ECE4740: Digital VLSI Design

Lecture 29: Final lecture

1094

No ones favorite but crucial!

Design for test (DFT)

1095

Testing is important

- Testing is among the most expensive aspects of digital VLSI design:
  - Logic verification accounts for more than 50% of the design effort for most VLSI designs
  - Debug after fabrication incurs extreme costs
- Example: Intel FDIV bug (1994)
  - Logic error found after >1M parts shipped
  - Recall cost $450M; image loss even worse

1096

Logic verification

- Does the design simulate correctly?
  - Verification engineers develop test-bench
  - Exhaustive tests often not possible
  - Test critical corner cases
- Example: 32-bit adder
  - Exhaustive testing: $>10^{19}$ combinations
  - Test only corner cases: $0,1,2,2^{31}-1,-1,-2^{31}$ etc.

designing good tests is difficult

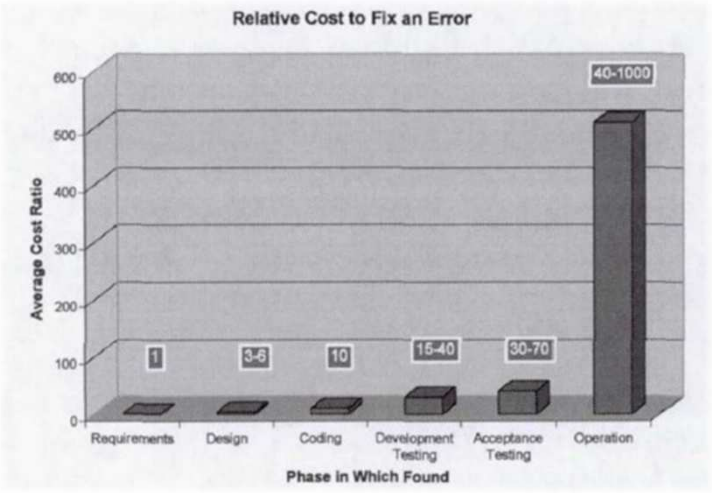
1097

Silicon debugging

- Logic bugs vs. electrical failures
  - Most failures are logic bugs from insufficient simulations, bad testing strategy, etc.
  - Some are electrical failures: crosstalk, leakage, charge sharing, ratio failures, etc.
  - Some fabrication errors: process variation
- Fix the bugs and fabricate corrected chip
- Main goal: *First time right!*

1098

Finding errors late is expensive

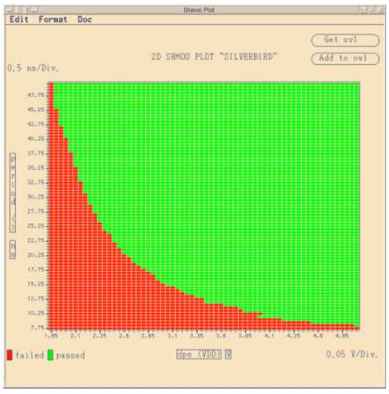


http://www.power-supply-designer.com/review/

1099

How to diagnose failures?

- Hard to access chips:
  - Picoprobes on chip
  - Electron beam
  - Built-in self test (BIST)
- "Schmoo plots"
  - Vary voltage/frequency
  - Look for cause of electrical failures



From P. Luethi (http://www.electronic-engineering.ch/study/silverbird/silverbird.html)

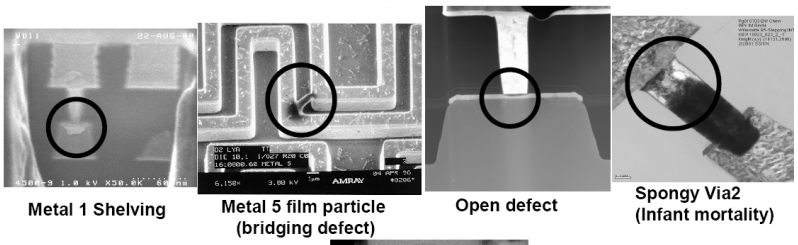
1100

Manufacturing test

- Yield of any chip is <100%
  - Test chips before delivered to customers
- Manufacturing tests are expensive
  - Minimize test time/chip
  - Maximize # of failures you can detect

