EEC170
Computer Architecture

Lecture 3
Design Process

October 12, 2005
Soheil Ghiasi
Electrical and Computer Engineering
University of California, Davis

Recap: Salient features of MIPS I

- **32-bit fixed format inst** (3 formats)
- **32 32-bit GPR** (R0 contains zero) and 32 FP registers (and HI LO)
  - – partitioned by software convention
- **3-address, reg-reg arithmetic instr.**
- **Single address mode for load/store:** base+displacement
  - – no indirection, scaled
- **16-bit immediate plus LUI**
- **Simple branch conditions**
  - – compare against zero or two registers for =,≠
  - – no integer condition codes
- **Delayed branch**
  - – execute instruction after a branch (or jump) even if the branch is taken
    (Compiler can fill a delayed branch with useful work about 50% of the time)

Now what?

- We have a description (contract) of what we need to build
  - MIPS ISA
  - Software writers will give us programs in this form (MIPS machine code)
- Need to design the hardware to fulfill the contract
  - Top down design, bottom up construction
- Questions:
  - What are the basic components that we are given?
  - What are the basic components needed to make a microprocessor?

The Design Process

"To Design Is To Represent"

Design activity yields description/representation of an object

-- **Traditional craftsman does not distinguish between the conceptualization and the artifact**

-- **Separation comes about because of *complexity***

-- **The concept is captured in one or more *representation languages***

-- **This process IS design**

Design Begins With Requirements

-- **Functional Capabilities: what it will do**

-- **Performance Characteristics: Speed, Power, Area, Cost, . . .**

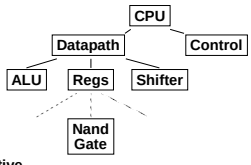
Design Process (cont.)

Design Finishes As Assembly

-- **Design understood in terms of components and how they have been assembled**

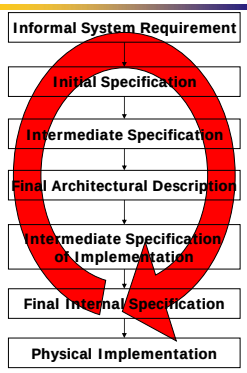
-- **Top Down *decomposition* of complex functions (behaviors) into more primitive functions**

-- **bottom-up *composition* of primitive building blocks into more complex assemblies**



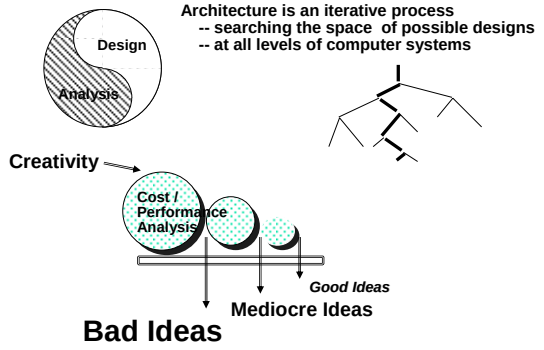
Design is a "creative process," not a simple method

Design Refinement

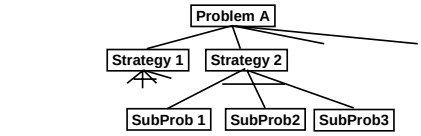


Refinement increasing level of detail

Measurement and Evaluation



Design as Search



Solution 1 | Solution 2 | Solution 3 | Solution *n*

Design involves educated guesses and verification

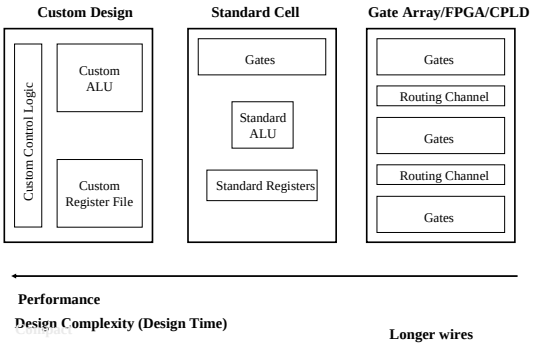
-- **Given the goals, how should these be prioritized?**

-- **Given alternative design pieces, which should be selected?**

-- **Given design space of components & assemblies, which part will yield the best solution?**

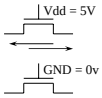
Feasible (good) choices vs. Optimal choices

