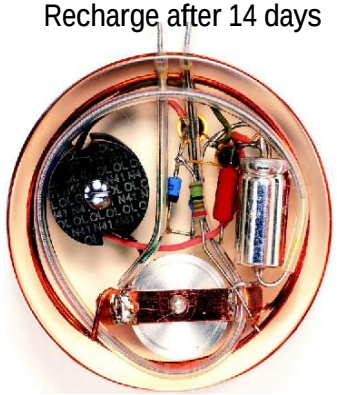


- 90 nm process
- 8 processors

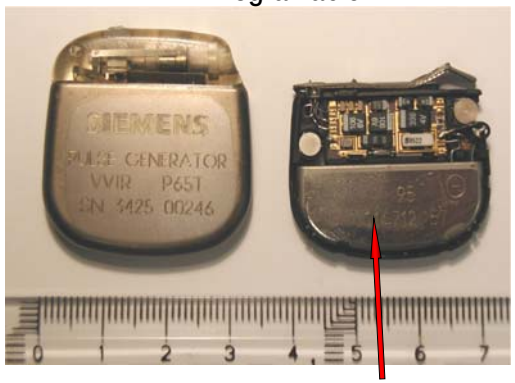
Playstation

Cardiac Pacemaker

1958
Recharge after 14 days



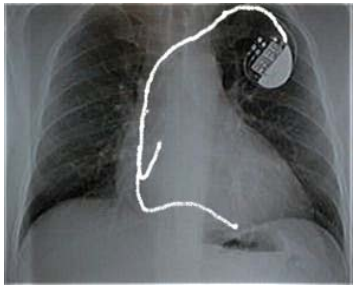
2000
Replacement after 7-20 years
Programable



Battery

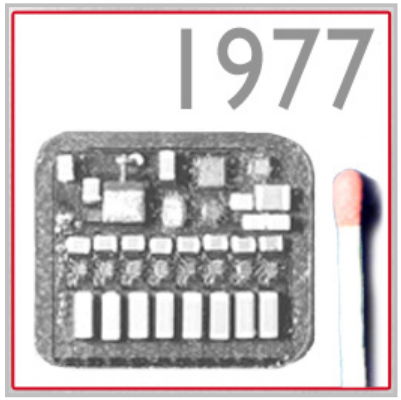
Chest X-Rays

Dual chamber pacemaker



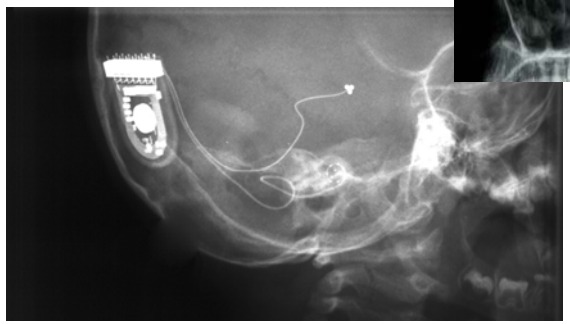
Single chamber pacemaker

Bionic Ear



MED EL

Skull X-Rays



MED EL

How to implement a digital system

- No two applications are identical and every one needs certain amount of customization
- Basic methods for customization
  - "General-purpose hardware" with custom software
    - General purpose processor: e.g., performance-oriented processor (e.g., Pentium), cost-oriented processor (e.g., PIC micro-controller)
    - Special purpose processor: with architecture to perform a specific set of functions: e.g., DSP processor (to do multiplication-addition), network processor (to do buffering and routing), "graphic engine" (to do 3D rendering)

Digital sytems

  - Custom hardware
  - Custom software on a custom processor (known as hardware-software co-design)
- Trade-off between Programmability, Coverage, Cost, Performance, and Power consumption
- A complex application contains many different tasks and use more than one customization methods

