VLSI Integrated Circuits and Systems: Principles and Design Methods

Olivier Sentieys
ENSSAT - Université de Rennes 1
IRISA/INRIA
sentieys@irisa.fr

Outline

Principles and Design Methods of VLSI Integrated Circuits and Systems: from the idea to the chip

Introduction
I. Integrated Circuit (IC) Technologies
II. Design of CMOS Cells
III. IC Design Methods
IV. Synchronous Design of IC
V. Logic Synthesis from VHDL
VI. Power Estimation and Reduction
VII. IC Design Project

http://people.rennes.inria.fr/Olivier.Sentieys/?page_id=95
Courses in circuit design at ENSSAT/EII

1st Year
- Digital Systems
  - H. Dubois
- Electronics
  - H. Chuberre
- VHDL
  - D. Chillet (16h)
- Principles and Design
  - Methods of VLSI
  - Integrated Circuits (66h)
- Low Power Digital Systems
  - Sentieys/Casseau (20h)

2nd Year
- System-on-Chip Design
  - and Verification
  - Sentieys/Casseau

3rd Year
- OPTION ISE
  - High-Level Synthesis
  - Test, Analog Circuit Design
  - Multiprocessor
  - TPs et projet d’intégration
- Project EII3
- Internship
- MASTER SISEA

Computer-Aided-Design (CAD) Tools for IC Design at ENSSAT

- Synopsys (Linux): logic synthesis (design compiler), power/timing analysis (prime time, power compiler)
- Cadence (Linux): IC place and route, analog and full-custom digital circuit design
- Altera Quartus (Linux/Windows): FPGA/CPLD design
- Xilinx ISE (Linux/Windows): FPGA/CPLD design
- MentorGraphics/ModelSim (Linux/Windows): HDL simulation and verification
- MentorGraphics/FPGA Advantage (Linux/Windows): FPGA design
- Cadence/Orcad/Pspice (Linux/Windows): Printed-Circuit-Board (PCB) design and verification
- MentorGraphics/Eldo (Linux): analog simulation
Detailed Outline (1/2)

I. Integrated Circuit (IC) Technologies
   – MOS Technology
   – (Bipolar Technology)
   – IC Fabrication
   – Silicon Technological Evolutions

II. Design of CMOS Cells
    – Combinatorial Logic Cells
    – Layout Design
    – Sequential Logic Cells
    – Delay and Power

III. IC Design Methods
     – IC Classification
     – Design Methods and CAD Tools
     – IC Specification

Detailed Outline (2/2)

IV. Synchronous Design of IC
    – Synchronous Design Rules and Principles
    – Finite State Machine (FSM) plus Datapath Model
    – Arithmetic Operators

V. Logic Synthesis from VHDL
   – Methods and Tools
   – Register-Transfer Level (RTL) Models using VHDL
   – RT and Logic Synthesis CAD Algorithms

VI. Power Estimation and Reduction (now part of another course)
    – Why Power is an Issue?
    – Power Estimation
    – Power Reduction

VII. IC Design Project
     – It will be your turn to work!
     – And to learn a lot about team-work!
Introduction

- A little bit of history and on the Moore law
- The Semi-Conductor Business
- ASIC, SOC et FPGA
- Advantages?

ENIAC – the first electronic computer

- ENIAC : Electronic Numerical Integrator and Computer
- 1946, J. Eckert et J. Mauchly
- Computation of ballistic tables
- Decimal arithmetic
- 18,000 electronic valves
- 150 square meters, 30 tons, 150,000 watts
- 200,000 Hz
- 5000 additions/subtractions per second
- 350 multiplications and 50 divisions per second
**1958: The First Integrated Circuit**

- 1947: W. Schockley (Bell Labs) invents the transistor (Nobel Price in 1956)
- 1958: J. Kilby (Texas Inst.) design the first IC (Nobel Price in 2000!)
  - Transistors, diodes, capacitors, wires assembled on a Silicon substrate
  - “I perceived that a method for low-cost production of electronic circuits was in hand... that instead of merely being able to build things smaller, we could fabricate entire networks in one sequence, and that we had extended the transistor’s capability as a fundamental electronics tool.” Jack Kilby, 1958

---

**The First Microprocessor**

- Intel 4004
- 1971
- 400 kHz
- 4 bits
- 200 US$ (1200FF)
- 0.06 MOPS
- 10 microns
- 2300 transistors
- 640 addressable bytes
Then, everything will accelerate…

- 1970 4Kbits MOS Memory
- 1972 First Processor 4004 (Intel), NMOS technology
- 1977 16K DRAM et 4K SRAM produced
- 1979 64K DRAM produced
- 1980 Intel® first x86 Processor
- 1984 Intel® 80286 Processor (PC AT)
- 1986 1 megabit DRAM
- 1988 Ti/Hitachi 16-megabit DRAM
- 1990 Intel® 80286 Processor with multimedia processing
- 1990 20 cm Wafer for production
- 1991 4 megabit DRAM produced
- 1993 Intel® Pentium® Processor
- 1997 Intel® Pentium® II Processor
- 1999 Intel® Pentium® III Processor
- 2000 Intel® Pentium® 4 Processor
- 2002 Intel® Itanium™ 2 Processor
- 2003 Intel® Pentium® M Processor, 1 gigabit DRAM
- ...

25 Years of Technological Advances

INTEL 4004 (1971)
4-bit data
2300 transistors, 10 microns
0.06 MOPS, 108 kHz

INTEL Pentium II (1996)
32-bit data
5.5M transistors, 0.35µ, 2 cm²
200 MHz, 200 MOPS, 3.3V, 35W
Intel’ Microprocessor Gallery

1999: Intel® Pentium® III Processor
9.5M Tr, 0.25um, 450MHz – 1GHz

2000: Intel® Pentium® 4 Processor
42M Tr, 0.18um, 1.5GHz – 3.6GHz

Number of Transistors

• G. Moore’s Law (INTEL corp.)

Number of transistors double every 2 years

<table>
<thead>
<tr>
<th>Year</th>
<th>CPU Type</th>
<th>Transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1970</td>
<td>4004</td>
<td>2.3K</td>
</tr>
<tr>
<td>1980</td>
<td>8088</td>
<td>29K</td>
</tr>
<tr>
<td>1990</td>
<td>386</td>
<td>275K</td>
</tr>
<tr>
<td>2000</td>
<td>Pentium</td>
<td>1.2M</td>
</tr>
<tr>
<td></td>
<td>Pentium Pro</td>
<td>3.3M</td>
</tr>
<tr>
<td></td>
<td>Pentium II</td>
<td>5.5M</td>
</tr>
<tr>
<td></td>
<td>Pentium III</td>
<td>7.5M</td>
</tr>
<tr>
<td></td>
<td>Pentium 4</td>
<td>42M</td>
</tr>
</tbody>
</table>
Clock Speed

- G. Moore’s Law (INTEL corp.)

Frequency is increasing 58% per year (x4 every 3 years)
Processor performance doubles every 2 years

Why care about Power?

- Processor Power Evolution: x4 every 3 years
And then came the “Power Wall”

- Power Density: 100 W/cm² is a limit

![Power Density Graph]

and the “Multicore Era”

- Increasing performance by increasing # of cores

![Multicore Trajectory Graph]
Semiconductor Market

- Semiconductor Industry Association (SIA) reported: “The global semiconductor market hit a new record in 2006 with a sales volume of $247.7 billion, up 8.9 percent from 2005”
- Driven by consumer products such as cell-phones, MP3 players and HDTV receivers
- As the semiconductor industry completes one of its most successful years in 2010, worldwide semiconductor revenue is forecast to total $314 billion in 2011, up 4.6% from 2010’s (Gartner Inc.)

- 235 million units of PC were shipped in 2006,
  - but more than 1 billion cellphones…
    - market is in DSP, MCU and memory

Semiconductor Business

- Foundry cost is high
  - lithography, test, packaging
  - investment increase of 19% every year

- More and more people to design complex ICs
What is an ASIC?

• ASIC: Application Specific Integrated Circuit
  – An ASIC is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use
  – Examples: video recorder, camera, phone modem, speech processing, etc.

System-on-Chip (SoC)

- Analog
  - A/D
  - RF, modulation
- µP/µC core
  - control
  - user interface
- DSP core
  - low-complexity DSP processing
  - but with flexibility
- ASIC/IP core
  - hardware accelerator
- Memory
- On-Chip Bus
Ex. 1: 2G cellphone

Your father’s GSM

Now
All is SoC Inside the iPhone!

Ex. 2: IXP1200 Intel NPU

6.5M Transistors

StrongARM Core
PCI
SRAM I/F
SDRAM I/F
IX Bus
6 Micro-RISC
Ex. 3: FPGA Xilinx VIRTEX II Pro

- Up to 4 embedded processors IBM PowerPC 405, 300MHz
- Rocket I/O Multi-Gigaset Transceiver

Ex. 3: FPGA Xilinx VIRTEX 7

<table>
<thead>
<tr>
<th>Virtex-7 FPGAs</th>
<th>Optimized for Highest System Performance and Capacity (1.0V, 0.9V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part Number</td>
<td>KCV7K410T</td>
</tr>
<tr>
<td>Logic Resources</td>
<td></td>
</tr>
<tr>
<td>EasyPath™</td>
<td>162,700</td>
</tr>
<tr>
<td>Silicon™</td>
<td>467,700</td>
</tr>
<tr>
<td>Logic Cells</td>
<td>246,680</td>
</tr>
<tr>
<td>CLB Flip-Flops</td>
<td>317,600</td>
</tr>
<tr>
<td>Memory Resources</td>
<td></td>
</tr>
<tr>
<td>Maximum Distributed RAM (Kbits)</td>
<td>3,475</td>
</tr>
<tr>
<td>Block RAM/FIFO at 400 ns (Kbits)</td>
<td>410</td>
</tr>
<tr>
<td>Total Block RAM (Kbits)</td>
<td>14,760</td>
</tr>
<tr>
<td>Clock Resources</td>
<td>Mixed Mode Clock Managers (MHz/50)</td>
</tr>
<tr>
<td>I/O Resources</td>
<td>Maximum Single-Ended I/O</td>
</tr>
<tr>
<td>DSP/BRAM Blocks</td>
<td></td>
</tr>
<tr>
<td>700</td>
<td>700</td>
</tr>
<tr>
<td>Embedded Hard IP Resources</td>
<td></td>
</tr>
<tr>
<td>Configuration AES</td>
<td>HMAC Blocks</td>
</tr>
<tr>
<td>QTI 16-Bit I/O Transceivers</td>
<td>20</td>
</tr>
<tr>
<td>QTI 15.625 Gbps Transceivers</td>
<td>20</td>
</tr>
<tr>
<td>GTZ 25 Gbps Transceivers</td>
<td>20</td>
</tr>
<tr>
<td>Speed Grade</td>
<td>Commercial</td>
</tr>
<tr>
<td>Configuration</td>
<td>Configuration Memory (MHz)</td>
</tr>
</tbody>
</table>

- 22M CLBs
- 45MB RAM
- 4000 MUX

3.125 Gb Serial World’s Fastest Logic & Routing
IBM PowerPC RISC CPU
XtremeDSP™
Synchronous Dual-Port RAM
SystemQ™

Embedded Processor

Virtex II Pro Architecture

Virtex II Pro

Rocket I/O Multi-Gigaset Transceiver

Olivier Sentieys, INRIA/IRISA - ENSSAT
Ex. 4: Set Top Box (STb) STMicroelectronics

- STB Product is one chip solution for:
  - Dual H264-MPEG2-VC1 HD decoder, Triple TV display
    - MPEG2 MP@HL
    - ISO/IEC 14496-10/ITU Rec. H.264 Main profile level 4.1
    - VC1
  - Communications
    - 4 external transport streams (and three playbacks/timeshift from HDD or network)
    - 2 MII Ethernet, 3 USB2.0 and 2 SATA ports
    - Channel 3/4 mod
    - HD digital HDMI, 1 HD analog, 2 SD analog I/F
    - 1 Software modem including analog interface

- 65nm LP 7ML
- 150M transistors
- 886 pads
- Top+5 BE partitions
- 18 FE subsystems
- 115 propagated clocks
- (19 for interconnect)
- 36 soft IPs
- 2 hard blocks
- 16 analog IPs
- 19 IOLIBs
- 29 internal blocks/glues
- 140 memory cuts
Ex. 5: NVIDIA Tegra 2 chip

- Smartphone market
- 40 nm, TSMC, 260M transistors
- Die size: 49mm²
- 2 ARM Cortex A9 1Ghz
- 1 ARM A7 (low-power)
- 300 Mhz (rest of the chip)
- Capable of processing images at a rate of 80 Mpixels/s
- Supports two cameras (12+5 Mpixels)
- Real time H.264 video encoding
- Decode 1080p H.264 videos at up to 20Mbps

Advantages and Drawbacks

- Advantages of ASIC/FPGA solutions
  - Size and Power are reduced while Speed is increased: Energy Efficiency
  - Cost may be reduced: depends on volume and on ASIC/FPGA choice
  - Confidentiality, Technological Advantage
- Drawbacks of ASIC/FPGA
  - Cost of the first produced chip is more than 1M$ (Not true for FPGA
  - Requires highly-qualified people (but you will be there for that...)
  - Design Complexity: an SoC is HW + SW (Verification is a nightmare!)
  - Partial loss of control in delays and costs; Higher risk
    - Partial unpredictability
    - Risk and cost overheads in case of design errors in the first prototype
I. Integrated Circuit (IC) Technologies

1. MOS Technology
   1. The MOSFET Transistor
   2. Transistor Performance Models
   3. MOS Technologies (nMOS, pMOS, CMOS)

2. (Bipolar Technology)

