System-On-Chip Technologies

1. General

$10^{\pm}$	21	18	15	12	9	6	3	2	1
+	Z zetta	E exa	P peta	T tera	G giga	M mega	k kilo	h hecto	da deca
−	z zepto	a atto	f femto	p pico	n nano	µ micro	m milli	c centi	d deci

2. SoC Paradigm

2.1. Moore's Law

Chip capacity (transistors, performance) doubles every 18–24 month

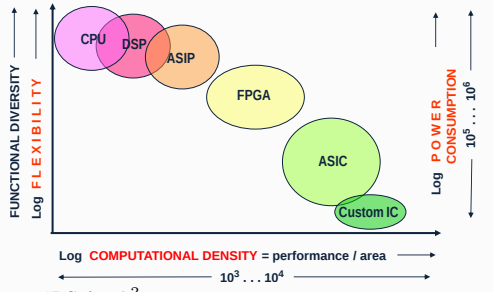
2.2. Challenges

Optimization: Time-To-Market, Price, Performance, Power Cons.
Productivity: reuse components, shorter development cycles, higher chances for (first time) fault-free design

2.3. Chip Platforms

Computation Density (CD) vs Functional Diversity (FD)

Platform	CD	FC	units	costs
CPU	40 – 80	256 – 16k	ALU	
DSP			Multiplier	very few
ASIP			special exec units	few
FPGA	400	1	LUTs	thousands
ASIC	4 000	10	standard cells	millions
Cust. IC	> 10 000	≈ 0	transistors	10 millions

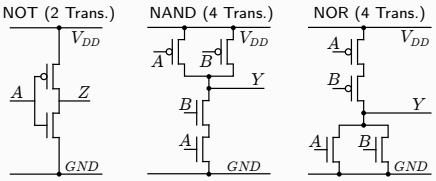


$$CD = \frac{\text{IPC} \cdot f \cdot w \cdot \lambda^2}{A}$$

Instructions per cycle IPC, structure size λ, area A
Frequency f, Wordsize w (e.g 32 Bit)

2.4. CMOS

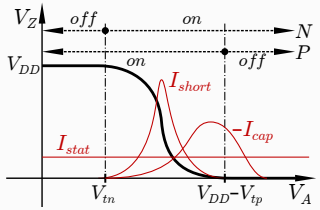
Complementary Metal (Poly-Si) Oxide (SiO2) Semiconductor
Why? Low power dissipation, Noise immunity, Clean logic levels, One supply voltage, Cascadable, Easy to design, Fabrication well understood



2.5. MOSFET

channel width	$W_{n/p}$
channel length	$L_{n/p}$
gate oxide thickness	t_{ox}
electron mobility	$\mu_n \approx 250 \times 10^{-4} \frac{m^2}{V\,s}$
	$\mu_p \approx 200 \times 10^{-4} \frac{m^2}{V\,s}$
rel. permittivity of gate oxide	$\epsilon_{ox} \approx 3,9$
dielectric constant	$\epsilon_0 = 8.854\,187\,8 \times 10^{-12} \frac{A\,s}{V\,m}$
specific oxide capacity	$C'_{ox} = \frac{\varepsilon_{ox}\varepsilon_0}{t_{ox}}$
oxide capacity	$C_{ox} = C'_{ox} \cdot WL$
gain (also β)	$K_n = \mu_n C'_{ox} \frac{W_n}{L_n}$
	$K_p = (-1)\mu_p C'_{ox} \frac{W_p}{L_p}$
propagation delay	$t_{pHL} \propto \frac{C_L t_{ox} L_p}{W_p \mu_p \varepsilon_{ox}(V_{DD} - \lvert V_{th}\rvert)}$

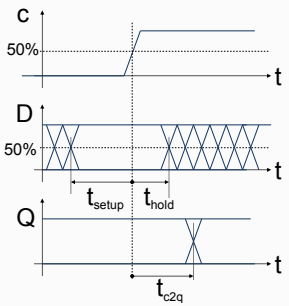
2.6. Inverter

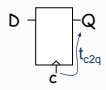


Dynamic Power Consumption	$P_{dyn} = P_{cap} + P_{short}$
Capacity Power	$P_{cap} = \alpha_{01} f C_L V_{DD}^2$
Short Circuit Power:	$P_{short} = \alpha_{01} f \beta_n \tau (V_{DD} - 2V_{th})^3$

