Integrated Circuits and Systems

Zoran Stamenković

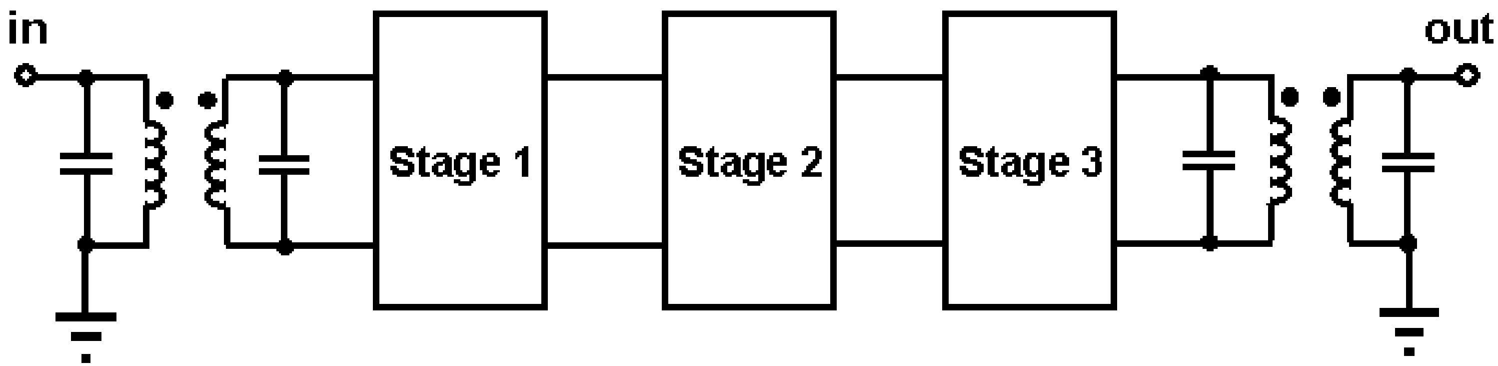
stamenkovic@ihp-microelectronics.com

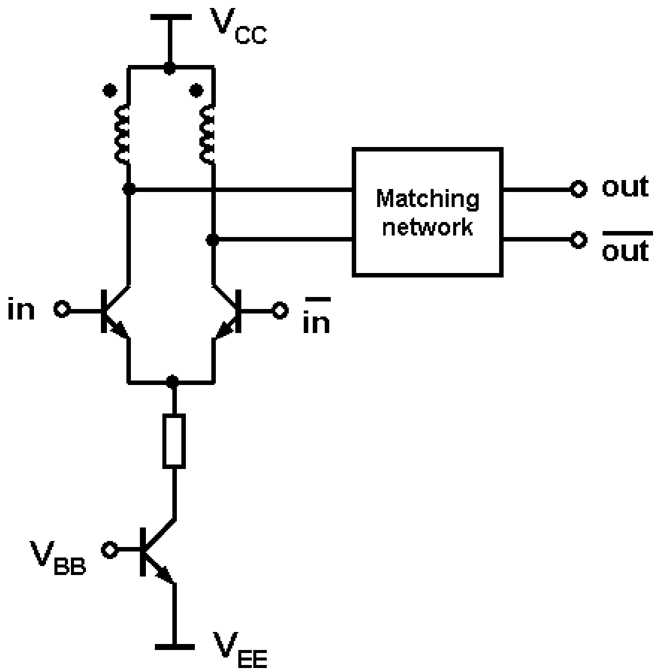
International Organization for Migration (IOM)
Organisation internationale pour les migrations (OIM)
Organización Internacional para las Migraciones (OIM)

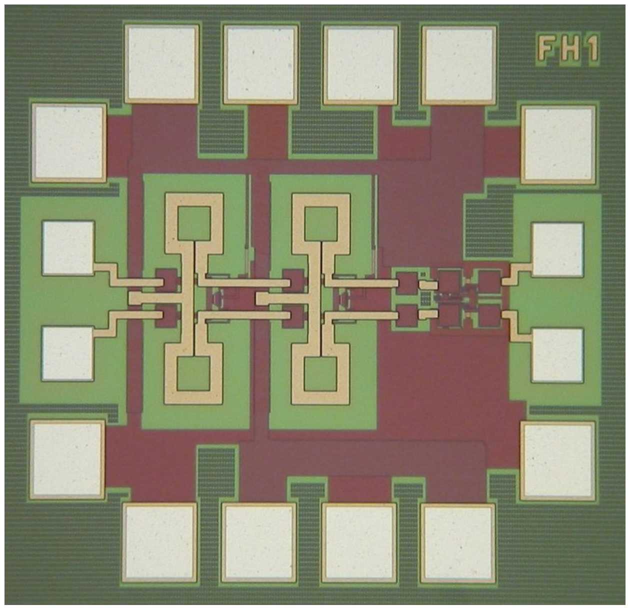
Topics

- **Analog Circuits**
  - **Amplifier**
  - **Oscillator**
  - **Modulator/Demodulator**
- **Standard Cells and Memories**
  - **Inverter, Switch, NAND, NOR, AOI/OAI**
  - **Delay and Power Consumption**
  - **ROM, SRAM, DRAM**
  - **Memory Generators**
- **Digital Circuits**
  - **Adders**
  - **Multipliers**
- **Digital Systems**
  - **General Purpose Processor**
  - **Digital Signal Processor**
  - **Peripherals**

