Xilinx Virtex-Ⅱ Pro를 이용한 SoC 설계(Ⅰ)

Virtex-II Pro PPC405의 디자인 플로 개관

이 글은 Virtex-II Pro의 PPC405를 처음 쓰는 엔지니어를 대상으로 EDK(Embedded Development Kit) 소프트웨어와 PPC405의 일반적인 사용법에 대해 설명한다. 이 글을 읽는 독자들은 C 언어에 대한 기본 지식이 필요하며 FPGA에 대한 설계 경험이 어느 정도 있어야 한다.

글: 김 혁/Insight Korea Xilinx FAE 과장
hyukkim@memec-korea.com

자일링스(Xilinx)는 지난해 Virtex-Ⅱ Pro™ FPGA를 출시했다. 이 제품은 FPGA 내부에 PPC405 코어와 MGT (Multi-gigabit transceiver)를 하드웨어 IP로 내장하고 있다. FPGA 내부에 CPU가 내장되어 있다는 것은 기존에 익숙해 있던 디자인 플로와는 다른 하드웨어와 소프트웨어 디자인 플로를 요구한다. 이제는 소프트웨어와 하드웨어를 한 개의 칩에 구현해야 하기 때문에 소프트웨어 엔지니어와 하드웨어 엔지니어가 서로 다른 방에서 근무하기가 어려워질지도 모르겠다.

EDK는 Virtex-Ⅱ Pro의 PPC405와 FPGA 로직에 관련된 모든 하드웨어와 소프트웨어 관련 작업을 해주는 통합 환경으로 synthesis, implementation, simulation과 같은 하드웨어 관련 작업과 compile, linking, debugging과 같은 소프트웨어 작업을 통합된 GUI 환경에서 수행할 수 있다.

디자인 플로

일반적으로 프로세서를 사용하는 시스템을 설계할 때 해야 할 많은 작업이 있지만, 그중 초반부에 하는 것은 ROM, RAM 및 퍼리퍼럴(Peripheral)의 메모리 맵을 정하는 것이다. 만일, OS를 사용하지 않는 시스템이라면 각 퍼리퍼럴을 제어하기 위한 디바이스 드라이버 프로그램을 직접 만들어줘야 하며 컴파일된 코드가 적절한 메모리 위치에 있도록 하기 위해서 Linker Script로 만들어줘야 한다.

그리고 나면 RAM, ROM 및 퍼리퍼럴들이 디코딩 될 수 있도록 PLD를 프로그램하고 PCB 입고 후 조립절차를 걸쳐 소프트웨어를 CPU에서 구동시켜서 제대로 동작하는지 검사를 해야 한다. 따라서 한번 검증된 하드웨어는 설계 스펙이 크게 바뀌지 않는 한 계속 그 설계를 유지하는 것이 일반적인 경향이다.

예를 들어 CPU의 RESET 회로, Memory Map, PLD 프로그램 소스, 전원회로 등은 한번만 검증되면 계속 사용하는 경향이 있다.

Virtex-II Pro™ PPC405의 디자인 플로 개관

Virtex-Ⅱ Pro의 PPC405를 사용할 경우, 시스템의 모든 하드웨어는 mhs(Microprocessor Hardware Specification) 파일이라는 텍스트 파일에 정의한다. 일반적으로 Virtex-Ⅱ Pro의 PPC405를 사용할 경우 이 파일의 이름은 system.mhs로 정하고, 이 파일에는 Virtex-Ⅱ Pro의 PPC405 자체와 Virtex-Ⅱ Pro의 PPC405에 연결하는 각종 퍼리퍼럴, 버스 구조, 메모리 맵이 정해진다.

EDK의 핵심 프로그램은 XPS(Xilinx Platform Studio)

이다. XPS는 system.mhs 파일을 읽어 Virtex-Ⅱ Pro의 PPC405의 버스를 구성하고 어떤 퍼리퍼럴이 사용됐는지 판단해 해당 퍼리퍼럴의 netlist와 mhs 파일에 있는 메모리 맵을 근거로 decoding 로직을 자동으로 만들어준다. 그리고 퍼리퍼럴을 액세스하기 위한 header file, c source file을 자동으로 만들어 주기 때문에 개발자가 일부러 각 퍼리퍼럴을 사용하기 위한 디바이스 프로그램을 만들 필요가 없다. 각 함수의 return type과 argument type은 software reference guide를 보면 보다 자세히 알 수 있다.

MHS 파일이란 무엇인가?