3. SoC Components

3.1. Sequential Logic





Flip-flop characteristics:
setup-time: Data must be stable t_{setup} before clock edge
hold-time: Data must be stable for t_{hold} after clock edge
clock-to-output delay: Data will be visible at output t_{c2q} after clock edge

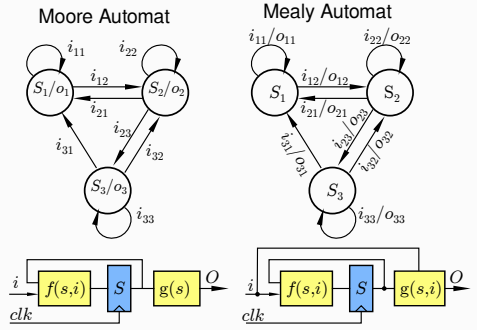
t_{Setup}	setup before clock edge
t_{hold}	hold after clock edge
t_{c2q}	output valid after t_{c2q}
Max. clock period	$t_{clk} \geq t_{1,c2q} + t_{logic,max} + t_{2,setup}$
Max. clockfrequency	$f_{max} = \lfloor \frac{1}{t_{clk}} \rfloor$ (Nicht aufrunden)
hold time condition	$t_{hold} \leq t_{c2q} + t_{logic,min}$ → Dummy Gate
Durchsatz	$\frac{1\text{Sample}}{t_{clk,pipe}} = f$
Latenz	$t_{clk} \cdot$ #Pipelinestages (#FFs - 1)
Slack	$t_{slack} = t_{available} - t_{required}$

3.2. Karnaugh-Maps

$z \backslash xy$	00	01	11	10
0	1	0	0	0
1	X	1	1	0

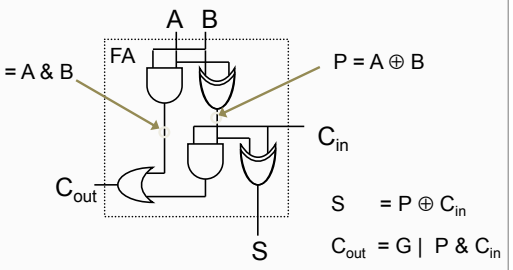
Combine equal cells:
e.g. $\overline{xy} + y \cdot z$
Use don't care values!

3.3. Finite State Machines



Synchronous System Design paradigm: Essentially all control functions in state-of-art digital IC's consist of "communicating FSMs".
Avoid combinatorial logic through paths!
Stick to one FSM design style across SoC!

3.4. Adder



3.4.1 Ripple-Carry

Worst-Case$(G, P = 1)$: $t_{add} = (N-1)t_{carry} + t_{sum}$
Number of input Bits/Full Adders N

3.4.2 Carry-Bypass

Fast carry propagation (useful if $N > 4$)
$t_{CBA} = t_{setup} + Bt_{carry} + (\frac{N}{B} - 1)t_{skip} + (B-1)t_{carry} + t_{sum}$
witch group size in bits B. t_{CBA} still $\mathcal{O}(N)$, but with more graduate slope

3.4.3 Carry-Select

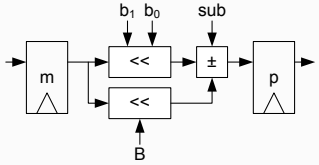
Precompute C_{out} for $C_{in} = 0$ and $C_{in} = 1$ for all blocks in parallel. Then select the correct one. $t_{CSA} = t_{setup} + Bt_{carry} + \frac{N}{B} t_{mux} + t_{sum}$
Square Root Carry-Select Adder:
$t_{SCS} = t_{setup} + Mt_{carry} + \sqrt{2N} t_{mux} + t_{sum}$

3.5. Multiplier (Addition of partial products)

$$x \cdot y = \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} x_i y_j \cdot 2^{i+j}$$ Result requires $N + M$ bit

