Programmable Logic: Introduction

- Digital electronics: a few reminders to basic ideas and concepts
  - Combinational logic
  - Sequential logic
  - Synchronous vs. Asynchronous designs
- Programmable logic devices (PLD)
  - Hardware overview
- Field-programmable gate arrays (FPGA)
  - Basics
  - Design flow

Course Material at [http://pcweb.physik.uni-giessen.de/fpgaprak/](http://pcweb.physik.uni-giessen.de/fpgaprak/)
Basic Logic Gates

AND: $C = A \cdot B$

OR: $C = A + B$

NOT: $C = A'$

EXCLUSIVE OR: $C = A \oplus B$
<table>
<thead>
<tr>
<th>Positive Logic</th>
<th>Negative Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV</td>
<td>INV</td>
</tr>
<tr>
<td>AND</td>
<td>OR</td>
</tr>
<tr>
<td>NAND</td>
<td>NOR</td>
</tr>
<tr>
<td>OR</td>
<td>AND</td>
</tr>
<tr>
<td>NOR</td>
<td>NAND</td>
</tr>
</tbody>
</table>

Example
Implementation of NAND and NOR gates is easier than that of AND and OR gates (e.g., CMOS)

NAND:

\[ A \quad \text{NAND} \quad B \equiv C \]

\[ C = (AB)' = A' + B' \]

NOR:

\[ A \quad \text{NOR} \quad B \equiv C \]

\[ C = (A+B)' = A'B' \]
Combinational Logic

- Has no memory =>
  present state depends only on the present input

\[ X = x_1 x_2 \ldots x_n \]

\[ Z = z_1 z_2 \ldots z_m \]

\[ Z(t) = F(X(t)) \]
Combinational-Circuit Building Blocks

- Multiplexers
- Decoders
- Encoders
- Code Converters
- Comparators
- Adders/Subtractors
- Multipliers
- Shifters
Example: Multiplexers: 2-to-1 Multiplexer

- Have number of data inputs, one or more select inputs, and one output
  - It passes the signal value on one of data inputs to the output

(a) Graphical symbol

(b) Truth table

<table>
<thead>
<tr>
<th>$s$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$w_0$</td>
</tr>
<tr>
<td>1</td>
<td>$w_1$</td>
</tr>
</tbody>
</table>

(c) Sum-of-products circuit

\[ f = s'w_0 + sw_1 \]
Example: Full Adder

Module

Truth table

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Cin</th>
<th>Cout</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Sum = $X'Y'Cin + X'YCin' + XY'Cin' + XYCin$

Cout = $X'YCin + XY'Cin + XYCin' + XYCin$
Sequential Circuits

- **Circuits with Feedback**
  - Outputs = \( f(\text{inputs, past inputs, past outputs}) \)
  - Basis for building "memory" into logic circuits
    - Door combination lock is an example of a sequential circuit
  - State => memory
    - State is can be "output" and "input" to combinational logic or to other sequential logic
Simplest Circuits with Feedback

- Two inverters form a static memory cell
  - Will hold value as long as it has power applied

- How to get a new value into the memory cell?
  - Selectively break feedback path
  - Load new value into cell
Clocks

- Used to keep time
  - Wait long enough for inputs to settle
  - Then allow to have effect on value stored
- Clocks are regular periodic signals
  - Period (time between ticks)
  - Duty-cycle (time clock is high between ticks - expressed as % of period)

![Diagram showing period and duty cycle with a 50% duty cycle example]
Edge-Triggered Flip-Flops

- Positive edge-triggered
  - Inputs sampled on rising edge; outputs change after rising edge
- Negative edge-triggered flip-flops
  - Inputs sampled on falling edge; outputs change after falling edge
Comparison of Latches and Flip-Flops

- **positive edge-triggered flip-flop**
- **transparent (level-sensitive) latch**

behavior is the same unless input changes while the clock is high
Timing Methodologies

- Rules for interconnecting components and clocks
  - Guarantee proper operation of system when strictly followed
- Approach depends on building blocks used for memory elements
  - Focus on systems with edge-triggered flip-flops
    - Found in programmable logic devices
- Basic rules for correct timing:
  - (1) Correct inputs, with respect to time, are provided to the flip-flops
  - (2) No flip-flop changes state more than once per clocking event
Definition of terms
- **clock**: periodic event, causes state of memory element to change; can be rising or falling edge, or high or low level
- **setup time**: minimum time before the clocking event by which the input must be stable (Tsu)
- **hold time**: minimum time after the clocking event until which the input must remain stable (Th)

There is a timing "window" around the clocking event during which the input must remain stable and unchanged in order to be recognized.
Typical Timing Specifications

- Positive edge-triggered D flip-flop
  - Setup and hold times
  - Minimum clock width
  - Propagation delays (low to high, high to low, max and typical)

All measurements are made from the clocking event that is, the rising edge of the clock.
Synchronous vs. Asynchronous Designs

- **Clocked synchronous circuits**
  - Inputs, state, and outputs sampled or changed in relation to a common reference signal (the clock)

- **Asynchronous circuits**
  - Inputs, state, and outputs sampled or changed independently of a common reference signal (glitches/hazards a major concern)
    - *Stay away from asynchronous designs!*

- **Asynchronous inputs to synchronous circuits**
  - Inputs can change at any time, will not meet setup/hold times
  - Dangerous, synchronous inputs are greatly preferred
  - Cannot be avoided (e.g., reset signal, memory wait, user input)
    - *Solution: synchronize with clock as early as possible!*
Overview: IC Technology

- In the early 80s:
  - Generic logic circuits (Example TTL: SN7400)
  - Complex applications assembled from basic building blocks: chips with few (< 10) hardwired logic functions
    - Many PCBs, interconnects, inflexibility, cost ...