MHS 파일은 Virtex-Ⅱ Pro의 PPC405나 Microblaze를 사용하는 시스템에 꼭 필요한 텍스트 파일이다. 이 파일에는 시스템의 메모리 맵, 각 퍼리퍼럴의 스펙 및 버스 구조와 같은 하드웨어를 정의한다.

버스나 퍼리퍼럴은 각각 자기만의 파라미터가 있는데, 이 파라미터 값을 시스템에 맞게 정해주게 된다. 이 파라미터들은 특정 값으로 정하지 않거나 빠진 경우에는 디폴트 값으로 정의되며 버스와 퍼리퍼럴의 파라미터 종류와 파라미터 디폴트 값은 MPD(Microprocessor Peripheral Definition)라는 텍스트 파일에 있다.

이 중 자주 사용되는 퍼리퍼럴을 정의한 system.mhs 파일을 가지고 개발자가 필요한 퍼리퍼럴만 system.mhs에 포함시키면 쉽게 시스템을 설계할 수 있다(**리스트 1**).

디폴트 하드웨어 연결

그림 1은 여기서 설계하려는 시스템의 블록 다이어그램이다. 이 시스템은 Virtex-Ⅱ pro와 DIP Switch, 4개의 LED, UART 1 Channel 그리고 4M의 SDRAM으로 구성되어 있다. 따라서 FPGA 내부에는 UART 컨트롤러, LED 드라이브 GPIO, DIP sw read GPIO와 SDRAM 컨트롤러가 필요하다. 그리고 PPC405 CPU와 프로그램 및 데이터가 저장될 블록 메모리(Block Memory), PLB(Processor Local Bus), 64-bit PLB 메모리 컨트롤러, PLB2OPB 브리지, OPB(OnChip Peripheral Bus), JTAG 블록, 리셋 블록이 있어야 한다.

PLB는 프로세스 로컬 버스로 64bit 데이터 버스 크기를 가지며 OPB는 32bit 데이터 버스 크기를 가지고 있다.

```
BEGIN opb_gpio
PARAMETER INSTANCE = mygpio1
PARAMETER HW_VER = 1.00.a
PARAMETER C_GPIO_WIDTH = 4
PARAMETER C_ALL_INPUTS = 0
PARAMETER C_BASEADDR = 0xF0000300
PARAMETER C_HIGHADDR = 0xF00003ff
PORT OPB_Rst = peripheral_rst
PORT GPIO_IO = leds
BUS_INTERFACE SOPB = myopb
END

BEGIN opb_uartlite
PARAMETER INSTANCE = myuart
PARAMETER HW_VER = 1.00.b
PARAMETER C_DATA_BITS = 8
PARAMETER C_CLK_FREQ = 100000000
PARAMETER C_BAUDRATE = 19200
PARAMETER C_USE_PARITY = 0
PARAMETER C_BASEADDR = 0xf0000100
PARAMETER C_HIGHADDR = 0xf00001ff
PORT OPB_Rst = peripheral_rst
PORT RX = rx
PORT TX = tx
BUS_INTERFACE SOPB = myopb
END
```

