(기계로 한국어 번역)

Getting started with Xillinux for Cyclone V SoC (SoCKit)

Xillybus Ltd.

www.xillybus.com

Version 2.0

이 문서는 영어에서 컴퓨터에 의해 자동으로 번역되었으므로 언어가 불분명할 수 있습니다. 이 문서는 원본에 비해 약간 오래되었을 수 있습니다.

가능하면 영문 문서를 참고하시기 바랍니다.

This document has been automatically translated from English by a computer, which may result in unclear language. This document may also be slightly outdated in relation to the original.

If possible, please refer to the document in English.

1

소개

중요한:

상대적으로 낮은 대중의 관심으로 인해 SoCKit용 Xillinux는 단계적으로 중단됩니다. 알려진 문제 없이 즉시 사용할 수 있지만 구현은 FPGA 번들을 빌드하기 위한 다소 오래된 Quartus II 13.0sp1로 제한됩니다. 이것은 앞으로 변경되지 않을 것으로 예상됩니다. Cyclone V SoC, 특히 SoCKit에 대한 전반적인 지원은 제한적입니다.

1.1 Xillinux 배포판

Xillinux는 혼합 소프트웨어/logic 프로젝트의 신속한 개발을 위한 플랫폼으로 의도된 SoCKit 보드용 완전한 그래픽 Ubuntu 12.04 LTS 기반 Linux 배포판입니다. 다른 Linux 배포판과 마찬가지로 Xillinux는 Linux를 실행하는 개인용 데스크톱 컴퓨터와 거의 동일한 기능을 지원하는 소프트웨어 모음입니다. 일반적인 Linux 배포판과 달리 Xillinux에는 일부 하드웨어 logic, 특히 VGA 어댑터도 포함되어 있습니다.

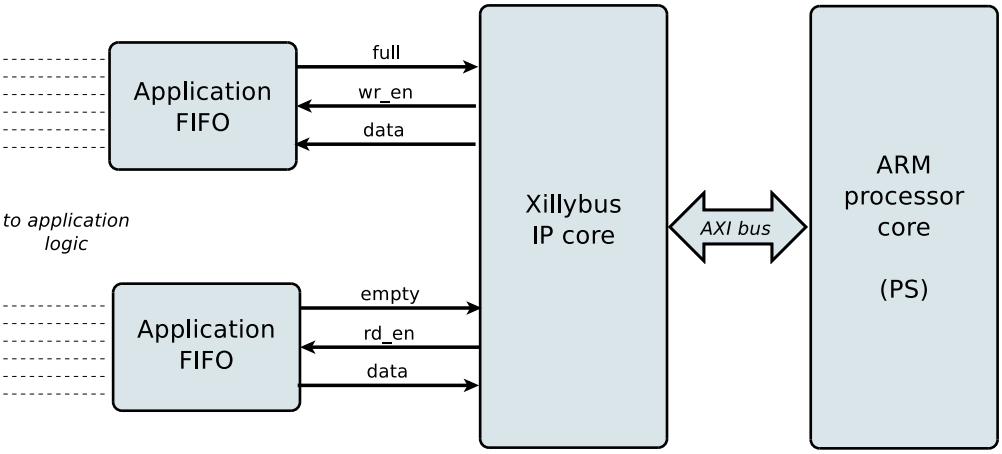
배포판은 클래식 키보드, 마우스 및 모니터 설정에 맞게 구성되어 있습니다. 또한 USB UART 포트에서 명령줄 제어를 허용하지만 이 기능은 대부분 문제 해결에 사용할 수 있습니다.

Xillinux는 또한 장치의 FPGA logic fabric와 ARM processor에서 실행되는 일반 user space applications 간의 통합을 위한 킥스타트 개발 플랫폼입니다. 포함된 Xillybus IP core 및 driver를 사용하면 FPGA logic 및 Linux 기반 소프트웨어가 함께 작동하는 애플리케이션의 design을 완성하는 데 프로그래밍 및 logic design의 기본 기술만 있으면 됩니다.