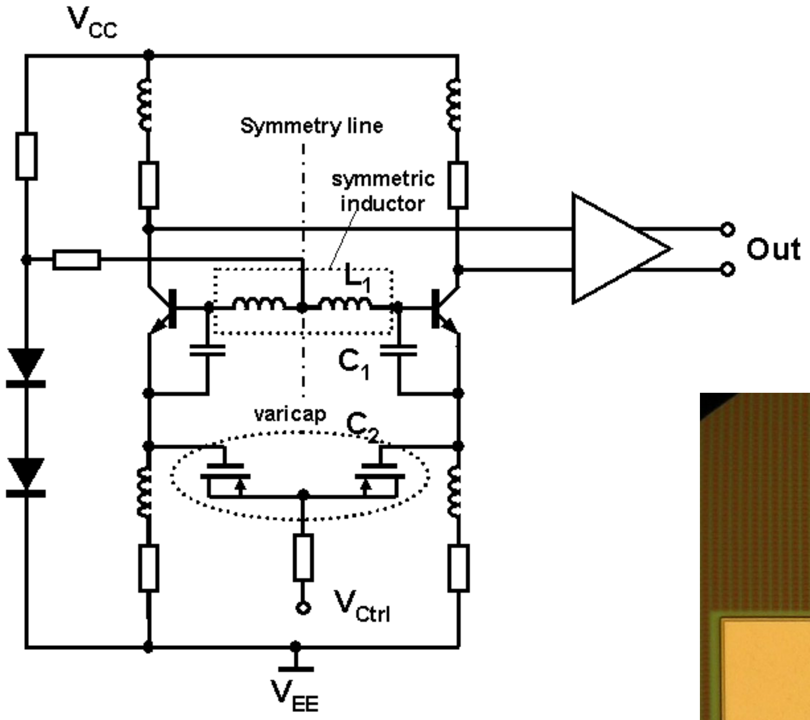
Low Noise Amplifier

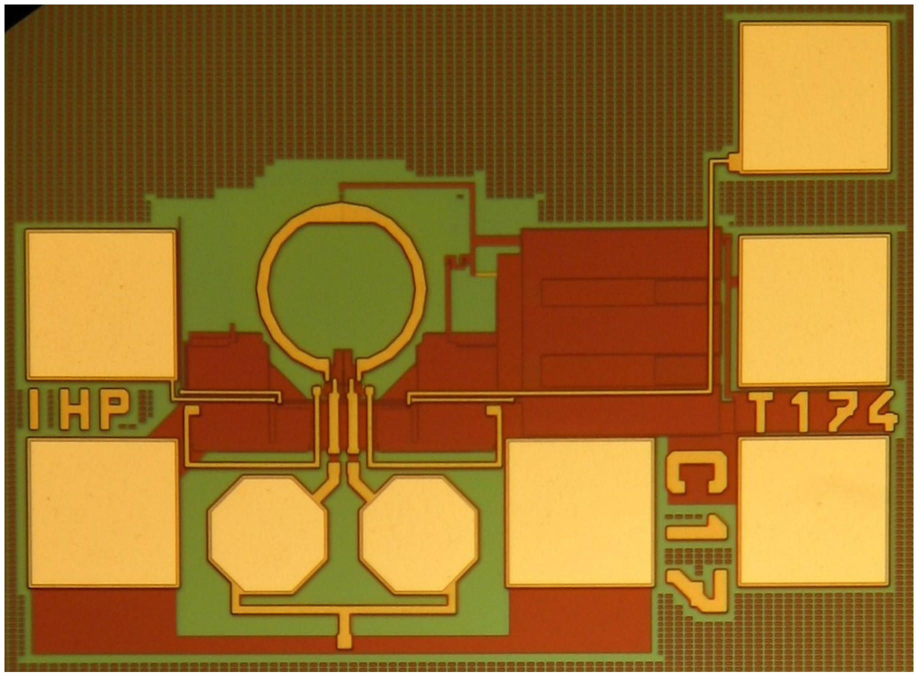




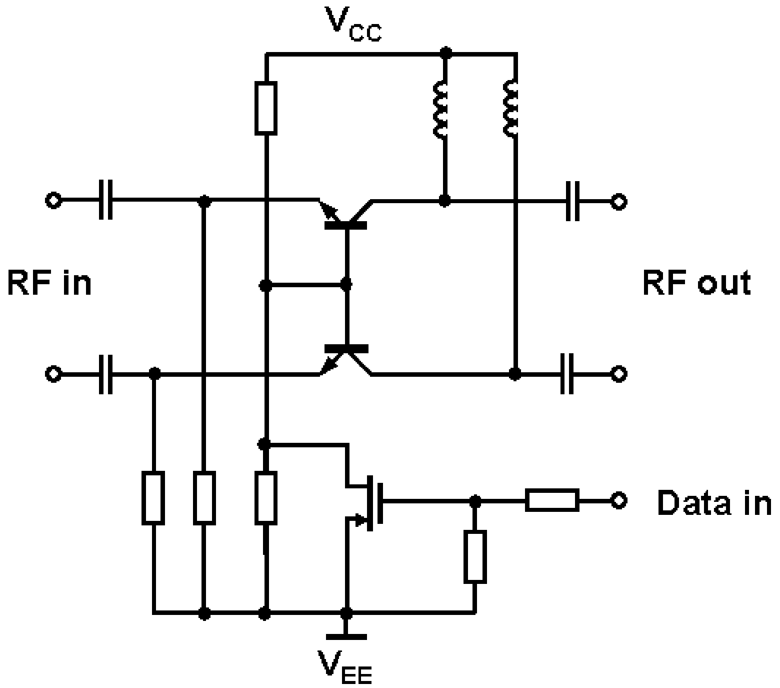


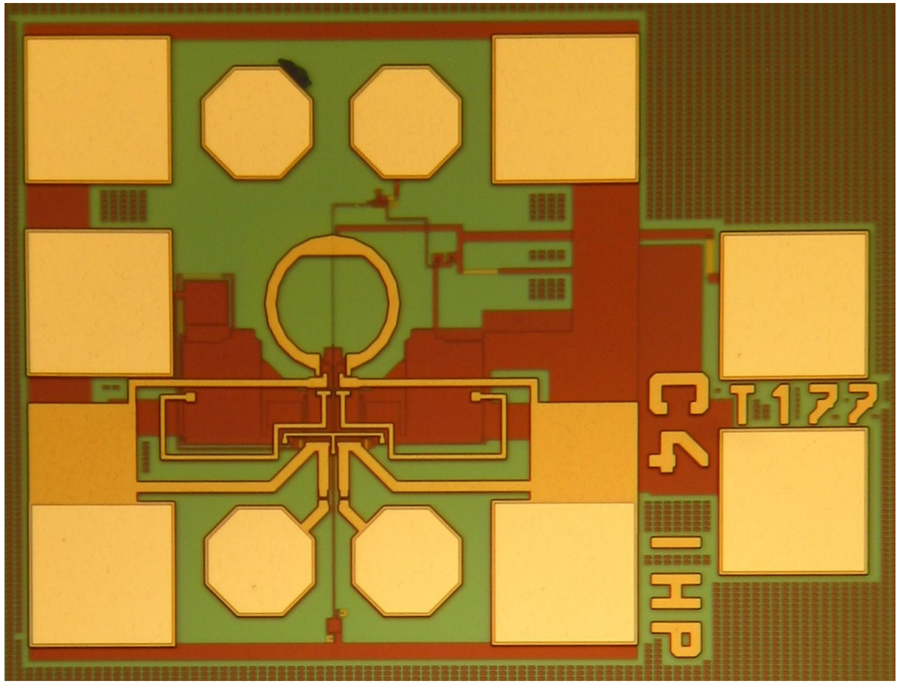
Voltage Control Oscillator





Modulator/Demodulator

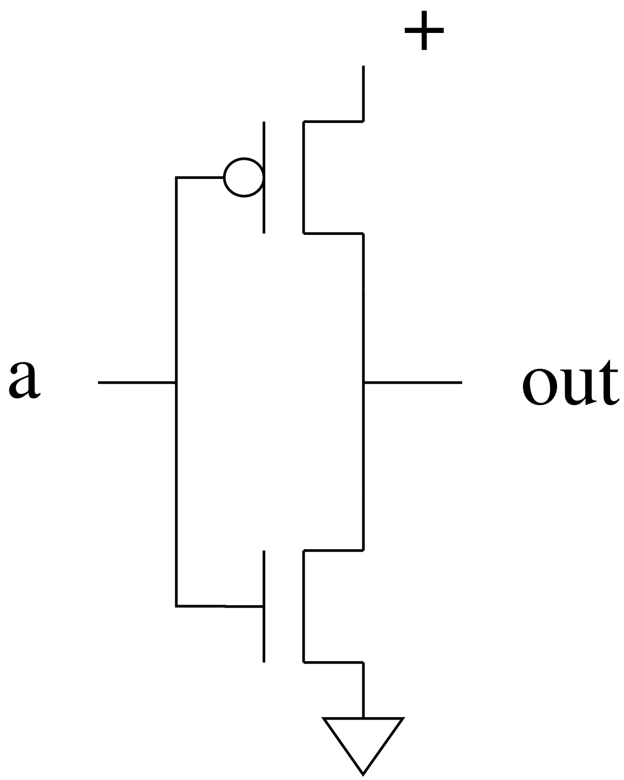


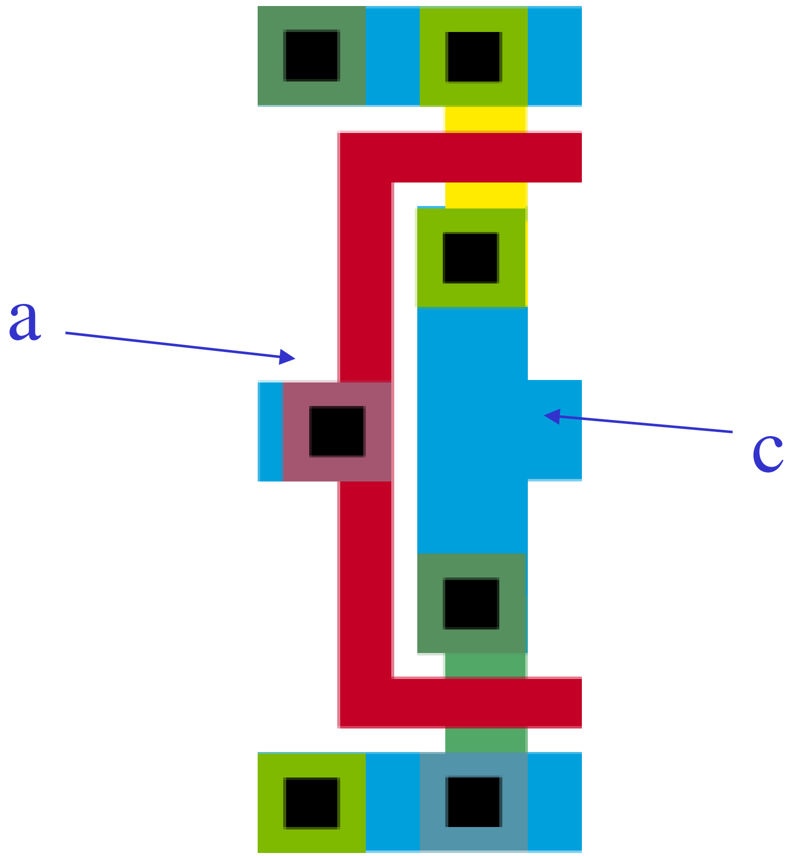


Logic Functions

- **Function**
  - *f = a'b + ab'*: *a* is a variable, *a* and *a'* are literals, *ab'* is a term
- ***Irredundant* Function**
  - No literal can be removed without changing its value
- **Implementing logic functions is non-trivial**
  - No logic gates in the library for all logic expressions
  - A logic expression may map into gates that consume a lot of area, time, or power
- **A set of functions f1, f2, ... is complete if every Boolean function can be generated by a combination of the functions from the set**
  - NAND and NOR are complete sets
  - AND and OR are not complete
  - Transmission gates are not complete
- **Incomplete set of logic gates**
  - No way to design arbitrary logic

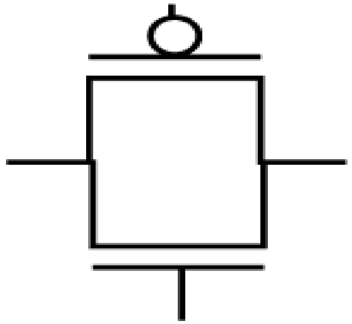
Inverter



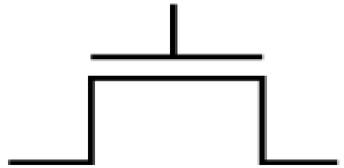


Switch

- **Complementary switch produces full-supply voltages for both logic 0 and logic 1**
  - **n-type transistor conducts logic 0**
  - **p-type transistor conducts logic 1**

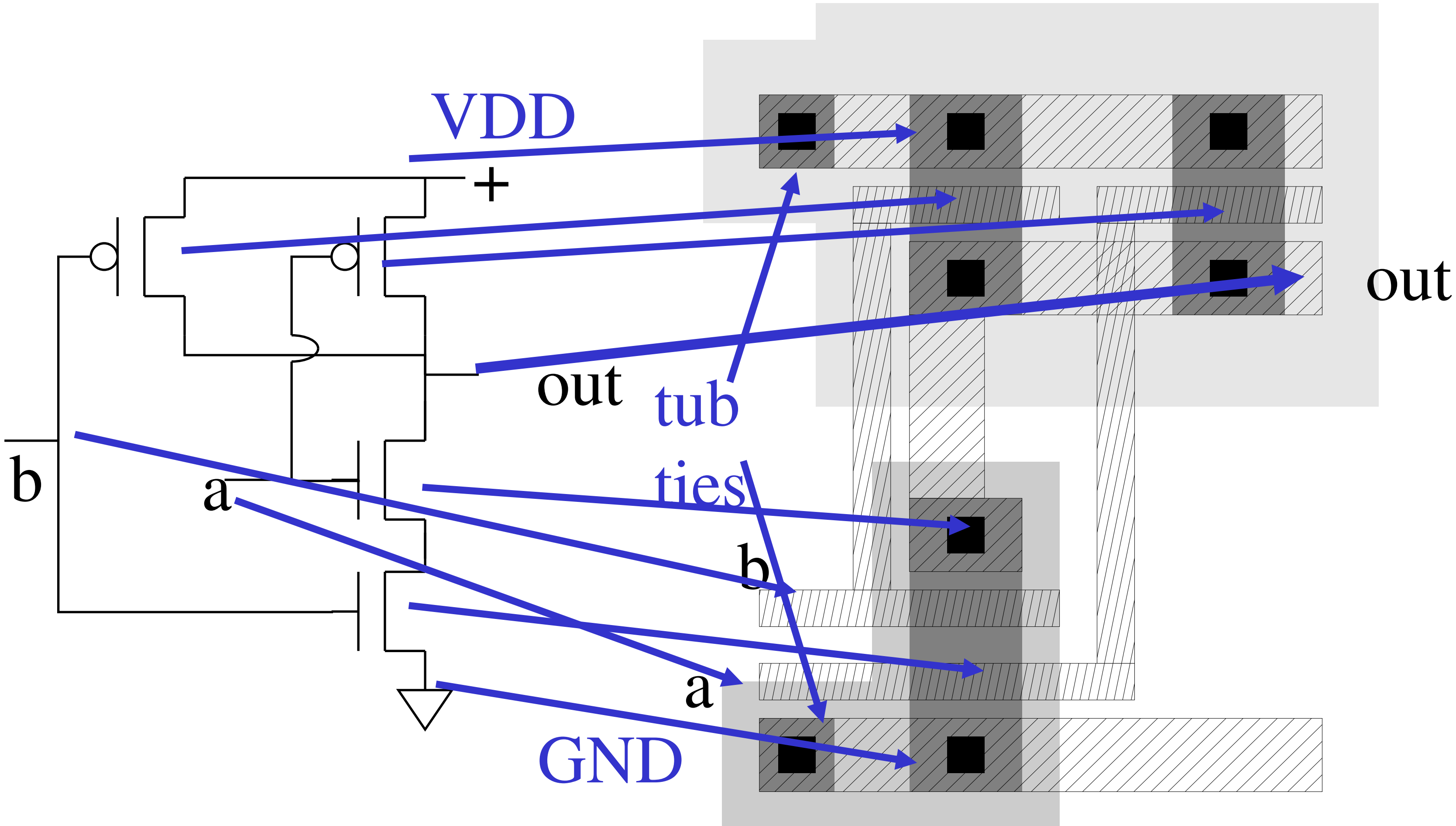


complementary

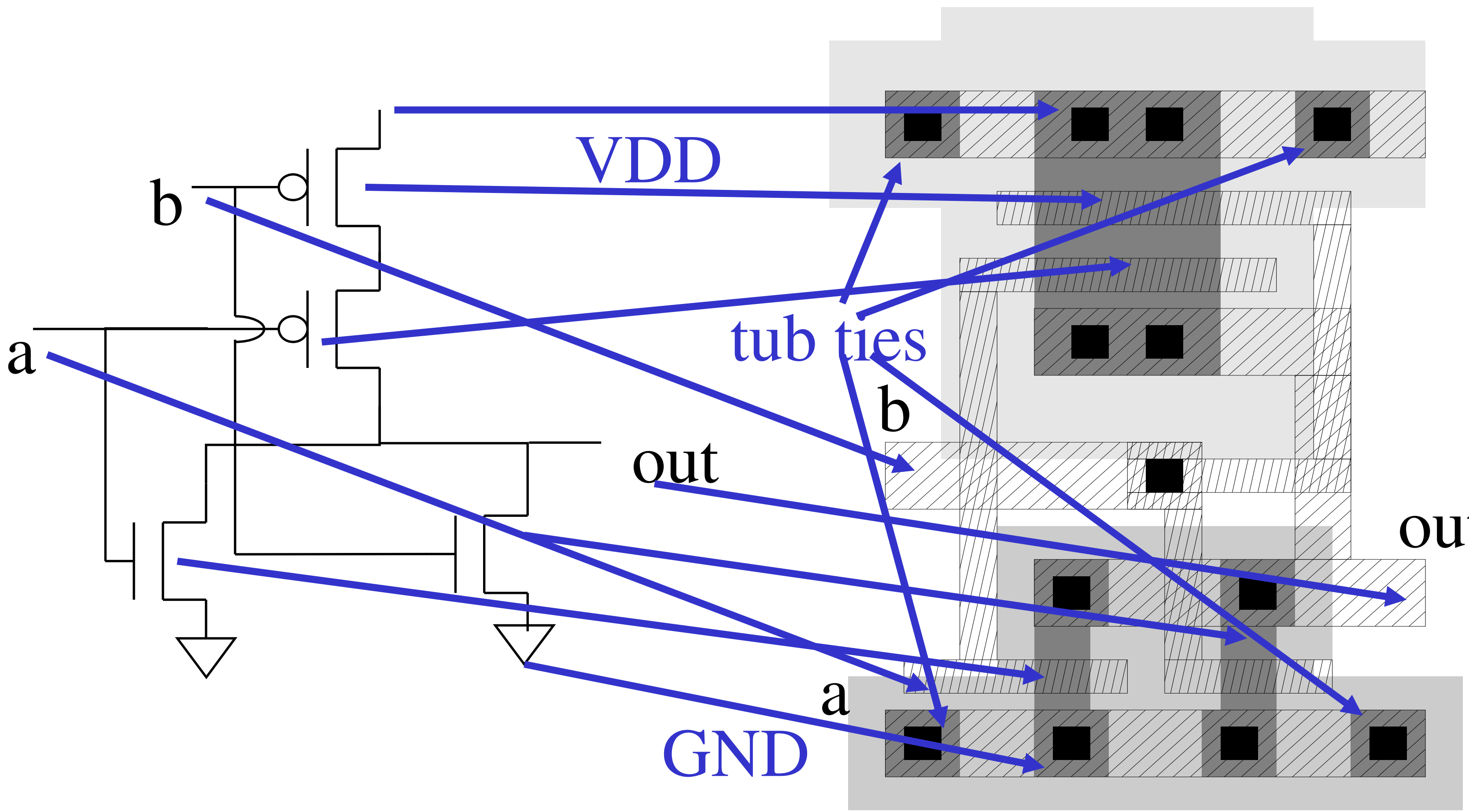


n-type

NAND Gate

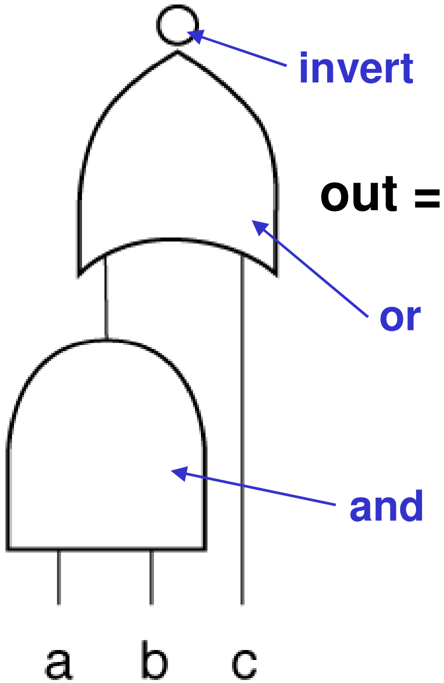


NOR Gate

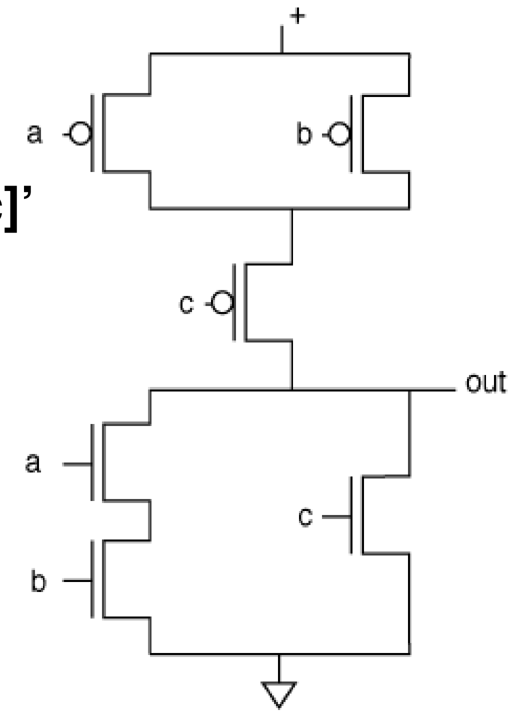


AOI/OAI Gates

- AOI = and/or/invert
- OAI = or/and/invert
- Implement larger functions
- Pull-up and pull-down networks are compact
  - Smaller area, higher speed than NAND/NOR network equivalents
- AOI312
  - And 3 inputs
  - And 1 input (dummy)
  - And 2 inputs
  - Or together these term
  - Invert