Classification of device technologies

- Classification:
  - Full-custom ASIC
  - **Standard cell ASIC**
  - Gate array ASIC (80's)
  - **Complex field programmable logic device (FPGA)**
  - Simple field programmable logic device
  - Off-the-shelf SSI (Small Scaled IC)/MSI (Medium Scaled IC) components

Standard-Cell ASIC

- Circuit made of a set of pre-defined logic, known as standard cells
- E.g., basic logic gates, 1-bit adder, D FF etc
- Layout of a cell is pre-determined, but layout of the complete circuit is customized
- Masks needed for all layers

- Processors are build as standard cell ASICS

Complex Field Programmable Device

- Device consists of an array of generic logic cells and general interconnect structure
- Logic cells and interconnect can be “programmed” by utilizing “semiconductor fuses or “switches”
- Customization is done “in the field”
- Two categories:
  - CPLD (Complex Programmable Logic Device)
  - FPGA (Field Programmable Gate Array) **will be used in the Lab**
- No custom mask needed

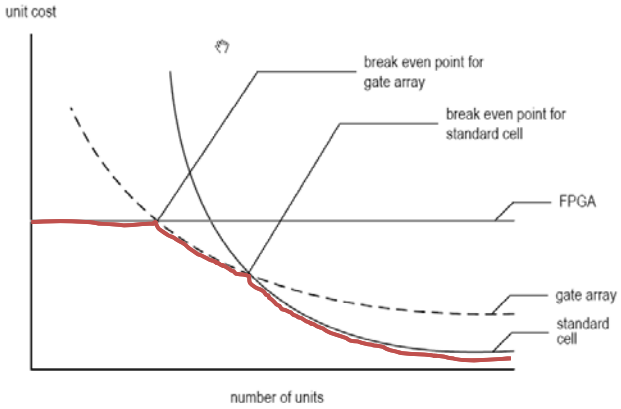
Comparison of technology

- Area (Size): silicon “real-estate”
  - Standard cell is the smallest since the cells and interconnect are customized
  - FPGA is the largest
    - Overhead for “programmability”
    - Capacity cannot be completely utilized
- Speed (Performance)
  - Time required to perform a task
- Power
- Cost

Cost

- Types of cost:
  - NRE (Non-Recurrent Engineering) cost: one-time, per-design cost
  - Part cost: per-unit cost
  - Time-to-market “cost” loss of revenue
- Standard cell: high NRE, small part cost and large lead time (up to several years)
- FPGA: low NRE, large part cost and small lead time

Graph of per-unit cost



$$C_{per\_unit} = C_{per\_part} + \frac{C_{nre}}{\text{units produced}}$$

Summary of technology

	FPGA	Gate array	Standard cell
tailored masks	0	3 to 5	15 or more
area			best (smallest)
speed			best (fastest)
power			best (minimal)
NRE cost	best (smallest)		
per part cost			best (smallest)
design cost	best (easiest)		
time to market	best (shortest)		
per unit cost		depend on volume	

- Trade-off between optimal use of hardware resource and design effort/cost
- No single best technology
- Application (medical, mobile, …)

Recap

Following slides should fresh up your memory

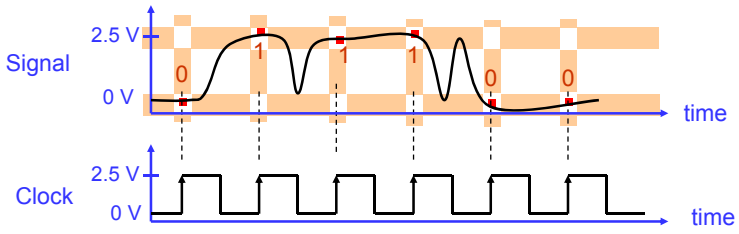
”Digital” is an Abstraction

Only if we guarantee to meet the timing requirements

... do the components guarantee to behave as intended.