번들로 제공되는 Xillybus IP cores는 애플리케이션 설계자에게 간단하면서도 효율적인 작업 환경을 제공함으로써 application logic와 processor 간의 인터페이스는 물론 kernel programming의 하위 수준 내부를 처리할 필요가 없습니다.

1.2 Xillybus IP core

Xillybus는 FPGA와 Linux 또는 Microsoft Windows를 실행하는 host 간의 데이터 전송을 위한 DMA 기반 종단 간 솔루션입니다. FPGA logic의 설계자와 소프트웨어 프로그래머에게 간단하고 직관적인 인터페이스를 제공합니다. PCI Express bus를 기본 전송으로 사용하는 개인용 컴퓨터 및 embedded 시스템과 AMBA bus(AXI3/ AXI4)와 인터페이스하는 ARM 기반 processors에서 사용할 수 있습니다.



위에 표시된 것처럼 FPGA의 application logic은 표준 FIFOs와만 상호 작용하면 됩니다.

예를 들어, 다이어그램에서 하위 FIFO에 데이터를 쓰는 것은 Xillybus IP core가 FIFO의 다른 쪽 끝에서 데이터를 전송할 수 있음을 감지하게 합니다. 곧 IP core는 FIFO에서 데이터를 읽고 host로 보내 userspace software에서 읽을 수 있도록 합니다. 데이터 전송 메커니즘은 FIFO와만 상호 작용하는 FPGA의 application logic에 투명합니다.

반면 Xillybus IP core는 AXI bus를 활용하여 데이터 흐름을 구현하여 processor core의 bus에서 DMA 요청을 생성합니다.

host의 응용 프로그램은 named pipes처럼 동작하는 device files와 상호 작용합니다. Xillybus IP core 및 driver는 FPGAs의 FIFOs와 host의 관련 device files 간에 데이터를 효율적이고 직관적으로 전송합니다.

IP core는 온라인 웹 애플리케이션을 사용하여 고객의 사양에 따라 즉시 구축됩니다. streams의 수, 방향 및 기타 속성은 design의 대역폭 성능, 동기화 및 단순성 간의 최적 균형을 달성하기 위해 고객이 정의합니다.

이 가이드에 설명된 대로 준비를 마친 후 http://xillybus.com/custom-ip-factory에서 사용자 지정 IP core를 빌드하고 다운로드하는 것이 좋습니다.

이 가이드는 Xillybus IP core가 포함된 Xillinux 배포판을 빠르게 설정하는 방법을 설명합니다. 이 IP core는 실제 애플리케이션 시나리오 테스트를 위해 사용자 제공 데이터 소스

및 data sinks에 연결할 수 있습니다. 데모 키트는 아니지만 유용한 작업을 있는 그대로 수행할 수 있는 완전한 기능의 starter design입니다.

기존 IP core를 특수 응용 프로그램에 맞게 조정된 것으로 교체하는 것은 빠른 프로세스이며 하나의 바이너리 파일을 교체하고 하나의 단일 모듈을 인스턴스화해야 합니다.

궁금하신 분들을 위해 Xillybus IP core의 구현 방법에 대한 간략한 설명은 Xillybus host application programming guide for Linux의 부록 A에서 찾을 수 있습니다.

2

전제 조건

2.1 하드웨어

Sockit Linux 배포용 Xillybus(Xillinux)는 현재 SoCKit 보드만 지원합니다.

다른 보드의 소유자는 자신의 하드웨어에서 배포판을 실행할 수 있지만 일부는 중요하지 않을 수 있는 특정 변경이 필요할 수 있습니다. 이에 대한 자세한 내용은 5.2 섹션을 참조하십시오.

다음 장비도 필요합니다.