리스트 1.system.mhs의 일부분

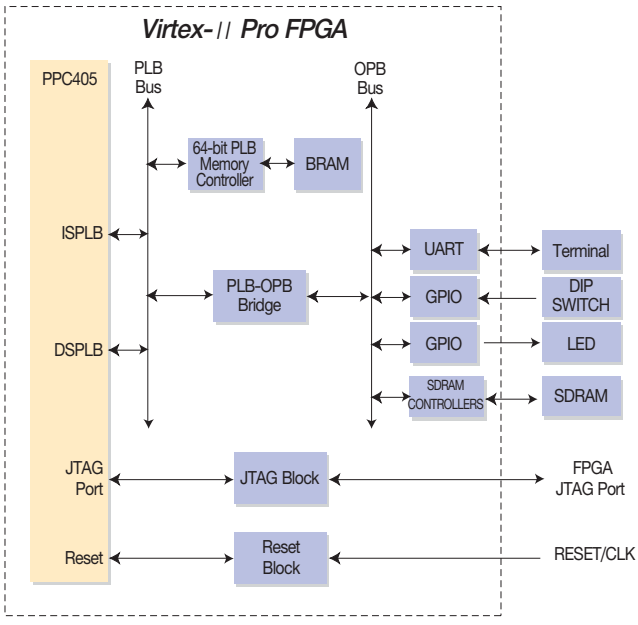

그림 1. 시스템 블록 다이어그램

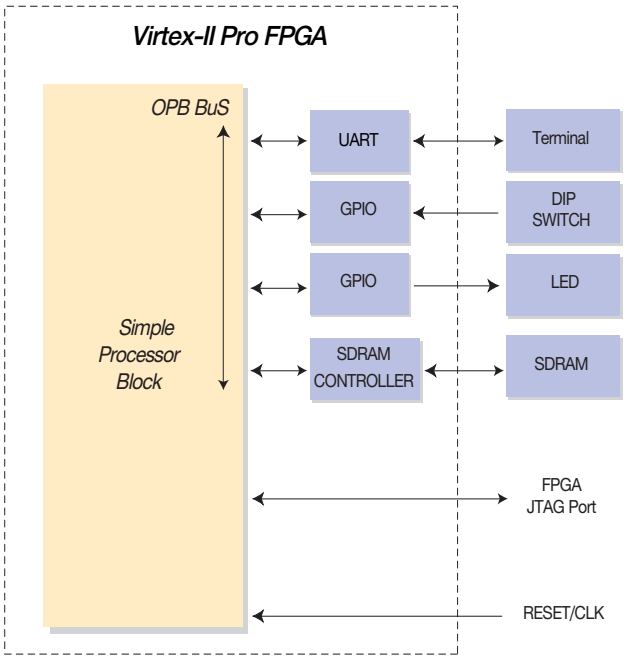

그림 2. 단순한 시스템 블록 다이어그램

CPU에 필요한 프로그램과 데이터는 64-bit PLB 메모리 컨트롤러와 연결된 BRAM(16Kbytes)에 저장되어 있다. 따라서 프로그램과 데이터를 저장하기 위한 ROM이나 플래시 메모리가 시스템에는 존재하지 않는다.

PPC405는 리셋 벡터 값이 0xffff_fffc로 정해져 있다. 전원이 인가되고 리셋이 릴리스되면 PPC405는 프로그램 카운터를 리셋 벡터 값으로 초기화한다. 따라서, user code나 boot code를 FPGA 내부에 있는 블록 RAM(BRAM)에 넣기 위해서는 BRAM이 0xffff_c000~0xffff_ffff에서 디코딩되도록 한다. 나머지 UART, GPIO, SDRAM 컨트롤러는 PLB보다는 응답속도가 느린 퍼리퍼럴을 연결하는 버스로 반드시 PLB2OPB 브리지를 통해서만 액세스할 수 있다.

JTAG 블록은 FPGA의 JTAG 체인과 PPC405의 JTAG 체인을 연결하는 회로이며 소프트웨어 디버깅할 때 디버깅 프로그램과 PPC405를 연결한다. 리셋 블록에서는 3가지 리셋 신호가 출력되는데, 용도에 따라 PPC405만 리셋하기 위한 core_reset_req, FPGA를 리셋하기 위한 chp_reset_req, 그리고 FPGA와 외부에 있는 모든 칩을 리셋하기 위한 system_reset_req가 있다.

PPC405는 Havard 아키텍처를 사용하기 때문에 데이터를 패치하는 부분과 명령어를 패치하는 하드웨어가 따로 존재한다. 따라서 PLB와 연결된 PPC405를 보면 ISPLB와 DSPLB가 있는데, ISPLB는 Instruction Side PLB이고, DSPLB는 Data Side PLB를 의미한다.

시스템을 설계하다보면 특정 CPU를 계속 사용할 경우 RESET 회로, POWER 회로, CLK 회로, Memory 인터페이스 회로 등은 계속 사용하는 경우가 많다. 이럴 경우 시스템에 다른 퍼리퍼럴을 추가하거나 변경할 때는 이런 회로는 그대로 사용하고 나머지 자기가 관심있는 퍼리퍼럴을 수정, 보완하는 경우가 많다. Virtex-Ⅱ Pro의 PPC405를 가지고 시스템을 설계할 때도 이런 개념이 적용된다.

Simple Processor Block

Virtex-Ⅱ Pro의 PPC405를 사용하는 대부분의 시스템에서 PPC405, PLB, OPB, PLB2OPB, JTAG BLOCK, RESET BLOCK 등은 이미 정해진 것을 변경없이 사용하는 경우가 많이 있다. 이렇게 미리 정해진 하드웨어에 UART, GPIO, SDRAM 컨트롤러 등을 추가해 시스템을 설계하면 **그림 2**와 같이 단순한 시스템을 설계할 수 있다.