Range of Design Styles

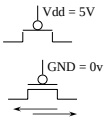


Basic Technology: CMOS

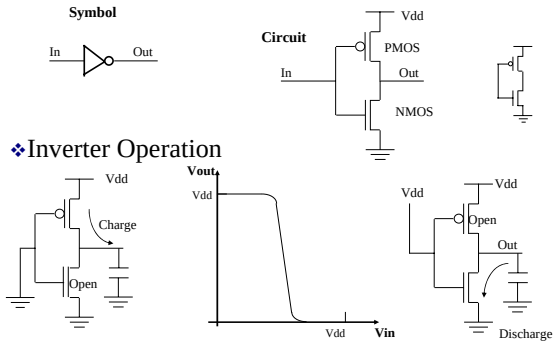
- CMOS: Complementary Metal Oxide Semiconductor
  - NMOS (N-Type Metal Oxide Semiconductor) transistors
  - PMOS (P-Type Metal Oxide Semiconductor) transistors
- NMOS Transistor
  - Apply a HIGH (Vdd) to its gate turns the transistor into a "conductor"
  - Apply a LOW (GND) to its gate shuts off the conduction path



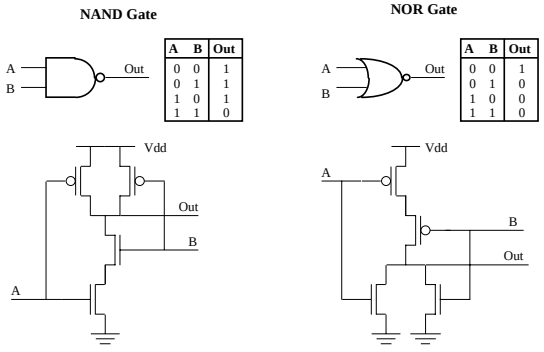
- PMOS Transistor
  - Apply a HIGH (Vdd) to its gate shuts off the conduction path
  - Apply a LOW (GND) to its gate turns the transistor into a "conductor"



Basic Components: CMOS Inverter

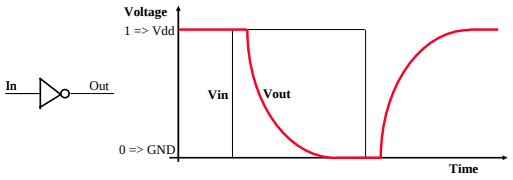


Basic Components: CMOS Logic Gates

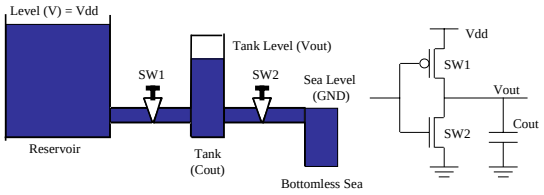


Ideal (digital) versus Reality (analog)

- When input 0 -> 1, output 1 -> 0 but NOT instantly
  - Output goes 1 -> 0: output voltage goes from Vdd to 0v
- When input 1 -> 0, output 0 -> 1 but NOT instantly
  - Output goes 0 -> 1: output voltage goes from 0v to Vdd
- Voltage does not like to change instantaneously



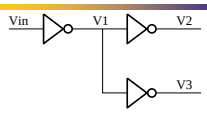
Fluid Timing Model



- Water <-> Electrical Charge Tank Capacity <-> Capacitance (C)
- Water Level <-> Voltage Water Flow <-> Charge Flowing (Current)
- Size of Pipes <-> Strength of Transistors (G)
- Time to fill up the tank ~ C / G

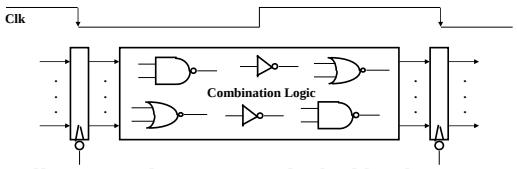
Delay = Internal Delay + (Load Dependent Delay x Output Load)

Calculating Aggregate Delays; Critical Path



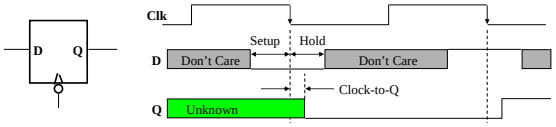
- Sum delays along serial paths
- Delay (Vin -> V2) ! = Delay (Vin -> V3)
  - Delay (Vin -> V2) = Delay (Vin -> V1) + Delay (V1 -> V2)
  - Delay (Vin -> V3) = Delay (Vin -> V1) + Delay (V1 -> V3)
- $Capacitance_1$ = Wire Capacitance
  + $Capacitance_{in}$ of $Invertor_2$
  + $Capacitance_{in}$ of $Invertor_3$
- Critical Path = The longest among the N parallel paths