- 아날로그 VGA 입력으로 VESA 호환 1024x768 @ 60Hz를 표시할 수 있는 모니터(즉, 거의 모든 PC 모니터).
- 모니터용 아날로그 VGA 케이블
- USB 키보드
- USB 마우스
- 키보드와 마우스가 단일 USB 플러그에 결합되지 않은 경우 Linux 3.8.0에서 인식하는 USB 허브
- SoCKit 보드 카드에 대한 USB 케이블, USB Micro B 플러그에 대한 유형 A 소켓(암). 이 케이블은 SoCKit 보드 구매 시 **포함되어 있지 않습니다** .
- GB가 4개 이상 있는 안정적인 microSD 카드, 가장 바람직하게는 Sandisk에서 제조한 카드입니다.

  다른 브랜드는 Xillinux 사용 시 문제가 보고되었으므로 권장하지 않습니다.
- 이미지와 boot file을 카드에 쓰기 위한 microSD 카드와 PC 사이의 USB 어댑터. PC 컴퓨터에 SD 카드용 내장 슬롯이 있는 경우 이 작업이 필요하지 않을 수 있습니다.

SD와 microSD의 차이점은 물리적 폼 팩터일 뿐이므로 SD 카드 대 USB 어댑터는 SD 대 microSD 어댑터와 함께 MicroSD 대 USB 어댑터 대신 사용할 수 있습니다.

무선 키보드/마우스 콤보는 USB 허브가 필요 없고 실수로 USB 케이블을 잡아당겨 보드의 USB 포트가 물리적으로 손상되는 것을 방지하므로 권장됩니다.

2.2 배포판 다운로드

Xillinux 배포판은 Xillybus 사이트의 다운로드 페이지: http://xillybus.com/xillinux/에서 다운로드할 수 있습니다.

배포판은 두 부분으로 구성됩니다. 부팅 시 Linux에서 볼 수 있는 file system로 구성된 microSD 카드의 raw image와 first-stage boot image를 생성하기 위한 Intel FPGA 도구로 구현하기 위한 파일 세트입니다. 이에 대한 자세한 내용은 섹션 3입니다.

배포판에는 processor와 logic fabric 간의 손쉬운 통신을 위한 Xillybus IP core 데모가 포함되어 있습니다. 이 demo bundle의 특정 구성은 간단한 테스트를 위한 것이므로 특정 응용 프로그램에서 상대적으로 성능이 저하될 수 있습니다.

사용자 지정 IP cores는 IP Core Factory 웹 애플리케이션을 사용하여 구성, 자동 구축 및 다운로드할 수 있습니다. 이 도구를 사용하려면 http://xillybus.com/custom-ip-factory를 방문하십시오.

Xillybus IP core 및 Xillinux 배포를 포함하여 다운로드한 모든 번들은 이 사용이 “evaluation”라는 용어와 합리적으로 일치하는 한 무료로 사용할 수 있습니다. 여기에는 최종 사용자 designs에 IP core 통합, 실제 데이터 실행 및 현장 테스트가 포함됩니다. IP core 사용의 유일한 목적이 특정 애플리케이션에 대한 기능과 적합성을 평가하는 것이라면 IP core 사용 방법에는 제한이 없습니다.

2.3 개발 소프트웨어

Quartus II 13.0sp1(Web 또는 Subscription Edition)만 Xillinux를 빌드하는 데 사용할 수 있습니다. 다른 버전으로 번들을 빌드하려고 하면 오류와 함께 실패합니다. ARM processor가 Intel FPGA 장치의 새로운 기능이므로 지원 소프트웨어 도구가 매우 빠르게 변경되고 있으므로 신뢰할 수 있는 결과를 보장하는 유일한 방법은 Xillinux가 제대로 테스트된 것과 정확히 동일한 개정판을 사용하는 것입니다.

Cyclone V SoC용 Xillinux가 단계적으로 중단됨에 따라 불행히도 향후 이 제한을 해제할 계획이 없습니다.

도구가 RAM의 2GB 이상을 할당하므로 Quartus II의 64비트 버전이 특히 Windows 시스템에서 선호되며, 이는 32비트 Quartus II에서 실패할 수 있습니다. 32비트 Windows

XP를 사용하는 경우 boot.ini에 `/3gb` 플래그를 추가하면 도구를 성공적으로 실행할 수 있습니다. http://msdn.microsoft.com/en-us/library/windows/hardware/gg487508.aspx를 참조하십시오.