Simple Processor Block의 프로그램과 데이터를 합한 크기는 16KByte 이내가 되어야 하고, 이 크기는 최대 64KByte까지 확장할 수 있다. 이 메모리 맵 상에서 0xffff_c000~0xffff_ffff에 위치한다. 디버깅 프로그램이 프로그램을 디버깅할 수 있는 JTAG 포트를 제공하고 있으며 PPC405만 리셋할 수 있는 포트를 가지고 있다. OPB를 연결하는 퍼리퍼럴의 숫자나 종류는 상관이 없으며 FPGA의 로직 한도 내에서는 계속 OPB에 연결할 수 있다.

XPS란 무엇인가?

XPS(Xilinx Platform Studio)는 **그림 3**과 같이 system.mhs 파일을 읽어서 Virtex-II Pro의 PPC405와 Bus 로직, 퍼리퍼럴을 연결하고 implementation하는 HWPlatGen과 Simulation Model을 만들어 주는 SimPlatgen, 그리고 개발자의 프로그램을 컴파일, 디버깅하는 프로그램이 내장되어 있는 통합 환경이다.

그림 4는 XPS를 사용하는 모습으로 왼쪽에 있는 창에서

는 system.mhs 파일에 정의된 하드웨어를 표시하고 있으며, 오른쪽 창은 system.mhs 파일을 오픈시켜 놓은 모습을 보여준다.

UART 구현

UART를 구현하기 위한 스펙으로는 **그림 5**와 같이 어떤 버스에 연결할 것인지 결정하고 메모리 맵에 맞춰 base address와 high address를 결정한다. 그리고 입력 클록 주파수, BuadRate와 Parity Bit, Stop Bit, Data Bit를 정의해야 한다.

Base Address도 메모리 맵 상에 나와 있는 것처럼 0xF000_0100로 하고 High Address는 0xF000_01FF로 정의한다. 참고로, 한번 정해진 UART는 FPGA 로직으로 구현되기 때문에 프로그램이 실행되는 도중에 BuadRate와 같은 하드웨어 스펙을 바꿀 수는 없다.

```
PARAMETER C_BASEADDR = 0xf0000100
PARAMETER C_HIGHADDR = 0xf00001ff
```

UART로 공급되는 CLK은 시스템에서 사용되는 클록의 스피드에 의해 결정되는데, 현재는 100MHz로 FPGA에 공급되고 있다. 이 값을 100MHz로 정해준다.

```
PARAMETER C_CLK_FREQ = 100000000
```

UART가 통신을 하기 위해서는 Tx, Rx 신호가 UART로부터 나와야 하며, 이 신호는 그대로 FPGA의 핀으로 연결되어야 한다. 따라서 UART의 Tx, Rx를 FPGA의 핀과 연결시켜 주어야 한다. BEGIN과 END 사이에 있는 PORT를 local port라고 하고 BEGIN, END 밖에 있는 PORT를 global port라고 한다. 이 글로벌 핀은 FPGA의 IO 핀이다.

따라서 UCF(User constraint file)에서는 rx, tx 핀이 FPGA의 몇 번 핀에 할당되어 있는지 반드시 정의되어 있어야 한다. 만일 정의되지 않은 경우라면 implementation 과정에서 임의의 핀이 할당되게 된다.