1101

Manufacturing failures

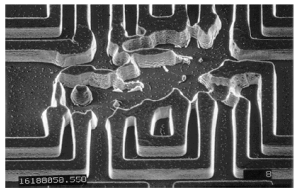


Metal 1 Shelving

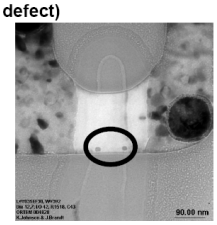
Metal 5 film particle (bridging defect)

Open defect

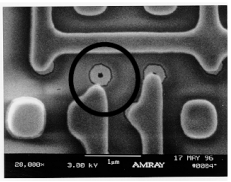
Spongy Via2 (Infant mortality)



Metal 5 blocked etch (patterning defect)



Spot defects "Co" Defect under Gate



Metal 1 missing pattern (open at contact)

1102

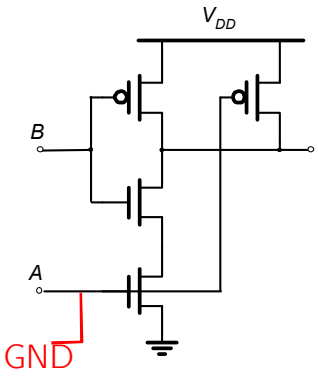
Image adapted from: CMOS VLSI Design: A Circuits and Systems Perspective by Weste, Harris

The "stuck-at" fault model

- Common failures are shorts between two conductors or opens in a conductor
  - Effect of such failures can be very complex
- Use a simpler model:
  - Assume all failures cause nodes to be "stuck-at" 0 or 1, i.e., short to GND or V_{DD}
  - Two types:
    - Stuck-at 1 (SA1)
    - Stuck-at 0 (SA0)

1103

Example stuck-at fault: NAND2

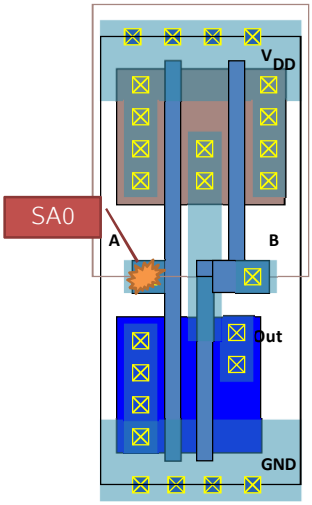


No faults:

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

SA0:

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	1



1104

Observability and controllability

- Observability: able to observe internal node by observing external output (=pins) of chip
- Controllability: able to force internal node to 0 and 1 via inputs (=pins) of chip

- Combinational logic: nodes are rather easy to control and observe
- Sequential logic: extremely difficult or even impossible to control and observe

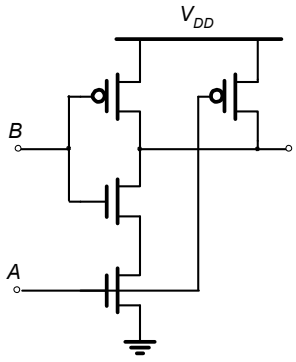
1105

Test pattern generation

- Generate inputs (=stimuli) and expected outputs (=responses) of a given design
- Apply smallest sequence of test vectors to prove that each node is not stuck
- You have to ensure good observability and controllability of internal nodes:
  - Reduces number of test vectors (lower cost)
  - Increases number of nodes that can be tested

1106

Example: NAND2



- Min. test vector set: {01,11,10}

A=SA0

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	1

A=SA1

A	B	Out
0	0	1
0	1	0
1	0	1
1	1	0

B=SA0

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	1

B=SA1

A	B	Out
0	0	1
0	1	1
1	0	0
1	1	0

1107

Design for test (DFT)