32비트 Windows 7 이상은 `increaseuserva` 매개변수로 수정할 수 있습니다. 다음을 참조하십시오.

http://msdn.microsoft.com/en-us/library/windows/hardware/ff542202(v=vs.85).aspx.

Xillinux 빌드에는 Soc EDS 패키지가 **필요하지 않습니다** . user space programs 및/또는 kernel modules의 compilation은 보드의 processor에서 직접 수행할 수 있습니다.

이 소프트웨어는 Intel FPGA의 웹사이트(https://www.intel.com)에서 직접 다운로드할 수 있습니다.

2.4 FPGA design 사용 경험

SoCKit 보드로 작업할 때 플랫폼에서 배포판을 실행하기 위해 FPGA design에 대한 이전 경험이 필요하지 않습니다. 다른 보드로 작업하려면 Intel FPGA 도구 사용에 대한 약간의 지식이 필요하고 Linux kernel와 관련된 몇 가지 기본 기술이 필요할 수 있습니다.

배포판을 최대한 활용하려면 logic design 기술을 잘 이해하고 HDL 언어(Verilog 또는 VHDL)를 마스터해야 합니다. 그럼에도 불구하고 Xillybus 배포판은 실험할 간단한 스타터 design을 제공하므로 이러한 학습을 위한 좋은 출발점입니다.

3

건물 Xillinux

3.1 개요

Xillinux 배포판은 단순한 데모가 아닌 개발 플랫폼으로 고안되었습니다. 맞춤형 logic 개발 및 통합을 위한 즉시 사용 가능한 환경은 하드웨어에서 실행하기 위한 준비 과정에서 구축됩니다. 따라서 첫 번째 테스트 실행을 위한 준비 시간은 약간 길다(일반적으로 30분, 대부분은 Xilinx의 도구를 기다리는 시간으로 구성됨). 그러나 이러한 긴 준비는 맞춤형 logic 통합 주기를 단축시킵니다.

microSD 카드에서 Xillinux 배포의 boot를 수행하려면 세 가지 구성 요소가 있어야 합니다.

- U-boot 로더로 구성된 특수 partition에 있는 초기 boot image 환경
- FPGA 부분에 대한 구성 bitstream, Linux kernel 바이너리 및 해당 device tree를 포함하는 파일이 있는 작은 FAT 파일 시스템.
- Linux에 탑재된 root file system.

FPGA의 bitstream을 제외한 이 모든 것은 이미 Xillinux용 microSD 이미지에 포함되어 있습니다.

microSD 준비를 위한 다양한 작업은 이 섹션에서 단계별로 자세히 설명합니다. 대부분의 시간은 FPGA bitstream을 준비하는 데 사용됩니다.

이 절차에서는 SoCKit 보드가 사용된다고 가정합니다. 다음 단계로 구성되며 아래에 설명된 순서대로 수행해야 합니다.

- boot image 키트 압축 풀기
- processor의 래퍼 및 bus 인프라 생성
- 기본 FPGA 프로젝트를 구현하고 bitstream을 올바른 형식으로 변환합니다.

- 원시 Xillinux 이미지를 microSD 카드에 쓰기
- bitstream 파일을 microSD 카드에 복사

다른 보드와 작업하는 방법은 5.2 단락에서 설명합니다.

3.2 boot image 키트 압축 풀기

이전에 다운로드한 xillinux-eval-sockit-*XXX*.zip 파일을 작업 디렉토리에 압축을 풉니다.

중요한:

작업 디렉토리의 경로에는 공백이 포함되지 않아야 합니다. 특히 Desktop은 해당 경로에 "Documents and Settings"가 포함되어 있으므로 적합하지 않습니다.

번들은 다음 디렉토리로 구성됩니다.