3. IC Fabrication
   1. Fabrication Process
   2. Example of a Diode and a MOSFET

4. Silicon Technological Evolutions
   1. Processor and Technology Roadmaps
   2. Technology Scaling

5. Interconnects
1.1 MOSFET: Metal–Oxide–Semiconductor Field-Effect Transistor

Transistor Types

NMOS Transistor

1.1 MOS Transistor

demo
1.1 NMOS Transistor

- MOSFET: Metal (Polysilicium) Oxide Silicium Field Effect Transistor
- Cutoff or sub-threshold mode:
  \[ V_{GS} < V_t, \quad R_{SD} \approx +\infty \]
- Linear mode:
  \[ V_{GS} > V_t \text{ and } V_{DS} < V_{GS} - V_t \]
  - a channel is created which allows current to flow between the drain and the source
- Saturation mode:
  \[ V_{GS} > V_t \text{ and } V_{DS} > V_{GS} - V_t \]

NMOS/PMOS Transistors

- **NMOS**
  - \( V_{GS} < V_t \)
  - \( V_{GS} > V_t \)
- **PMOS**
  - \( V_{SG} < |V_l| \)
  - \( V_{SG} > |V_l| \)

**Voltage Level Transfer (Source > Drain)**
- A ‘0’ is well transmitted
- A degraded ‘1’ is transmitted \( (V_{dd} - V_t) \)
- A ‘1’ is well transmitted
- A degraded ‘0’ is transmitted \( (V_{ss} + |V_t|) \)

**Charge Carriers**
- **Electrons**
  - Substrate/Bulk/Well Voltage: \( V_{ss} \)
- **Holes**
  - Substrate/Bulk/Well Voltage: \( V_{dd} \)
### 1.2 MOS Transistor Models

- **W**: gate width
- **L**: gate length
- **tox**: oxide width (#L/10)
- **Cox**: gate oxide capacitance per unit area
- **K = \mu \cdot W/(tox \cdot L) = \mu \cdot Cox \cdot W/L**
- **\mu**: charge-carrier effective mobility
  - NMOS (electrons) \( \mu_n = 500 \text{ cm}^2/\text{V-sec} \)
  - 2 \( \mu \text{P} \)
  - PMOS (holes) \( \mu_p = 270 \text{ cm}^2/\text{V-sec} \)
- **\varepsilon**: oxide permittivity \( \varepsilon_0 = 3.5 \times 10^{-13} \text{ F/cm} \)

\[
I_{ds} = \begin{cases} 
0 & \text{cut-off} \\
K \left( V_{gs} - V_t \right)^2 / 2 & \text{linear} \\
K_0 \left( V_{gs} - V_t \right)^2 / 2 & \text{saturated}
\end{cases}
\]

- \( K \) is a function of \( W/L \)
- \( I_{ds \ max} \) depends on \( W \)
- Temperature \( \uparrow \): influence on \( \mu \) and \( I_{ds \ max} \)
NMOS Parasitic Elements

Drain-Source Resistance: \( R_{ds} = \frac{1}{K(Vdd - Vt)} = \frac{1}{k(Vdd - Vt)W} \)

Gate Capacitor: \( C_g = \frac{W}{L} \times W \times L \times C_{ox} \)

Drain/Source-Bulk Capacitor: \( C_{gs} = C_{sb} = W \times L \times C_i \)

Delay: \( \tau = R_{ds}C_g = \frac{L^2}{\mu(Vdd - Vt)} \)

MOS SPICE model 1

• Expression of \( I_d \)

<table>
<thead>
<tr>
<th>Mode</th>
<th>Condition</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut-off</td>
<td>( Vgs &lt; 0 )</td>
<td>( I_d = 0 )</td>
</tr>
<tr>
<td>Linear</td>
<td>( Vds &lt; Vgs - Vt )</td>
<td>( I_d = KP \frac{W}{L} (Vgs - Vt) Vds - \frac{Vds^2}{2} )</td>
</tr>
<tr>
<td>Saturated</td>
<td>( Vds &gt; Vgs - Vt )</td>
<td>( I_d = KP \frac{W}{2L} (Vgs - Vt)^2 )</td>
</tr>
</tbody>
</table>

• Threshold Voltage

\( V_t = V_{T0} + \text{GAMMA} + \sqrt{\text{PHI}} - V_b - \sqrt{\text{PHI}} \)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>NMOS 0.25u</th>
<th>PMOS 0.25u</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{T0} )</td>
<td>Threshold Voltage</td>
<td>0.4V</td>
<td>-0.4V</td>
</tr>
<tr>
<td>( KP )</td>
<td>Transconductance Coefficient</td>
<td>300( \mu )A/V²</td>
<td>120( \mu )A/V²</td>
</tr>
<tr>
<td>( \text{PHI} )</td>
<td>Surface Inversion Potential</td>
<td>0.3V</td>
<td>0.3V</td>
</tr>
<tr>
<td>( \text{GAMMA} )</td>
<td>Bulk Threshold Parameter</td>
<td>0.4V²/3</td>
<td>0.4V²/3</td>
</tr>
<tr>
<td>( W )</td>
<td>Channel Width</td>
<td>0.5-20( \mu )m</td>
<td>0.5-40( \mu )m</td>
</tr>
<tr>
<td>( L )</td>
<td>Channel Length</td>
<td>0.25( \mu )m</td>
<td>0.25( \mu )m</td>
</tr>
</tbody>
</table>

Olivier Sentieys, INRIA/IRISA - ENSSAT

VLSI Integrated Circuits and Systems: Principles and Design Methods
MOS SPICE model 3

\[ I_{ds} = 0 \quad \text{Cut-Off} \quad V_{gs} < 0 \]

\[ I_{ds} = \frac{W}{L_{eff}} (1 + K_{app}) V_{ds} \left( V_{ds} - \frac{V_{de}}{2} \right) \quad \text{Normal} \quad V_{gs} > V_{on} \]

\[ V_{on} = 1.2V, \quad V_{t} = V_{TO} + \Gamma \left( \sqrt{P_{hi} - V_{b}} - \sqrt{P_{hi}} \right) \]

\[ V_{de} = \text{MIN}(V_{ds}, V_{dsat}), \quad V_{dsat} = V_{c} + V_{sat} - \sqrt{V_{c}^2 + V_{sat}^2}, \quad V_{sat} = V_{gs} - V_{t} \]

\[ V_{c} = V_{MAX} \frac{L_{eff}}{0.06}, \quad L_{eff} = L - 2L_{D}, \quad K_{eff} = \frac{K_{P}}{1 + \Gamma (V_{gs} - V_{t})} \]

\[ I_{ds} = K_{app} \frac{W}{L_{eff}} (1 + K_{app}) V_{on} \left( V_{gs} - V_{t} \right) \left( \frac{V_{de}}{2} \right)^{\frac{V_{gs} - V_{on}}{V_{t}}} \]

\[ V_{de} = \text{MIN}(V_{on}, V_{dsat}) \]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>NMOS 0.25u</th>
<th>PMOS 0.25u</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD</td>
<td>Lateral diffusion into channel</td>
<td>0.01(\mu)m</td>
<td>0.01(\mu)m</td>
</tr>
<tr>
<td>KAPPA</td>
<td>Saturation field vector</td>
<td>0.01(V^{-1})</td>
<td>0.01(V^{-1})</td>
</tr>
<tr>
<td>VMAX</td>
<td>Maximum drift velocity</td>
<td>150km/s</td>
<td>150km/s</td>
</tr>
<tr>
<td>THETA</td>
<td>Mobility degradation factor</td>
<td>0.3(V^{-1})</td>
<td>0.3(V^{-1})</td>
</tr>
<tr>
<td>NSS</td>
<td>Subthreshold factor</td>
<td>0.07(V^{-1})</td>
<td>0.07(V^{-1})</td>
</tr>
</tbody>
</table>

1.3 MOS Technologies

- NMOS (or PMOS) Technologies (1970-1980)
  - Only one type of transistor NMOS or PMOS
  - Resistances made with depleted NMOS transistors

- Good integration, but...
- Rising/Falling delays unbalanced
- ‘1’ level is degraded
- Static power consumption when \(S = 0\)
1.3 MOS Technologies

• CMOS Technology

- Perfect Voltage Level Transmission
  - PMOS transmits $V_{dd}$ (‘1’) when $E=0$
  - NMOS transmits $V_{ss}$ (‘0’) when $E=1$
- Noise Margin:
  - noise level at input without modifying logic level
- Excellent Noise Margins
  - $V_{OH}=V_{dd}$; $V_{OL}=V_{ss}$

• BiCMOS
  - Combine advantages of bipolar (speed) and MOS (density, power)
  - Push-Pull bipolar amplifier at the output of the cell

• Gallium Arsenide (GaAs) Technology
  - Compound of the elements gallium and arsenic, a III/V semiconductor
  - Higher electron mobility than Si: high frequency
  - No P-channel MESFET
  - Used in RF, LEDs or lasers
1.3 MOS Technologies

• SOI (Silicon On Insulator) Technology
  – In Bulk Silicon CMOS substrate and well imply parasitic capacitors and leakage currents
  – Use of a layered silicon-insulator-silicon substrate (silicon dioxide or sapphire) in place of conventional silicon
  – Lower parasitic capacitance due to isolation from the bulk silicon, which improves power consumption
  – Resistance to latchup due to complete isolation of the n- and p-well structures
  – Compatible with CMOS
  – Integration density is higher

I. Integrated Circuit (IC) Technologies

1. MOS Technology
   1. The MOSFET Transistor
   2. Transistor Performance Models
   3. MOS Technologies (nMOS, pMOS, CMOS)

2. (Bipolar Technology)

3. IC Fabrication
   1. Fabrication Process
   2. Example of a Diode and a MOSFET

4. Silicon Technological Evolutions
   1. Processor and Technology Roadmaps
   2. Technology Scaling

5. Interconnects
1.2 How ICs are made?

From sand to silicon
From silicon to integrated circuit

Integrated Circuit Fabrication

- Silicon ingot (#100kg) pure at 99.9999999%
Integrated Circuit Fabrication

- Wafer: is a thin slice of Si material (substrate) – 450 mm (17.7 inch), thickness 925 µm
- Set of identical chips (die) printed on a Wafer

Integrated Circuit Fabrication

- Wafer testing of chips using a wafer prober
Integrated Circuit Fabrication

- Wafer is then cut into dies
  - Dicing
- Packaging
- Final testing with package

Photolithography is like printing a tiny 3D book
Integrated Circuit Fabrication

• Extremely clean operating conditions

Lithography

Start with a wafer with SiO2 (surface oxidation)
1. Spin on a photoresist
2. Pattern photoresist with a mask
3. Etching of photoresist
4. Wash off photoresist

1. Photoresist
2. Mask
3. Etched Photoresist
4. Washed Off
**Etching**

- Pattern material on the surface
- Desired shape is patterned with photoresist through a mask
- Chemical (wet) or plasma (dry) etching of the material

**Oxide Deposit/Growth**

- Oxidation of the silicon surface creates a SiO$_2$ layer that acts as an insulator
- Oxide layers are also used to isolate metal interconnections
- Wafer is placed in a high-temperature furnace to make the silicon react with oxygen or water vapor
  - Si + 2H$_2$O $\rightarrow$ SiO$_2$ + 2H$_2$
- PVD: Physical Vapor Deposition
- CVD: Chemical Vapor Deposition
Ion Implantation

- Ion implantation is used to add doping materials to change the electrical characteristics of silicon locally.
- Dopant ions penetrate the surface, with a penetration depth that is proportional to their kinetic energy.
- P+ implant: boron
- N+ implant: arsenic or phosphorous

Example: Diode Fabrication

- a) Oxide deposit
  - SiO₂
  - Substrate
- b) Spin on photoresist
  - photoresist
- c) UV light exposure through mask
  - UV
  - mask
- d) Pattern photoresist
- e) Pattern SiO₂
- f) Ion implant (P-type)
CMOS Inverter

CMOS Gate Fabrication

Implant N-Well

Etch away silicon where there won’t be transistors

Deposit oxide in trenches

Polish top of water flat and expose silicon
CMOS Gate Fabrication

1. Implant gate and grow oxide
   - Implant sets turn-on voltage of transistor to the correct value.

2. Deposit and Etch Polysilicon for gate

3. Implant source and drain
   - Diffusion self-aligns to edges of gate.

4. Coat Poly and Silicon with metal to reduce Resistance

CMOS Gate Fabrication

1. Deposit Dielectric

2. Etch metal trenches

3. Etch via trenches
CMOS Gate Fabrication

- Deposit metal
- Polish surface back

I/O Pads

- I/O pads use large transistors

Bonding Pad

GND

NMOS

PMOS

VDD

VSS

vDD

vSS

100 µm
Real Transistors and Metal Layers

3D FinFET Transistors (Intel) ...

• At 22nm transistors go 3D
Ultra Thin Body (FD) – SOI
- Total dielectric isolation
  - Lower S/D capacitances & leakages
  - Latch-up immunity
- Improved VT variation

Gate Material
Junctions
Film & Box
Isolation
Body Bias
Ground-Plane
Hybrid / Bulk

For the gate dielectrics, since the high process temperature (C) or the plastic melting point (T) is usually greater than the glass transition temperature (C14), the gate dielectric layer (SiN) retains its crystalline properties, leading to a high defect density. This high defect density can trap charges near the SiN/a-Si:H interface, causing stability issues.

Two mechanisms are responsible for this electrical instability. First is carrier trapping in the gate insulator SiN due to the high electric field near the SiN/a-Si:H interface. This trapping is caused by the thin (<10 nm) SiN film and the sharp electrical field gradient. The trapped charges can shift the threshold voltage (VTH) and affect the sub-threshold slope, leading to a distortion in the DC characteristics of the TFTs.

The second mechanism is related to the high process temperature that occurs during the deposition (PECVD). This dielectric layer, however, has many defects that can trap charges near the a-Si:H/SiN interface when a positive gate-source voltage is applied to the gate terminal. Note that because these traps are not evenly distributed, the threshold voltage may vary with the drain voltage.

Despite these difficulties, with rapid improvements in TFT technologies and materials, many novel applications have now become feasible and attracted much attention from interdisciplinary researchers. TFT technologies in recent years, many novel applications have now become feasible and attracted much attention from interdisciplinary researchers. Whether these difficulties can be overcome is still unknown, but as we will explain.