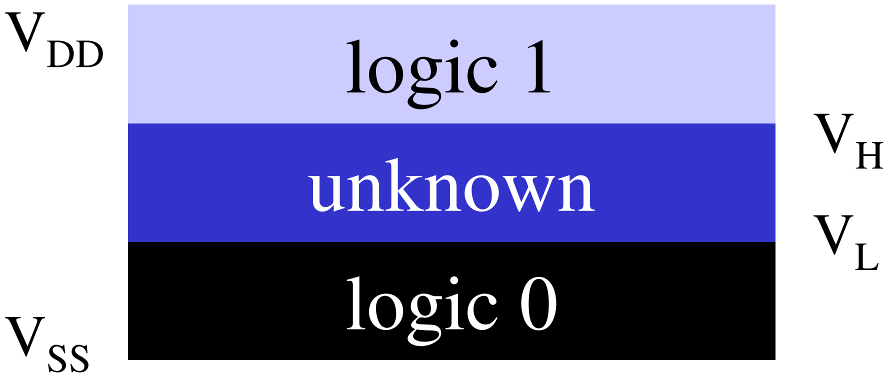


out = [ab+c]’

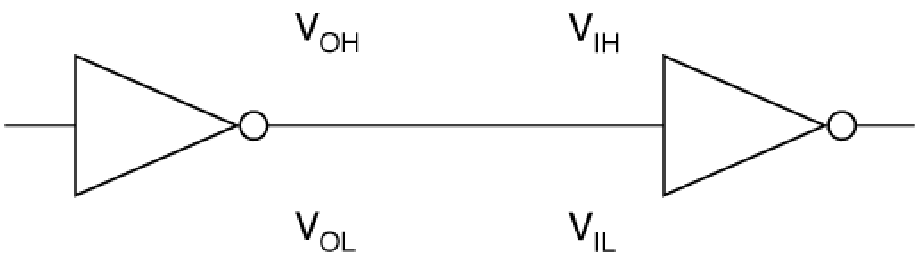


Logic Levels

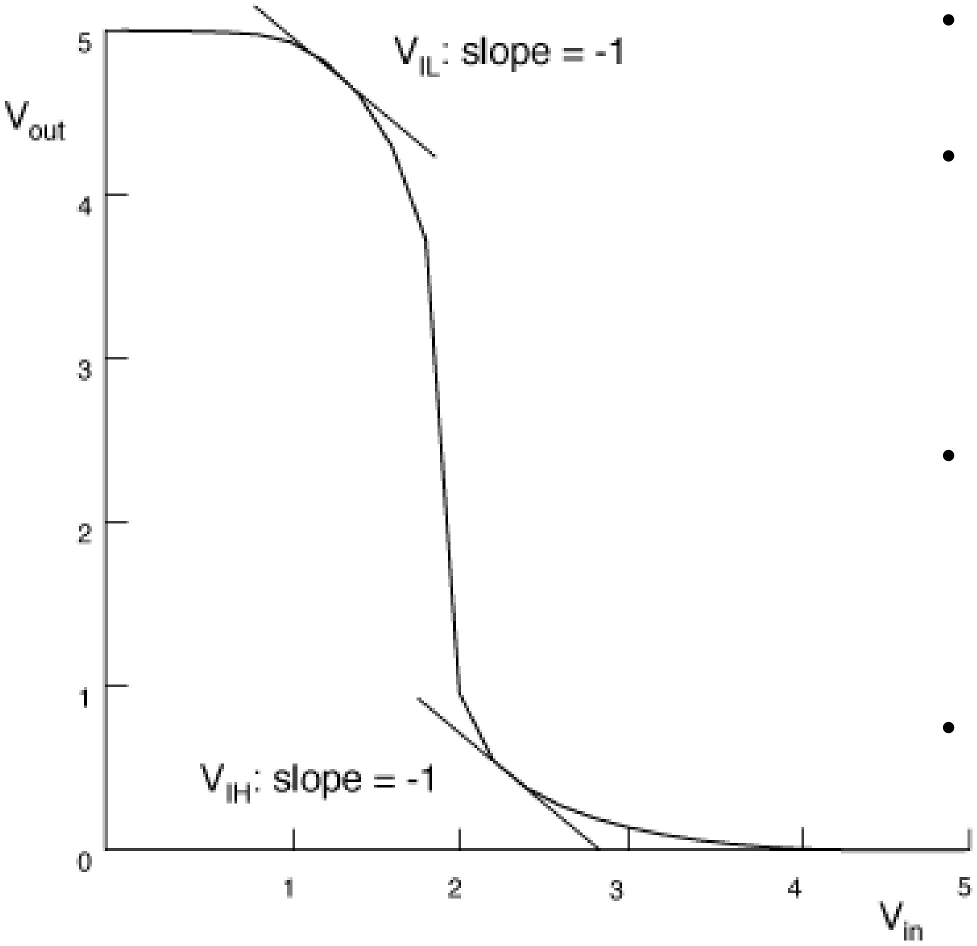
- Solid logic 0/1 defined by V_{SS}/V_{DD}
- Inner bounds of logic values V_L/V_H are not directly determined by circuit properties, as in some other logic families



- Levels at output of one gate must be sufficient to drive next gate



Inverter Transfer Curve



- **Choose threshold voltages at points where slope of transfer curve is -1**
- **Inverter has**
  - **High gain between V_{IL} and V_{IH} points**
  - **Low gain at outer regions of transfer curve**
- **Note that logic 0 and 1 regions are not equally sized**
  - **In this case, high pull-up resistance leads to smaller logic 1 range**
- **Noise margins are V_{DD}-V_{IH} and V_{IL}-V_{SS}**
  - **Noise must exceed noise margin to make second gate produce wrong output**

RC Model for Delay

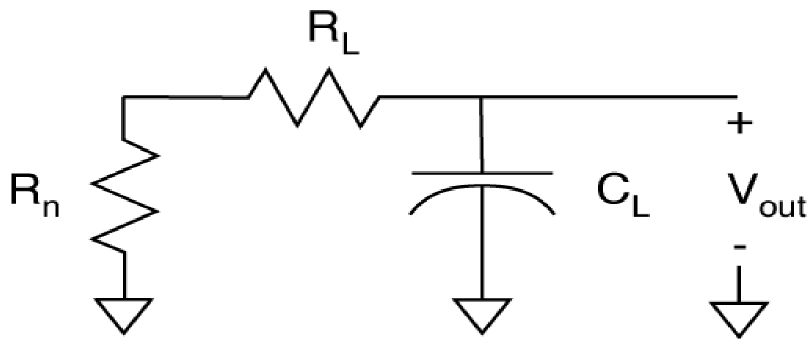
- **Delay**

  Time required for gate's output to reach 50% of final value
- **Transition time**

  Time required for gate's output to reach 10% (logic 0) or 90% (logic 1) of final value
- **Gate delay based on RC time constant**

  $V_{out}(t) = V_{DD} \exp\{-t/[(R_n+R_L)C_L]\}$

  $t_d = 0.69\ R_nC_L$, $t_f = 2.3\ R_nC_L$



- **0.5 μm process**

  $R_n = 3.9\ k\Omega$, $C_L = 0.68\ fF$

  $t_d = 0.69 \times 3.9 \times .68E\text{-}15 = 1.8\ ps$

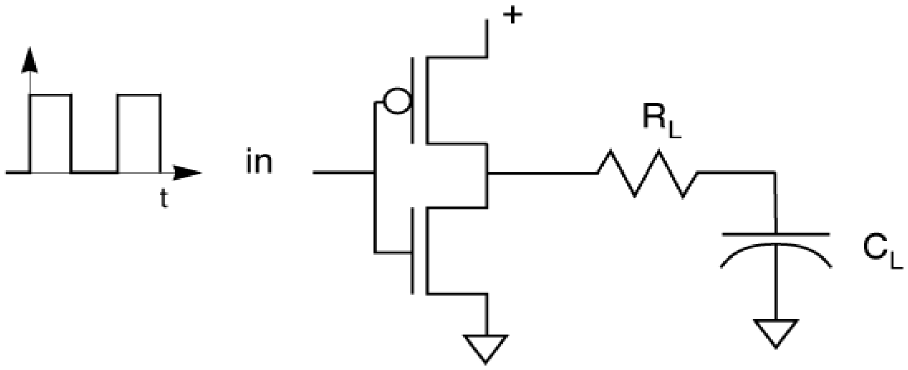
  $t_f = 2.3 \times 3.9 \times .68E\text{-}15 = 6.1\ ps$
- **For pull-up time, use pull-up resistance**
- **Current source model (in power/delay studies)**

  $t_f = C_L\ (V_{DD}\text{-}V_{SS})/[0.5\ k'\ (W/L)\ (V_{DD}\text{-}V_{SS}\ \text{-}V_t)^2]$
- **Fitted model**

  Fit curve to measured circuit characteristics

Power Consumption

- Clock frequency
  - $f = 1/t$
- Energy
  - $E = C_L(V_{DD} - V_{SS})^2$
- Power
  - $E \times f = f\, C_L(V_{DD} - V_{SS})^2$



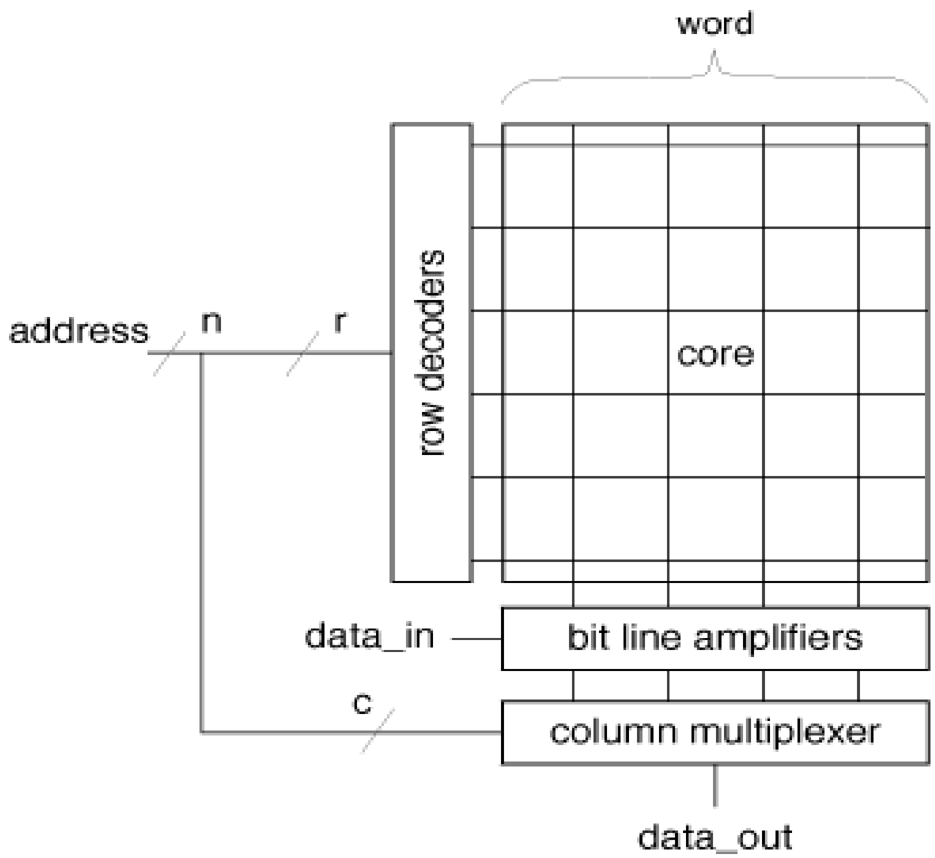
- Almost all power consumption comes from switching behavior
  - A single cycle requires one charge and one discharge of capacitor
- Static power dissipation
  - Comes from leakage currents
- Surprising result
  - Resistance of the pull-up/pull-down transistor drops out of energy calculation
  - Power consumption is independent of the sizes of the pull-up and pull-down transistors
- Static CMOS power-delay product is independent of frequency
  - Voltage scaling depends on this fact

Memory Architecture

- **Address is divided into row and column**

  Row may contain full word or more than one word

- **Selected row drives/senses bit lines in columns**
- **Amplifiers/drivers read/write bit lines**



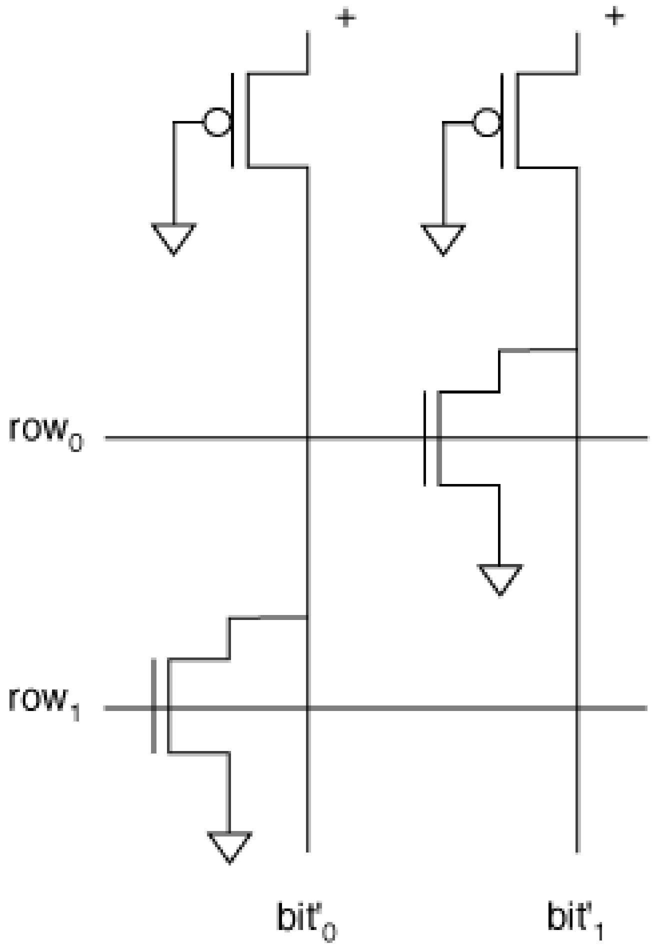
Read-Only Memory (ROM)