- 90’s: VLSI Circuits + “glue logic”

- Now three types of IC technologies
  - Full-custom ASIC
  - Semi-custom ASIC (gate array and standard cell)
  - PLD (Programmable Logic Device)
NRE and unit cost metrics

- Unit cost
  - the monetary cost of manufacturing each copy of the system, excluding NRE cost
- NRE cost (Non-Recurring Engineering cost)
  - The one-time monetary cost of designing the system
- total cost = NRE cost + unit cost * # of units
- per-product cost = total cost / # of units
  = (NRE cost / # of units) + unit cost
General-purpose processors

- Programmable device used in a variety of applications
  - Also known as “microprocessor”
- Features
  - Program memory
  - General datapath with large register file and general ALU
- User benefits
  - Low time-to-market and NRE costs
  - High flexibility
- Example: Pentium, ARM, ...

```
Assembly code for:
total = 0
for i = 1 to ...
```
Application-specific processors

- Programmable processor optimized for a particular class of applications having common characteristics
- Features
  - Program memory
  - Optimized datapath
  - Special functional units
- Benefits
  - Some flexibility, good performance, size and power
- Example: DSP, Media Processor

```
total = 0
for i = 1 to ...
```
Single-purpose hardware

- Digital circuit designed to execute exactly one program
  - coprocessor, accelerator
- Features
  - Contains components needed to execute a single program
  - No program memory
- Benefits
  - Fast
  - Low power
  - Small size

![Diagram showing components of a single-purpose hardware system.](image-url)
Full-custom/VLSI

- All layers are optimized for an embedded system’s particular digital implementation
  - Placing transistors
  - Sizing transistors
  - Routing wires
- Benefits
  - Excellent performance, small size, low power
- Drawbacks
  - High NRE cost (e.g., $300k), long time-to-market
Semi-custom

- Lower layers are fully or partially built
  - Designers are left with routing of wires and maybe placing some blocks
- Benefits
  - Good performance, good size, less NRE cost than a full-custom implementation (perhaps $10k to $100k)
- Drawbacks
  - Still require weeks to months to develop
PLD (Programmable Logic Device)

- All layers already exist
  - Designers can purchase an IC
  - Connections on the IC are either created or destroyed to implement desired functionality
  - Field-Programmable Gate Array (FPGA) very popular

- Benefits
  - Low NRE costs, almost instant IC availability

- Drawbacks
  - Bigger, expensive (perhaps $30 per unit), power hungry, slower
## Comparison of different technologies

<table>
<thead>
<tr>
<th>Technology</th>
<th>Performance/Cost</th>
<th>Time until running</th>
<th>Time to high performance</th>
<th>Time to change code functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIC</td>
<td>Very High</td>
<td>Very Long</td>
<td>Very Long</td>
<td>Impossible</td>
</tr>
<tr>
<td>FPGA</td>
<td>Medium</td>
<td>Medium</td>
<td>Long</td>
<td>Medium</td>
</tr>
<tr>
<td>ASIP/DSP</td>
<td>High</td>
<td>Long</td>
<td>Long</td>
<td>Long</td>
</tr>
<tr>
<td>Generic</td>
<td>Low-Medium</td>
<td>Very Short</td>
<td>Not Attainable</td>
<td>Very Short</td>
</tr>
</tbody>
</table>

**Flexibility**

**Speed**
Roadmap for Programmable Logic

- PROM
- PLA
- PAL
- CPLD
- FPGA
Programmable Logic Device (PLD):
- An integrated circuit chip that can be configured by end use to implement different digital hardware
- Also known as “Field Programmable Logic Device (FPLD)"
PLD Advantages

- Short design time
- Less expensive at low volume
PLD Categorization

- PLD
  - SPLD
    - Simple PLD
  - PAL
    - Programmable Array Logic
  - CPLD
    - Complex PLD
  - FPGA
    - Field Programmable Gate Array
  - HCPLD
    - High Capacity PLD

- PLA
  - Programmable Logic Array
Programmable ROM (PROM)

- Address: N bits; Output word: M bits
- ROM contains $2^N$ words of M bits each
- The input bits decide the particular word that becomes available on output lines
Logic Diagram of 8x3 PROM
## Combinational Circuit Implementation using PROM

### Truth Table

<table>
<thead>
<tr>
<th>I0</th>
<th>I1</th>
<th>I2</th>
<th>F0</th>
<th>F1</th>
<th>F2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Diagram

- **3 X 8 Decoder**
- **Programmable OR array**
- Inputs
- Outputs: F0, F1, F2
PROM Types

- **Programmable PROM**
  - Break links through current pulses
  - Write once, Read multiple times
- **Erasable PROM (EPROM)**
  - Program with ultraviolet light
  - Write multiple times, Read multiple times
- **Electrically Erasable PROM (EEPROM)/ Flash Memory**
  - Program with electrical signal
  - Write multiple times, Read multiple times