Flexible Electronics
- Plastic substrate
  - Organic PV cells
  - MOS Transistors
  - polysilicon thin-film transistors (LTPS-TFTs) on a plastic substrate

Table 1: Comparison of LTPS and Thin-Film transistor (TFT) device technologies

<table>
<thead>
<tr>
<th>Technology</th>
<th>Channel Doping</th>
<th>Channel Material</th>
<th>ID/IO</th>
<th>Microwave</th>
<th>I SO</th>
<th>DEFENSE</th>
<th>SOI</th>
<th>SiGe/SiGe</th>
<th>SiGe/SiGe</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTPS</td>
<td>Low</td>
<td>Glass</td>
<td>3.5</td>
<td>1000</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Thin-Film transistor</td>
<td>Low</td>
<td>Plastic</td>
<td>3.5</td>
<td>1000</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Table from: Huang et al., Robust Circuit Design for Flexible Electronics, IEEE Design and Test of Computers, 2011

World's First Flexible 8-Bit Asynchronous Microprocessor

Fight against FDSOI Transistors (ST)

- Ultra Thin Body (FD) – SOI
  - Total dielectric isolation
  - Lower S/D capacitances & leakages
  - Latch-up immunity
  - Improved VT variation

STMicro’s roadmap

Feature size: 32 nm
Process technology: Lithography
Process temperature: 1000°C
Roll-to-roll lithography
Shadow mask
Ink-jet printing

800°/Vs: 1500
100
0.01 or 0.5

To have a compelling technology offer for the mobile application, bridging the gap with uncertain 14nm node roadmap

20FDSOI, bridging the gap with uncertain 14nm node roadmap

28nm bulk roadmap

2012 February 2012
2013
2014
2015
2016

Table 2: Comparison of UTBB and FDSOI transistor (FDSOI) device technologies

<table>
<thead>
<tr>
<th>Technology</th>
<th>Channel Doping</th>
<th>Channel Material</th>
<th>ID/IO</th>
<th>Microwave</th>
<th>I SO</th>
<th>DEFENSE</th>
<th>SOI</th>
<th>SiGe/SiGe</th>
<th>SiGe/SiGe</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTBB</td>
<td>Low</td>
<td>Glass</td>
<td>3.5</td>
<td>1000</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>FDSOI</td>
<td>Low</td>
<td>Plastic</td>
<td>3.5</td>
<td>1000</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Table from: Huang et al., Robust Circuit Design for Flexible Electronics, IEEE Design and Test of Computers, 2011
I. Integrated Circuit (IC) Technologies

1. MOS Technology
   1. The MOSFET Transistor
   2. Transistor Performance Models
   3. MOS Technologies (nMOS, pMOS, CMOS)
2. (Bipolar Technology)
3. IC Fabrication
   1. Fabrication Process
   2. Example of a Diode and a MOSFET
4. Silicon Technological Evolutions
   1. Processor and Technology Roadmaps
   2. Technology Scaling
5. Interconnects

Silicon Technology

- 0.35 µm in 1995, 0.25 µm in 1998, 0.18 µm in 2000
- 130 nm in 2002, 90 nm in 2004, 65 nm in 2007
- 45 nm in 2010 (first chip 2008)
  - 11-15 metal levels, wafer 30cm
  - 0.6-0.9 Volts
  - 700 MHz (ASIC) - 9 GHz (on-chip 12 inverters) - 5 GHz (off-chip)
  - 3-4 (MPU), 1 (DRAM) - 4-8 (ASIC) cm²
  - DRAM: 4Gbits, 4Gbits/cm², 0.005 $/Mbits
  - 300 (MPU) - 6000 (ASIC) MTr/cm², 0.05-0.1 $/MTr (MPU)
  - SRAM: 1500MTr/cm², 250Mbits/cm²
  - 6000 RISC processors (e.g. ARM7)

- 28 nm in 2013 (first chip in 2010)
- 11 nm in 2019-2021 and then ?
- Post-Silicon Technologies (nanotechnologies)
**Silicon in 2015**

- Power Supply: 0.6-0.8 V
- Technology: 25 nm CMOS (200 Ang.)
  - 20 GTransistors, wafer 45 cm, 2-4 cm2, 13-17 metal levels
  - Inverter 2.5 ps, 0.6 Volt
  - 33 GHz (on-chip 12 inverters) - 29 GHz (off-chip)
  - DRAM 16 GBits at 10ns, 0.006 $/Mbits
  - SRAM (cache) 1 GBits at 1.5ns
  - 256-bit Bus

- More than 8500 Person.Month Design Cycle
- Software
- Mask set is few Millions US$

---

**Technology Scaling**

- Scaling factor: $s$
- Between two successive generations: $s \# 0.7$

![Bar chart showing technology scaling from 250 nm to 130 nm](chart.png)
Technology Scaling

• Supply Voltage Scaling (Vdd)

Technology Scaling

• Number of transistors
  – Logic: x2 every 3 years
  – Memory: x4 every 3 years

• Speed (before power wall...)
  – Logic: x2 every 3 years
  – Memory: x4 every 10 years

• Processor Performance
  – 50% per year

• Transistor density doubles and chip area increases by 25% at each new generation
Technology Scaling

- Power Density
  - Frequency is increased by 43%
  - Total capacitance and supply voltage (Vdd) are decreased by 30%
  - Then Energy is decreased by 65%
    - \[ E = C \cdot Vdd^2 = 0.7C' \cdot (0.7Vdd')^2 = 0.35C' \cdot Vdd'^2 = 35\% \cdot E' \]
  - Power is decreased by 50%
    - \[ P = f \cdot C \cdot Vdd^2 = 1.43f' \cdot 0.35C' \cdot Vdd'^2 = 50\% \cdot P' \]
    - Activity is supposed constant
  - But this is for a constant transistor count!
  - Power Density therefore increases by 2
  - Power supply current increases a lot
    - 100W at 1v equals to...?

Technology Scaling

- Scaling factor: s
- Between two successive generations: s # 0.7

<table>
<thead>
<tr>
<th>Device dimensions :</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>W, L, tox, junction depth</td>
<td>s</td>
</tr>
<tr>
<td>Transistor area (W.L)</td>
<td>s^2</td>
</tr>
<tr>
<td>Capacitance per unit area : Cox</td>
<td>1/s</td>
</tr>
<tr>
<td>Capacitances : C=WlCox</td>
<td>s</td>
</tr>
<tr>
<td>Vdd, Vt</td>
<td>s</td>
</tr>
<tr>
<td>Gate delay</td>
<td>s</td>
</tr>
<tr>
<td>Power/gate</td>
<td>s^2</td>
</tr>
<tr>
<td>Power.delay product</td>
<td>s^3</td>
</tr>
<tr>
<td>Power density</td>
<td>1</td>
</tr>
</tbody>
</table>
Power Supply Voltage Evolution

• Power and Substrate Noises
  – Vdd scaling $\rightarrow$ SNR $\downarrow$

[© R. Rutenbar, CMU]

1.5 Interconnections

• Why worry about interconnections?
  – Important gate load
  – Technology scaling $\Rightarrow$ more parasitic effects
  – Increase of chip area $\Rightarrow$ more interconnect

• Capacitance model: $C_{int} = \frac{\varepsilon_{ox} WL}{t_{ox}}$

• Side capacitance
• Crosstalk capacitance
Interconnections

- Resistance model
  - With technology scaling, parasitic resistance grows in $1/H$
  - Solutions:
    - Keep $H$ constant
    - Increase number of wiring levels
    - Optimised contact resistance

\[ R = \rho \frac{L}{HW} \]

- Interconnections in Deep Submicron
  - Delay is fundamentally changed
    - Wire interconnections dominate delay and power
      - 60% of delay critical path is due to wires
    - Wire routing prediction is necessary at gate level
  - Use of copper, increase metal levels

Example: delay of a 2-input NAND
- connected to 2mm of metal: 280ps
- connected to 0.5mm of metal: 119ps
**Interconnection Length**

- Light Speed: 300μm/ps
- Diagonal: 30 mm (21mm side)
- 100 ps
- 1 clock cycle @ 10GHz
- In real 5-10 clock cycles

[Source: Intel]

---

**Reducing wire delay**

- Height of wires
- Copper
- Repeaters

[Source: Intel]

---

[Source: IBM]
Circuits go 3D

- 3D Integrated Circuits
- Stack Multiple Dies
- Connect Dies with Through Silicon Vias (TSV)
- Examples
  - Image Sensors
  - Processor + Memory

Why 3D?

- Wire Length Reduction
  - Replace long, high capacitance wires by TSVs
  - Low Latency, Low Energy
- Small footprint
Why 3D?

- Integration
  - From “Off-Chip” to “On-Chip”
  - Improved Communication
    - Low Latency, High Bandwidth, and Low Energy
  - Heterogeneous Integration
    - E.g. Emerging Devices

3D Technologies
Circuits go 2.5D...

- Smaller chip size = increase yield!

Circuits go 3D

- Tire Pressure Monitoring
- Health & Fitness

Target: batterless → Vibration

STMicroelectronics
Circuits go 3D

Source: IBM

Cooling!

• Thermal effects
II. Design of CMOS Cells

1. Combinatorial Logic Cells
   1. Complementary Logic (CMOS)
   2. (NMOS/Pseudo-NMOS Logic)
   3. Switch-Based Logic
2. Layout Design
   1. Design Rules
   2. Layout Design
3. Sequential Logic Cells
   1. Elementary Memory Cells
   2. ROM and RAM Memory
4. Delay and Power
   1. Delay Models
   2. Fan-In and Fan-Out
   3. Static and Dynamic Power

II.1 Combinatorial Logic Cells

- 1.1 Complementary Logic (CMOS)
   - CMOS Static Logic

\[ E \rightarrow \text{Logic Cell} \rightarrow S = f(E) \]


**NAND and NOR**

NAND and NOR gates are fundamental in digital electronics, with applications ranging from simple logic functions to complex digital circuits.

- **NAND Gate**
  - **Function:** $S = \overline{A \cdot B}$
  - **Truth Table:**
    | A | B | S |
    |---|---|---|
    | 0 | 0 | 1 |
    | 0 | 1 | 1 |
    | 1 | 0 | 1 |
    | 1 | 1 | 0 |

- **NOR Gate**
  - **Function:** $S = A + B$
  - **Truth Table:**
    | A | B | S |
    |---|---|---|
    | 0 | 0 | 1 |
    | 0 | 1 | 1 |
    | 1 | 0 | 1 |
    | 1 | 1 | 0 |

**N-Input Gates possible, but with Fan-In issues**

**Complex gates**

- One CMOS stage can generate any sum-of-product or product-of-sum:
  $$S = f(E_1, E_2, ..., E_N) = \Sigma [\Pi] = \Pi [\Sigma]$$

- **Example:** $S = \overline{A \cdot B} + C \cdot D$
  - AOI (And-Or-Invert) gate

---

Olivier Senteys, INRIA/IRISA - ENSSAT
General rules for constructing $F(X)$

<table>
<thead>
<tr>
<th>$F$</th>
<th>$N$ network</th>
<th>$P$ network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
</tr>
<tr>
<td>$F_1.F_2$</td>
<td>$F_1$ $F_2$</td>
<td>$F_1$ $F_2$</td>
</tr>
<tr>
<td>$F_1+F_2$</td>
<td>$F_1$ $F_2$</td>
<td>$F_1$ $F_2$</td>
</tr>
</tbody>
</table>

### Static Logic

- **Examples**
  - Direct application of the design rules
  - $S_1 = (/A.B.D+CE+CD)$
  - $S_2 = /[A.D.B+C(E+D)]$

- **Multiple-Stage Complex Functions**
  - Optimisation of the logic equation
  - Trade-off between speed and area
  - $S_3 = A.B.C.D$
  - $S_4 = !A.B+A.!B$ (XOR)
II.2 Pass-Transistor Logic

• Switch or Transmission Gate

NMOS

PMOS

\[ \begin{align*}
E & \quad \overline{C} & \quad S \\
0 & \quad 0 & \quad 0 \\
1 & \quad \#1 & \quad 1 \\
\end{align*} \]

\[ \begin{align*}
E & \quad C & \quad S \\
0 & \quad 0 & \quad 0 \\
1 & \quad 1 & \quad 1 \\
\end{align*} \]

• Example: 2-input multiplexer

\[ S = \begin{cases} 
A & \text{if } C = 0 \\
B & \text{if } C = 1 
\end{cases} \]

\[ S = A \overline{C} + B.C \]

• Example: XOR

General Rules

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>U</th>
<th>W</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Simplifications
  – if \( V = 0 \) PMOS can be omitted
  – if \( V = 1 \) NMOS can be omitted
General Rules

- Avoid numerous serial transistors
  - voltage loss, propagation time
- Level restoring or Pull-up
  - e.g. XOR
- Avoid conflicts

II.2 MOS Layout Design

- $X\ nm$ Technology
  - $X\ nm \# 2\lambda$
  - $\lambda$: smallest mask size
  - Transistor Channel $L \# 2\lambda$
- Stick Diagram
II.2 MOS Layout Design

Layout Design Rules

- Interface between designer and process engineer
- Guidelines for constructing process masks
- Unit dimension: e.g. min. width
  - scalable design rules: \( \lambda \)
  - absolute dimensions (microns)
- Intra-Layer Design Rules
  - width: min. of each layer (poly, metal, diffusion)
  - spacing: min. space between two materials
- Inter-Layer Design Rules
  - Via, Contact, Overflow

Layer = Color
- Well (P,N): Yellow
- Diffusion (P+,N+): Green
- Polysilicon: Red
- Metal (1,2): Blue
- Contact to Poly: Black
- Contact to Diff.: Black
- Via: Black

\[ \begin{align*}
  & \geq 3 \lambda \\
  & \geq 2 \lambda \\
  & \geq 5 \lambda \\
  \end{align*} \]
Layout Design Rules: Example

Layer Usage

- Possible Interconnection between Layers
  - Metal1 - Poly
  - Metal1 - Diffusion N
  - Metal1 - Diffusion P
  - Metal1 - Metal2 (via)
- Metal 1 and 2: low resistivity
  - Long Interconnect
  - Vdd, Vss
- Polysilicon: higher resistivity
  - Middle-length Interconnect, resistor
  - e.g. intre-cell gate interconnections
- Diffusion: very-high resistivity
  - Source/Drain short interconnections
  - High Capacitance
II. Design of CMOS Cells