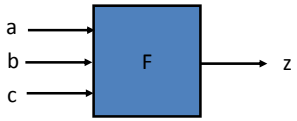
Digital Circuits

- Binary numbers: ”0” and ”1” or ”False” and ”True”
- Represented by voltages:
  - ”0” / ”False” is 0 V
  - ”1” / ”True” is e.g. 2.5 V (much lower (<1V) in newer technologies)
- Digital is an abstraction
  - Discrete values: 0 or 1
  - Discrete time (clock defines when valid):



Two Basic Digital Components

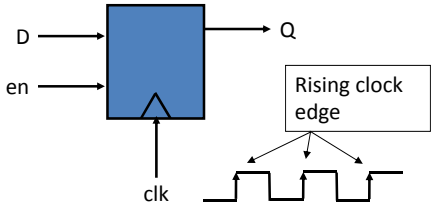
Combinatorial Logic



Always:
z <= F(a, b, c);

i.e. a function that is always evaluated when an input changes.
Can be expressed by a truth table.

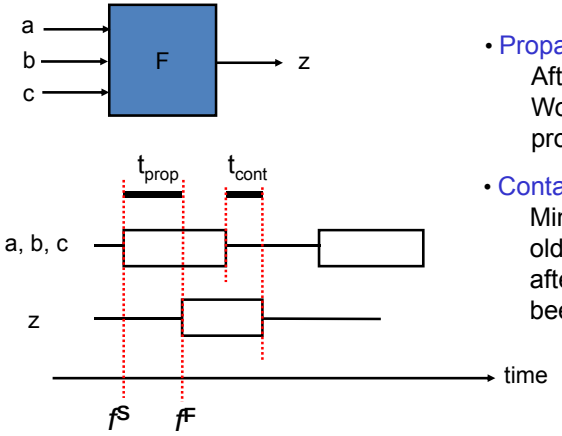
Register



if clk'event and clk='1' then
  if en='1' then
    Q <= D;

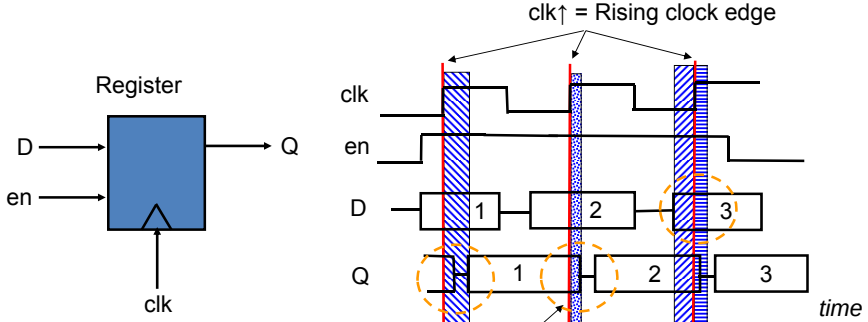
i.e. a stored variable,
Edge triggered D Flip-Flop with enable.

Combinatorial Logic Timing



- Propagation delay:
  After presenting new inputs Worst case delay before producing correct output
- Contamination delay:
  Minimum guaranteed time old output remains valid after new inputs have been provided.

Register timing



Propagation delay (clk_to_Q):
Worst case (maximum) delay **after** clk↑ before new **output** data is valid on Q.

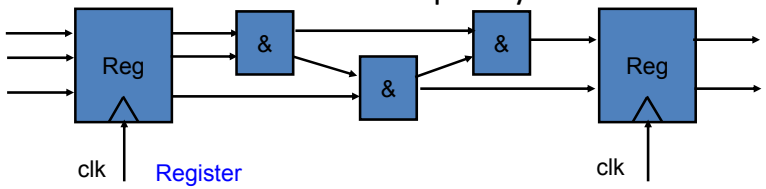
Contamination delay (cont_clk_to_Q):
Minimum guaranteed time old **output** remains valid **after** clk↑

Setup time:
Minimum time **input** must be stable **before** clk↑

Hold time:
Minimum time **input** must be stable **after** clk↑

A small exercise/problem

- What is the maximum clock frequency?