3.5.1 Repeated Addition

3.5.2 Sequential: Right Shift and Add



3.5.3 Array Multiplier

All partial products generated in parallel and organized in adder array with respective offset
$t_{mul} = [(M-1)(N-2)]t_{carry} + (N-1)t_{sum} + t_{and}$

3.6. Shifter

Single-bit left/ right shift operations through individual pass transistors
Barrel Shifter: Words pass through maximum one transmission gate

3.7. Multiplexer

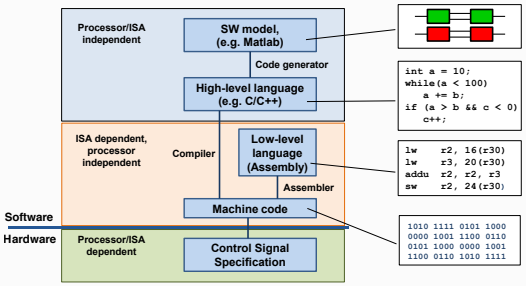
Mux: $Z = \overline{S}A_1 + SA_1$ DeMux: $Z_1 = \overline{S}A, \quad Z_2 = SA$

4. Processor Structure

4.1. Processor Classification

	Type	Application	Characteristic	Remark
Instruction complexity	RISC	Embedded control	Load/store instructions for memory access	MIPS, ARM, PowerPC
	CISC	Personal Computer/ Servers	Complex, variable-length instructions	Intel x86-based
Instruction-level parallelism (ILP)	Superscalar	Personal Computer/ Embedded	Instruction parallelism on run-time	Intel, ARM, PowerPC
	VLIW	Image Processing	Instruction parallelism on compile-time	Parallel video pixel processing
Application-specific area	ASIP	Embedded	Application-specific intructions	Tensilica
	DSP	Signal Processing	HW multiply for digital filters	TI

4.2. Software Levels

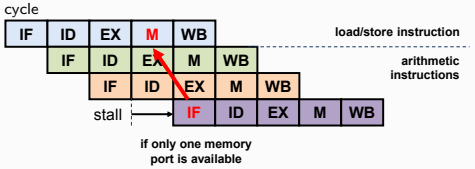


4.3. Multi Cycle Core

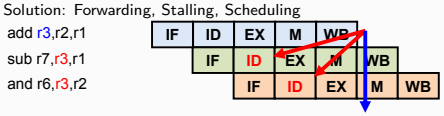
1. **Instruction Fetch (IF):** increase PC
2. **Instruction Decode (ID):** read OP and register
3. **Execution (EX):** ALU executes command
4. **Memory Stage (M):** read/write to memory
5. **Write Back (WB):** load from memory to register?

4.4. Hazards (problems due to pipelining)

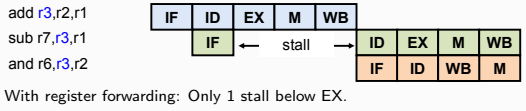
Structural Hazard: same resource is needed multiple times in the same cycle



Data Hazard: data dependencies (read-after write, write-after-write, write-after-read).
Solution: Forwarding, Stalling, Scheduling

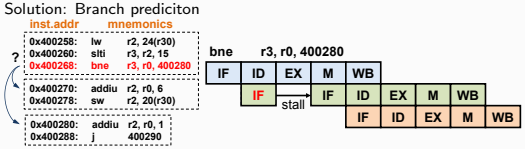


Stalling is required



With register forwarding: Only 1 stall below EX.

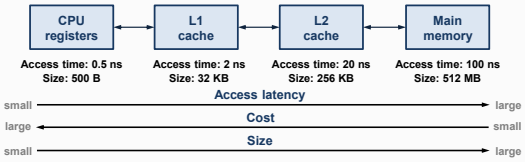
Control Hazard: next executed instruction is not the next specified instruction due to jump, branch, exception.
Solution: Branch prediciton



4.5. Processor Performance

$$\text{CPUTime} = \underbrace{\frac{\text{Instructions}}{\text{Program}}}_{\text{Estimate}} \cdot \underbrace{\frac{\text{ClockCycles}}{\text{Instruction}}}_{\text{CPI}} \cdot \frac{\text{Seconds}}{\text{ClockCycle}}$$

