

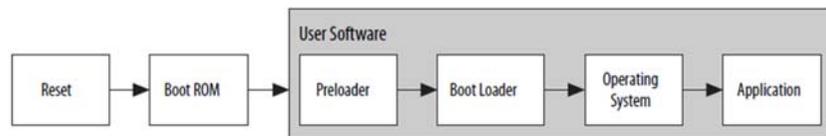
## 교차 개발 환경

## 교차 개발 환경

- 임베디드 시스템 플랫폼
  - 커널을 올리기 전엔 target 시스템에 아무 프로그램도 올라가 있지 않음
  - target 시스템은 일반적으로 메모리 용량이 적어서 compiler를 포함한 프로그램 개발환경을 올리기가 어려움
- 교차개발환경
  - target 시스템용 프로그램은 대개 별도의 host시스템에서 개발됨
  - host와 target에 사용되는 processor가 다른 경우, 컴파일러의 실행은 host에서 되지만 실행 코드는 target system에서 실행됨
  - cross compiler가 필요함
    - 다른 target system용 실행파일을 생성하는 컴파일러 (intel-FPGA monitor program에서 사용)

2

## Typical ARM Cortex-A9 Boot Sequence



- Boot ROM
  - Hard coded by Altera(Intel)
  - Determines the boot source by reading the boot select pins
- Preloader
  - In (1) **Flash/SD Card** or (2) **FPGA**
  - Typically initializes the DDR3 SDRAM and HPS I/O pins
- Boot loader
  - Loads and starts the operating system

3

## Boot Sequence with Altera Monitor Program

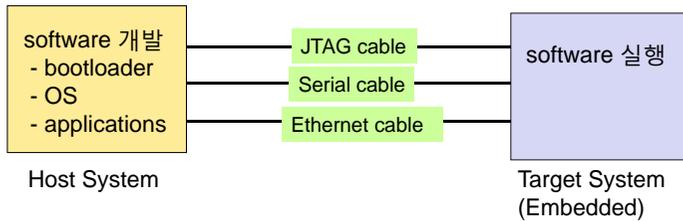
- Resets the HPS
- Halts the code in the Boot ROM
- Loads a **preloader** into the HPS on-chip RAM
  - **Preloader** comes pre-compiled with the Monitor Program
  - Initializes the DDR3 SDRAM and HPS I/O pins
  - Enables the HPS-FPGA bridges
  - Disables the watchdog timers
  - Re-maps the DDR3 SDRAM to address 0x00000000
- Loads the **user's code** into the DDR3 SDRAM instead of OS
  - Pauses the processor at the first instruction

4

## 시스템 개발 환경 구성

- **JTAG cable:** flash메모리 저장용
- **Serial cable:** terminal용
- **Ethernet cable:** network용 (tftp, NFS)

직접 연결: cross cable  
허브에 연결: direct cable



5

## 부트로더 설치

### ■ 부트로더(Bootloader) 설치 방법

- 부트로더가 설치되는 디바이스에 따라서 설치 방법이 다름

### ■ ROM

- EPROM 칩을 ROM 전용 writer(programmer)를 이용하여 프로그래밍 한 후에 시스템의 ROM용 소켓에 장착함
- 예전에 사용되던 방식



6

## 부트로더 설치(2)

### ■ 플래시 메모리 (보드에 영구히 장착된 것)

- 기존에 설치된 부트로더가 있는 경우
  - target의 부트로더에서 host로 부터 new 부트로더를 다운로드
  - 부트로더의 플래시 메모리 프로그래밍 기능을 이용하여 프로그래밍

프로그래밍(programming) = ROM이나 플래시 메모리에 내용을 기록하는 것

- 기존에 설치된 부트로더가 없는 경우 (최초 또는 부트로더가 망가짐)
  - **JTAG, BDM, ICE**(in circuit emulator)장비 이용

### ■ 플래시 메모리 (탈부착 가능한 것) – SD, microSD, CF card 등

- 호스트 시스템에서 부트로더 등의 image를 플래시 메모리에 기록한 후에 target 시스템에 장착하여 사용 (ex) DE1-SoC
- 영구 장착된 것에 비해서 속도 제약이 있음

7

## JTAG

### ■ JTAG 이란

- PCB와 IC를 테스트 하기 위한 목적으로 1985년 조직된 JTAG(Joint Test Action Group)에 의해 제정된 표준이다.
- Boundary Scan Test 이용하여 외부의 핀을 조사하거나 제어할 수 있도록 함
- 전체적인 인터페이스는 5개의 핀에 의해서 제어.
  - TDI, TMS, TCK, nTRST, TDO