- ROM core is organized as an array of NOR gates

  Pull-down transistors of NOR determine programming

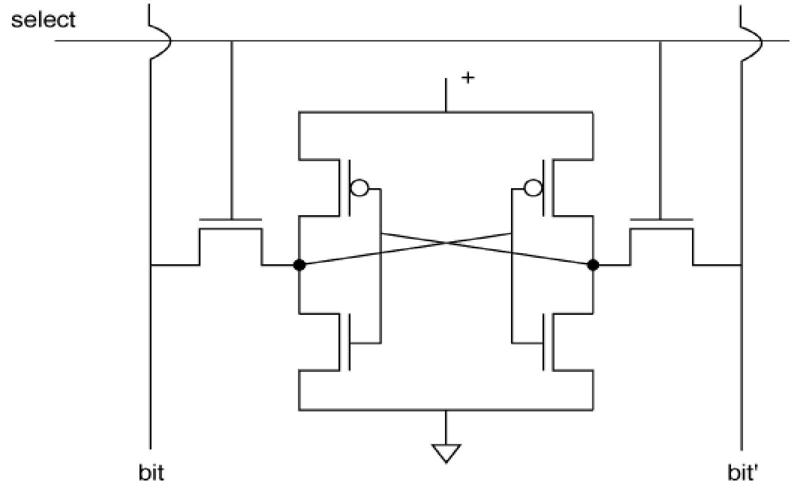
- Erasable ROMs require special processing that is not typically available
- ROMs on digital ICs are generally mask-programmed

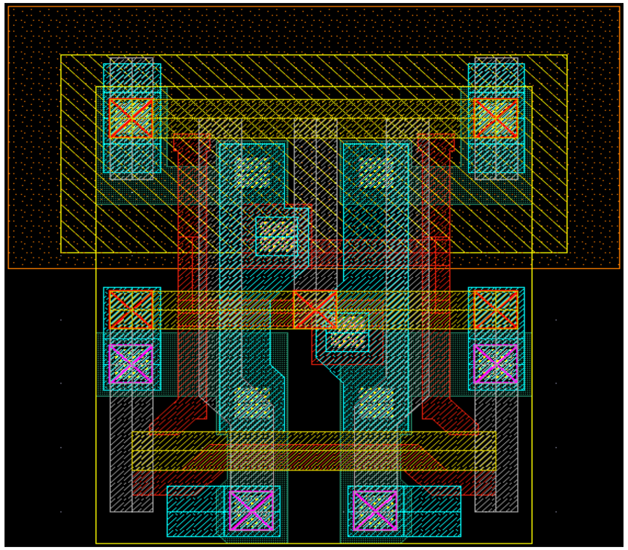
  Placement of pull-downs determines ROM contents



Static Random-Access Memory (SRAM)

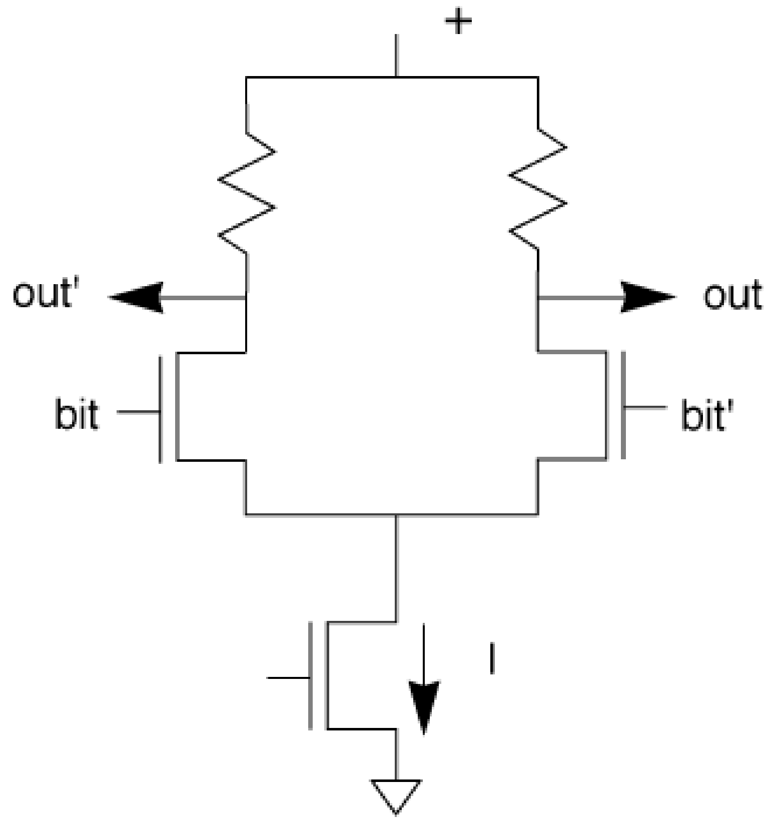
- Core cell uses six-transistor circuit to store value
- Value is stored symmetrically
  - Both true and complement are stored on cross-coupled transistors
- SRAM retains value as long as power is applied to circuit
- Read
  - Precharge bit and bit' high
  - Set select line high from row decoder
  - One bit line will be pulled down
- Write
  - Set bit/bit' to desired (complementary) values
  - Set select line high
  - Drive on bit lines will flip state if necessary





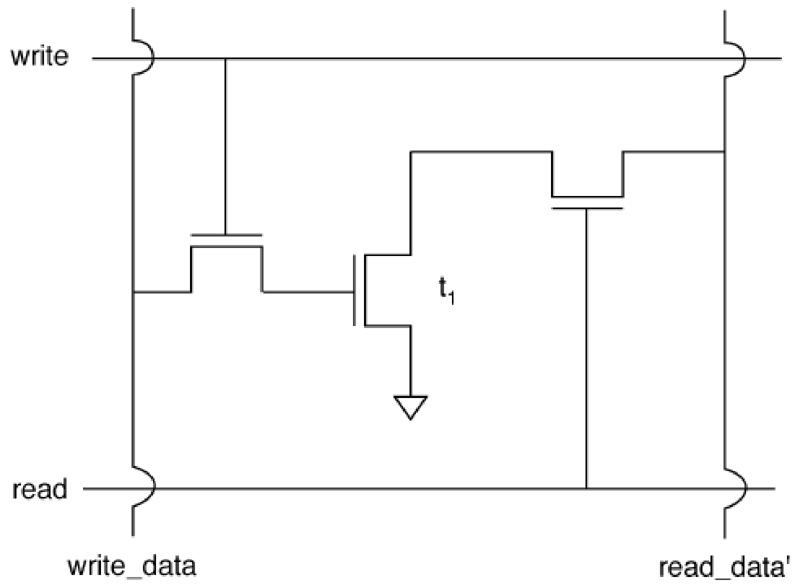
SRAM Sense Amplifier

- Differential pair
  - Takes advantage of complementarity of bit lines
- One bit line goes low
  - One arm of diff pair reduces its current, causing compensating increase in current of another arm
- Sense amp can be cross-coupled to increase speed



Dynamic Random-Access Memory (DRAM)

- First form of DRAM
  - Modern commercial DRAMs use one-transistor cell
- Cell can easily be made with a digital technology process
- Dynamic RAM loses value due to charge leakage
  - Must be refreshed
- Value is stored on gate capacitance of transistor t_1
- Read
  - read = 1, write = 0, read_data' is precharged
  - t_1 will pull down read_data' if 1 is stored
- Write
  - read = 0, write = 1, write_data = value
  - Guard transistor writes value onto gate capacitance



Memory Generators

- A software tool which can create memories (ROM or RAM blocks) in a range of sizes as needed
  - The customer usually wants a particular number of words (depth) and bits (width) for each memory ordered
  - Each of the final building blocks (physical layout) will be implemented as a stand-alone, densely packed, pitch-matched array
- Complex layout generators and state-of-the-art logic and circuit design techniques offer
  - Embedded memories of extreme density and performance
- Each memory generator is a set of various, parameterized generators
  - Layout generator generates an array of custom, pitch-matched leaf cells
  - Schematic generator and Net-lister extracts a net-list used for both layout vs. schematic and functional verification
  - Function and Timing model generators create models for gate level simulation, dynamic/static timing analysis and synthesis
  - Symbol generator generates schematic
  - Critical Path generator is used for both circuit design and timing characterization

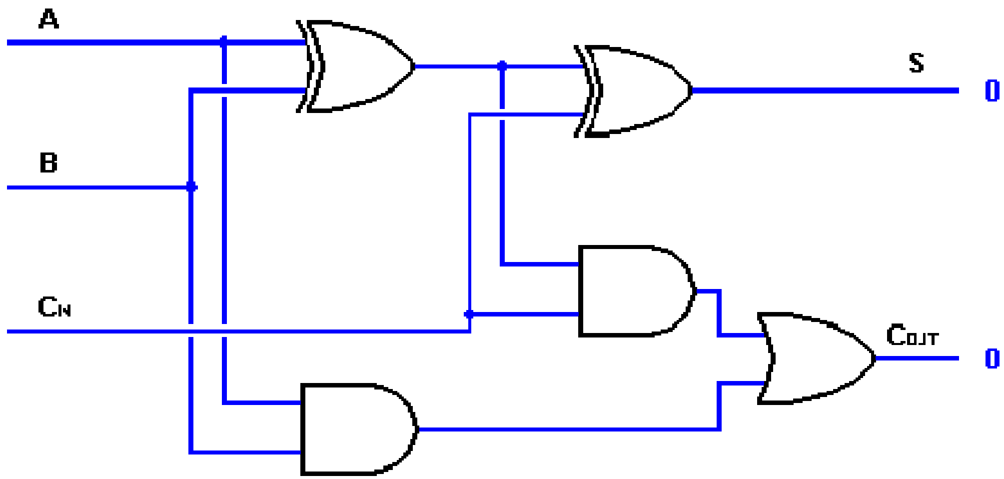
Full Adder

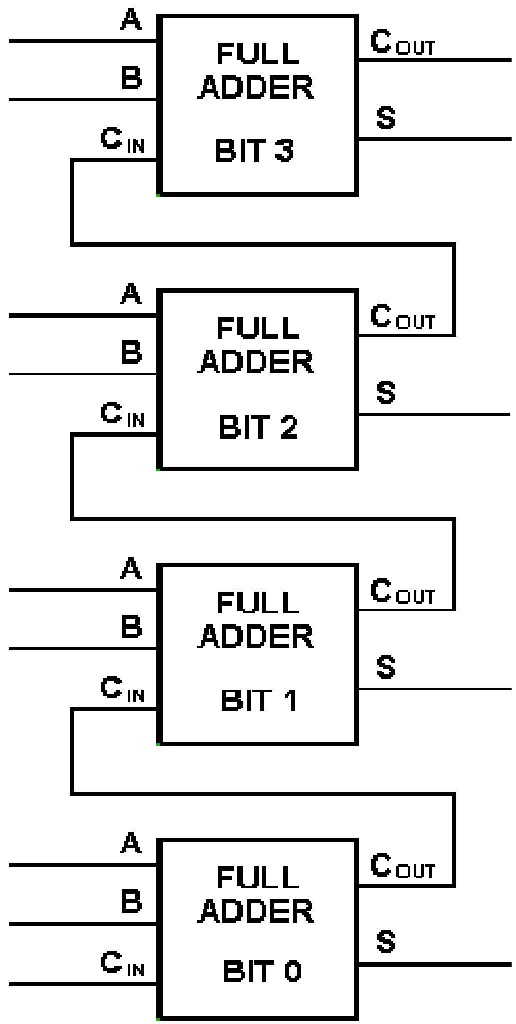
- Computes one-bit sum and carry

  $s_i = a_i \oplus b_i \oplus c_{in}$

  $c_{out} = a_i b_i + a_i c_i + b_i c_{in}$

- Ripple-carry adder: n-bit adder built from full adders
- Delay of ripple-carry adder goes through all carry bits