5. Memory



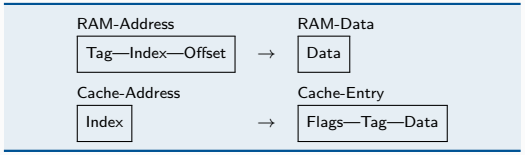
Type	Used	Speed	Density
Register	CPU Registers $32 \cdot 64$ bit	$<$ ns	
On-Chip SRAM	Cache 32 kByte	ns	
DDR3 SDRAM	Main Memory $\approx$ GByte	$2 \cdot 800$ MHz	
HDD	Mass Storage $>$ TB	$150 \frac{\text{MB}}{\text{s}}$	
ROM	Sys. Config few kByte	$\approx$ kB/s	

5.1. CMOS Memory

Register: 2 Inverters (Q,$\overline{\text{Q}}$)
Latch: 4 NANDS (e,D,Q,$\overline{\text{Q}}$)
Flip-Flop: (Q,$\overline{\text{Q}}$, clk)

5.2. Cache

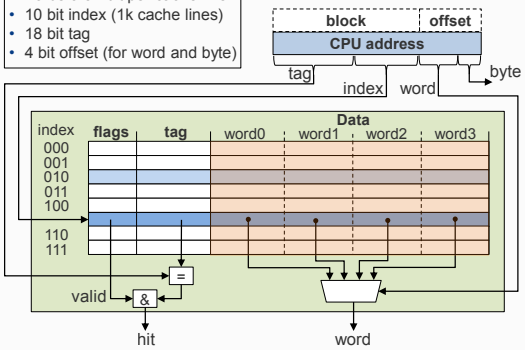
Caches store only small share of main memory. The Cache maps RAM-Addresses to Cache-Entries. One Cache-Entry can contain several Data Bytes. The Tag verifies the mapping, the Valid-Flag verifies the actuality of the cached data.



#RAMAddressBits = #TagBits + #IndexBits + #OffsetBits
#CacheEntries = $2^{\text{Indexbits}}$
#CacheDataBytes = $2^{\text{Offsetbits}}$ (Byte accurate Cache access)
CacheSizeInByte = #CacheEntries · #CacheDataBytes

5.2.1 Direct Mapped Cache

Example: 16 KB direct mapped
4 words à 32 bit per cache line
- 10 bit index (1k cache lines)
- 18 bit tag
- 4 bit offset (for word and byte)



Replace Strategy: Replace old with new.

5.2.2 Set-Associative-Cache

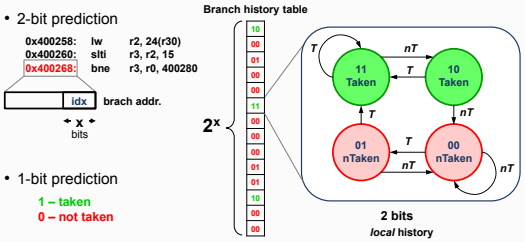
Blocks with equal Index can be stored in n cache entries. Tag needed to distinguish. Replace Strategy: Replace if all sets are full. Random, FIFO or LRU (least recently used)

$\frac{\#\text{Zugriffe}}{\text{Zeit}}$ = Hit-Rate · Hit-Time + Miss-Rate · Miss-Time

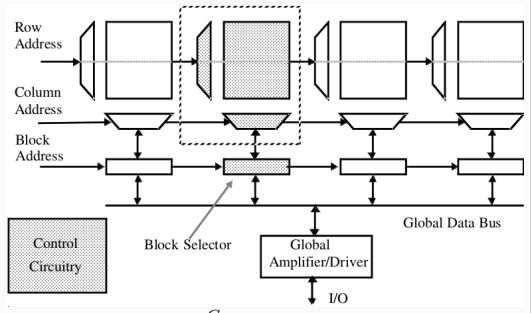
5.2.3 Fully Associative Cache

A memory block can be stored in any cache entry.

