Programmable Logic Device
설계 특성
2006. 4. 6.

이재홍

한밭대학교 정보통신컴퓨터공학부
발표 순서

1. PLD의 개요 및 구조
2. CPLD/FPGA의 구조
3. CPLD/FPGA 설계 및 검증방법
4. Embedded SW와 FPGA Design

질의 & 응답
ASIC vs PLD

- **Standard ICs**
  - General-purpose processors, memory chips and other standard components

- **ASIC (Application Specific IC)**
  - Semi-custom chip design
  - Dedicated to single function, or limited range of functions
  - A typical ASIC is a circuit, where functions are designed by the customer and layout and the fabrication is done by the silicon vendor
  - Personalized at the factory

- **PLD (Programmable Logic Device)**
  - Personalized at the client site
  - (EE)PROM, PLA, PAL, CPLD/FPGA
What's the PLDs?

- CAD S/W를 이용하여 설계한 회로를 다운로드하여 칩으로 즉시 제작 가능한 소자

- PLD의 구성 요소
  - Configurable Logic Block
  - Programmable Routing Switch Block
  - Configurable Memory Block
  - Input/Output Block

- PLD의 장점
  - Fast Prototyping
  - Re-Programmable
  - In-System Programmable
  - Easy Transfer of Design Results
    - HDL → Text File Format
  - Design Security in Chips
Classifications of PLDs

- SPLD (Simple PLD)
  - PLA (Programmable Logic Array)
    - PLA is a relatively small FPD that contains two levels of logic, an AND-plane and an OR-plane, where both levels are programmable.
  - PAL (Programmable Array Logic)
    - PAL is a relatively small FPD that has a programmable AND-plane followed by a fixed OR-plane.

- CPLD (Complex PLD)
  - A more complex PLD that consists of an arrangement of multiple SPLD-like blocks on a single chip.

- FPGA (Field-Programmable Gate Array)
  - FPGA is an FPD featuring a general structure that allows very high logic capacity.
Basic Configuration of PLDs(I)

(a) Programmable read-only memory (PROM)

(b) Programmable array logic (PAL) device

(c) Programmable logic array (PLA) device
Basic Configuration of PLDs(II)

PROM

PLA

PAL

Φ INDICATES PROGRAMMABLE CONNECTION
+ INDICATES FIXED CONNECTION
Logic Allocation: PLA

**PAL: Requires 4 pt’s!**

**PLA: Requires only 3 pt’s!**

---

**Can NOT share common logic**
- Indicates 'used' junction
- Indicates 'unused' junction
- Indicates 'fixed' junction

---

**Common logic may be shared in CoolRunner-II**

\[
X = A \& B + C \\
Y = A \& B + !C
\]
A CPLD comprises multiple PAL-like blocks on a single chip with programmable interconnect to connect the blocks.

CPLD Architecture
FPGA consists of an array of programmable basic logic cells surrounded by programmable interconnect.

FPGA Structure
FPGA Chip Architecture

[Diagram of FPGA architecture with various components labeled in Korean]

- I/O pad
- Program circuit
- Logic module
- Program circuit
- FPGA structure
- 5x5 routing switch matrix
- SRAM switch cell
- SRAM
CPLD/ FPGA Design Process

Design Flow

- Design Specification
- Define I/O
- Design Entry
- Functional Simulation
- Map, Place, Route
- Timing Simulation
- Download
- Debug
Mentor VHDL Simulator
Documentation for FPGA Design

- **Design Specification**
  - Functional Spec., I/O Signal Spec., Test plan

- **Define I/O**
  - 신호별 세부 기능에 관한 기술문서

- **Design Entry**
  - HDL(VHDL/VerilogHDL) design files
  - Schematic diagram

- **Functional Simulation**
  - Test vector에 대한 기술 문서
  - Simulation waveform 및 분석 결과 문서

- **Map, Place, Route**
  - 중간 출력 파일
    - User constraints file

- **Timing Simulation**
  - Test vector에 대한 기술 문서
  - Timing simulation 분석 결과 문서
  - Report file 문서

- **Download**
  - JTAG file

- **Board-level Test**
  - 보드레벨 검증문서
CPLD/FPGA 설계 문서화

- CPLD/FPGA 설계 및 검증에 관한 문서화 내용
  - CPLD/FPGA 설계 방법 및 설계 과정
    - 설계환경 (HW, SW), 구체적인 설계 과정
  - CPLD/FPGA 디바이스 및 내부구조
    - 디바이스선택의 타당성 검토 문서
  - CPLD/FPGA 설계에 대한 검증 방법론
  - 설계에 대한 Functional/Timing Simulation
    - Simulation Waveform 및 결과 분석서
  - 설계 단계별 중간 생성 자료 (파일) 분석서
    - User Constraints Files
  - 검증에 필요한 테스트 시나리오 및 테스트 벡터 집합
  - 보드레벨 검증 문서
    - 검증 환경, 실측 입력출력 신호파형, 검증결과 분석서
ASIC/FPGA 구현을 위한 CAD Tool 흐름도

- **RTL 회로 설계**
  - 회로도 추출 및 효율적 VHDL 표현
  - Functional Simulation에 의한 회로 검증

- **Gate Level 회로 설계**
  - 면적, 속도, Power를 고려한 Design Constraints 설정
  - Design Library의 설정
    - FPGA, ASIC
  - Component Delay를 고려한 기능 및 Timing Simulation에 의한 회로 검증
  - Physical Design을 위한 EDIF 파일 생성
ASIC/FPGA 구현을 위한 CAD Tool 흐름도