- Idea: Design the chip to increase observability and controllability

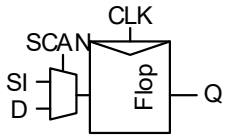
add test structures already during design

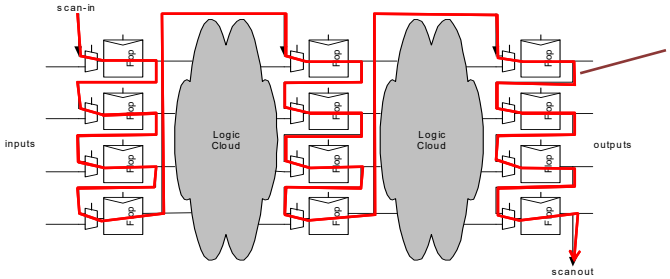
- If each register could be observed and controlled, testing reduces to testing combinational logic between registers
- Even better: logic could enter test mode where they test themselves automatically

1108

Scan chain

- Convert each flip-flop to a scan register
  - Cost = one MUX
  - Scan mode: behaves as a shift register

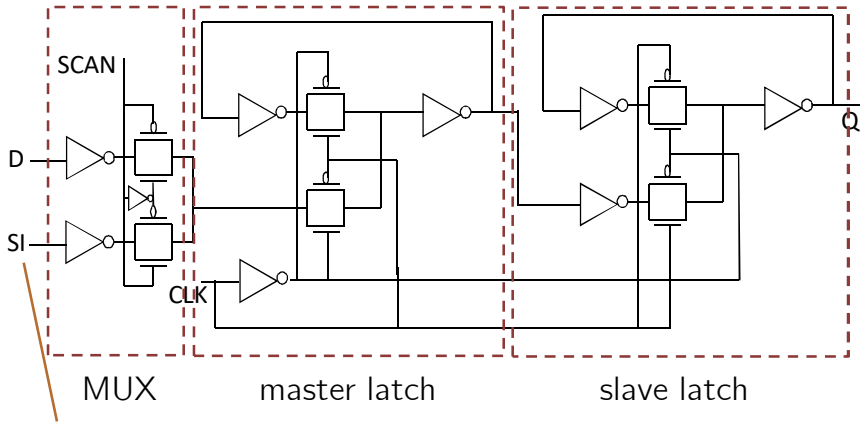




in scan mode, we can set and read the state of each flip-flop

1109

MS ET flip-flop with scan capability

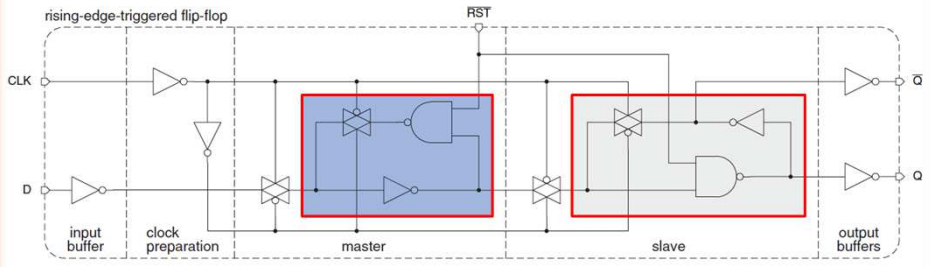


I could easily give one lecture only about scan chains

1110

Be able to reset your circuit

- You must be able to set your circuit in a predefined state (think reset button!)
- Add reset functionality to each flip-flop



From H. Kaeslin, "Digital Integrated Circuit Design," Cambridge Univ. Press, 2008