### ■ JTAG의 기능

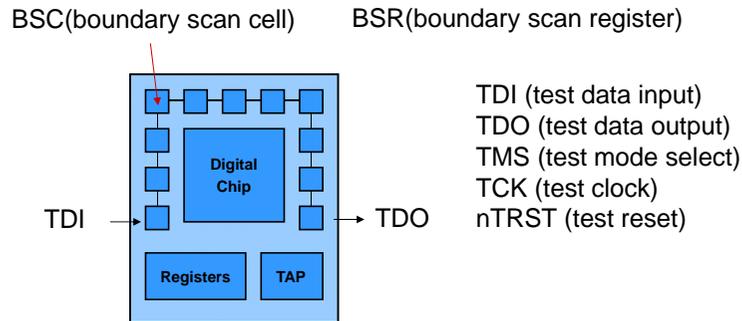
- 프로세서(CPU)의 상태와는 상관없이 디바이스의 모든 외부 핀을 구동 시키거나 값을 읽어 들일 수 있는 기능을 제공
- 회로의 배선과 소자의 전기적 연결상태 test
- 디바이스간의 연결상태 test
- **Flash memory programming(fusing)**

8

## JTAG – Boundary Scan Test Interface

### Boundary Scan Test Interface

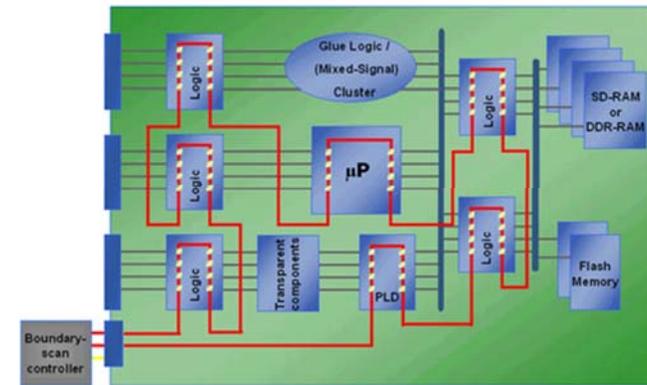
- 디바이스 내에서 모든 외부와의 연결점을 가로챈. (즉 외부로 나가는 각각의 핀들과 일대일로 연결)
- 각각의 셀(BSC)은 시리얼 쉬프트 레지스터(BSR)를 형성하기 위해서 서로 연결



9

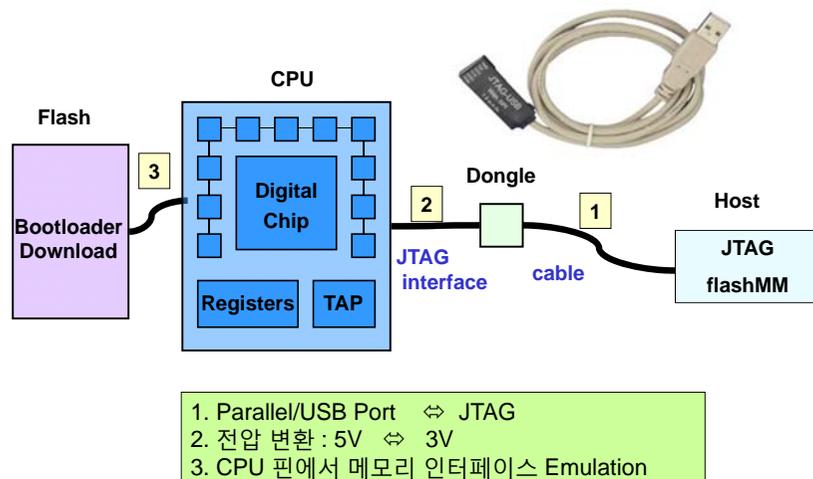
## JTAG – PCB boundary scan test

### PCB boundary scan test



10

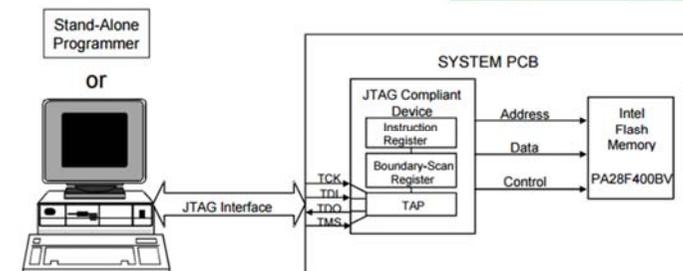
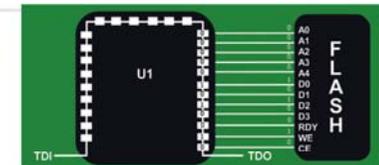
## JTAG을 이용한 Flash Memory fusing



11

## JTAG Fusing 프로그램

- JTAG compliant device를 통하여 Flash memory의 fusing 수행



- bootloader 프로그램을 최초로 flash 메모리에 저장할 때 사용
- 그 이후에는 bootloader의 명령어를 사용하여 flash 내용을 변경함