Combinational Multiplier

0 1 1 0 multiplicand

x 1 0 0 1 multiplier

0 1 1 0

+ 0 0 0 0

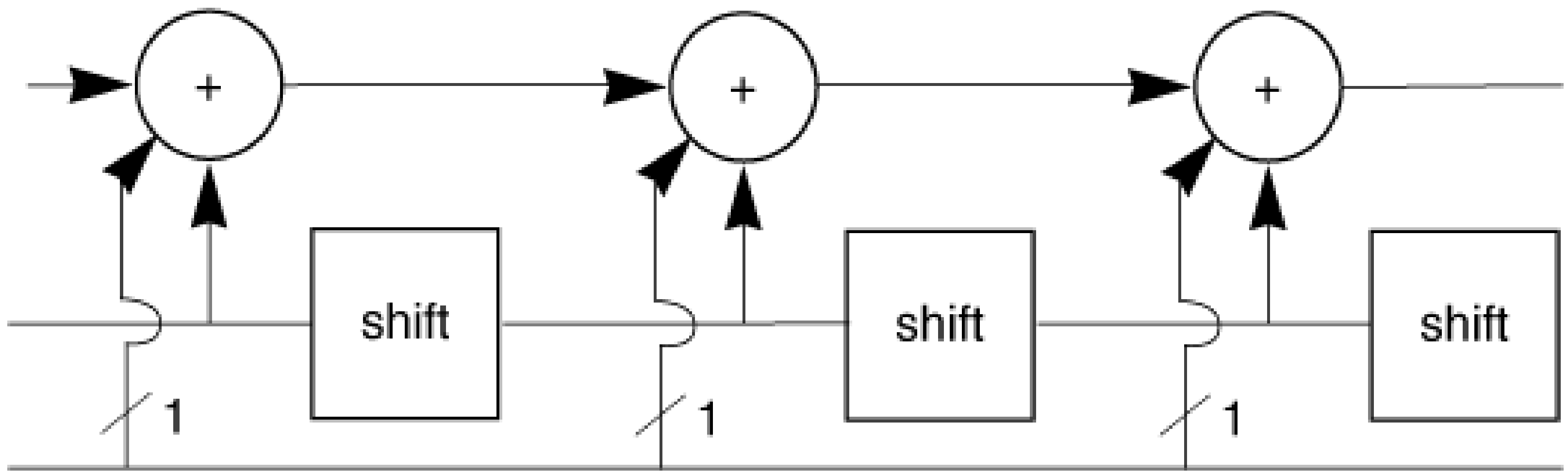
partial product

0 0 1 1 0

+ 0 0 0 0

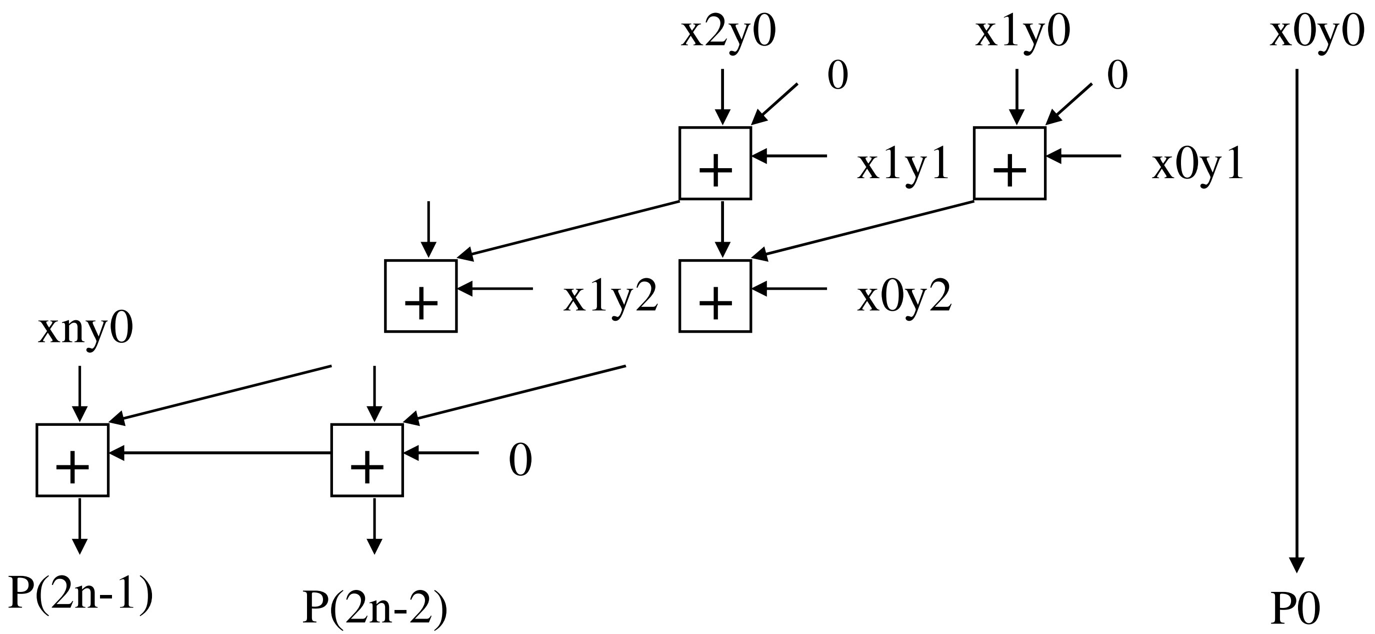
0 0 0 1 1 0

+ 0 1 1 0



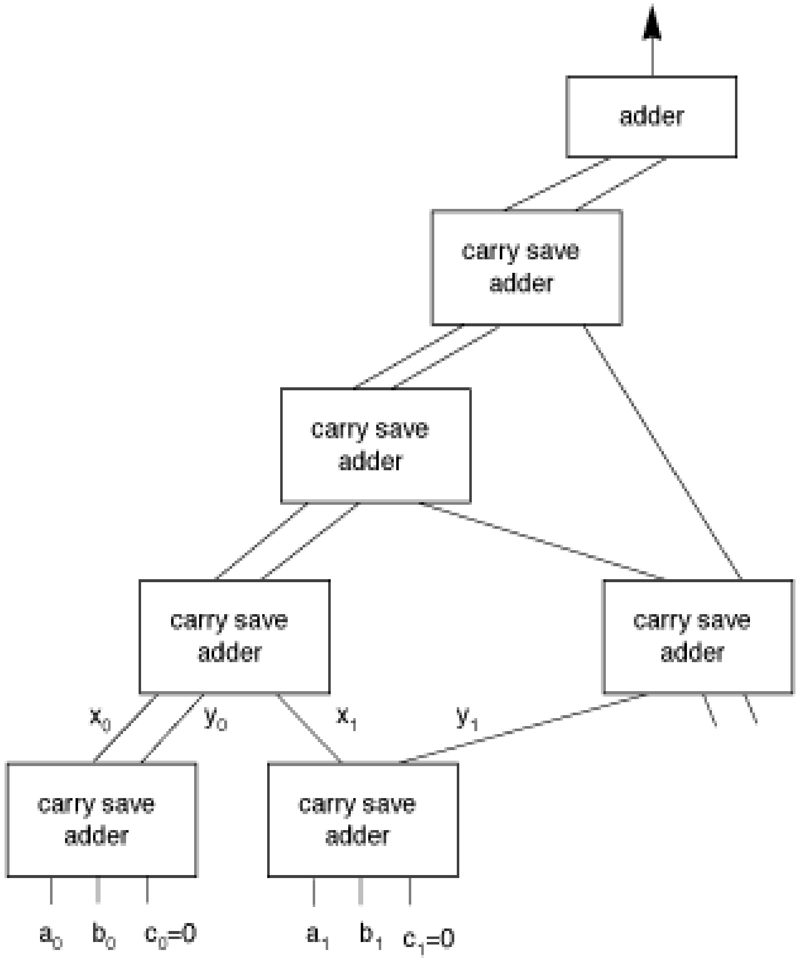
Array Multiplier

- Array multiplier is an efficient layout of a combinational multiplier
- Array multipliers may be pipelined to decrease clock period at the expense of latency

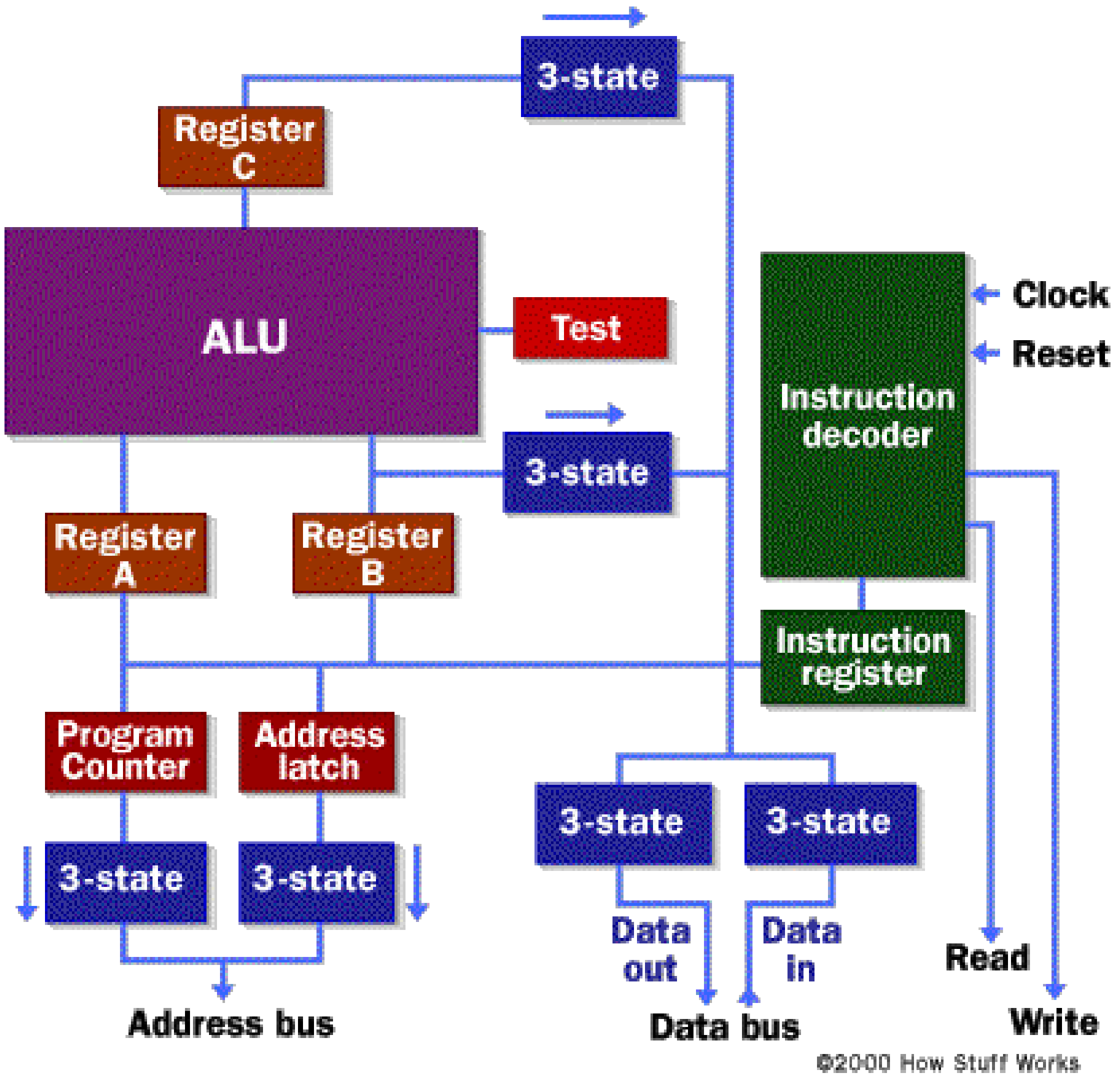


Wallace Tree

- Reduces depth of adder chain
- Built from carry-save adders
  - Three inputs a, b, c
  - Produces two outputs y,
  - y + z = a + b + c
- Carry-save equations
  - y_i = parity (a_i,b_i,c_i)
  - z_i = majority (a_i,b_i,c_i)
- At each stage, i numbers are combined to form 2i/3 sums
- Final adder completes the summation
- Wiring is more complex



Basic Processor Architecture



Processor Units

- **Registers keep data and instructions**

  Registers A, B, C and the address register are made of flip-flops

  The program counter is a register that can increment by 1 and reset to 0

- **An ALU might add, subtract, multiply and divide n-bit numbers**

  The ALU could be as simple as an 8-bit adder

- **The test register holds values of comparisons performed in the ALU**

  An instruction decoder uses these values to make decisions

- **A tri-state buffer passes a 1, a 0 or disconnects its output**

  Multiple outputs can be connected to a wire but only one actually drives a 1 or a 0 onto the line

- **The instruction register and instruction decoder are responsible for controlling all of the other components**

Instruction Operators

Operator Type	Examples
Arithmetic and Logical	Integer Operations: Add, Subtract, And, Or
Data Transfer	Loads/Stores (Moves With Memory Addressing)
Control	Branch, Jump, Procedure Call and Return
System	Operating System Call, Virtual Memory Instructions
Floating Point	Floating-point Operations: Add, Multiply
Decimal	Decimal Operations: Add, Multiply, Decimal-to-character
String	Move, Compare, Search
Graphics	Pixel Operations, Compression/Decompression

← **Instructions**

Percentage of Executed →

		Total Executed on Integers
1	Load	22%
2	Conditional Branch	20%
3	Compare	16%
4	Store	12%
5	Add	8%
6	And	6%
7	Sub	5%
8	Reg-Reg Move	4%
9	Call	1%
10	Return	1%

An Example: The ADD Instruction

- **Instruction can be broken down as a set of sequenced operations**
  - **They manipulate the components of the processor**
- **During the first clock cycle, the instruction decoder needs to**
  - **Activate the tri-state buffer for the program counter**
  - **Activate the RD line**
  - **Activate the data-in tri-state buffer**
  - **Latch the instruction into the instruction register**
- **During the second clock cycle, the ADD instruction is decoded**
  - **Set the operation of the ALU to addition**
  - **Latch the output of the ALU into the register**
- **During the third clock cycle, the program counter is incremented**
  - **This could be overlapped into the second clock cycle**

Pipelining

- **Pipelining increases instruction throughput by breaking up the instruction into different stages**

  Instruction fetch (IF) **- obtaining the requested instruction from memory**

  Instruction decode (ID) **- decoding the instruction and sending out the various control lines**

  Execution (EX) **- performing any calculations**

  Memory and IO (M) **- storing and loading values to and from memory**

  Write back (WB) **- writing the result of a calculation, memory access or input into the register**

  There are more instructions in various execution stages simultaneously

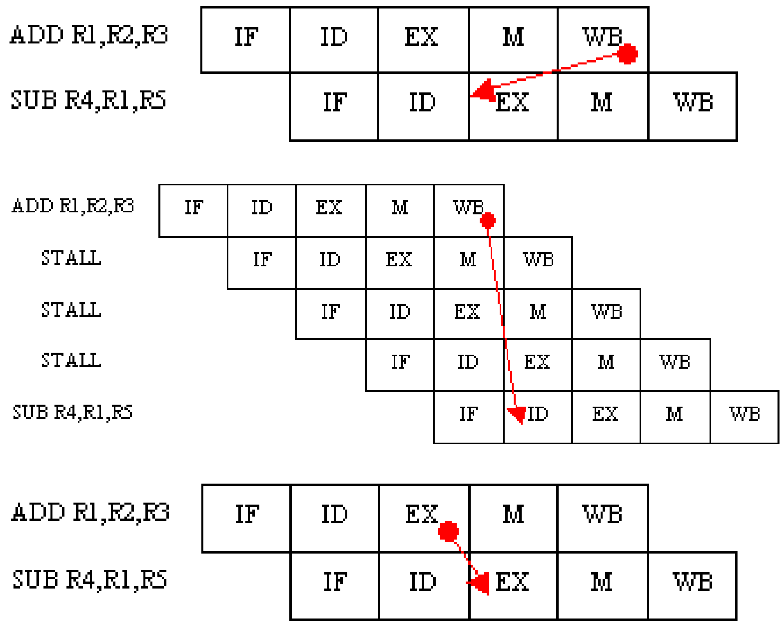
  It looks like one instruction completes every clock cycle