1. Combinatorial Logic Cells
   1. Complementary Logic (CMOS)
   2. (NMOS/Pseudo-NMOS Logic)
   3. Switch-Based Logic
2. Layout Design
   1. Design Rules
   2. Layout Design
3. Sequential Logic Cells
   1. Elementary Memory Cells
   2. ROM and RAM Memory
4. Delay and Power
   1. Delay Models
   2. Fan-In and Fan-Out
   3. Static and Dynamic Power
5. Interconnections
II.3 Sequential Logic Circuits

3.1 Elementary Memory Cells

1. Static Memory Basic Cell: Latch

\[ D \xrightarrow{\phi} Q \]

\[ \phi = 0: D \text{ is disconnected}, Q \text{ is fed back} \]

\[ \phi = 1: Q \text{ samples } D \]

With Static Gates

2. Dynamic Memory Cell – MOS Capacitor: \( C = f(\text{area}) \)
   - State (‘1’) (voltage level) is stored for few ms
     - Leakage Current
     - Need for refreshing state

Ex. Shift Register

\[ D \xrightarrow{\phi_1} \xrightarrow{\phi_2} Q \]
II.3 Sequential Logic Circuits

• 3. D Flip-Flop (edge-triggered)
  – Two serial latches
  – D is sampled in inverter (1) when clk = 0
  – Latch (1) and (2) keeps D value when clk = 1 until !D is transferred to second latch (3) and (4)
  – Asynchronous clear signal: replace inv. (1) and (4) by NAND

• D flip-flop optimisations
  – C²MOS Flip-Flop – negative edge triggered
    • Higher speed
    • Less sensible to clock skew
II.3.4 ROM and RAM Memory

• Memory Classification

<table>
<thead>
<tr>
<th>Random-Access Read-Write Memory</th>
<th>Non-Random Access Read-Write Memory</th>
<th>Non-Volatile Read-Write Memory</th>
<th>Read-Only Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>FIFO</td>
<td>EPROM</td>
<td>Mask-Programmed ROM</td>
</tr>
<tr>
<td>DRAM</td>
<td>LIFO</td>
<td>EEPROM</td>
<td>Programmable ROM</td>
</tr>
<tr>
<td></td>
<td>Shift-Register CAM</td>
<td>Flash</td>
<td></td>
</tr>
</tbody>
</table>

• Parameters
  – Address Size, Word Size, Access Time R/W, ...


Memory Architecture

Intuitive architecture for N x M memory
Too many select signals: N words == N select signals

Decoder reduces the number of select signals
K = log₂N

Memory Architecture

Problem: ASPECT RATIO or HEIGHT >> WIDTH


Memory Architecture: Hierarchy

Advantages:
1. Shorter wires within blocks
2. Block address activates only 1 block => power savings

Memory Timing


Elementary Memory Cell

- Non-volatile memory
  - ROM: diode, mask programmed
  - PROM: transistor, one-time programmable (metallization, avalanche effect)
  - EPROM: floating-gate MOS, programmable (avalanche), erasable by UV
  - Flash, EEPROM: floating-gate MOS
    - avalanche injection of electrons in the floating gate results in higher Vt, erasable electrically (tunnel effect)
- Read-write memory
  - DRAM: dynamic latch (capacitance)
    - simple cells, refreshing cycle (# ms)
  - SRAM: static latch
    - complex cells
Read-Only Memory Cells

- Memory Cell $P[i,j]$: row $WL[i]$ (word line) and column $BL[j]$ (bit line) selected
  - With a transistor: $WL[i]=1$, transistor is ON, $BL[j]=0$
  - Without a transistor: $BL[j]=1$
  - Mask programmed par including or not transistors
ROM Memory (NAND type)

- All word lines high (WL[j]=1) by default with exception of selected row WL[i]=0
- Without a transistor: BL[j]=1
- With a transistor: WL[i]=0, transistor is ON, BL[j]=0
- No Vdd or Vss connections: size is decreased
- Performance is decreased w.r.t. NOR-type ROM

Floating-gate transistor

- Floating gate
- Gate
- Source
- Drain
- Substrate
- Device cross-section
- Schematic symbol
Floating-Gate Transistor Programming

Avalanche injection

High voltage creates an Avalanche Effect.
Electrons are trapped on the floating-gate

Removing programming voltage leaves charge trapped
Electrons stay trapped for a lower voltage

Programming results in higher $V_T$
Applying Vdd=5V, transistor effect will not happen since threshold voltage $V_T$ is now 7.5V

Read-Write Memories (RAM)

• Static RAM (SRAM)
  – Data stored as long as supply is applied
  – Large (6 transistors/cell)
  – Fast
  – Differential

• Dynamic RAM (DRAM)
  – Periodic refresh required
  – Small (1-3 transistors/cell)
  – Slower
  – Single Ended
6-Transistor CMOS SRAM Cell

- Latch where WL replaces clock
- Dual-rail bit-lines required to increase noise margin during R/W
- WL selection: WL[i] = 1
- Write 0: BL=0 et IBL=1 ↔ Reset of Latch
- Read: BL et IBL pre-charged to 1, WL selection -> BL=Q and IBL=!Q
  - Sense amplifiers will act as a comparator to increase speed of Latch value to output

1-Transistor DRAM

- Refresh: read followed by write (every 2-4ms)
- Write: WL = 1, input on BL, Cs is charging/discharging
- Read: WL = 1, BL pre-charged to Vdd/2, charge exchange between Cs and Cbl, read is destructive (need for refresh)
- Amplification is necessary after read
- Cs is a junction capacitor (transistor)
3-Transistor DRAM

- 2 lines WL and BL: read and write
- No amplification

II. Design of CMOS Cells

1. Combinatorial Logic Cells
   1. Complementary Logic (CMOS)
   2. (NMOS/Pseudo-NMOS Logic)
   3. Switch-Based Logic
2. Layout Design
   1. Design Rules
   2. Layout Design
3. Sequential Logic Cells
   1. Elementary Memory Cells
   2. ROM and RAM Memory
4. Delay and Power
   1. Delay Models
   2. Fan-In and Fan-Out
   3. Static and Dynamic Power
5. Interconnections
II.4.1 Propagation Time

CMOS Inverter

\[ E(t) = 0: S(t) = Vdd \cdot [1 - e^{-t/\tau_p}] \quad \tau_p = Rp \cdot C_L \]

\[ E(t) = 1: S(t) = Vdd \cdot e^{-t/\tau_n} \quad \tau_n = Rn \cdot C_L \]

\[ T_p = \text{MAX}(T_{p\text{HL}}, T_{p\text{LH}}) \]

I/O Transfer Characteristic

- Stable states: \( E = Vss \) or \( Vdd \)
  - Static power \# 0 (only transistor leakage)
- Transition state: \( Vt < E < Vdd - Vt \)
  - NMOS and PMOS partially opened
  - Short circuit power: \( P = F T \cdot C_{\text{eff}} \cdot Vdd^2, C_{\text{eff}} = C_L \cdot \text{Prob}(0 \leftrightarrow 1) \)
  - Signal slopes have to be fast
**NMOS Parasitic Elements**

**NMOS Enhancement**

\[
R_{ds} = \frac{1}{k(V_{dd} - V_t)} = \frac{1}{k(V_{dd} - V_t) W} L
\]

Gate Capacitor: \(C_g = \frac{eWL}{\text{tox}} = WLC_{ox}\)

Drain/Source-Bulk Capacitor: \(C_{sb} = C_{ds} \approx WLC_i\)

Delay: \(\tau = R_{ds} C_g = \frac{L^2}{\mu(V_{dd} - V_t)}\)

---

**Gate/drain/source/bulk capacitances**

- \(C_{db2}\): Pmos drain capa. \((C_{OP})\)
- \(C_{db1}\): Nmos drain capa. \((C_{ON})\)
- \(C_{gs12}\): Nmos/Pmos gate-drain capa.
- \(C_{OP} + C_{ON} = C_{ON}(1+a)\)
  - with \(a = (W/L)_{P} / (W/L)_{N}\)
- \(C_{int} = C_{db1} + C_{db2} + 2. C_{gs12}\)
- \(C_{int}\): wire capa. = \(C_{itx}\)
- \(C_{gs3}\): gate capa.
- \(C_{ext} = C_{gs} + C_{int}\)
- \(C_L = C_{int} + C_{itx} + C_{ext}\)

Internal Capacitance: \(C_{int}\)
External Capacitance: \(C_{ext}\)
Interconnect Capacitance: \(C_{itx}\)
**Inverter Propagation Time**

\[
\begin{align*}
\tau_{nH} &= \frac{1}{k_n} \left( V_{DD} - V_{TN} \right) \\
\tau_{pH} &= \frac{1}{k_p} \left( V_{DD} - V_{TP} \right) \\
\tau_p &= \frac{\tau_{pH} + \tau_{nH}}{2} = \frac{C_L}{2(V_{DD} - \left| V_T \right|)} \left( \frac{1}{k_n} + \frac{1}{k_p} \right) \\
R_m &= \frac{1}{2(V_{DD} - \left| V_T \right|)} \left( \frac{1}{k_n} + \frac{1}{k_p} \right)
\end{align*}
\]

\[
t_p = C_L \int_{V_{GS}}^{V_{DS}} \frac{dV}{i_{DS}(V)} = C_L \frac{V_{DS} - V_{GS}}{2}
\]

**Simplified model**

\[t_p = R_{DS}\cdot C_L = R_{DS}[C_{int} + C_{itx} + C_{ext}]\]

- \(R_{DS}\) is transistor drain-source resistance
- Temperature ↑: mobility ↓; propag. time ↑; \(I_{ds\ max}\) ↓

\[t_{phl} = Rp \cdot C_l \quad \text{and} \quad t_{ph} = Rn \cdot C_l\]

\(Rn\): NMOS resistance
\(Rp\): PMOS resistance (\(Rp \neq 2Rn\))
Exemples

k-input NAND

\[ t_{phh} = t_{phl} = \]

k-input NOR

\[ t_{phh} = t_{phl} = \]

Transistor Sizing

• CMOS inverter
  – A given technology gives, for a NMOS transistor with size \( 2\lambda:2\lambda \) (L:W), values for \( R_{nu} \) et \( C_{gn} \) of 1250\( \Omega \) and 0.3fF.
  – For NMOS transistors with size \( 2\lambda:6\lambda \), and PMOS with size \( 2\lambda:12\lambda \), give the values of:
    – \( R_{n} = \) \( R_{p} = \)
    – \( C_{gn} = \) \( C_{gp} = \)
  – What is the delay of this inverter loaded by the same inverter?
Transistor Sizing

- Complex function
  - Same technology as previous inverter
  - \( T_{plh} = \)
  - \( T_{phl} = \)
- Indicate critical path
- Which input values give the best/worst case delay?

Propagation Delay of Cascaded Cells

- Give \( S_1 \) as a function of \( A \) and \( B \).
- Give \( S \) as a function of \( S_1, A \) and \( B \).
- Give gate count and propagation delay of the cell.
II.4.2. Fan-in and Fan-out

- **Fan-In (or Drive):** relative to size of transistors
  - Basic inverter is 1x
- **Fan-Out:** ratio between load capacitance and drive
- **Relative Fan-Out (RF):** ratio between fan-out and next-stage fan-in

\[
\begin{array}{c|c|c|c|c|c}
\text{Porte} & \text{FIN} & \text{FOUT} & \text{RF} \\
A & 3 & 4 & 4/3 \\
B & 2 & 2 & 1 \\
C & 1 & 1 & 1/2 \\
D & 1 & 1 & 1/2 \\
\end{array}
\]

\[
\begin{align*}
\text{Tp} &= \text{TD} + \text{ID} \\
\text{Tp} &= R_{DS}[C_{\text{int}} + C_{\text{ext}}]
\end{align*}
\]

Logic-level model

- **Tp** = transport delay + inertial delay = TD + ID
- **Tp** = \(R_{DS}[C_{\text{int}} + C_{\text{ext}}]\)

![Graph showing the relationship between transport delay (TD) and fan-out, with Tp = TD + RF.UD and UD: unit delay]
Cell delay

- 3-input NAND

Example

- Clock tree delay
  - Give Relative Fan-Out (RF) for all nodes
  - Estimate propagation time from E to S: $T_{PE}\_S$
  - Inverter transport delay: $TD=0.29\text{ns}$
  - Inverter unit delay: $UD=0.17\text{ns}$
  - Give an optimized schematic with less $T_{PE}\_S$
II.4.3 Power

- \( P = P_{\text{dyn}} + P_s = P_c + P_{\text{sc}} + P_s \)
- Charging power: \( P_c \)
  - Charge and discharge of node capacitance
- Short-circuit power: \( P_{\text{sc}} \)
  - Short circuit during the commutation of logical structures
- Static power: \( P_s \)
  - Junctions, sub-threshold behaviour

\[ P_{\text{dy}n} = \alpha \cdot C_L \cdot V_{dd}^2 \cdot f \]

Static power (1)

- Sub-threshold Leakage Current
  - Even if \( V_{gs} < V_{t} \) MOS transistors MOS are not completely off
  - If \( V_{dd} \) decreases (towards \( V_{t} \)) leakage currents increase quickly
- Source/Drain-Bulk junction leakage (diodes)

\[ I_{off} = I_e \frac{V_T}{n^*U_T} \]

\[ P_s = N_{Tr} \cdot V_{dd} \cdot I_{o.e} \frac{V_T}{n^*U_T} \]
Static power (2)

- Impact of threshold voltage

- Recent technologies use transistors with 2 Vt
  - Low-Leakage or High-Performance cells

Static power (3)

- 130 nm Technology

<table>
<thead>
<tr>
<th>$I_{\text{static}}(A)$</th>
<th>slow-slow</th>
<th>typical</th>
<th>fast-fast</th>
</tr>
</thead>
<tbody>
<tr>
<td>-10°C</td>
<td>2.1E-06</td>
<td>1.2E-05</td>
<td>7.0E-05</td>
</tr>
<tr>
<td>25 °C</td>
<td>1.7E-05</td>
<td>8.2E-05</td>
<td>3.9E-04</td>
</tr>
<tr>
<td>50 °C</td>
<td>6.1E-05</td>
<td>2.5E-04</td>
<td>1.1E-03</td>
</tr>
</tbody>
</table>

- Circuit with 5 million transistors
  - 0.6 Volt, 1 mA leakage current
  - Higher than total specified current...
- Cannot be still neglected!