12

## Tool Chain

- Tool chain이란?
  - software의 개발에 사용되는 프로그래밍 도구의 집합
  - 대개 target 시스템의 software의 개발에 사용되는 host 시스템의 cross compile 환경을 말함
- Tool chain의 구성:
  - source code 을 compile하고 build하여 binary 실행 파일을 생성하는데 필요한 각종 Utility 및 Library들로 구성됨
- GNU Tool Chain
  - GNU GCC (compiler collection)
    - a set of programming language compilers: gcc, g++
  - GNU binary utilities: binutils
    - assembler, loader, 기타 tools
  - GNU C library: glibc
    - cross compiler 구축을 위한 library 및 일반 library

## GNU Toolchain 명령어

- gcc, g++ – C,C++ compiler
- as – portable GNU assembler
- ld – GNU linker
- ar – create, modify, extract from archives
- ranlib – generate index to archive (= ar -s)
- nm – list symbols from object files
- strip – discard symbols from object files
- objdump – display information from object files
- objcopy - copy and translate object files
- size – list section sizes and total size
- strings – display the sequences of printable characters
- readelf – display information about ELF files

## ld - GNU linker

- ld - GNU linker
  - combines a number of object and archive files, relocates their data and ties up symbol references.
  - Usually the last step in compiling a program is to run **ld**  
  
# ld -o *outfile* /usr/lib/crt1.o hello.o -lc
  - linker script 파일을 지정하면 default linker script를 대신하여 사용  
-T *scriptfile*
  - link map 파일을 출력함  
-Map *mapfile*

## objdump

- displays information about one or more object files
  - 일반적인 dump  
# objdump -s *objfile*
  - ELF binary file의 특정 section dump  
# objdump -s -j .interp *objfile*
  - binary로 dump  
# objdump -s -b binary *objfile*
  - **disassembly**  
# objdump -d *objfile* # objdump -S *objfile* (소스코드포함)
  - file header 출력  
# objdump -f *objfile*
  - -b, -m으로 지정 가능한 모든 object format 출력  
# objdump -i *objfile*

## objcopy

- copy and translate object files
- It can write the destination object file in a format different from that of the source object file

```
# objcopy hello hello.new          원본과 같음

# objcopy -O binary hello hello.new  ELF 헤더가 없는 binary
                                     bootloader 생성에 사용

# objcopy -S hello hello.new        strip(심볼,재배치정보삭제)
# objcopy -R .note -R .comment hello hello.new
                                     .note와 .comment 섹션 제거
```

교차개발환경

17

## ABI와 EABI

- ABI (Application Binary Interface)
    - 응용 프로그램과 운영체제 또는 라이브러리의 표준화된 인터페이스 제공
    - 실행 파일 형식, 자료형, 레지스터 사용방법, 스택 사용방법, 변수 사용방법 등이 정의됨
    - 서로 다른 컴파일러가 만든 object 파일을 연결하여 사용할 수 있게 함
  - EABI (Embedded ABI)
    - ARM이나 PPC와 같은 임베디드 시스템에 적합하도록 개정된 ABI 표준
    - EABI를 지원하기 위해서는 cross-compile을 할 때에 EABI 표준을 지원하는 교차 컴파일러를 사용해야 함
- (예) intelFPGA Monitor Program에서 사용하는 교차개발환경
- 위치 - C:\intelFPGA\16.1\University\_Program\Monitor\_Program\arm\_tools\baremetal\bin

```
arm-altera-eabi-g++.exe
arm-altera-eabi-gcc.exe
```

18

## 사용 예

- Altera monitor program에서의 cross compile (assembly)

```
$ rm-altera-eabi-as -mfloat-abi=soft -march=armv7-a -mcpu=cortex-a9 --
gstabs -I "$GNU_ARM_TOOL_ROOTDIR/arm-altera-eabi/include"
... part1.s" -o part1.s.o

$ arm-altera-eabi-ld -e _start -u _start --defsym arm_program_mem=0x40
--defsym arm_available_mem_size=0x3ffffc0 --section-start .vectors=0x0
-T"C:/intelFPGA/16.1/University_Program/Monitor_Program/build/
altera-socfpga-unhosted-as.ld" -g
-o "C:/prog/altera_monitor/Exercise5/part1/part1.axf" ...

$ arm-altera-eabi-objcopy -O srec part1.axf part1.srec
// 실행파일을 S-record 형식으로 변환
```

19

- Altera monitor program에서의 cross compile (C program)

```
$ arm-altera-eabi-gcc -g -O1 -mfloat-abi=soft -march=armv7-a
-mtune=cortex-a9 -mcpu=cortex-a9 -Wall -Wl,--defsym
-Wl,arm_program_mem=0x40 -Wl,--defsym
-Wl,arm_available_mem_size=0x3ffffbc -Wl,--defsym -
Wl,__cs3_stack=0x3fffffc -Wl,--section-start -Wl,.vectors=0x0
-T"C:/intelFPGA/16.1/University_Program/Monitor_Program/build/altera-
socfpga-hosted-with-vectors.ld"
-o "C:/prog/altera_monitor/Exercise7/part1/part1.axf"

▪ -Wl,option      linker용 옵션 지정
▪ -T script       linker script 파일 지정
```

20