5.3. Branch Prediction



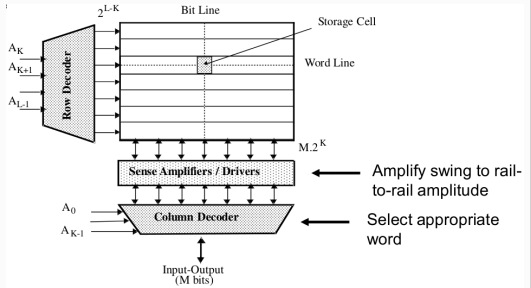
5.4. Main Memory



$\Delta V = (V_X - V_{\text{Pre}}) \cdot \frac{C_S}{C_S + C_{\text{BL}}}$ $V_{\text{Pre}} \approx 0.5 V_{\text{DD}}$

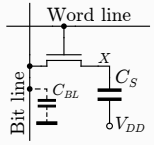
$\Delta V_{\text{BL}} = V_{\text{eq}} - V_{\text{BL}} = \pm \frac{C_S}{C_S + C_{\text{BL}}} \cdot \frac{V_{\text{DD}}}{2}$

5.5. Memory Block

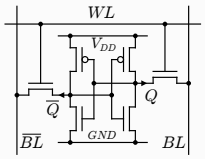


5.6. Memory cell types

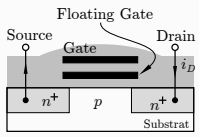
DRAM



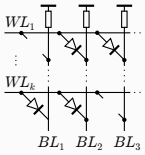
SRAM



Flash



ROM



DRAM Timings: t_{CAS}–t_{RCD}–t_{RP}–t_{RAS}
Access Latency: $t_{\text{Lat}} = t_{\text{CAS}} + t_{\text{RCD}}$
Min. row cycle: $t_{\text{RC}} = t_{\text{RP}} + t_{\text{RAS}}$

Access Times:

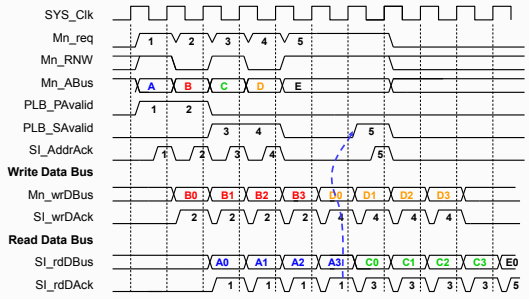
DRAM:	Single:	(cLatency + 1c(amp)) · #reads
	Burst:	cLatency + (#reads-1) + 1c(amp)
SRAM	Single:	cLatency · #reads
	Burst:	cLatency + (#reads-1)

6. Interconnect

6.1. Interconnection

6.1.1 Processor Local Bus (PLB)

Bus-Transaction: Request → Addr. Trans. → Data Trans.→ Data Ack
Burst Transfer: Reduction of Req./Addr. signaling overhead for read/write transactions to consecutive addresses. Burst transfers with implicit address increment



Bus Standard: AMBA (ARM), CoreConnect (IBM), OCP (Sonics), VSIA

6.2. AMBA AHB

Advanced **M**icrocontroller **B**us **A**rchitecture **A**dvanced e**X**tensible **I**nterface

6.3. FIFOs

Are use for decoupling clock domains or word widths.
Pointer: Read (RP) and Write (WP)
Control Flags: Almost Full (AF) and almost empty (AE)

6.4. Network-on-Chips (NoC)

Benefits: Scalability, Synchronization, short point-to-point links
Drawbacks: Latency, Area

7. Low Power Design

7.1. Motivation – Why?

- Reliability: Plus 10°C doubles failure rate
- High currents destroy on-chip wires
- Cooling: higher costs and power consumption

Leakage current: $I_{\text{leak}} \propto \exp(V_{\text{GS}} - V_{\text{th}})$
Gate Delay: $t_{\text{d}} \propto \frac{C_L}{V_{\text{DD}} - V_{\text{th}}}$

7.2. Techniques and Hierarchy

Trade in Power with Performance, Area, Cost

Frequency Scaling and Voltage Scaling (DFS, DVS)
Algorithmic Optimization: $x^2 + ax = x(x + a)$
Power Gating: Switch components off if not needed.
Clock Gating: toggle registers only when outputs can change
Threshold Control: bias threshold voltage V_{th}