[Piguet03]
Dynamic power (1)

- Charging current: $I_c$

  \[ P_c = \alpha f C_L Vdd^2 \]
  
  $\alpha$: activity, $C_L$: total load capacitance, $f$: frequency

Dynamic power (2)

- Energy per transition = $C_L Vdd^2$
- Power = Energy per transition x rate of transition
  
  \[ P_c = C_L Vdd^2 f_{0\to1} \]
  
  \[ P_c = C_L Vdd^2 f \text{Prob}_{0\to1} \]
  
  \[ P_c = \alpha C_L Vdd^2 f \]
  
  \[ P_c = C_{EFF} Vdd^2 f \]

- Effective capacitance $C_{EFF} = \alpha C_L$
- Power does not directly depend on transistor size of the considered cell

Power \[\Rightarrow\] Data dependant
Activity dependant
Dynamic power (3)

- Short-circuit current: $I_{sc}$
  - Short-circuit path during commutation where NMOS and PMOS are simultaneously on
  - For correctly designed circuits: $\approx 15\%$ of total power
  - Slow slopes?

\[ P_{cc} = \alpha.f.t.K.(W/L).(V_{dd} - 2V_t)^{3/2} \]

$\alpha$: activity, $t$: rise/fall time, $K$: techno., $W/L$: size

Dynamic power (4)

- Short-circuit current: impact of rise/fall
  - Short-circuit time
  - Depends on gate load

\[ P_{cc} = \alpha.f.t.K.(W/L).(V_{dd} - 2V_t)^{3/2} \]
Dynamic power is data dependant

- Example: inverter
  \[ P_0 \rightarrow 1 = P(\text{OUT} = 0) \times P(\text{OUT} = 1) = 1/4 \]

- Example: 2-input static NOR gate
  - Transition probability of a NOR gate
    \[
    P_A = P(A=1); \quad P_B = P(B=1) \\
    P_1 = P(S=1) = (1-P_A)(1-P_B) \\
    P_{0 \rightarrow 1} = P_0 \cdot P_1 = (1-(1-P_A)(1-P_B))(1-P_A)(1-P_B)
    \]
    \[
    C_{eff} = 3/16.CI
    \]

  - Depends on input statistics
    \[
    \begin{array}{c|c|c|c}
    & A & B & \text{OUT} \\
    \hline
    0 & 1 & 0 & 0 \\
    1 & 0 & 0 & 0 \\
    \end{array}
    \]

    \[
    \begin{array}{c|c|c}
    & P_{0 \rightarrow 1} \\
    \hline
    \text{AND} & (1-P_AP_B) P_A P_B \\
    \text{OR} & (1-P_A)(1-P_B)(1-P_A)(1-P_B) \\
    \text{XOR} & (1-(P_A+P_B-2 P_A P_B)) (P_A+P_B-2 P_A P_B) \\
    \end{array}
    \]
Transition probabilities

- Probability propagation

P(A) = ½
P(B) = ½
P(C) = ½

Transition probabilities

- Re-convergence: conditional probabilities

P(Z=1) = P(B=1) \times P(X=1|B=1)

- Becomes quickly complex!
Glitches

- Glitch
  - Dynamic hazards
  - Useless behaviour
  - Important useless power

Example 1: Chain of NAND gates
III. IC Design Methods

1. IC Classification
   1. Full-Custom
   2. Standard-Cells
   3. Gate Array
   4. Programmable Logic (CPLD, FPGA)
   5. Metrics

2. Design Methods and CAD Tools
   1. Design flow
   2. Top-Down or Bottom-Up
   3. CAD Tools

3. IC Specification
   1. Detailed specifications
   2. Outline of a specification document
III.1. IC Classification

- Trade-off between: flexibility, cost, design time (time-to-market), prototyping, foundry, tools

Digital Circuit Implementation Approaches

Custom • Semicustom

Cell-based • Array-based

Standard Cells Compiled Cells • Macro Cells • Pre-diffused (Gate Arrays) • Pre-wired (FPGA’s)

1. Full-Custom Circuits

- Each transistor can be optimized
- Advantages
  - density, performance, power
- Drawbacks
  - cost
  - design time (6/17 trans./day)
  - re-engineering cost
- Used in:
  - processors
  - memory
  - spatial domain
2. Cell-Based (or Standard Cells) Circuits

- Based on cells in a technological library
- Use place&route for metal routing

Library of standard cells defined at layout level: design kit from foundry
- Logic gates
- Flip-flop, latches
- Buffers, mux, decoders
- Adder/subtractor
- I/O Pad, level shifter...
- CAN, CNA, PLA, UART, μP

Variable width
Fixed height

I/O Pads
Compiled Blocks
ROM, RAM multiplier

Standard Cell Library

- Cell example: 3-input NAND

<table>
<thead>
<tr>
<th>Truth Table</th>
<th>Symbol</th>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>NAND</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>NAND</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>NAND</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>NAND</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>NAND</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>NAND</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>NAND</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>NAND</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

AC Characteristics

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Symbol</th>
<th>( V_{OL} )</th>
<th>( V_{OH} )</th>
<th>( I_{OL} )</th>
<th>( I_{OH} )</th>
<th>( f_{MAX} )</th>
<th>( t_{PD} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay &amp; Rise</td>
<td>( t_{PD} )</td>
<td>2.0 ns</td>
<td>3.0 ns</td>
<td>1.0 ns</td>
<td>3.0 ns</td>
<td>1.0 ns</td>
<td>3.0 ns</td>
</tr>
<tr>
<td>Output Drive</td>
<td>( I_{OH} )</td>
<td>2.0 mA</td>
<td>3.0 mA</td>
<td>1.0 mA</td>
<td>3.0 mA</td>
<td>1.0 mA</td>
<td>3.0 mA</td>
</tr>
<tr>
<td>Delay &amp; Fall</td>
<td>( t_{PD} )</td>
<td>2.0 ns</td>
<td>3.0 ns</td>
<td>1.0 ns</td>
<td>3.0 ns</td>
<td>1.0 ns</td>
<td>3.0 ns</td>
</tr>
<tr>
<td>Output Drive</td>
<td>( I_{OH} )</td>
<td>2.0 mA</td>
<td>3.0 mA</td>
<td>1.0 mA</td>
<td>3.0 mA</td>
<td>1.0 mA</td>
<td>3.0 mA</td>
</tr>
<tr>
<td>Output Drive</td>
<td>( I_{OL} )</td>
<td>2.0 mA</td>
<td>3.0 mA</td>
<td>1.0 mA</td>
<td>3.0 mA</td>
<td>1.0 mA</td>
<td>3.0 mA</td>
</tr>
<tr>
<td>Output Drive</td>
<td>( I_{OL} )</td>
<td>2.0 mA</td>
<td>3.0 mA</td>
<td>1.0 mA</td>
<td>3.0 mA</td>
<td>1.0 mA</td>
<td>3.0 mA</td>
</tr>
<tr>
<td>Output Drive</td>
<td>( I_{OL} )</td>
<td>2.0 mA</td>
<td>3.0 mA</td>
<td>1.0 mA</td>
<td>3.0 mA</td>
<td>1.0 mA</td>
<td>3.0 mA</td>
</tr>
<tr>
<td>Output Drive</td>
<td>( I_{OL} )</td>
<td>2.0 mA</td>
<td>3.0 mA</td>
<td>1.0 mA</td>
<td>3.0 mA</td>
<td>1.0 mA</td>
<td>3.0 mA</td>
</tr>
</tbody>
</table>
Cell-Based (or Standard Cells) Circuits

- **Advantages**
  - Reduction of cost and design time
    - reuse pre-designed cells
    - design kit
  - Chip includes only needed cells
  - But challenge is on place and route
- **Drawback**
  - Cost is still high for low volume
- **Usually combines**
  Full-Custom and Standard-Cells

```
Feedthrough Cell
Logic Cell
```

Place and Route

- 4-Bit Counter (up/down) with enable
3. Gate Array

- High-volume production of transistors
  - Regular and dense organization
  - Fabrication on wafers
  - Reduces cost of fabrication
- Customized metal levels to match design of customer
- Fabrication of the customized chip at lower volume and cost
Gate Array

- Advantage: cost and design time
- Drawbacks: lower integration density
  - Unused transistors
  - Number of transistors is fixed
4. PLD: Programmable Logic Devices

PLD: Programmable Logic Device

CPLD: Complex PLD

I/O Block

EPLD (Altera)

Primary inputs

Macrocell

Courtesy Altera Corp.
FPGA: Field-Programmable Gate Arrays

- Matrix of Logic Blocks and Programmable Interconnects
- FPGA types
  - SRAM FPGA
  - Antifuse/Flash FPGA
- Examples:
  - Altera: Stratix, Cyclone
  - Xilinx: Virtex, Spartan
  - Actel/Microsemi: Igloo

SRAM-based FPGAs

- Xilinx Architecture
SRAM-based FPGAs

• Xilinx Basic Cell (CLB)

Combinational logic

Storage elements

Courtesy of Xilinx

SRAM-based FPGAs

Xilinx XC4025

Xilinx Virtex
FPGA Xilinx VIRTEX 7

<table>
<thead>
<tr>
<th>Virtex-7 FPGAs</th>
<th>XC7V585T</th>
<th>XC7V900T</th>
<th>XC7V1100T</th>
<th>XC7V1300T</th>
<th>XC7V1500T</th>
<th>XC7V2000T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin Count</td>
<td>27,536</td>
<td>49,456</td>
<td>71,400</td>
<td>92,352</td>
<td>112,304</td>
<td>132,256</td>
</tr>
<tr>
<td>Logic Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LUTs</td>
<td>42,700</td>
<td>70,450</td>
<td>91,200</td>
<td>111,950</td>
<td>132,700</td>
<td>153,450</td>
</tr>
<tr>
<td>Slice Cells</td>
<td>296,080</td>
<td>496,080</td>
<td>696,080</td>
<td>896,080</td>
<td>1,096,080</td>
<td>1,296,080</td>
</tr>
<tr>
<td>CLB Flip-Flops</td>
<td>517,000</td>
<td>861,000</td>
<td>1,161,000</td>
<td>1,461,000</td>
<td>1,761,000</td>
<td>2,061,000</td>
</tr>
<tr>
<td>Memory Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block RAM/FIFOs</td>
<td>3,475</td>
<td>5,686</td>
<td>8,898</td>
<td>11,110</td>
<td>13,322</td>
<td>15,534</td>
</tr>
<tr>
<td>Timer Block Memory (bits)</td>
<td>410</td>
<td>910</td>
<td>1,410</td>
<td>1,910</td>
<td>2,410</td>
<td>2,910</td>
</tr>
<tr>
<td>PCI Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI Express Interface Blocks</td>
<td>14</td>
<td>22</td>
<td>30</td>
<td>38</td>
<td>46</td>
<td>54</td>
</tr>
<tr>
<td>Embedded User IP Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analog Front End (AFED)</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Configuration AES/IMAC Blocks</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>QTH 16.050Gbps Transceivers</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>QTH 13.925Gbps Transceivers</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>GTZ 26.050Gbps Transceivers</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Speed Grades</td>
<td>Commercial</td>
<td>Industrial</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration Memory (Megibits)</td>
<td>75.4</td>
<td>115.4</td>
<td>165.4</td>
<td>215.4</td>
<td>265.4</td>
<td>315.4</td>
</tr>
</tbody>
</table>

5. Metrics

<table>
<thead>
<tr>
<th>Integration</th>
<th>Full Custom</th>
<th>Standard Cell</th>
<th>Gate Array</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Performances</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Generic/Reuse</td>
<td>❌</td>
<td>❌</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Design Time</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>CAD Tools</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Complexity</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Cost (low volume)</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Cost (high volume)</td>
<td>✔️</td>
<td>✔️</td>
<td>❌</td>
<td>❌</td>
</tr>
</tbody>
</table>
Cost

• Total Cost = NRE + UP x volume
  – NRE: Non Recurring Expenses
  – UP: Unit Price
• Cost per chip = NRE/volume + UP
• Yield: \( \rho = (1+N_0.A)^{-n} \)
  – A: Chip Area
  – \( A_{tot} \): Wafer Area
  – n: number of masks
  – \( N_0 \): defect density per mask
• Unit Price = Wafer Price / (number of chip x \( \rho \))
  = Wafer Price \( (1+N_0.A)^n \) \( (A/A_{tot}) \)
  = K.A.(1+n.N_0.A) if \( N_0.A << 1 \)

NRE: Non Recurring Expenses

• Mask set: up to several M$
• Fabrication: circuit fabrication costs; eventually cost of P&R and DRC; test options, yield options, performance, analog precision, etc.
• CAD Tools: 10-100k€ plus maintenance
• Hardware: servers, disks, etc.