Hazards

- **Data hazards - Instruction attempts to use the result of previous instructions not yet finished**

  Stalling **- Halting the instruction flow until the required result is ready to be used**

  Forwarding **- The result is forwarded from the EX stage of previous to the EX stage of next instruction**

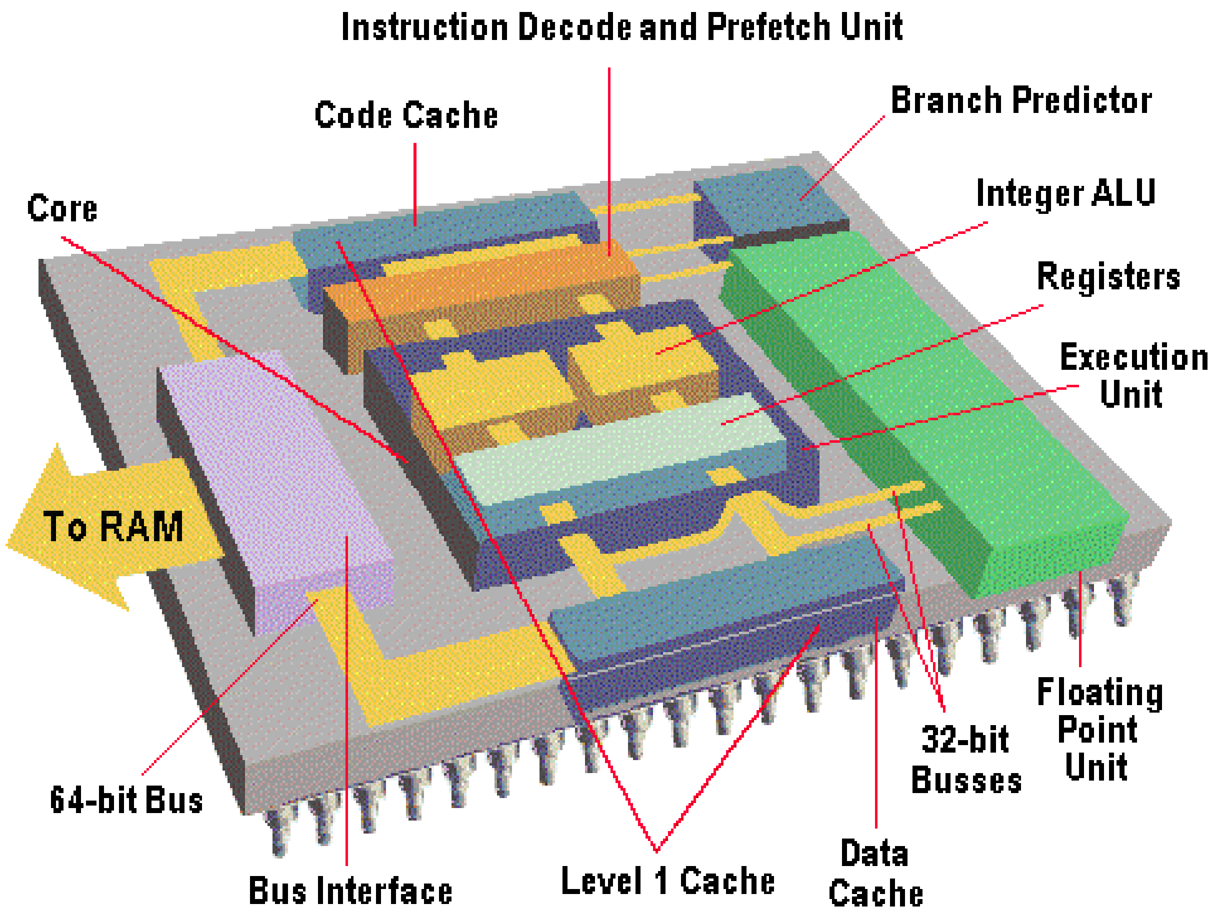


- **Control hazards - Changes in the normal program execution flow**

  Events such as branches, interrupts and exceptions

- **Structural hazards - The hardware is unable to handle certain combinations of instructions simultaneously**

Full Processor Architecture



Processor Program Properties

- **Programs tend to reuse already used data and instructions**

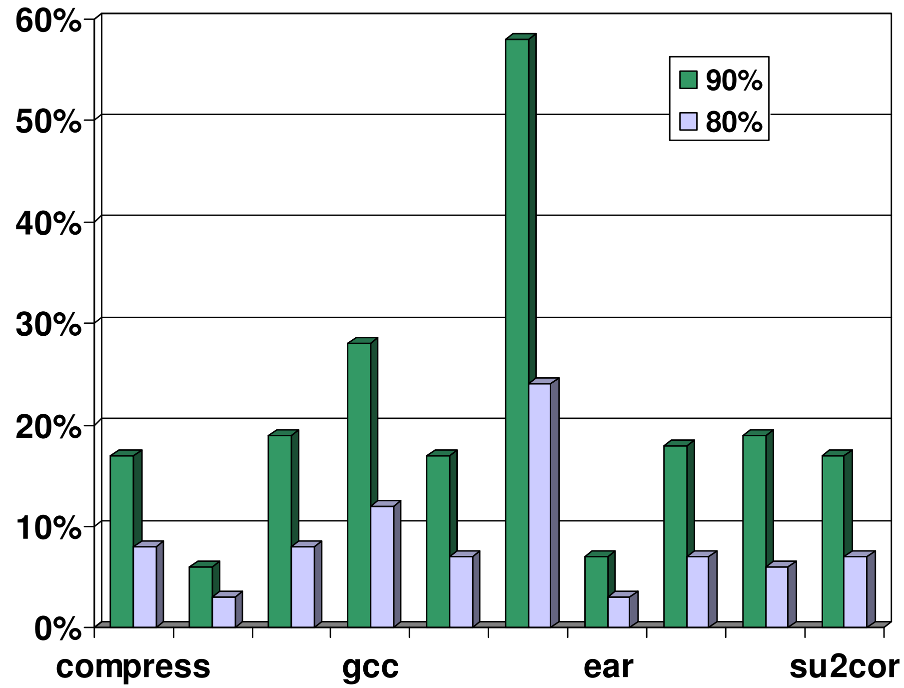
  A rule of thumb: A program spends 90% of its execution time in only 10% of the code

- **Temporal locality**

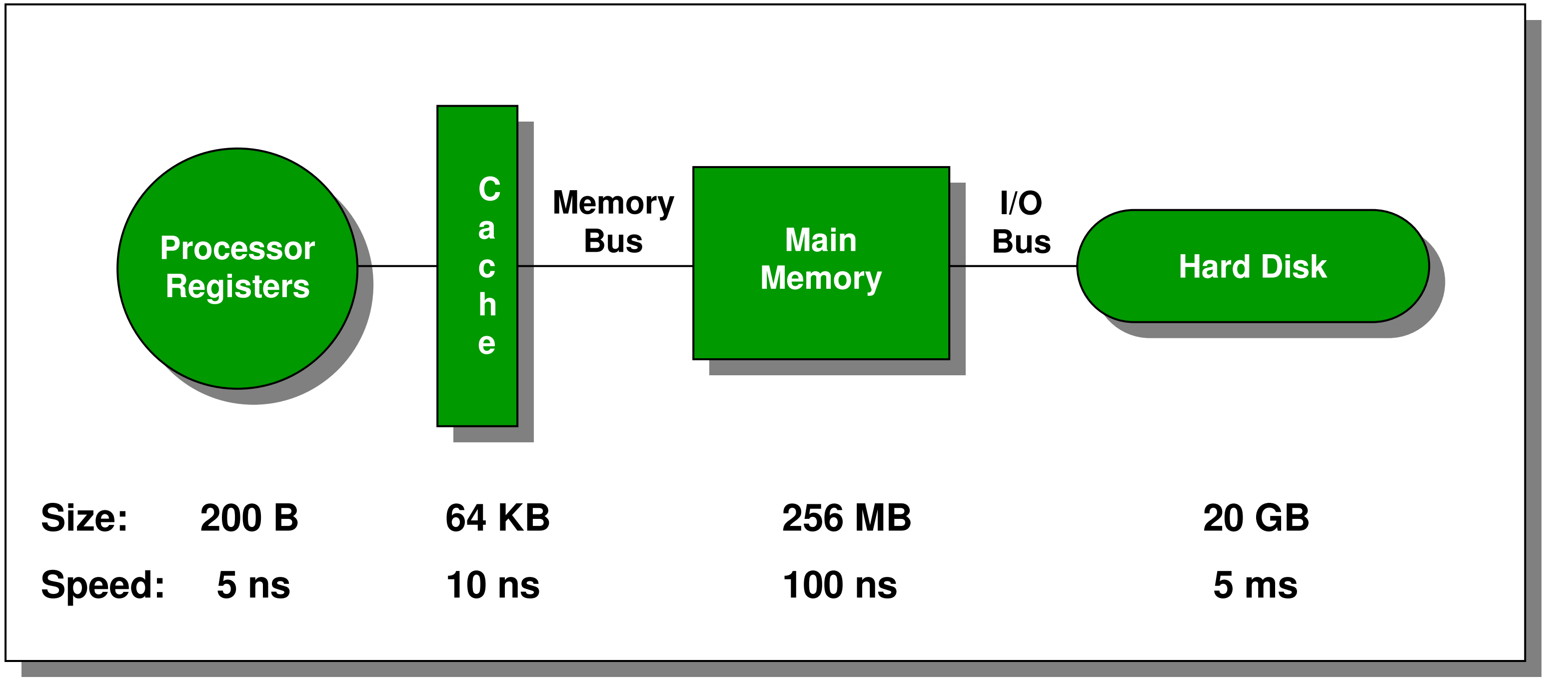
  Recently used items are likely to be used in the near future

- **Spatial locality**

  Items with addresses near each other tend to be used close together in time



Memory Hierarchy



$$Speedup = \frac{1}{(1 - Fraction_{cache}) + Fraction_{cache} / Speedup_{cache}}$$

Cache Features

- **A cache is a small, fast memory located close to the processor that holds the most recently accessed instructions or data**

  When requested data are found in the cache, it is a *cache hit*

  If the access fails, a *cache miss* occurs and the request is forwarded to the main memory

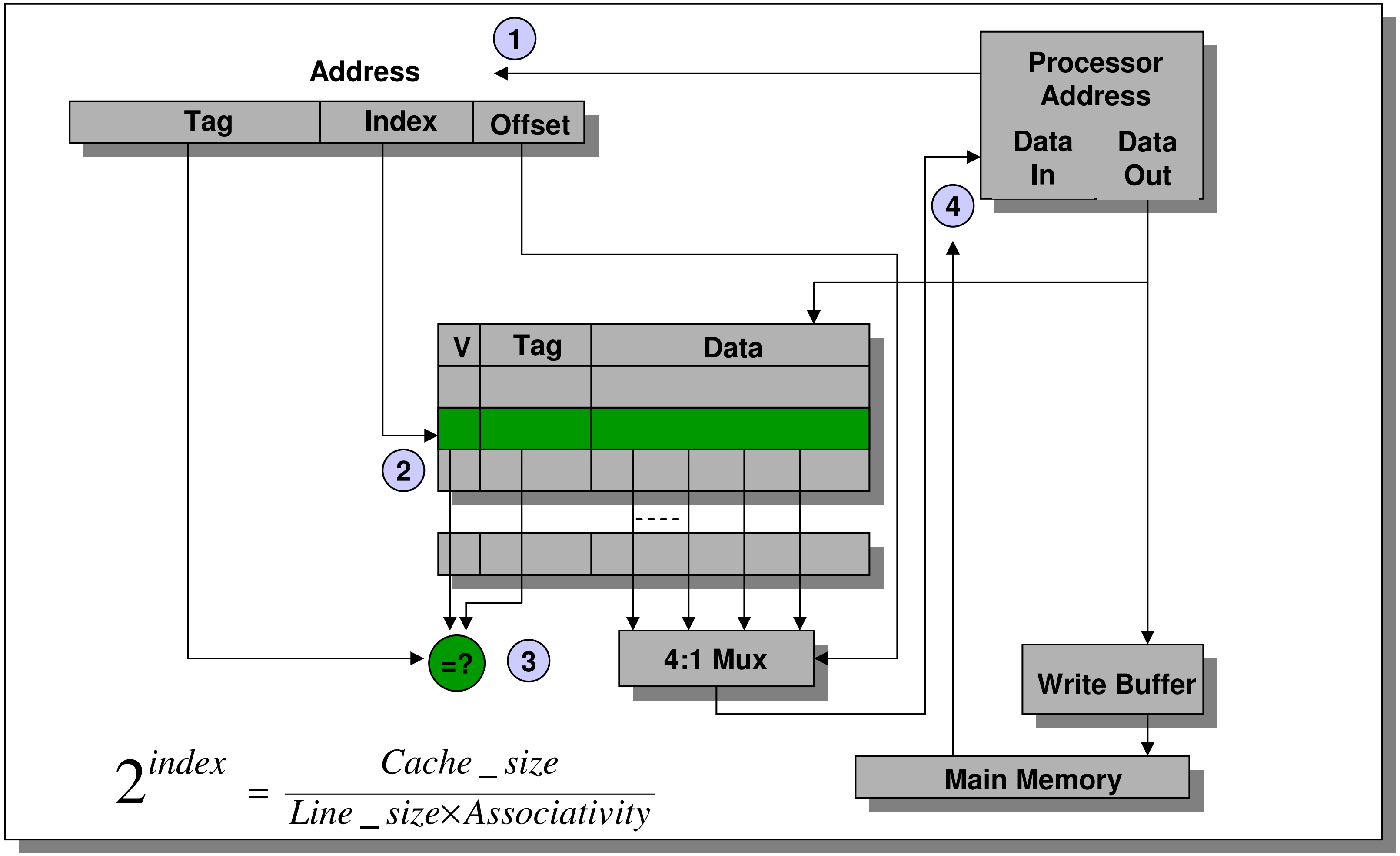
- **A cache is of use only if it reduces the mean access time significantly (the hit-rate must be high)**