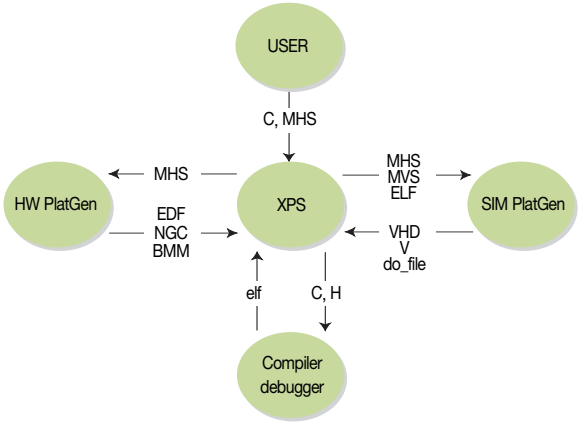

그림 3. XPS 플로

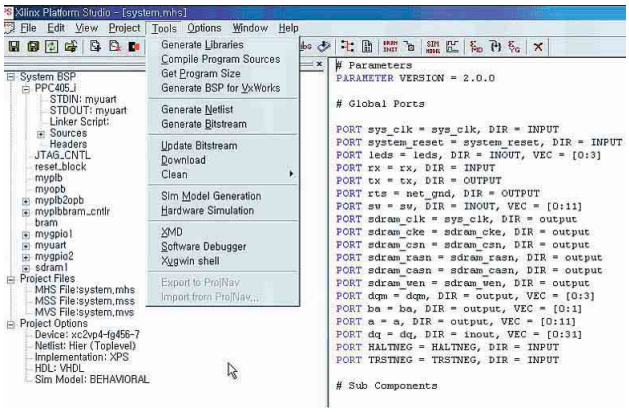

그림 4. XPS Program

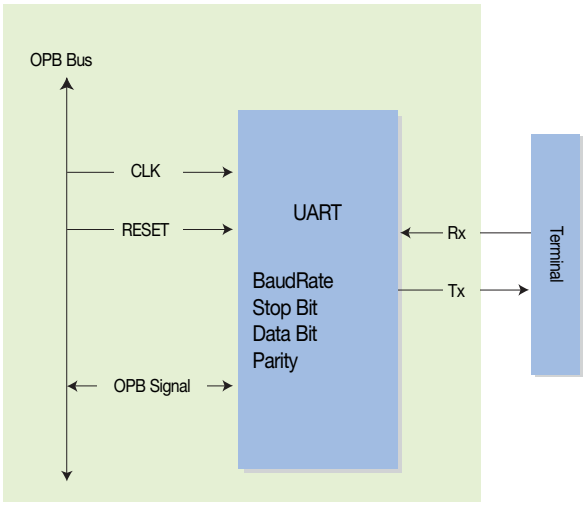

그림 5. UART 연결

```
PORT rx = rx, DIR = INPUT
PORT tx = tx, DIR = OUTPUT

BEGIN opb_uartlite
  PORT RX = rx
 PORT TX = tx
 BUS_INTERFACE SOPB = myopb
END
```

그리고 boudrate, parity, data bits와 같은 값을 주어진 스펙대로 초기화한다.

```
PARAMETER C_DATA_BITS = 8
PARAMETER C_BAUDRATE = 19200
PARAMETER C_USE_PARITY = 0
```

이런 UART를 설계하기 위해 자일링스에서는 opb_uartlite라는 퍼리퍼럴을 만들었는데, 이 퍼리퍼럴은 netlist로 제공된다. 따라서 netlist가 업데이트 될 때마다 버전을 관리할 필요가 있다.

```
PARAMETER HW_VER = 1.00.b
```

UART를 한 개만 사용하는 시스템도 있지만, 2개 이상 사용할 때도 같은 UART 하드웨어를 사용한다. 따라서 각 퍼리퍼럴을 사용할 때마다 고유의 instance 이름을 넣어서 구별해준다. 이 이름은 나중에 소프트웨어에서도 #define 문에 의해서 새롭게 정의되어 사용되며 장치 드라이버에서 사용하고 있다.

```
PARAMETER INSTANCE = myuart
```

UART 리셋 신호를 연결해 주는데, 이 리셋 신호는 리셋 블록에서 입력된다.

```
PORT OPB_Rst = peripheral_rst
```

현재 OPB는 myopb라는 instance를 사용하고 있다. 이 OPB에 UART를 연결해주면 UART와 OPB 사이에 버스 프로토콜은 자동적으로 만들어진다.

```
BUS_INTERFACE SOPB = myopb
```

리스트 2는 완성된 uart mhs file이다.

```
PORT rx = rx, DIR = INPUT
PORT tx = tx, DIR = OUTPUT

BEGIN opb_uartlite
PARAMETER INSTANCE = myuart
PARAMETER HW_VER = 1.00.b
PARAMETER C_DATA_BITS = 8
PARAMETER C_CLK_FREQ = 100000000
PARAMETER C_BAUDRATE = 19200
PARAMETER C_USE_PARITY = 0
PARAMETER C_BASEADDR = 0xf0000100
PARAMETER C_HIGHADDR = 0xf00001ff
PORT OPB_Rst = peripheral_rst
PORT RX = rx
PORT TX = tx
BUS_INTERFACE SOPB = myopb
END
```

리스트 2. UART_LITE 하드웨어 스펙

다음 호에서는 GPIO와 SDRAM에 관한 mhs file을 정의하고 XPS에서 만들어주는 장치 드라이버를 살펴보겠다. 그리고 실제 C언어로 코드를 만들고, 그 동작을 Virtex-Ⅱ pro 데모 보드에서 확인해 보도록 하겠다. Real Time