• Engineers...
• Re-design costs...
Cost

• Total Cost = NRE + UP x volume
  – NRE: Non Recurring Expenses
  – UP: Unit Price
• Cost per chip = NRE/volume + UP

III. IC Design Methods

1. IC Classification
   1. Full-Custom
   2. Standard-Cells
   3. Gate Array
   4. Programmable Logic (CPLD, FPGA)
   5. Metrics

2. Design Methods and CAD Tools
   1. Design flow
   2. Top-Down or Bottom-Up
   3. CAD Tools

3. IC Specification
   1. Detailed specifications
   2. Outline of a specification document
Design Productivity Gap

- Productivity is limited by design methods and CAD tools

![Design Flow Diagram]

Design Flow

Top-Down Design

- Specifications
- High-Level Design
- RTL/Logic Design
- Placement/Routing
- Physical Level

Bottom-Up Design

- Specifications
- High-Level Design
- RTL/Logic Design
- Placement/Routing
- Physical Level

Refinement of Specifications

Abstraction of Specifications
Design Flow: from algorithm to circuit

Algorithm

Gate Level

Register Transfer Level (RTL)

Layout Level

Transistor or Circuit Level

Hardware vs. Software

- Hardware
  - Layout-Level
  - Logic-Level
  - Register-Transfer Level (VHDL, Verilog)
  - Algorithmic Level (C/C++, System Verilog)
  - System Level (e.g. UML, Matlab/Simulink)

- Software
  - Binary Code
  - Assembly Code
  - Machine Dependent Languages (e.g. C)
  - Virtual Machine (e.g. Java)
  - System Specifications (e.g. UML)
The Verification Loop

IC Design: a Real Team Work

- Teams of several engineers to design an IC (10-100)
  - Accurate task/block decomposition
    - Clear definition of interface and data exchange
  - Task/work distribution
    - Blocks to People
    - Front End, Back End, Verification Tasks
  - Frequent Meetings for
    - Specification Updates
    - Design Follow-Up

One Person One Block
Each Block Communicates with Others
III. IC Design Methods

1. IC Classification
   1. Full-Custom
   2. Standard-Cells
   3. Gate Array
   4. Programmable Logic (CPLD, FPGA)
   5. Metrics

2. Design Methods and CAD Tools
   1. Design flow
   2. Top-Down or Bottom-Up
   3. CAD Tools

3. IC Specification
   1. Detailed specifications
   2. Outline of a specification document

---

Spécification d’un ASIC

• Le terme spécification regroupe toutes les informations qui caractérisent de l'extérieur le composant à réaliser. Les “spéc” sont indépendantes de l'utilisation qui est faite du circuit. Elles ont pour but de décrire ce que doit faire le composant (le QUOI) et pas du tout comment il le fait (le COMMENT).

• Généralement sous forme papier accompagné de modèles
Nature de la spécification d’un circuit

• Spécifications fonctionnelles
  – Description des fonctions que doit assurer le circuit (ou le bloc)
    • Équation logique, table de vérité, chronogramme
    • Diagramme état/transition, Graficet, Statechart (extension du diagramme d’état au parallélisme)
    • Réseau de Petri (comportement temporel)
    • Modèle mathématique, signal, commande
    • Description algorithmique

• Spécifications opératoires
  – Manière dont une fonction doit opérer, conditions et domaines de fonctionnement
    • Renseignements sur les grandeurs ou données utilisées dans les spécifications fonctionnelles (type, domaine de définition, précision)
    • Informations pour guider les concepteurs dans le choix des solutions à mettre en œuvre (expériences précédentes dans l’entreprise ou ailleurs)
    • Test à opérer sur le circuit

• Spécifications technologiques
  – Renseignements en rapport avec la réalisation matérielle
    • Définition électrique des E/S
    • Performances, contraintes
    • Spécifications liées à la réalisation (taille, coût, technologie, type de boîtier,...)
    • Contraintes de l’environnement
    • Qualité de test

Spécification détaillée d’un ASIC

• Une spécification détaillée (Detailed Design Specification) est un document écrit qui regroupe
  – La spécification de la définition
  – La notice descriptive de fonctionnement

• Elle doit fournir toutes les informations utiles aux concepteurs des cartes utilisatrices, ainsi qu’aux concepteurs de logiciels.
Plan type de spécification détaillée

1. Introduction
   - Rappel de l'utilisation du circuit sur la carte
   - Rôle principal et fonctions

2. Environnement du circuit
   - Présentation générale de l'environnement du circuit sur la carte

3. Organisation générale du circuit
   - Interfaces externes
   - Présentation générale des fonctions
     - Synoptique générale du circuit faisant apparaître les blocs fonctionnels
     - Description des liaisons inter-blocs

4. Fonctions réalisées
   - Description détaillée des fonctions réalisées par le circuit
     - fonctions d'entrées-sorties
     - fonction d'initialisation
     - fonctions pour le test du composant
     - fonctions pour le test en sortie de fabrication des cartes utilisant le circuit

Plan type (suite)

5. Caractéristiques matérielles
   - Interface physique
     - Packaging
     - Brochage, description des signaux d'E/S et des alimentations
     - Technologie des E/S
   - Chronogramme et Timings
     - Chronogrammes théoriques associés aux diverses fonctions
     - Timings à respecter
   - Caractéristiques électriques
     - Limites électriques, conditions transitoires et opérationnelles
     - Caractéristiques statiques et dynamiques des E/S (alimentations, tension, courant, capacités de charge, Slew Rate Control, ...)
     - Découplage d'alimentation à prévoir sur la carte

6. Interface Logiciel
   - Synthèse des informations relatives à l'interface matériel/logiciel
   - Description des registres et compteurs accessibles et de leur mode d'accès
   - L'utilisateur logiciel doit pouvoir se limiter à ce paragraphe pour le développement des logiciels
Spécification de réalisation

- Dans la phase de conception des blocs, cette spécification précise aux concepteurs les directives de réalisation
- Décrit l’architecture interne et fournit une description détaillée de chacun des blocs constituant le circuit
- Ce document est indispensable pour les circuit dont la complexité nécessite plusieurs concepteurs
- En cours de conception, la spécification fait l’objet de mises à jour

Plan type

- Présentation générale de la découpe en blocs
  - Diagramme général du circuit découpé
  - Interfaces inter-blocs ou avec l’extérieur
    - Bus et liaisons de contrôle principaux, distribution d’horloge interne, cycles d’échanges entre blocs
  - Pour chaque bloc :
    - description succincte de la fonction réalisée
    - estimation de la complexité
    - mémoires ou cellules spécifiques nécessaires (capacité, temps de cycle, simple/multi port...)
    - fréquence moyenne de fonctionnement et taux d’activité

Plan type (suite)

- Contraintes de réalisation
  - Boîtier : type de montage sur carte (soudé, support)
  - Nombre et type d’E/S
  - Fréquence aux accès
  - Contraintes principales de timing
  - Bilan des mémoires et/ou cellules nécessaires
  - Technologie (CMOS, bipolaire, ECL, ...)
  - Référence fondeur du circuit

- Description détaillée des interfaces inter-blocs
  - Fonctions et chronogrammes de chaque bus ou liaison de contrôle interne

- Description détaillée de chaque bloc
  - Fonctions réalisées
  - Accès E/S
  - Structure interne du bloc
  - Complexité du bloc (nombre de portes)
IV Synchronous Design of IC

1. Synchronous Design Rules and Principles
   1. General Issues and Parameters: critical path, clock skew
   2. Synchronous Rules (Ten Commandments)
   3. Recommended and Non-Recommended Circuits

2. Finite State Machine (FSM) plus Datapath Model
   1. Synchronous Models
   2. State Diagrams
   3. Moore/Mealy Machines

3. Arithmetic Operators
   1. Adders and Substracteurs
   2. Multipliers

Timing Parameters

- D Flip-Flop
  - Setup Time: $T_{\text{setup}}$
  - Hold Time: $T_{\text{hold}}$
  - Propagation Time: $T_p$
  - on Clock and Reset

<table>
<thead>
<tr>
<th></th>
<th>$D$</th>
<th>$CP$</th>
<th>$Q$</th>
<th>$QN$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truth Table</td>
<td>$D$</td>
<td>$CP$</td>
<td>$Q$</td>
<td>$QN$</td>
</tr>
<tr>
<td></td>
<td>$D$</td>
<td>$CP$</td>
<td>$Q$</td>
<td>$QN$</td>
</tr>
</tbody>
</table>

Physical Dimensions

<table>
<thead>
<tr>
<th>Property</th>
<th>FD1QLL</th>
<th>FD1QLLP</th>
<th>FD1QLLM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area (mm$^2$)</td>
<td>32.311</td>
<td>20.311</td>
<td>20.218</td>
</tr>
</tbody>
</table>

Capacitance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>$D$</th>
<th>$CP$</th>
<th>$Q$</th>
<th>$QN$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$D$</td>
<td>$CP$</td>
<td>$Q$</td>
<td>$QN$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>$D$</th>
<th>$CP$</th>
<th>$Q$</th>
<th>$QN$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$D$</td>
<td>$CP$</td>
<td>$Q$</td>
<td>$QN$</td>
</tr>
</tbody>
</table>

VLSI Integrated Circuits and Systems: Principles and Design Methods

Olivier Senteys, INRIA/IRISA - ENSSAT
Power

- Leakage and Dynamic Power

### Average Leakage Power

<table>
<thead>
<tr>
<th>Cell</th>
<th>bc_1.3V_2SC</th>
<th>bc_1.32V_12SC</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD1QLL</td>
<td>10767200</td>
<td>200860000</td>
</tr>
<tr>
<td>FD1QLP</td>
<td>12226500</td>
<td>236294000</td>
</tr>
<tr>
<td>FD1QLX4</td>
<td>15436600</td>
<td>300319000</td>
</tr>
</tbody>
</table>

### Internal Energy at minimum output load

UWMHz as a function of Tt (input transition time in ns)

<table>
<thead>
<tr>
<th>Cell</th>
<th>Cycles on pin</th>
<th>nom_1.2V_2SG</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD1QLL</td>
<td>Qrelated pin(CP)</td>
<td>0.029 + 0.013*Tr</td>
</tr>
<tr>
<td>FD1QLL</td>
<td>D(max)</td>
<td>0.020 + 0.007*Tr</td>
</tr>
<tr>
<td>FD1QLP</td>
<td>CP(max)</td>
<td>0.016 + 0.012*Tr</td>
</tr>
<tr>
<td>FD1QLP</td>
<td>Qrelated pin(CP)</td>
<td>0.038 + 0.013*Tr</td>
</tr>
<tr>
<td>FD1QXL</td>
<td>D(max)</td>
<td>0.021 + 0.008*Tr</td>
</tr>
<tr>
<td>FD1QLP</td>
<td>CP(max)</td>
<td>0.017 + 0.012*Tr</td>
</tr>
<tr>
<td>FD1QXLX4</td>
<td>Qrelated pin(CP)</td>
<td>0.060 + 0.013*Tr</td>
</tr>
<tr>
<td>FD1QLX4</td>
<td>D(max)</td>
<td>0.022 + 0.007*Tr</td>
</tr>
<tr>
<td>FD1QLX4</td>
<td>CP(max)</td>
<td>0.017 + 0.012*Tr</td>
</tr>
</tbody>
</table>

Synchronous Circuits

Data → Flip-Flop → Combinational Logic → Flip-Flop

200
Synchronous Circuits

- Two competing paths
  - Launching path
  - Capturing path

\[
T_{\text{launching\_path}} < T_{\text{capturing\_path}} + T_{\text{clk}}
\]

\[
C_{\text{clk\_tree}} + T_{\text{comb}} < C_{\text{clk\_tree}} + T_{\text{clk}}
\]

\[
T_{\text{comb}} < T_{\text{clk}} \quad \text{(no clk skew)}
\]

Critical Path

- All circuits have a maximal frequency, which is given by finding its critical path
  - Data must be stable when sampled by the clock

- Critical Path
  - Maximal combinational delay between:
    - Q output of a FF → logic → D input of a FF \( (a) \)
    - Q output of a FF → logic → output
    - input → logic → D input of a FF \( (b) \)
    - input → logic → output
Critical Path

- **Tcp**: Critical Path Delay of the Logic
  \[ Tcp = \text{MAX}(D_i), \text{ with } D_i \text{ Delay of path } i \]
  - For every logic path \( i \) in the considered circuit
- **Maximal Frequency**: 
  \[ F_{\text{clk}}^{\text{max}} = \frac{1}{Tcp + Tp + T_{\text{setup}}} \]
- **Example (1)**:

  ![Logic Diagram]

  \[ D_Q D_O D_{\text{setup}} = 1 \]

- **Example (2)**:

  ![Logic Diagram]

  \[ D_Q D_O D_{\text{setup}} = 1 \]
Meta-stability

- Analog state between two logic states
  - Setup or Hold violation
  - Minimum signal pulse width violation
    - e.g. Asynchronous Reset
- Generates and propagates slow signal slopes, unstable states
- Cascade of flip-flop reduces meta-stability probability

Clock Skew

- Every FF receives the clock edge at a different time
- Clock routing
- Slow slopes
- Clock load, clock phases, gates on clock path
**Clock Skew**

- DEC Alpha 21164 (1995)
  - 9.3 M. Transistors, 0.5µ
  - 300 MHz

**Clock Skew**

- Ideal Clock

- Slow Slopes

- With Clock Skew

Skew is the time during which \( \phi_1(t) \cdot \phi_2(t) = 1 \)
Clock Skew: problems

- Skew $\delta$ can be negative or positive
  - Reduction of maximal frequency
  - Maximal skew for circuit operation
    - Worst case is when receiving edge arrives late
      - Edge $f'$ of CLK2 should not violate hold time of D2
      - Race between data and clock

\[
\delta < t_{P1} + \min(t_{\text{comb}}) - t_{\text{hold}}
\]

Clock Trees

- Clock distribution
  - Geometric buffering
  - Tree-based

H-tree: constant skew in each block with equivalent number of flip-flops

Buffering: local reduction of skew

[Diagrams of clock skew and clock trees]
Synchronous Design Rules

• One unique clock phase is connected to all flip-flops of one clock domain
• One unique asynchronous reset is connected to all flip-flops

Don’t touch to clock and asynchronous reset signals!