- **FPGA를 이용한 Physical Design**
  - FPGA Tool에 의한 Compile 수행
  - 사용할 Device 및 Option의 설정
  - Layout Data를 이용한 Simulation
  - Target Board의 구성과 Emulation을 통한 검증
  - ASIC으로의 변환 여부에 따른 대응

- **ASIC 설계를 위한 Physical Design**
  - Design House와 연계 및 EDIF 전달
  - Wire Delay를 함께 고려한 Simulation
  - Sign-off와 ASIC에 대한 기다림
  - Test Board 구성, 검증 및 평가
Logic Synthesis (논리합성)

- Synthesis = Translation + Optimization + Mapping

```verilog
module fa(sum, c_out, a, b);
    output sum, c_out;
    input c_in, a, b;

    wire S0, S1, S2;

    xor(S0, a, b);
    and(S1, a, b);
    xor(SUM, S0, c_in);

    // Additional code...
```

- Translate into Boolean Representation
- Optimize + Map

<table>
<thead>
<tr>
<th>HDL Source code</th>
<th>Generic Boolean (GTECH)</th>
<th>Target Technology</th>
</tr>
</thead>
</table>
| module fa(sum, c_out, a, b);  
    output sum, c_out;  
    input c_in, a, b;  

    wire S0, S1, S2;  

    xor(S0, a, b);  
    and(S1, a, b);  
    xor(SUM, S0, c_in);  

    // Additional code... | | |
Why HDL?

- HDL is for writing models of a system
  - Growing complexity of designs

- Reasons for modeling
  - Requirements specification
  - Documentation
  - Testing using simulation
  - Formal verification
  - Synthesis

- Goal
  - Most reliable design process, with minimum cost and time
  - Avoid design errors before fabrication
Two Major HDLs

- Verilog and VHDL

- Both are programming languages
  - Text-based, easier to create a design over schematic entry/capture

- Verilog is Similar to C
- VHDL is closer to ADA

- Differences to regular programming languages
  - Regular programming is inherently sequential
  - HDLs have concurrency (parallel execution) and timing
  - HDLs have constructs to describe hardware
## GPL vs HDL

<table>
<thead>
<tr>
<th>Type of Input</th>
<th>General Programming Language</th>
<th>Hardware Description Language</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Text file (.c, .cpp, .java, .cs)</td>
<td>Text file (.vhd, .v, .sv)</td>
</tr>
<tr>
<td>Tools</td>
<td>Compiler, Debugger</td>
<td>Synthesizer(Silicon Compiler), Emulator, Logic analyzer</td>
</tr>
<tr>
<td>Target (Output)</td>
<td>Executable code (Binary)</td>
<td>Bit-Stream (Binary)</td>
</tr>
<tr>
<td>Execution</td>
<td>Sequential execution</td>
<td>Parallel execution</td>
</tr>
<tr>
<td>Reviewed material</td>
<td>Source code</td>
<td>Source code, Waveform</td>
</tr>
<tr>
<td>Verification</td>
<td>Debugging</td>
<td>Simulation, Emulation, Probing, Co-Verification</td>
</tr>
<tr>
<td>methodology</td>
<td>No consideration for HW</td>
<td>Controlled by Clock, Always think synthesized HW</td>
</tr>
</tbody>
</table>
Embedded SW vs FPGA Design Flow

**Standard Embedded SW Development Flow**
- C Code
- Compiler/Linker
- (Simulator)
- Object Code

**Standard FPGA HW Development Flow**
- VHDL/Verilog
- Synthesizer
- Simulator
- Place & Route

- Download to FPGA
- Debugger

Data2BlockRAM ➔ Bitstream ➔ Download to FPGA

MicroBlaze code in off-chip memory

MicroBlaze code in Spartan-3 on-chip memory

Download to Board & FPGA

OPB GPIO
OPB Arbiter
OPB UART

Embedded Development Kit
Hardware and Software Co-Design

- **Approach to develop Traditional Systems**
  - Generic Hardware
  - Software with Relaxed Resource
  - HW and SW are separated designed.

- **Approach to develop Embedded Systems**
  - SOFTWARE which is planned to run with resource constraints.
  - HARDWARE which is optimized and specialized to run the software

- HW and SW are co-designed.
  - Unified Design Process
디지털 시스템 설계

- TTL-based System design
  - 여러 종류의 TTL IC를 이용한 시스템 보드 설계
- System design with FPGA/CPLD
  - TTL IC들을 하나의 FPGA로 집적화 ⇒ 면적, 성능향상
- SoC (System On Chip) design
  - 프로세서가 포함된 시스템보드를 하나의 칩으로 구현
  - SoC 설계를 위한 HW/SW Codesign, Cosimulation 및 Coverification 개념 도입
SoC (System On Chip) Design

Core Platform

Memory Options

Peripheral Options

Design-Driven Auto-Generated Verification Environments

HW/SW Verification

SW Programmers Model

Emulation

Live Target

Others ...

CPU Core(s)

DSP Core(s)

USB & RS232

Multimedia Decryption Engine

RTOS Kernel

Drivers

Bluetooth

USB

MPEG4

MPEG4 Decoder

Core Apps

System

User Designed

RTOS & Application Options
질의 & 응답