Register

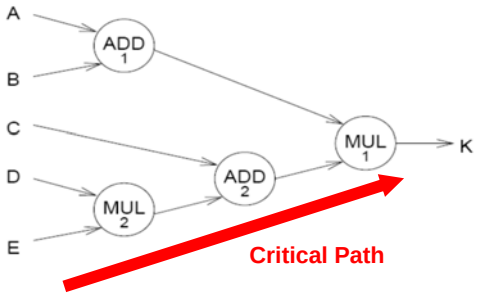
Propagation delay:	Tckl-Q	250ps
Contamination delay:	Tcont-clk-Q	150ps
Setup time:	Tsu	200ps
Hold time:	Th	100ps

AND-gate

Propagation delay:	Tprop	250ps
Contamination delay:	Tcont	150ps

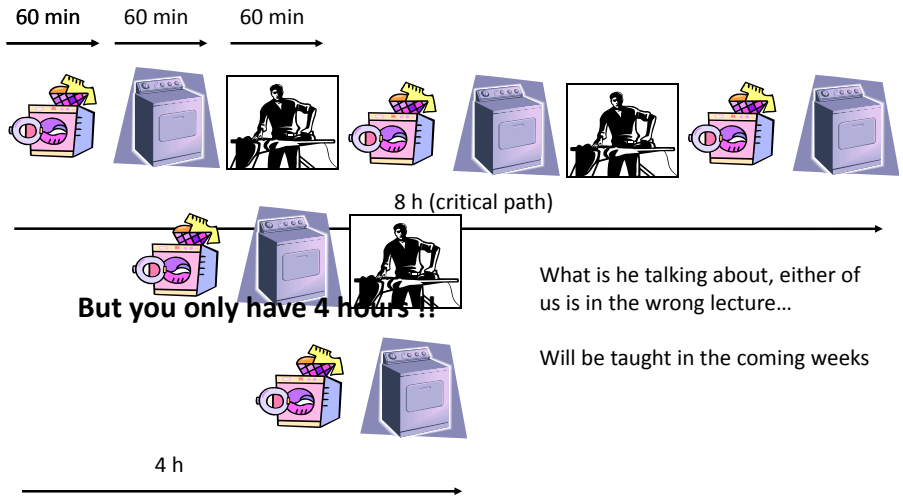
Critical path

- …begin to explore the construction of digital systems with complex behavior
  - Example: K = (A +$_1$ B) *$_1$ (C +$_2$ D *$_2$ E)
- Combinatorial circuit:



- Fast (low latency)
- Not flexible
- Expensive
- Latency of MUL?
- Throughput?

In the Laundry Processing Plant

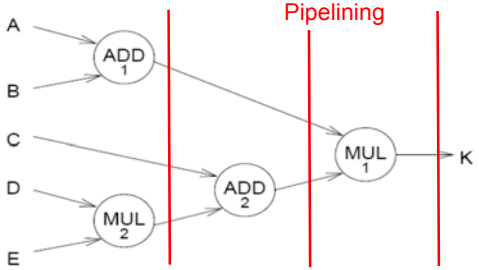


What is he talking about, either of us is in the wrong lecture…

Will be taught in the coming weeks

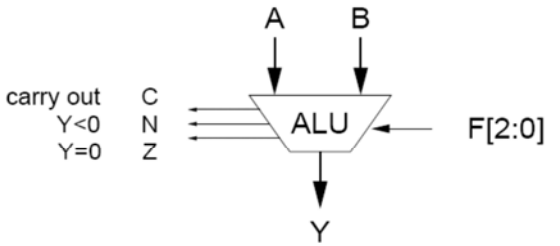
Pipelining

- …begin to explore the construction of digital systems with complex behavior
  - Example: K = (A +$_1$ B) *$_1$ (C +$_2$ D *$_2$ E)
- Combinational circuit:



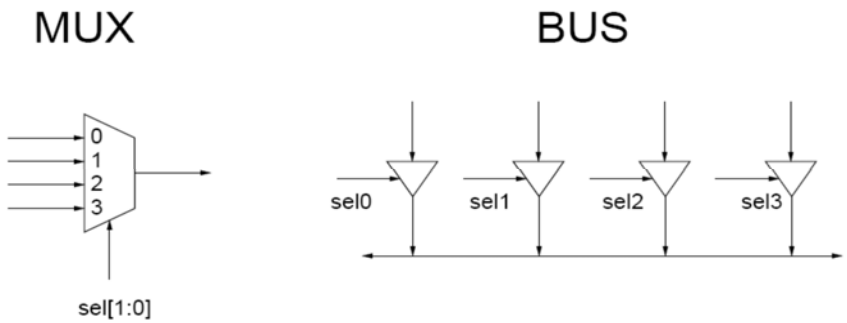
- Fast (low latency)
- Not flexible
- Expensive
- Latency of MUL?
- Throughput?

Combinational Logic



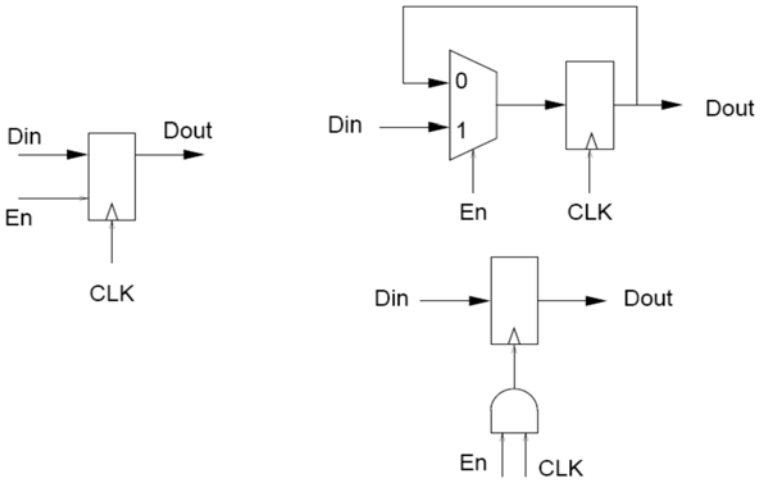
F	Y
000	A + B
001	A - B
010	A - 1
011	A and B
100	A or B
101	A * B
110	etc.
111	

Selection



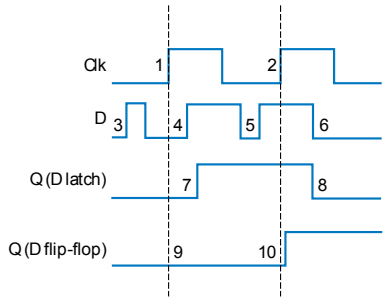
Shared bus with multiple drivers, only one driver may drive the bus, the others must be "disconnected" (high impedance / 'Z')

Registers (Flip-Flop's)



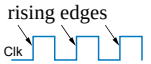
D Latch vs. D Flip-Flop

- Latch is level-sensitive: Stores D when C=1
- Flip-flop is edge triggered: Stores D when C changes from 0 to 1
  - Saying "level-sensitive latch," or "edge-triggered flip-flop," is redundant
  - Two types of flip-flops -- rising or falling edge triggered.
- Comparing behavior of latch and flip-flop:

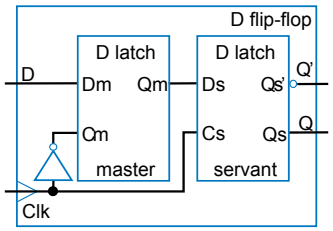


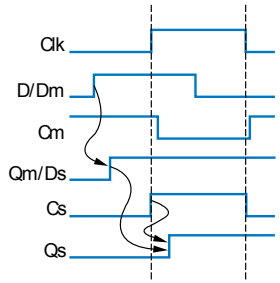
D Flip-Flop

- ***Flip-flop***: Bit storage that stores on clock edge, not level
- One design -- master-servant
  - Two latches, output of first goes to input of second, master latch has inverted clock signal
  - So master loaded when C=0, then servant when C=1
  - When C changes from 0 to 1, master disabled, servant loaded with value that was at D just before C changed -- i.e., value at D during rising edge of C