• A good clock tree and no slow slopes
• Some more rules
  – No combinatorial feedbacks
  – Centralized control of tri-state gates
  – Avoid using too many transmission gates in serial

Synchronous Design Rules

• Non-recommended circuits
  - gated clock
  - ripple clock

Use of asynchronous Reset

• Recommended circuits
  - Enable FF
  - Toggle FF

Use of synchronous Reset
Global asynchronous Reset
Synchronous Design Rules

- Non-recommended circuits

Pulse generation

Oscillators

RS Latches

Recommended circuits

Synchronous pulse generation

Synchronous RS

Les dix commandements en conception de circuits intégrés numériques

1. Une seule horloge maîtresse tu auras et tu ne construiras point de fausses horloges à partir de circuits stabilisés.
2. L’horloge tu ne manipuleras point et tu ne transmettras point à travers une porte logique, car cela causerait des aléas, des fausses transitions et des morts lentement.
3. Tu conserves tous les circuits selon les méthodes de conception synchrones à moins de pouvoir convaincre celui qui paie ton salaire, ou qui attribue ta note que, pour des raisons de rapidité, consommation de puissance ou publication d’articles scientifiques, les circuits synchrones ne peuvent faire affaire.
4. Tu ne t’associeras point avec des indésirables tels que les compteurs en cascade (‘ripple counter’) et les ‘multivibrateurs un-coup (‘monostable’ ou multivibrateur monostable), mais tu cultiveras des amitiés avec les compteurs Johnson, les compteurs pseudo-âlétaires et les bascules avec entrée d’activation (‘enable’).
5. Tous les éléments séquentiels tu raccorderas au RESET global afin que le circuit demeure toujours dans un état connu et défini et que tes simulations ne demeurent point indéfinies pour l’éternité.
6. Tu ne mélangeras point les mises à terre analogique et numérique, car une telle union ne peut mener qu’au désastre.
7. Il ne restera point impruni, celui qui laisse flotter des entrées CMOS.
8. Un reset asynchrone n’est pas prévu pour des tâches telles que remettre un compteur à zéro. Vraiment je te le dis, 6 circuits créés de manière se réinitialiseraient correctement et sembleraient d’appuyer gloire et honneur, mais le septième échouera lamentablement et te précipitera dans la honte et la disgrâce.
9. Les entrées asynchrones impropres tu purgeras en les passant dans au moins une bascule D avant de leur donner accès aux pures variables d’états.
10. Quiconque comprend parfaitement les motivations des précédents commandements saura aussi quelles libertés peuvent être tolérées avec eux. Que celui qui les vole dans l’ignorance prennent garde!
IV Synchronous Design of IC

1. Synchronous Design Rules and Principles
   1. General Issues and Parameters: critical path, clock skew
   2. Synchronous Rules (Ten Commandments)
   3. Recommended and Non-Recommended Circuits

2. Finite State Machine (FSM) plus Datapath Model
   1. Synchronous Models
   2. State Diagrams
   3. Moore/Mealy Machines

3. Arithmetic Operators
   1. Adders and Subtracteurs
   2. Multipliers

General Architecture

• Arithmetic Unit: datapath
  — adders, multipliers, shifters, comparators, etc.
• Memory Unit
  — RAM, ROM, register, register bank, FIFO, etc.
• Control Unit
  — Finite State Machines (FSM), counters, etc.
• Interconnections
  — Bus, tri-states, multiplexers
• Input/Output Unit
**FSM + Datapath Model**

- Control Unit: Moore FSM (or synchronized Mealy)
- Synchronized Inputs for avoiding meta-stability
- Asynchronous Reset on the FSM is mandatory
- One unique or different clock phases

**FSM Specification using State Diagram**

- State Transition Diagram
- Nodes: FSM states
  - Number of nodes: state register size
- Transitions: Boolean conditions of states and inputs
  - All transitions at the output of a state are complementary
- State encoding: numerical value of state
  - binary, gray, Johnson
- FSM outputs depends on states (Moore) and on transitions (for Mealy)
- Clock is implicit
Moore/Mealy

• Moore
  Outputs = f_0(Current State)
  Next State = f_1(Current State, Inputs)

• Mealy
  Outputs = f_0(Current State, Inputs)
  Next State = f_1(Current State, Inputs)

IV Synchronous Design of IC

1. Synchronous Design Rules and Principles
   1. General Issues and Parameters: critical path, clock skew
   2. Synchronous Rules (Ten Commandments)
   3. Recommended and Non-Recommended Circuits

2. Finite State Machine (FSM) plus Datapath Model
   1. Synchronous Models
   2. State Diagrams
   3. Moore/Mealy Machines

3. Arithmetic Operators
   1. Adders and Subtracteurs
   2. Multipliers
### Full-Adder

\[ S = A \oplus B \oplus C_i \]
\[ C_o = AB + BC_i + AC_i \]

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( C_i )</th>
<th>( S )</th>
<th>( C_o )</th>
<th>( Carry )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>delete</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>delete</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>propagate</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>propagate</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>propagate</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>propagate</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>generate</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>generate</td>
</tr>
</tbody>
</table>

### Ripple-Carry Adder

Worst case delay linear with the number of bits
\[ t_d = O(N) \]

\[ t_{adder} \approx (N - 1) t_{carry} + t_{sum} \]

Goal: Make the fastest possible carry path circuit
Complementary Static CMOS Full Adder

28 Transistors

Inversion Property

\[ \bar{S}(A, B, C_i) = S(\bar{A}, \bar{B}, \bar{C_i}) \]

\[ \bar{C}_o(A, B, C_i) = C_o(\bar{A}, \bar{B}, \bar{C_i}) \]
Ripple Carry Adder (V2)

- Exploiting inversion property
- Reduces propagation time

Mirror adder structure

24 transistors
Adder Optimization

- Generation
  - if \( A_i = B_i = 0 \) then \( C_{i+1} = 0 \)
  - if \( A_i = B_i = 1 \) then \( C_{i+1} = 1 \)
  - when \( A_i = B_i \) then \( C_{i+1} \) can be generated

- Propagation
  - when \( A_i \neq B_i \) then \( C_{i+1} = C_i \), carry is propagated

- Accelerating addition
  - \( P_i = A_i \oplus B_i \) propagation at \( i \)
  - \( G_i = A_i \cdot B_i \) generation at \( i \)
  - \( S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i \)

Carry Look Ahead (CLA) Adder

- CLA: avoid long carry propagation through \( N \) FAs by calculating \( P_i \), \( G_i \), \( C_i \) from inputs for each FA \( i \) using logic cells

\[
S_i = P_i \oplus C_i
\]
Carry Look Ahead (CLA) Adder

- At each stage $i$
  - Compute $P_i = A_i \oplus B_i$
  - Compute $G_i = A_i B_i$
  - Compute $C_i$
    
    $C_i = G_{i-1} + P_{i-1} C_{i-1}$
    
    
    $C_i = G_{i-1} + P_{i-1} (G_{i-2} + P_{i-2} C_{i-2}) = ...$
    
    $C_i = G_{i-1} + P_{i-1} G_{i-2} + P_{i-1} P_{i-2} G_{i-3} + ... + P_{i-1} ... P_0 G_0 + P_{i-1} ... P_0 C_0$
  - Compute $S_i = P_i \oplus C_i$
Look-Ahead Cell

Subtractor (two’s complement)

- $A - B = A + \text{NOT}(B) + 1$
Multiplier

- Multiplication of two unsigned numbers

- Signed numbers
  - A=-3, B=10
  - a) P=290
  - b) P=-30

\[
\begin{array}{c}
\text{Multiplier}\ \ 1\ 0\ 1\ 0\ 1\ 0 \\
\text{Multiplier}\ \ 1\ 0\ 1\ 0\ 1\ 0 \\
\text{Partial products}\ \\
\text{Result}\ \\
\end{array}
\]

\[
\begin{array}{c}
\times\ 1\ 1\ 1\ 0\ 1 \\
0\ 0\ 0\ 0\ 0 \\
1\ 1\ 1\ 0\ 1 \\
0\ 0\ 0\ 0\ 0 \\
1\ 1\ 1\ 0\ 1 \\
0\ 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 0 \\
\end{array}
\]

\[
\begin{array}{c}
\times\ 1\ 1\ 1\ 0\ 1 \\
0\ 0\ 0\ 0\ 0 \\
1\ 1\ 1\ 0\ 1 \\
0\ 0\ 0\ 0\ 0 \\
1\ 1\ 1\ 0\ 1 \\
0\ 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 0 \\
\end{array}
\]

\[
\begin{array}{c}
0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0 \\
1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0 \\
\end{array}
\]

Multiplier

- \(X\) (M bits) and \(Y\) (N bits) unsigned
- \(Z\) (M+N bits)

\[
Z = X \times Y = \sum_{i=0}^{M+N-1} Z_i \times 2^i
\]

\[
Z = \left( \sum_{i=0}^{M-1} X_i \times 2^i \right) \times \left( \sum_{j=0}^{N-1} Y_j \times 2^j \right)
\]

\[
Z = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} X_i Y_j \times 2^{i+j}
\]
Array Multiplier

Critical Path: $T_{mult} =$

Booth Algorithm

- Exploit series of 1 or 0
- Suppress null partial products
- Simplifying property: $111 = 1000-1$
Wallace Tree

- Wallace 3
  - \( S = X \oplus Y \oplus Z \); \( C = XY + XZ + YZ \)

- Wallace 5

- Addition of seven 4-bit data

V Logic Synthesis from VHDL

1. Methods, Design Flow, and Tools
2. Register-Transfer Level (RTL) Models using VHDL
   1. VHDL for Logic Synthesis
   2. Combinational Logic
   3. Sequential Logic
3. RT and Logic Synthesis CAD Algorithms
   Structuring, Flattening, Mapping
Logic Synthesis at a Glance

VHDL specification

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
entity nb2 is
  port(DIN: in std_logic_vector(7 downto 0);
       nb: out integer range 0 to 8);
end nb2;
architecture arch of nb2 is
begin
  process(DIN)
  variable nb_int: integer range 0 to 8;
  begin
    nb_int := 0;
    for i in 0 to 7 loop
      nb_int := nbint + conv_integer(DIN(i));
    end loop;
    nb <= nb_int;
  end process;
end arch;
```

Logic optimization

VHDL specification

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
entity nb2 is
  port(DIN: in std_logic_vector(7 downto 0);
       nb: out integer range 0 to 8);
end nb2;
architecture arch of nb2 is
begin
  process(DIN)
  variable nb_int: integer range 0 to 8;
  begin
    nb_int := 0;
    for i in 0 to 7 loop
      nb_int := nbint + conv_integer(DIN(i));
    end loop;
    nb <= nb_int;
  end process;
end arch;
```

Logic Synthesis

- Register-Transfer Level (RTL) Specifications
  - Specification of the logic/arithmetic behavior between clocked registers

- FSM
- decoding
- logic
- selection, multiplexing
- arithmetic operators
- registers, counters, ...
- memory
Advantages of Logic Synthesis (over gate-level design)

- Design flow automation
- Higher level of abstraction: more complex designs
- Hardware Description Language (HDL)
- Independent of technology (more or less...)
- Constrained design flow
  - timing, power, delay, area
- Optimized and analyzed results
This slide is to refresh your memory on VHDL

entity decoder is
  port ( A, B : in bit;
         S : out bit vector (0 to 3) );
end decoder;

architecture behavioral of decoder is
begin
  process (A,B)
  begin
    if A='0' then
      if B='0' then
        S<= "0001"
      else
        S<= "0010"
      end if;
    else
      if B='0' then
        S<= "0100"
      else
        S<= "1000"
      end if;
    end if;
  end process;
end behavioral;

architecture dataflow of decoder is
begin
  S(0) <= not(A) and not(B); 
  S(1) <= not(A) and B;
  S(2) <= A and not(B);
  S(3) <= A and B;
end dataflow;

architecture structural of decoder is
component DEC24
  port (I1, I2 : in bit;
        O1, O2, O3, O4 : out bit );
end component;
begin
  CELL : DEC24
  port map (A,B,S(0),S(1),S(2),S(3));
end structural;

VHDL RTL coding style for synthesis

entity counter is
  port (reset: in bit;
         clk: in bit;
         S: out integer range 0 to 15 );
end counter;

architecture RTL of counter is
begin
  signal count: integer range 0 to 15;
  process(reset,clk)
  begin
    wait until reset = '1' for 20 ns;
    if reset = '1' or count = 15 then
      count :=0;
    else
      count := count + 1 after 10 ns;
    end if;
    if clk'event and clk='1' then
      if count=15 then count <=0;
      elsif count=0 then
        count := count + 1;
      end if;
      S <= count;
    end if;
  end process;
end RTL;

entity counter is
  port (reset: in bit;
         clk: in bit;
         S: out integer := 0);
end counter;

architecture behavioral of counter is
begin
  process variable count: integer := 0;
  begin
    wait until reset = '1' for 20 ns;
    if reset = '1' or count = 15 then
      count :=0;
    else
      count := count + 1 after 10 ns;
    end if;
    S <= count;
  end process;
end behavioral;

Non Synthesizable

Synthesizable
VHDL RTL coding style for synthesis

Data types: Integer (with range)
             Enumerate, Record, Subtype
             1D array of finite dimension
             Bit, Bit_Vector or STD_LOGIC, STD_LOGIC_VECTOR
             Physical, Real, Access, File: ignored

Entity: in, out, inout, default values are ignored

Packages: Collection of resources: types, constants, functions, components
           Standard packages (STD_LOGIC) or technology (design kit)

Declarations: Constant, Signal, Variable, Component
              Register, Bus, Linkage, Alias: ignored

Operators: Logic (and, nand, or, nor, xor, not)
           Comparison (=, /=, <, >, <=, >=)
           Arithmetic (+, -, *, sign, abs)

Attributes: 'length, 'event, 'left, 'right, 'high, 'low, 'range, 'reverse_range

Sequential Instructions (process)
Wait: Supported at the first line of a synchronous PROCESS
      wait until clock = value;
      wait until clock'event and clock = value;
      wait until not clock'stable and clock = value;

Assignment: Assignment of variables and signals: supported
            Functions and procedures: supported
            Transport after: ignored

If/Case: Supported

Loops: for loop with static size: supported
       while loop: not supported

Parallel Instructions (architecture)
Process: Sensitivity list or wait for synchronization

Affectations: Conditional assignment of signals (when, select): supported

Block: Guarded blocks: not supported

Instantiation: Port map, generic map, generate: supported
IEEE Standard Logic 1164 Package

PACKAGE std_logic_1164 IS

  TYPE std_ulogic IS
    'U', -- Uninitialized
    'X', -- Forcing Unknown
    '0', -- Forcing 0
    '1', -- Forcing 1
    'Z', -- High Impedance
    'W', -- Weak Unknown
    'L', -- Weak 0
    'H', -- Weak 1
    '-' -- Don't care
  );

  TYPE std_ulogic_vector IS
    ARRAY (NATURAL RANGE <>) OF std_ulogic;