1111

Automatic test pattern generation

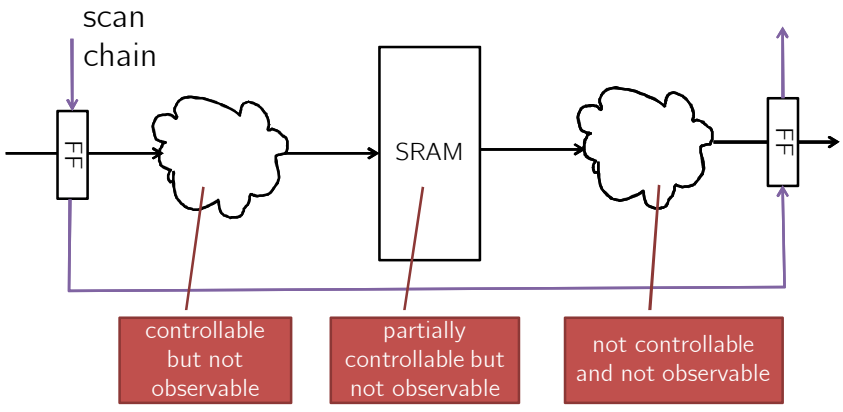
- ATPG produces good set of vectors for each block of combinational logic
  - Scan chains used to control and observe blocks
- Complete coverage of all nodes requires large number of test vectors → expensive
- Most products have >95% test coverage

- Macrocells (SRAMs, ROMs, etc.) make good test coverage difficult

1112

Block isolation

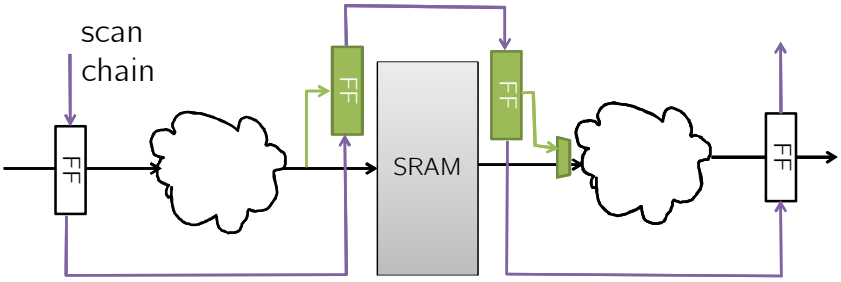
- Memories have usually no scan capabilities



1113

Block isolation (cont'd)

- Isolate macrocells (e.g., SRAMs)

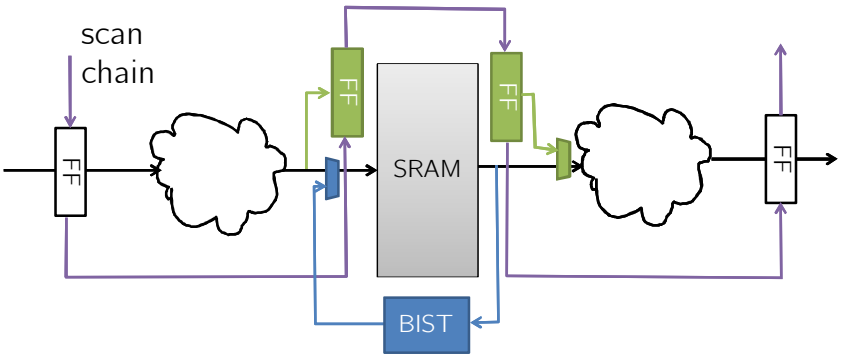


- Logic now controllable and observable
- How to test the isolated block (=SRAM)?

1114

Built-in self test (BIST)

- Add circuitry that tests the isolated block



- Usually write and/or read pseudo-random data

1115

Summary on design for test (DFT)

- Think about testing from the beginning
  - Simulate already during the design
  - Plan for test after fabrication
  - Add test structures (scan, BIST, isolation, etc.)
  - Try to get it "first time right"

"if you don't test it, it won't work (guaranteed!)"