  The hit-rate can be further increased by pre-fetch mechanisms

- **After the process change, the cache hit-rate is small (cold cache)**

  The cache warm-up time has to be taken into account for task switching

Cache Organization



$$2^{index} = \frac{Cache\_size}{Line\_size \times Associativity}$$

Cache Addressing

- **A 16 KB cache memory with a 16-byte cache line has 1024 cache lines**

  A system with 16 MB of main memory has 1024 different 16-byte lines of memory that must share a cache line

- **To address 16 MB of memory, 24 address lines are needed (2^{24} is 16 M), but 16 KB only requires 14 address lines**

  The 10 lines of difference (since 1024 is 2^{10}) represent an index

  The index selects one of the 1024 different addresses that can use a given cache line

- **No matter how fast the data RAM is, the tag RAM must be slightly faster**

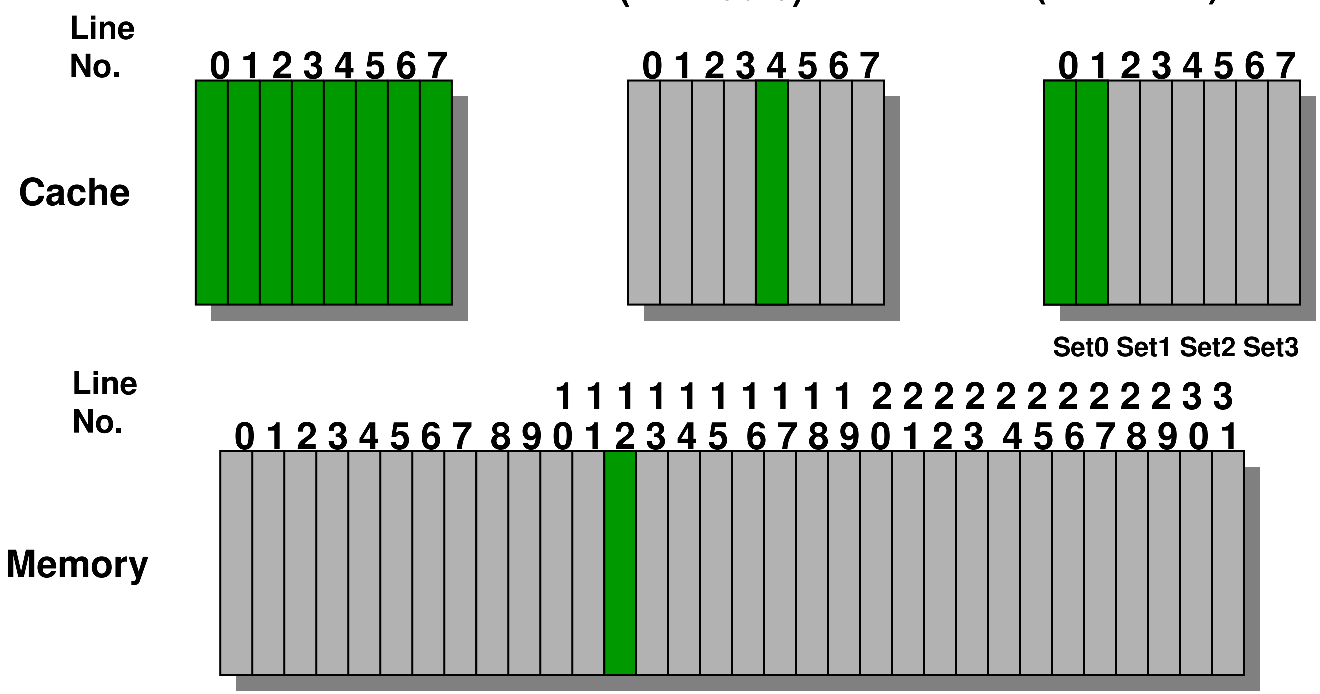
  Tag RAM access time has to be less than half of the clock cycle

Cache Associativity

Fully Associative:
Line12 can go anywhere

Direct Mapped:
Line12 can go only into Line4 (12 mod 8)

Set Associative:
Line12 can go anywhere in Set0 (12 mod 4)



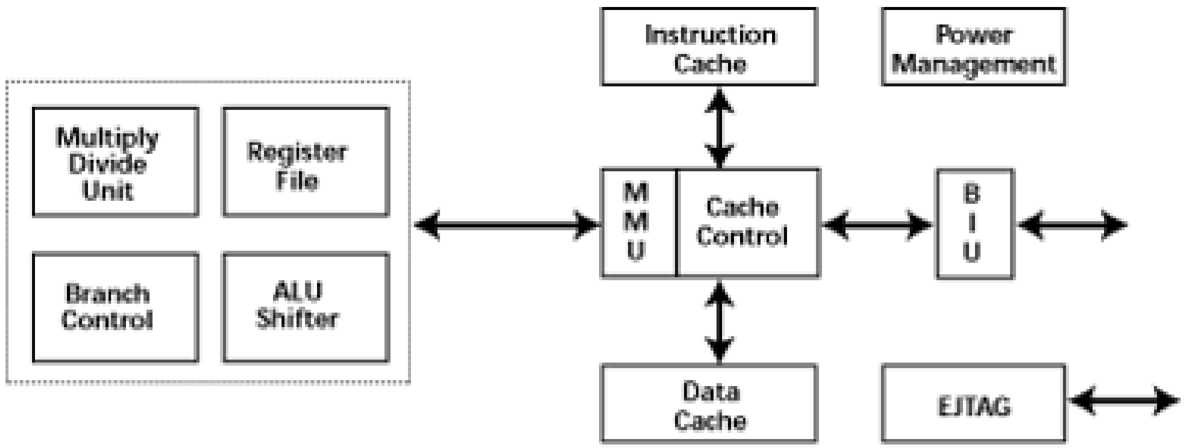
Fully Associative Cache

Best Hit Ratio, Moderate Search Speed

Direct Mapped Cache

Good Hit Ratio, Best Search Speed

General Purpose Processor



Technology	Area (mm^2) (8K I-cache and 8K D-cache)	Power/Frequency (mW/MHz)	Max Frequency (MHz)
0.25 µm	12	2.0	150
0.18 µm	4.7	1.7	260
0.13 µm	1.5	0.45	340

DSP versus GPP

- **Digital Signal Processor**
  - Designed to run *one program*
  - Recent DSPs have instruction caches but *no data caches*
  - *Specialized complex* instructions
  - Specialized hardware performs all key *arithmetic operations in 1 cycle*
  - Hardware support for managing numeric fidelity
  - *Dedicated* address generation unit
- **General Purpose Processor**
  - Designed to run *many programs*
  - May have *caches*
  - *Simple* instructions
  - Multiplies take *more than 1 cycle*
  - Shifts take *more than 1 cycle*
  - Other operations (saturation, rounding…) typically take *multiple cycles*
  - *No separate* address generation unit

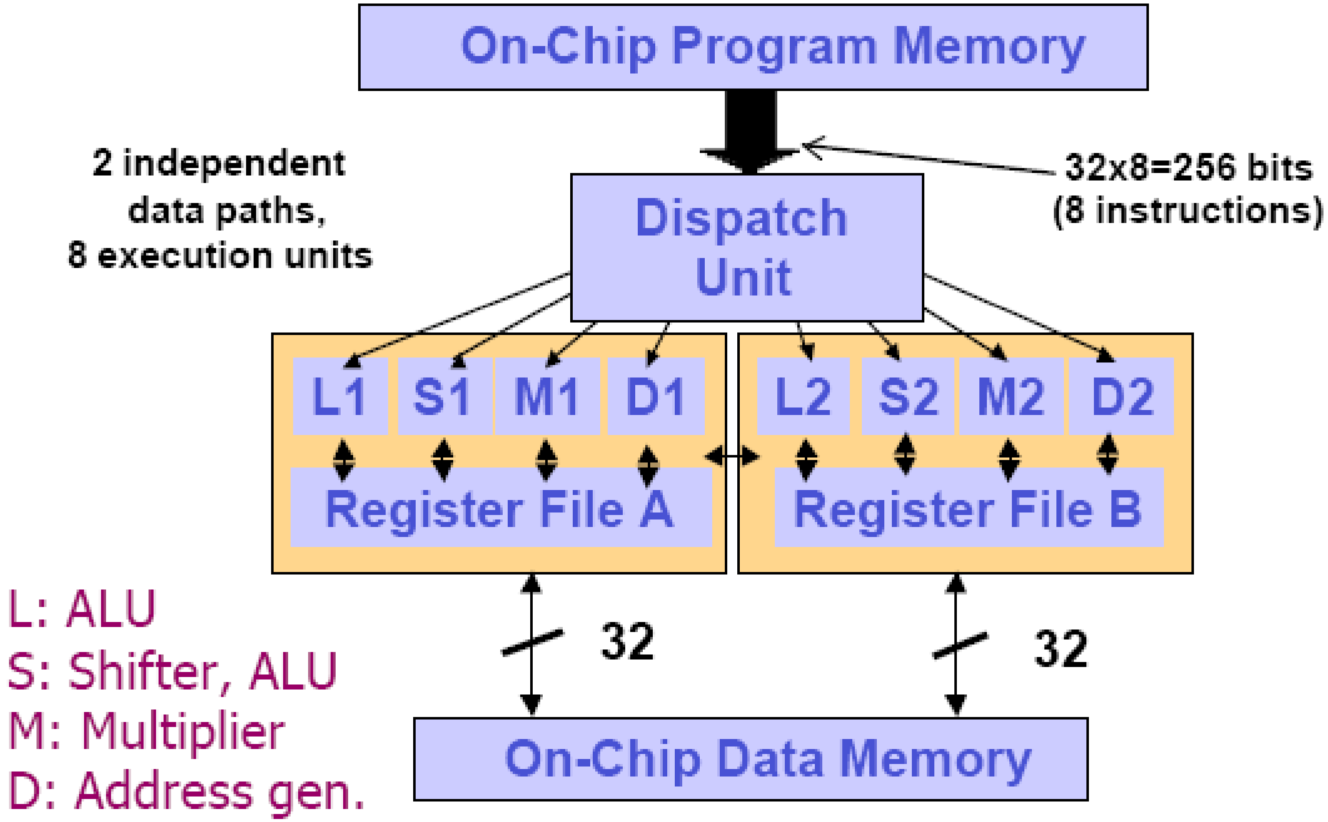
Basics of DSPs

- DSPs can have different data word sizes
  - Word size affects precision of fixed point numbers
- Speed defined by speed of Multiply-Accumulate (MAC) operations
  - A DSP should keep the multipliers busy 100% of the time
- Floating Point versus Fixed Point DSPs
  - More expensive (2-4 times)
  - Much slower
- Basic DSP algorithms
  - Infinite Impulse Response (IIR) Filters
  - Finite Impulse Response (FIR) Filters
  - Fast Fourier Transformers (FFT)
  - Convolvers
  - Turbo Decoders

VLIW DSPs

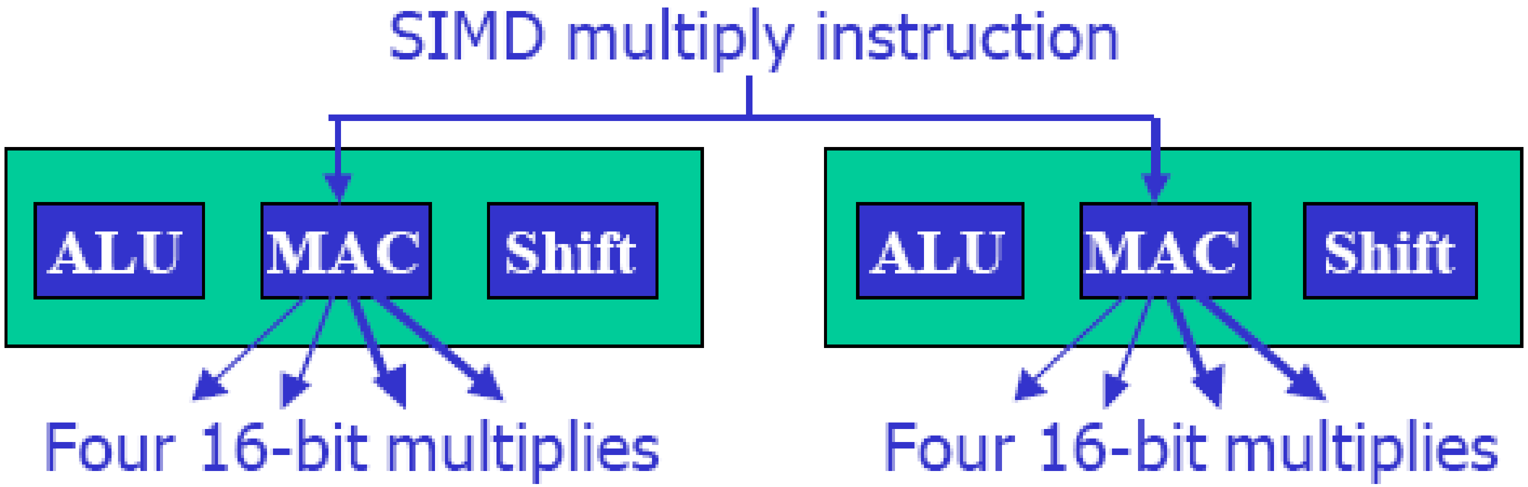
- VLIW stands for Very Long Instruction Word
  - Multiple independent instructions per cycle
  - Packed into single large "instruction word"
  - May be positional, or
  - May include routing information within sub-instructions
- Large complement of independent execution units
- More regular, orthogonal, RISC-like instructions
  - Usually wider than typical DSP instructions
  - Usually simpler than typical DSP instructions
- Large, uniform register sets
- Wide program and data buses