FUNCTION resolved (s : std_ulogic_vector) RETURN std_ulogic;

SUBTYPE std_logic IS
  resolved std_ulogic;

TYPE std_logic_vector IS
  ARRAY (NATURAL RANGE <>) OF std_logic;

SUBTYPE X01 IS
  resolved std_ulogic RANGE 'X' TO '1';
  ('X','0','1');

VHDL RTL coding style for synthesis

Conditional and parallel assignment of signals

S <= a+b;  S <= a and b  when a < "010"  with a select
      else a xor b when a < "101"  S <= a and b  when 2 downto 0,
      else a or b;

Combinational Process

add: process (a,b,c)
begin
  if c='1' then res <= a + b;
  else res <= a - b;
end if;
end process;

Synchronous Process

process (reset, clock) is begin
  if reset='0' then
    -- actions during async. reset
  elsif clock'event and clock='1' then
    -- synchronous statements
    -- no possible else
  else
    -- no more possible actions
  end if;
end process;

• All read signals must be placed in the sensitivity list of the Process
• All outputs must be assigned for all possible values of the conditions

Sensitivity list of the Process must contain the clock signal and eventually an asynchronous signal (reset)
• Sensitivity list can be replaced by a wait statement
• Outputs must be assigned both in synchronous and asynchronous manners
And then synthesis tools will do the job for you

```vhdl
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity counter is
  port (reset, clk, load, up: in Std_Logic;
        val: in Std_Logic_Vector(3 downto 0);
        count: buffer Std_Logic_Vector(3 downto 0));
end counter;

architecture RTL of counter is
begin
  synchronous: process(reset,clk)
  begin
    if reset='1' then
      count <= "0000";
    elsif clk'event and clk='1' then
      if load = '1' then
        count <= val;
      elsif up = '1' then
        count <= count + "0001";
      else
        count <= count - "0001";
      end if;
    end if;
  end process;
end RTL;
```

## V Logic Synthesis from VHDL

1. Methods, Design Flow, and Tools
2. Register-Transfer Level (RTL) Models using VHDL
   1. VHDL for Logic Synthesis
   2. Combinational Logic
      - Logic gates
      - Multiplexer, tri-state
      - Arithmetic, comparison
      - Decoding
   3. Sequential Logic
3. RT and Logic Synthesis CAD Algorithms
   - Structuring, Flattening, Mapping
Be very Careful with Latches and Logic

Latch

\[
\text{process}(A, S) \begin{cases} 
\text{begin} \\
\quad \text{if } S > 2 \text{ then} \\
\quad \quad X := A; \\
\quad \text{end if}; \\
\text{end process;}
\end{cases}
\]

Logic gates

\[
\text{process}(I) \begin{cases} 
\text{begin} \\
\quad \text{if } I = '0' \text{ then} \\
\quad \quad X := '1'; \\
\quad \text{else} \\
\quad \quad X := '0'; \\
\quad \text{end if}; \\
\text{end process;}
\end{cases}
\]

Complete specification of conditions to provide the full truth table

Multiplexers

With priority

\[
\text{process}(A, B, \ldots, C_1, \ldots) \begin{cases} 
\text{begin} \\
\quad \text{if } C_1 \text{ then} \\
\quad \quad X := A; \\
\quad \text{elsif } C_2 \text{ then} \\
\quad \quad X := B; \\
\quad \text{else} \\
\quad \quad \text{end if}; \\
\text{end process;}
\end{cases}
\]

Without priority

\[
\text{process}(A, B, \ldots, S) \begin{cases} 
\text{begin} \\
\quad \text{case } S \text{ is} \\
\quad \quad \text{when } C_1 => \\
\quad \quad \quad X := A; \\
\quad \quad \text{when } C_2 => \\
\quad \quad \quad X := B; \\
\quad \quad \text{else} \\
\quad \quad \quad \text{end if}; \\
\text{end process;}
\end{cases}
\]

Array Index

\[
X := A(\text{index});
\]
**Tri-states and Loops**

**Tri-State**

```vhdl
process (A, S)
begin
  if S = '1' then
    X <= A;
  else
    X <= 'Z';
  end if;
end process;
```

**Loop and Generate**

```vhdl
parité : process (A)
variable result : bit;
begin
  result := '0';
  for i in 0 to N-1 loop
    result := result xor A(i);
  end loop;
  X <= result;
end process;
```

**Example: data selection**

**Multiplexer Logic**

```vhdl
S <= A0 when sel=0 else
    A1 when sel=1 else
    'X' when sel=2;
```

**Tri-state Logic**

```vhdl
signal S, A0, A1, A2 : std_logic;
signal Sel_A0, Sel_A1, Sel_A2 : std_logic;
...
S <= A0 when sel_A0='1' else 'Z';
S <= A1 when sel_A1='1' else 'Z';
S <= A2 when sel_A2='1' else 'Z';
```
Example: Bidirectional Buffer

- Bidirectional buffer

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
entity transceive is
  port(
    A, B: inout std_logic_vector(7 downto 0);
    oeab, oeba: in std_logic);
end entity;
architecture RTL of transceive begin
  B <= A when oeab = '1' else "ZZZZZZZZ";
  A <= B when oeba = '1' else (others => 'Z');
end RTL;
```

Example: Signal Decoding

- Signal decoding

<table>
<thead>
<tr>
<th>count</th>
<th>0</th>
<th>1</th>
<th>54</th>
<th>55</th>
<th>56</th>
<th>154</th>
<th>155</th>
<th>254</th>
<th>255</th>
</tr>
</thead>
<tbody>
<tr>
<td>decode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Two coding styles

- Decoding of all possible values: Logic
- Decoding of state transitions: Latch
Example: Signal Decoding

Arithmetic Operators

- Arithmetic operations
  - $A \text{ op } B$;
    - Signed
      - integer range -128 to 127;
      - signed(7 downto 0);
    - Unsigned
      - integer range 0 to 255;
      - unsigned(7 downto 0);
- Division: divider is a power of 2
  - $S \leq A/2$;
**Parallel Statements and Assignments**

architecture A of E is begin
   S <= X + Y;
   Y <= Z + W;
end A;

- Dataflow specifications
- Order of equations has no influence
- Combinational logic

**V Logic Synthesis from VHDL**

1. Methods, Design Flow, and Tools
2. Register-Transfer Level (RTL) Models using VHDL
   1. VHDL for Logic Synthesis
   2. Combinational Logic
   3. Sequential Logic
      - Flip-flops, registers
      - FSM
      - Memory
3. RT and Logic Synthesis CAD Algorithms
   Structuring, Flattening, Mapping
Sequential Logic

• Edge-triggered flip-flop (FF) or registers
  – Sensitivity of the process is on the clock edge
  – Value ‘mem’ could be a vector of bit/integer

```vhdl
register: process(clock)
  if (clock'event and clock = '1') then
    reg <= input_val;
  end if;
end process;
```

• Latch
  – Sensitivity of the process is on a signal level

```vhdl
latch: process(enable, input_val)
  if enable = '1' then
    mem <= input_val;
  end if;
end process;
```

Sequential Logic

• Asynchronous or synchronous reset (or set)

**asynchronous**

```vhdl
process(clearb, clock)
  if clearb = '0' then
    reg <= 0;
  elsif (clock'event and clock = '1') then
    reg <= input_val;
  end if;
end process;
```

**synchronous**

```vhdl
process(clock)
  if (clock'event and clock = '1') then
    if clearb = '0' then
      reg <= 0;
    else
      reg <= input_val;
    end if;
  end if;
end process;
```
Variable or Signals?
Éléments de mémorisation

A signal assigned inside a [clk’event and clk=’1’] statement will always be a direct output of a flip-flop.

Example

Dessiner le schéma logique obtenu par synthèse de la description suivante :

```vhdl
process(clock)
variable int1 : std_logic ;
begin
  if (clock’event and clock=’1’) then
    int1 := A nand B ;
    int2 <= C nor D ;
    S <= int1 nand int2 ;
  end if ;
end process ;
```

Some more rules

Only one clock and only one edge

Do not touch the clock!

Synchronous register with load

Counting and Shifting

• Signal declared as
  — integer range 0 to N-1
  — signed/unsigned(n downto 0)
• Binary counter from 0 to N-1
  if (compteur = N-1) then compteur <= 0;
  else count <= count + 1;
  end if;
• N-bit shift register
  — regdec(N-1 downto 1) <= regdec(N-2 downto 0);
  — regdec(0) <= ‘0’;
  or
  — regdec <= regdec/2;
Example: 8-bit Register

- 8-bit register with bidirectional I/O and high-impedance output
- \( en = '1' \): read/write; \( en = '0' \): data = 'Z'
- \( en = '1' \) AND \( wb = '0' \): write
- \( en = '1' \) AND \( wb = '1' \): read

Example: Register Bank
Memory

- A memory is a 1D-array of words
- Type declaration
  ```
  type t_mem is array(0 to N-1) of std_logic_vector(nb_bits-1 downto 0);
  type t_mem is array(0 to N-1) of integer range 0 to 2**(nb_bits-1);
  ```
- Signal declaration
  ```
  signal mem: t_mem;
  ```
  - mem(i) is the $i^{th}$ element of the memory
- Read the memory
  ```
  data_out <= mem(i);
  ```
- Write to the memory
  ```
  if clk'event and clk='1 then
    mem(i) <= data_in;
  end if;
  ```

Finite State Machines

- Enumeration of states
  ```
  type states is (stat1, state2, state3, ...);
  ```
- State register declaration
  ```
  signal current_state: states;
  ```
- State register
  ```
  StateRegister: process(reset, clk)
  begin
    if reset='0' then
      current_state <= init_state;
    elsif (clk'event and clk='1') then
      current_state <= next_state;
    end if;
  end process;
  ```
Finite State Machines

- Decoding logic
  - combinational logic
    - remember to respect RTL coding rules
- Moore/Mealy style
  - for Mealy outputs assignments are inside the condition on inputs

\[
\text{moore:process}(\text{inputs}, \text{current\_state}) \begin{align*}
\text{begin} \\
\text{case current\_state is} \\
\text{when state1 } -> \\
\text{outputs } <= \ldots; \\
\text{if input1 } = \ldots \\
\text{then next\_state } <= \text{state2; } \\
\text{else next\_state } <= \text{state1; } \\
\text{end if;} \\
\text{when state2 } -> \\
\text{outputs } <= \ldots; \\
\text{if input2 } = \ldots \\
\text{then next\_state } <= \text{state5; } \\
\text{else next\_state } <= \text{state2; } \\
\text{end if;} \\
\ldots \\
\text{end case; } \\
\text{end process;} 
\end{align*}
\]

Exercice

Décrire la fonction "récepteur série asynchrone" de mots binaires de 4 bits

Le message binaire débute par un start bit (Din= 0) suivi des 4 bits d’information et se termine par 2 stop bits (Din= 1). Le circuit reconstitue le mot de 4 bits et le présentera après réception sur le bus de sortie Dout en activant le signal DR (Data Ready).

Définir les différentes phases du traitement, attente du start bit, réception des 4 bits, ... et décrire sous forme de machine d’états le contrôle.
V Logic Synthesis from VHDL

1. Methods, Design Flow, and Tools
2. Register-Transfer Level (RTL) Models using VHDL
   1. VHDL for Logic Synthesis
   2. Combinational Logic
   3. Sequential Logic
3. RT and Logic Synthesis CAD Algorithms
   Structuring, Flattening, Mapping
**Logic Synthesis Principle**

- **Logic Synthesis = Translation + Optimizations**
- **Translation**
  - VHDL source code is analyzed and transformed into a generic logic representation
- **Constrained optimizations**
  - Logic is mapped onto gates from the technological library and optimized under timing/power/area constraints

**Optimizations**

- **Structuring**: area oriented
  - Simplify and factorize Boolean equations
- **Flattening**: speed oriented
  - Distribute factors in Boolean equations
- **Mapping**
  - Selection of gates in the technological library
  - Specification of constraints (power/delay/load/area)
  - Generate several solutions
Optimizations

• Structuring: area oriented
  – Simplify and factorize Boolean equations
  – Share common factors

14 AND 2 NOT 4 OR
x = a.b.c + a.b.lc + a.d.e
y = a.b.c.d + a.b.c.ld
z = a.b + c.d

But delay is increased!

Simplify
x = a.b + a.d.e
y = a.b.c
z = a.b + c.d

Structuring
t = a.b
u = c.d
x = t + a.d.e
y = t.c
z = t + u

Optimizations

• Flattening: speed oriented
  – Distribute factors in Boolean equations
  – Reduces delay

4 AND 2 OR 1 NOT
x = a.b.t
y = a + t
z = b.c.lt

Delay is decreased!

Flattening
x = a.b.d + a.b.e
y = a + d + e
z = b.c.ld.le

7 AND 3 OR 2 NOT
Optimizations

- Mapping
  - Selection of gates in the technological library
  - Specification of constraints (power/delay/load/area)
  - Generate several solutions by graph covering
    - Evaluate delay/cost
    - Solution with lowest area respecting delay constraint is retained

<table>
<thead>
<tr>
<th>Gate</th>
<th>Delay</th>
<th>Cost</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>nand</td>
<td>2 ns</td>
<td>3</td>
<td>!(AB)</td>
</tr>
<tr>
<td>half adder</td>
<td>4 ns</td>
<td>14</td>
<td>A.B + A!B</td>
</tr>
</tbody>
</table>

Technological Library of Logic Gates
(many gates in ‘design kit’ library)

Resource Sharing

```plaintext
process (A,B,C,S)
begin
  if (S = '1') then
    X <= A + B;
  else
    X <= A + C;
  end if;
end process;

process (A,B,C,S)
begin
  variable opB : integer;
  if (S = '1') then
    opB := B;
  else
    opB := C;
  end if;
  X <= A + opB;
end process;
```
Pipeline

```vhdl
if (clk'event and clk = '1') then
  if (A + B) > (C + D) then
    S <= E;
  end if;
end if;
```

```vhdl
signal AB, CD: integer;

if (clk'event and clk = '1') then
  AB <= A + B;
  CD <= C + D;
  if AB > CD then
    S <= E;
  end if;
end if;
```