Clocking Methodology



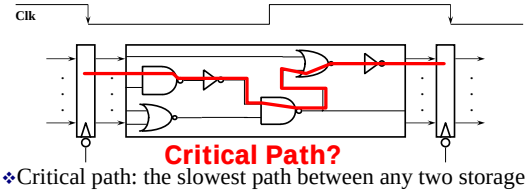
- All storage elements are clocked by the same clock edge (but there may be clock skews)
- The combination logic block's:
  - Inputs are updated at each clock tick
  - All outputs MUST be stable before the next clock tick

Storage Element's Timing Model



- Setup Time: Input must be stable BEFORE the trigger clock edge
- Hold Time: Input must REMAIN stable after the trigger clock edge
- Clock-to-Q time:
  - Output cannot change instantaneously at the trigger clock edge
  - Similar to delay in logic gates, two components:
    - Internal Clock-to-Q
    - Load dependent Clock-to-Q

Critical Path & Cycle Time

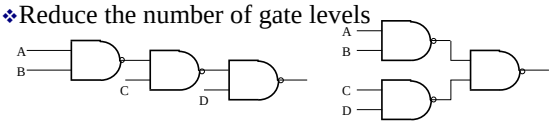


- Critical path: the slowest path between any two storage devices
- Cycle time is a function of the critical path
- must be greater than:

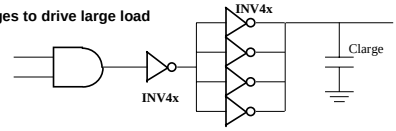
Clock-to-Q + Longest Path through Combination Logic + Setup (also clock skew)

Tricks to Reduce Cycle Time

- Reduce the number of gate levels



- Pay attention to loading
  - One gate driving many gates is a bad idea
  - Avoid using a small gate to drive a long wire
- Use multiple stages to drive large load



Problem: Design a "fast" ALU for the MIPS ISA

- Requirements?
- Must support the MIPS ISA Arithmetic / Logic operations
- Tradeoffs of cost and speed based on frequency of occurrence, hardware budget

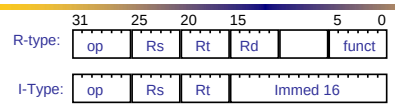
MIPS ALU requirements

- Add, AddU, Sub, SubU, AddI, AddIU
  - => 2's complement adder/sub with overflow detection
- And, Or, AndI, OrI, Xor, Xori, Nor
  - => Logical AND, logical OR, XOR, nor
- SLTI, SLTIU (set less than)
  - => 2's complement adder with inverter, check sign bit of result

Additional MIPS ALU requirements

- Mult, MultU, Div, DivU (coming soon)
  => Need 32-bit multiply and divide, signed and unsigned
- Sll, Srl, Sra (coming soon)
  => Need left shift, right shift, right shift arithmetic by 0 to 31 bits

MIPS arithmetic instruction format



Type	op	funct
ADDI	10	xx
ADDIU	11	xx
SLTI	12	xx
SLTIU	13	xx
ANDI	14	xx
ORI	15	xx
XORI	16	xx
LUI	17	xx

Type	op	funct
ADD	00	40
ADDU	00	41
SUB	00	42
SUBU	00	43
AND	00	44
OR	00	45
XOR	00	46
NOR	00	47

Type	op	funct
	00	50
	00	51
SLT	00	52
SLTU	00	53

- Signed arith generate overflow, no carry

Design Trick: divide & conquer

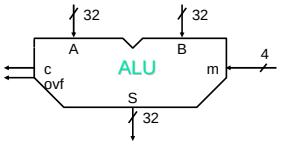
- Break the problem into simpler problems, solve them and glue together the solution
- Example: assume that immediates have been taken care of before the ALU
  - Now we only have to worry about 10 operations
  - Deal with others later (or someone else's problem)

00	add
01	addU
02	sub
03	subU
04	and
05	or
06	xor
07	nor
12	slt
13	sltU

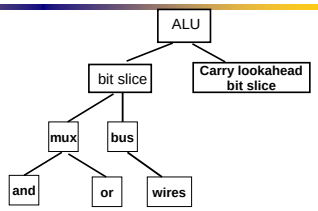
Refined Requirements

(1) Functional Specification
inputs: 2 x 32-bit operands A, B, 4-bit mode
outputs: 32-bit result S, 1-bit carry, 1 bit overflow
operations: add, addu, sub, subu, and, or, xor, nor, slt, sltU

(2) Block Diagram (e.g. VHDL entity)



Design Decisions



- Simple bit-slice
  - big combinational problem
  - many little combinational problems
  - partition into 2-step problem
- Bit slice with carry look-ahead
- . . .

Conclusion

- Critical Path is longest among N parallel paths
- Setup Time and Hold Time determine how long Input must be stable before and after trigger clock edge
- An Overview of the Design Process
  - Design is an iterative process, multiple approaches to get started
  - Do NOT wait until you know everything before you start