Texas Instruments TMS320C62

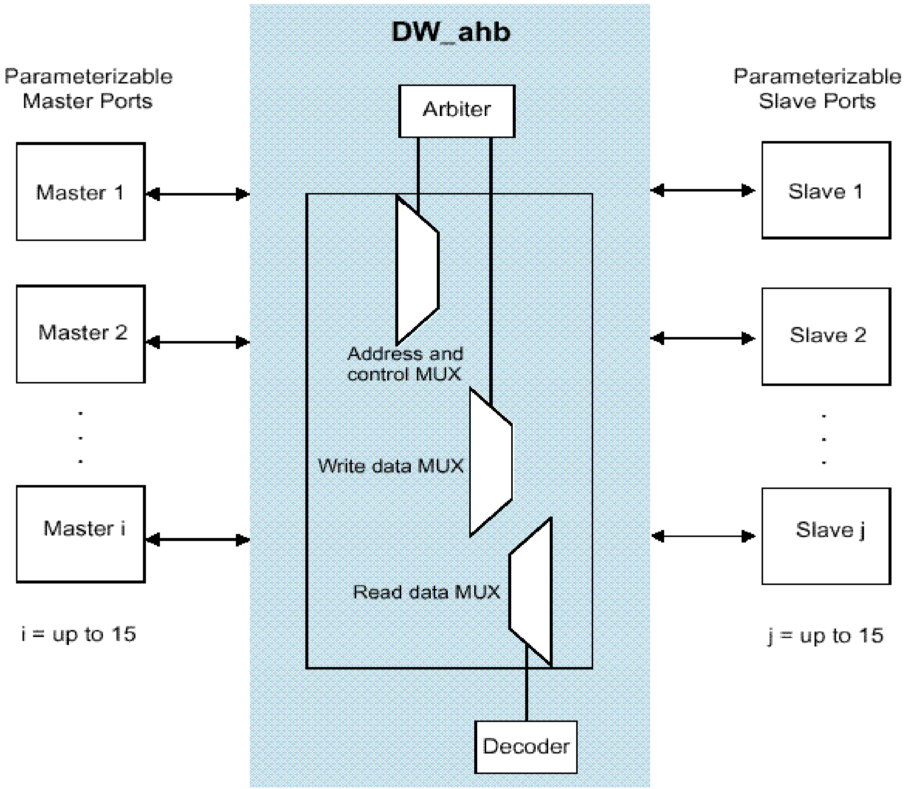


Analog Devices TigerSHARC

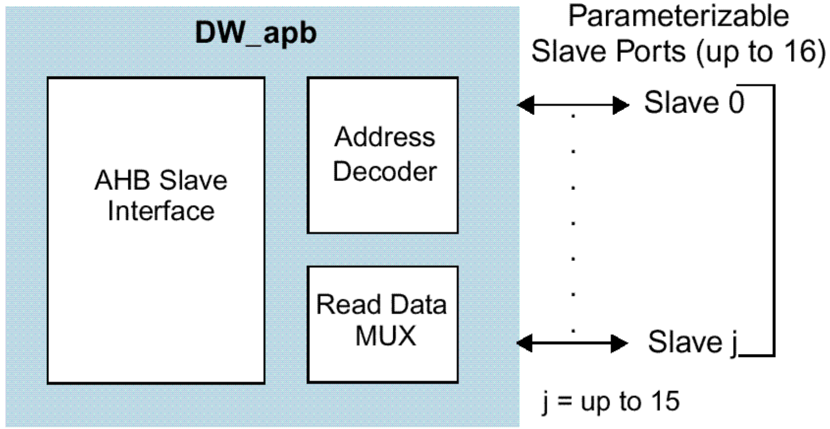
- 8-bit, 16-bit, 32-bit fixed point
- 32-bit floating point
- Single instruction, multiple data
- Executes up to four instructions per cycle
- Uses 32-bit instructions



Advanced Microcontroller Bus Architecture



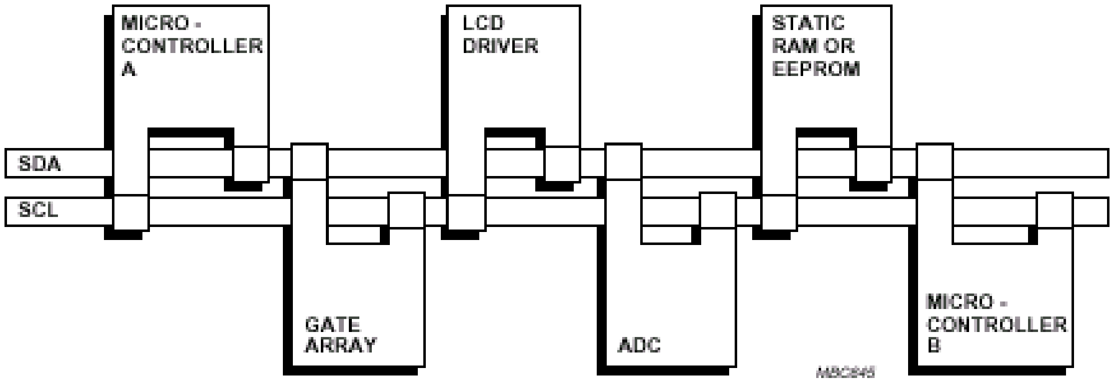
Advanced High-performance Bus (AHB)



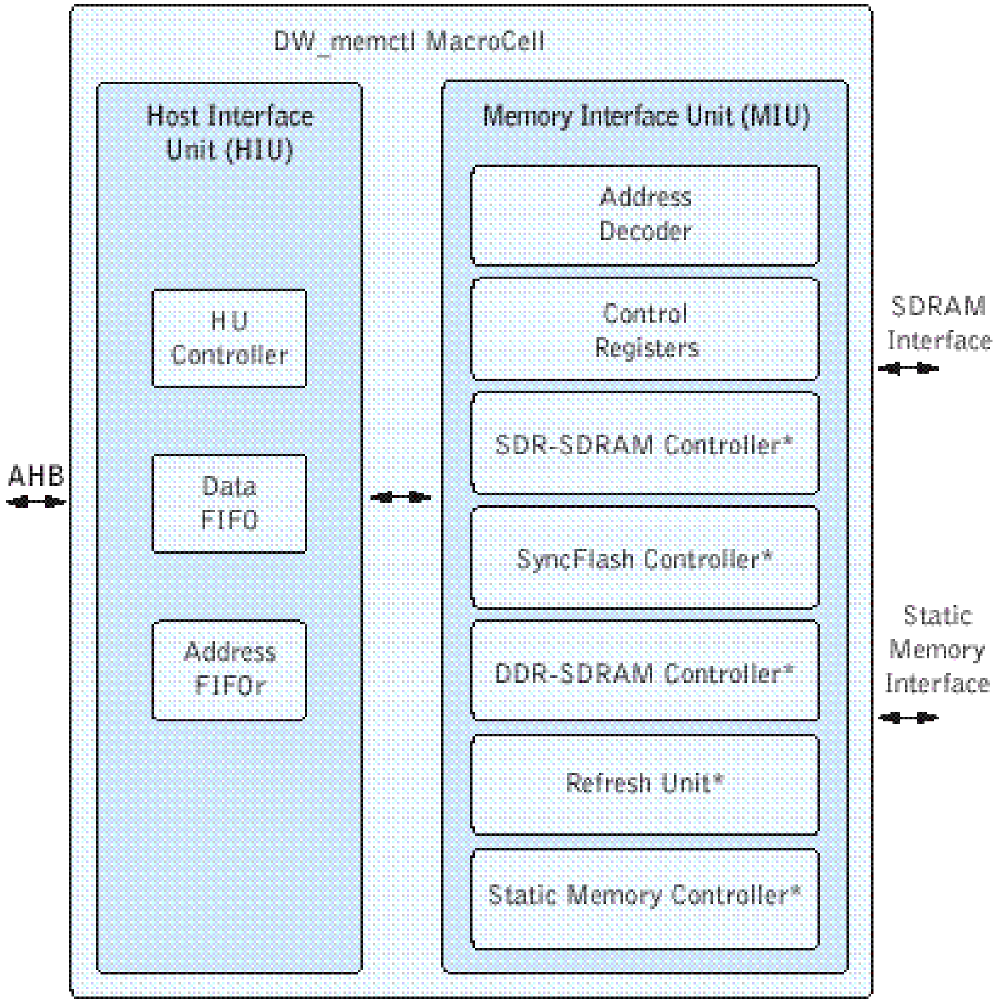
Advanced Peripheral Bus (APB)

Inter-IC Bus

- I^2C bus has two wires
  - Serial data (SDA)
  - Serial clock (SCL)
- A multi-master bus
  - More than one device with the controlling capability
- I^2C interface
  - Recognized by a unique address of 7 or 10 bits
  - Operates as either receiver only device or transmitter

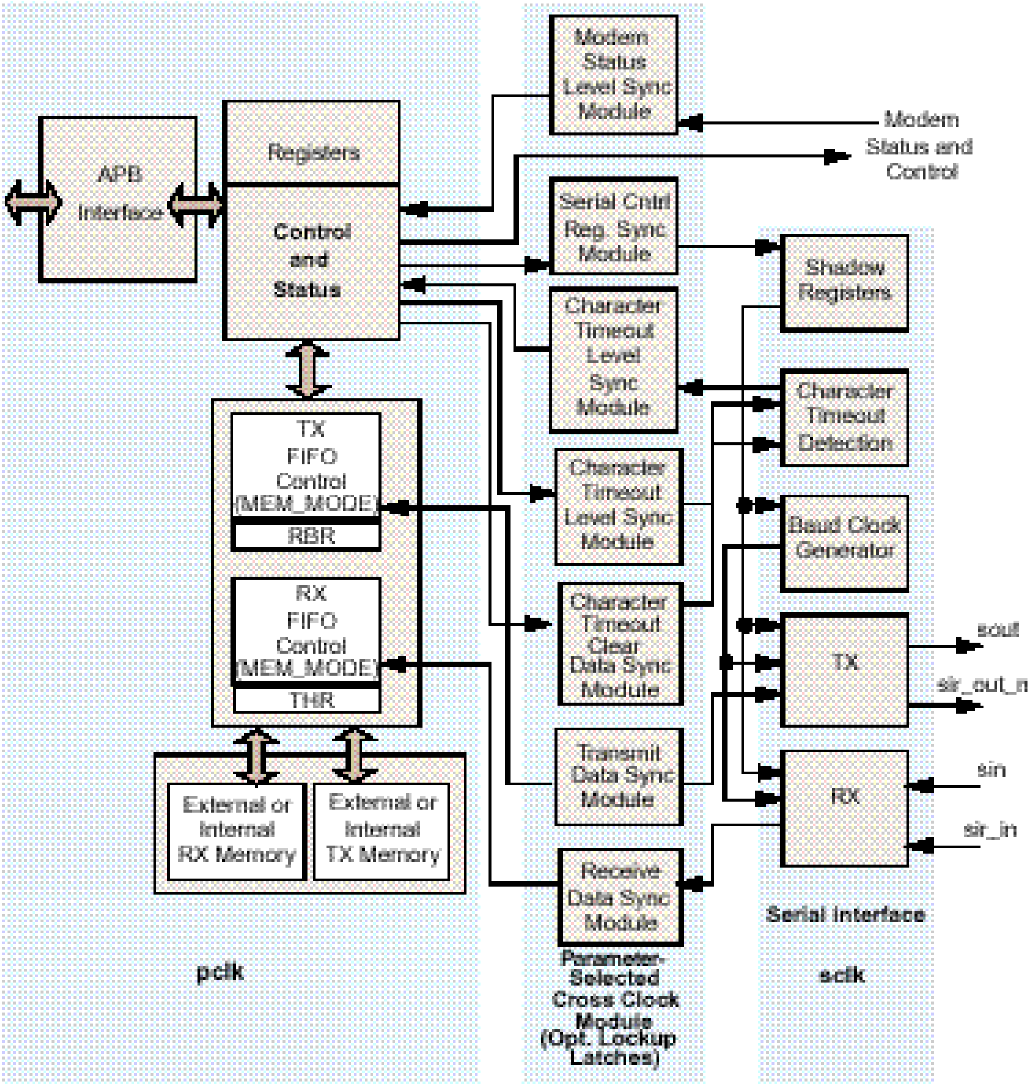


Memory Controller

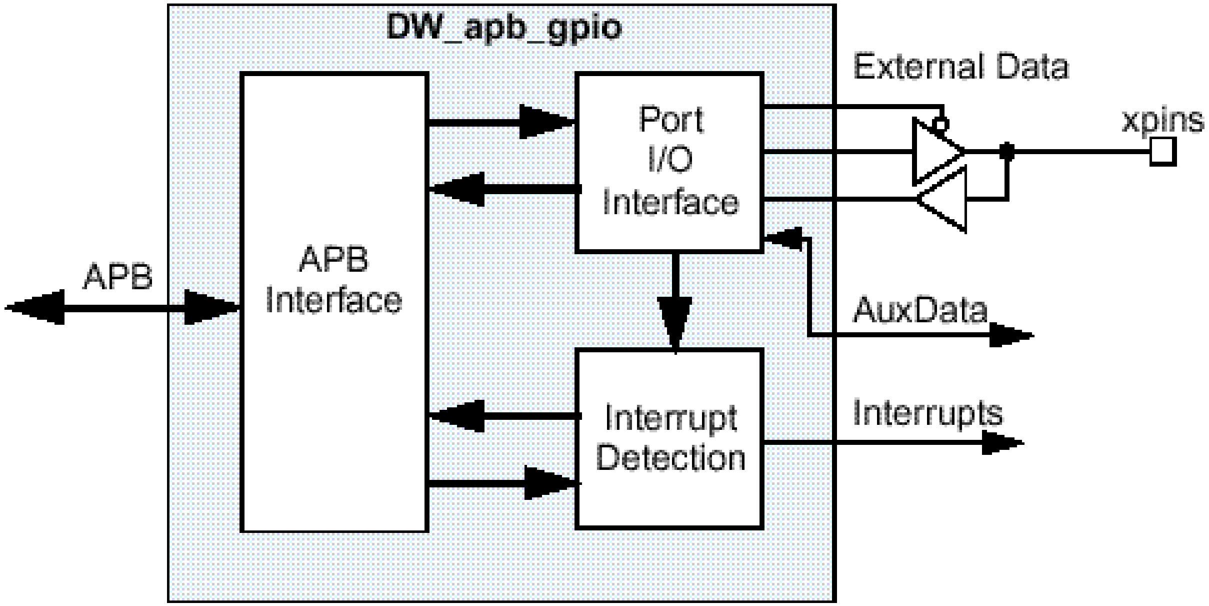


Note: *Conditional Instantiations

Universal Asynchronous Receiver and Transmitter



General Purpose Input/Output



JTAG Debug Link