1116

Just the basics

Low power design techniques

1117

Sources of power consumption

$$P = \sum_k P_{dyn,k} + P_{dp,k} + P_{stat,k}$$

- Dynamic power consumption
  - Switching: $P_{dyn,k} = V_{DD}^2 f_{clk} \alpha_k C_k$
  - Direct-path: $P_{dp,k} = \frac{\beta \alpha_k}{24}(V_{DD} - 2V_T)^2 t_{ra,k}$
- Static power consumption
  - Leakage: $P_{stat,k} = I_{stat,k} V_{DD}$

1118

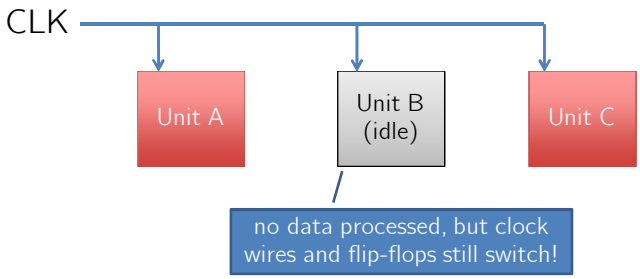
Reduce number of gates

- Reduces all 3 sources of power consumption
- Optimization on algorithm level
  - Use (or design new) algorithm that requires smallest number of gates for given throughput
  - Our research area
- Optimization on circuit level
  - Use logic optimization (tools)
  - Minimize circuit area in non-critical paths

1119

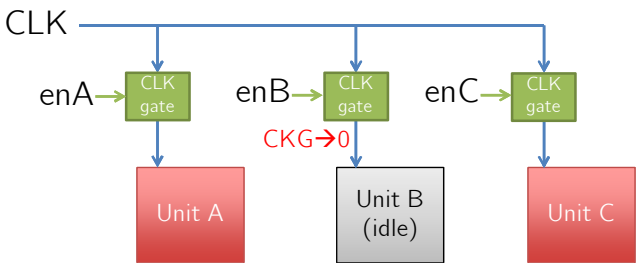
Reduce switching activity: Clock gating

- Most popular method for power reduction
- Gate clock signal if part of circuit is idle



1120

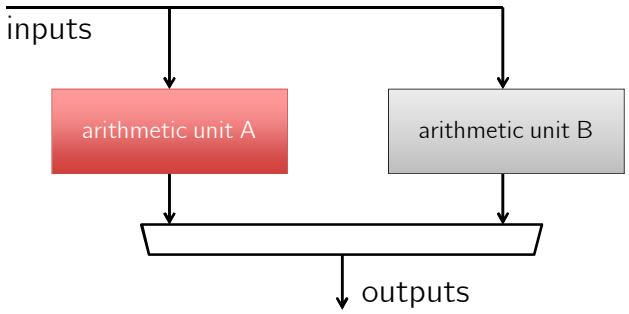
Clock gating



- Clock gates switch off clock signals
  - Be very careful with glitches on clock signal
  - Increases additional delay/skew on CLK signals

1121

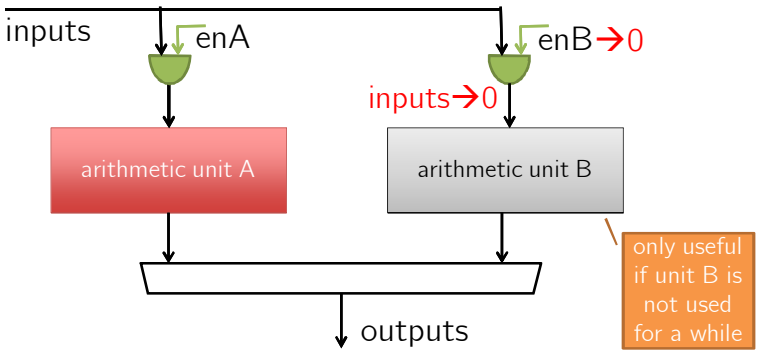
Reduce activity: Signal silencing



- In practice, only outputs of one unit needed
- Input data is propagating through both units