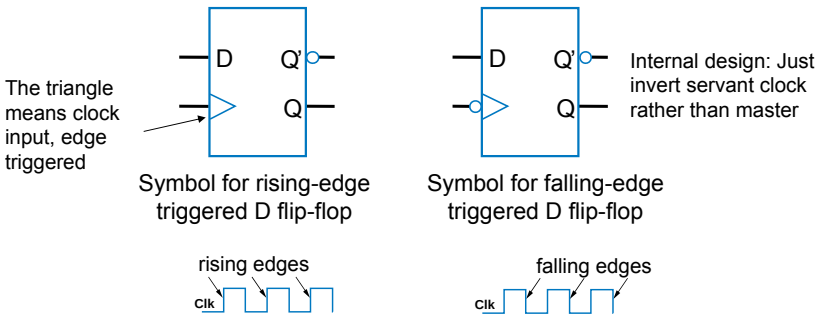


Note: Hundreds of different flip-flop designs exist





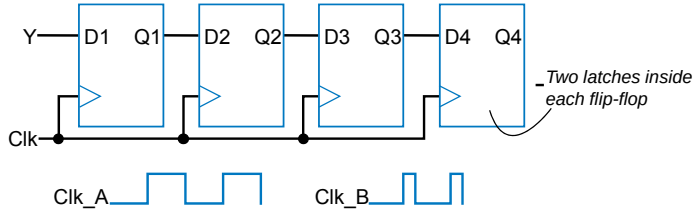
D Flip-Flop



We will use positive edge triggered FFs

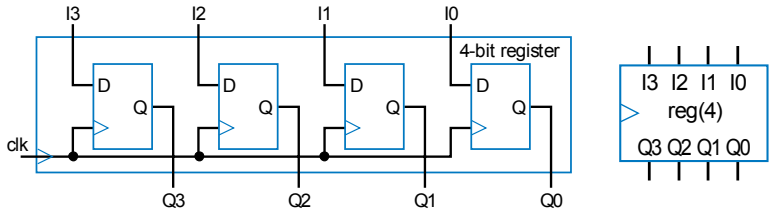
D Flip-Flop

- Solves problem of not knowing through how many latches a signal travels when C=1
  - In figure below, signal travels through exactly one flip-flop, for Clk_A or Clk_B
  - Why? Because on rising edge of Clk, all four flip-flops are loaded simultaneously -- then all four no longer pay attention to their input, until the next rising edge. Doesn't matter how long Clk is 1.

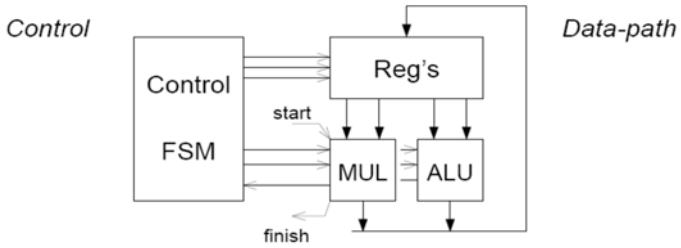


Basic Register

- Typically, we store multi-bit items
  - e.g., storing a 4-bit binary number
- ***Register***: multiple flip-flops sharing clock signal
  - From this point, we'll use registers for bit storage
    - No need to think of latches or flip-flops
    - But now you know what's inside a register



Programmable Structure



- Scheduling / ordering / sequencing of operations
- Mapping / allocation:
  - Variables -> {Reg1, ... ,RegN}
  - Operations -> {MUL, ADD, ALU, ... ,}
- Concurrency?
- Data dependent latency?

Questions?