- verilog– 기본 logic에 대한 프로젝트 파일과 Verilog의 일부 소스('src' 하위 디렉토리에 있음)를 포함합니다.
- vhdl– 기본 logic 및 일부 소스 파일에 대한 프로젝트 파일을 포함합니다. VHDL에서 편집할 파일은 'src' 하위 디렉토리에 있습니다.
- core– Xillybus IP cores의 미리 컴파일된 바이너리
- soc_system– ARM processor 래퍼 및 bus 인프라
- instantiation templates– Verilog 및 VHDL에 instantiation templates 포함

'verilog' 및 'vhdl' 디렉토리에는 모두 SoCKit 보드용 QSF 파일도 포함되어 있습니다. 다른 보드를 사용하는 경우 이 파일을 편집해야 합니다.

또한 vhdl 디렉토리에는 Verilog 파일이 포함되어 있지만 사용자가 편집할 필요는 없습니다.

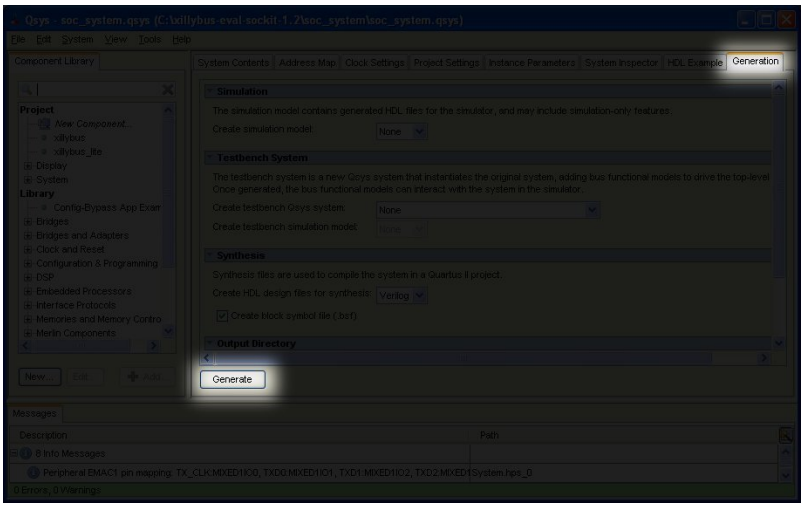
Xillybus IP core와의 인터페이스는 각 'src' 하위 디렉토리의 xillydemo.v 또는 xillydemo.vhd 파일에서 발생합니다. 이것은 자신의 데이터 소스와 sinks로 Xillybus를 시도하기 위해 편집할 파일입니다.

3.3 processor의 래퍼 생성

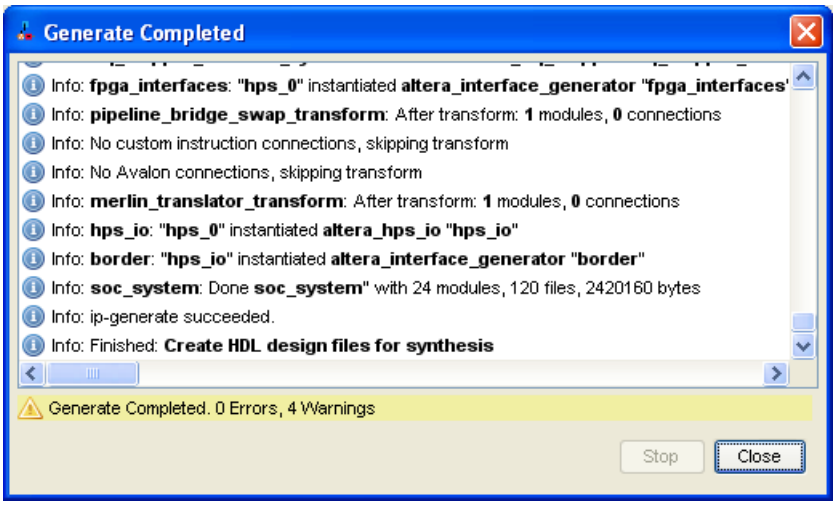
boot image 키트의 'verilog' 또는 'vhdl' 하위 디렉터리에서 ' xillydemo.qpf' 파일을 두 번 클릭하여 Quartus II를 엽니다.