1122

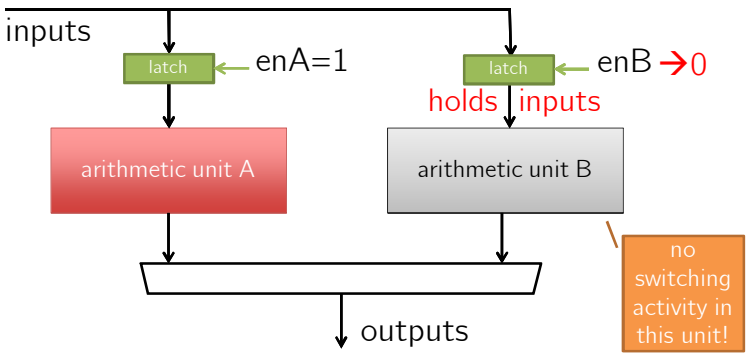
Signal silencing: method 1



- Similar idea as clock gating, but not a problem as logic is NOT on clock path
- Drawback: switching off unit causes activity

1123

Signal silencing: method 2

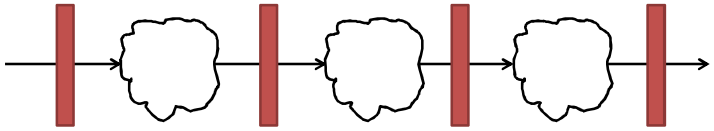


- Switching off unit causes NO activity
- Increases critical path and area!

1124

Suppress glitches by pipelining

- Glitches increase switching activity
- Glitches depend on logic depth of circuit
  - More logic gates increases likelihood of uneven arrival times at gate inputs
- Reduce logic depth by pipelining:



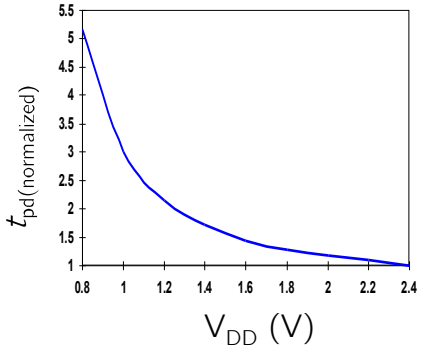
- Faster design → voltage/frequency scaling

1125

Voltage/frequency scaling

- Decreasing V_{DD} reduces power quadratically but increases gate delays!
- Having a design that is faster than necessary allows one to reduce V_{DD}

→ often substantial power reduction!



1126

Dynamic frequency/voltage scaling

- Dynamically adjust clock frequency and supply voltage to processing needs
  - High speed, high V_{DD} if needed → high power
  - Low speed, low V_{DD} → low power
- Used in virtually all modern microprocessors

1127

Reduce direct path currents

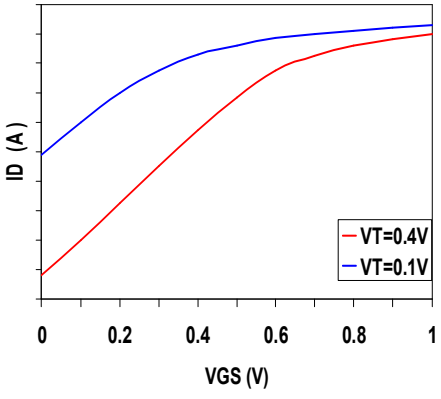
- Direct path currents depend on rise and fall times of clock signals in each gate
- Minimize rise and fall times:
  - Reduce clock loads in the design
  - Carefully place buffers in your clock tree

- Good CLK also reduces jitter → faster designs

1128

Reduce leakage currents

- Reducing V_T exponentially increases sub-threshold leakage currents
- Reducing V_T decreases gate delay → faster logic



1129

Image taken from: Digital Integrated Circuits (2nd Edition) by Rabaey, Chandrakasan, Nikolic

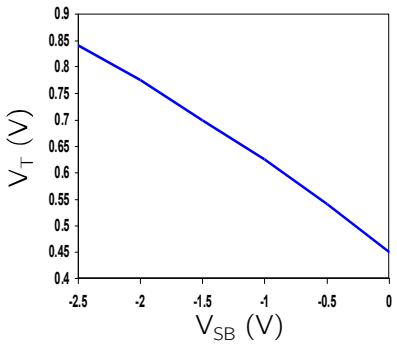
Reduce leakage currents (cont'd)

- Determine critical path(s) of design
- Leakage reduction:
  - Use low-V_T devices for timing-critical circuits
  - Use high-V_T devices for the rest
- Usually tool-assisted (if cell library contains both types of logic cells)

1130

Adaptive body biasing (ABB)

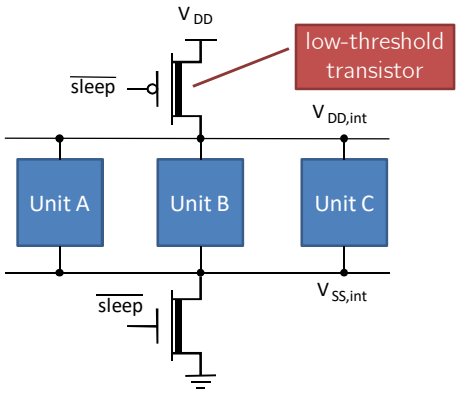
- Vary V_T at run-time by varying V_{SB}
- Negative bias increases V_T of NMOS
- Combine with with dynamic voltage/frequency scaling to lower leakage power

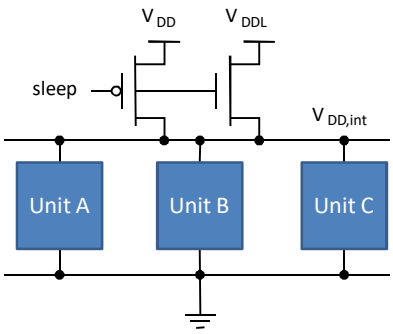


1131

Image taken from: Digital Integrated Circuits (2nd Edition) by Rabaey, Chandrakasan, Nikolic

Suppressing leakage





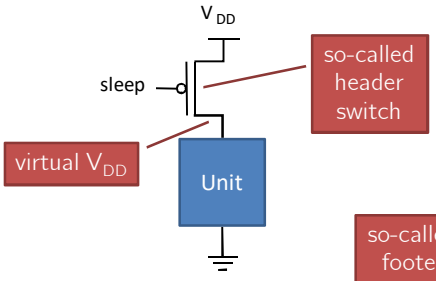
increasing resistance & reducing supply voltage

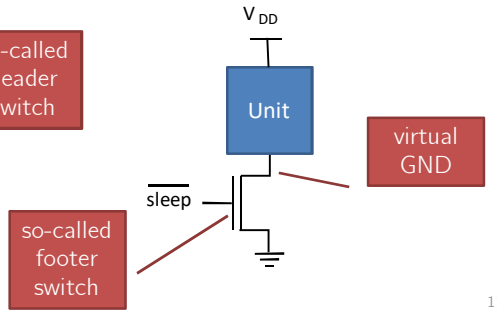
reducing supply voltage

1132

Power gating → eliminates leakage

- Completely switch off idle units
- Drawback: Overhead in storing state and resetting functionality → takes time

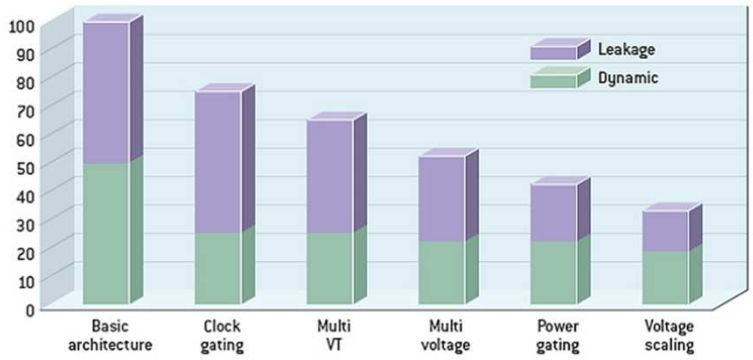




1133

What to do in practice?

- Combine all low-power techniques!



From http://www.newelectronics.co.uk/electronics-technology/power-optimisation-for-low-power-socs-targeting-mobile-devices/34896/

1134