Quartus 내에서 “File > Open...”을 선택하여 Qsys 프로젝트를 열고 한 디렉토리 위로 이동하여 soc_system 디렉토리로 들어가 soc_system.qsys를 선택합니다. Qsys 도구가 processor 래퍼 프로젝트를 시작하고 엽니다.

창 상단 근처에서 "Generation" 탭을 선택하고 “Generate”(하단 근처)를 클릭합니다.



프로세스는 다음과 같이 종료되는 진행 창과 함께 몇 분 정도 걸립니다.



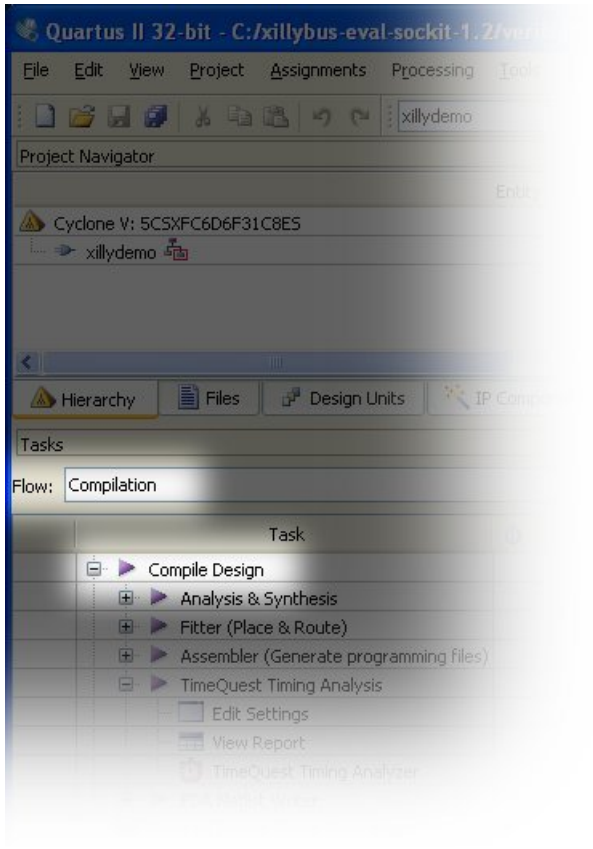
진행 창과 Qsys 창을 닫습니다. 앞으로 이 과정을 반복할 필요가 없습니다.

3.4 원시 bitstream 파일 생성

이 단계는 3.3 단락에 설명된 대로 processor 래퍼 생성을 완료한 후 수행됩니다.

기본 설정에 따라 'verilog' 또는 'vhdl' 하위 디렉토리에서 ' xillydemo.qpf' 파일을 두 번 클릭합니다. Quartus II는 올바른 설정으로 프로젝트를 시작하고 엽니다. Qsys로 방금 완료했다면 이미 Quartus II가 열려 있을 가능성이 있습니다. 이 경우 선호하는 언어가 선택되었는지 확인하십시오(상관 없는 경우 Verilog 선택).

이 이미지와 같이 "flow"가 "Compilation"로 설정되어 있는지 확인하고 "Compile Design"을 클릭하여 FPGA 프로그래밍 파일을 생성합니다.



이 프로세스는 약 100개의 경고를 생성하지만 "Full Compilation was successful"을 알리는 대화 상자로 끝나야 합니다. 심각한 경고가 생성되지만 오류가 허용되어서는 안 됩니다. 또한 compilation 이후에는 "Timing requirements not met" (332148)에 **대한** 경고가 없는지 확인하는 것이 항상 필수입니다. 이것은 특히 나중에 Verilog/VHDL 소스를 변경한 후 design의 compilation을 실행할 때 사실입니다.

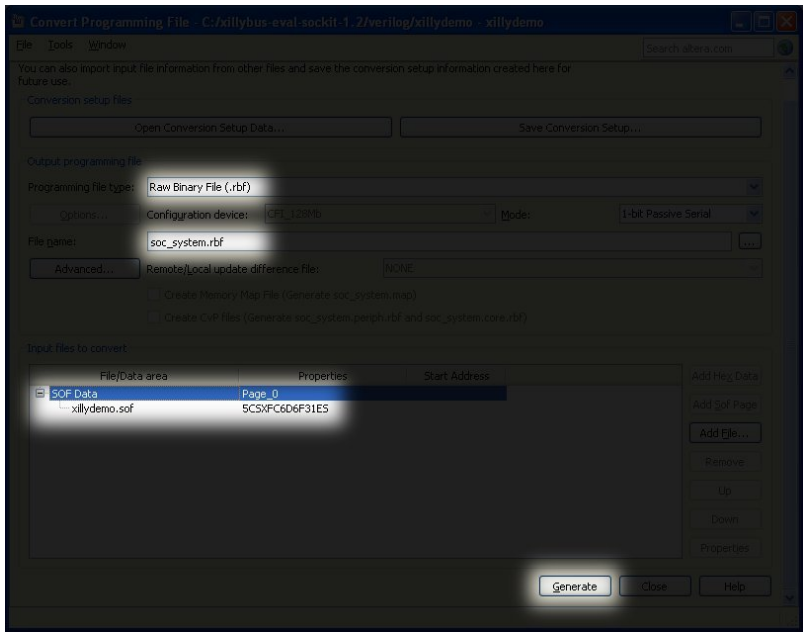
마지막으로 얻은 프로그래밍 파일을 필요한 형식으로 변환합니다. Quartus II에서 File > Convert Programming Files...를 선택하고 프로그래밍 파일 유형으로 Raw Binary File (.rbf)를 선택하십시오. 바로 아래에서 “File name”을 soc_system.rbf로 설정합니다.

“Input files to convert” 영역에서 “SOF Data”를 클릭한 다음 오른쪽에 있는 “Add File...”을 클릭합니다. xillydemo.sof를 선택합니다.

그런 다음 오른쪽 하단에서 “Generate”를 클릭합니다. 파일을 성공적으로 생성한 후 창을 닫습니다. 3.6 단락에 설명된 대로 soc_system.rbf를 MicroSD 카드에 복사해야 합니다.

중요한:

.rbf 파일을 생성할 때 압축을 활성화하지 마십시오(기본값 유지). soc_system.rbf는 6-7 MBytes여야 합니다.



3.5 이미지와 함께 microSD 로드

3.5.1 일반적인

이 작업의 목적은 다운로드한 microSD 카드 이미지 파일을 장치에 쓰는 것입니다. 이미지는 xillinux-1.1-sockit.img.gz(또는 이와 유사한)라는 파일로 다운로드되었으며 microSD 카드의 gzip 압축 이미지입니다(soc_system.rbf 파일을 추가해야 함).

이 이미지는 압축을 풀고 microSD 카드의 첫 번째 sector 이상에 기록해야 합니다. 이를 수행하는 몇 가지 방법과 도구가 있습니다. 몇 가지 방법이 다음에 제안됩니다.

이미지에는 partition table, boot image를 배치하기 위한 부분적으로 채워진 FAT file system, raw boot partition 및 ext4 유형의 Linux root file system이 포함되어 있습니다. 거의 모든 Windows 컴퓨터는 FAT partitions만 감지하므로 microSD 카드는 용량이 매우 작은 것처럼 보입니다(47 MB 정도).

전체 디스크 이미지를 작성하는 것은 일반 컴퓨터 사용자를 위한 작업이 아니므로 Windows 컴퓨터의 특수 소프트웨어와 Linux에 대한 각별한 주의가 필요합니다. 다음 단락에서는 두 운영 체제에서 이 작업을 수행하는 방법을 설명합니다.

중요한:

microSD에 이미지를 기록하면 포함되어 있을 수 있는 이전 콘텐츠가 복구 불가능하게 삭제됩니다. 이미지 작성에 사용된 것과 동일한 도구를 사용하여 기존 콘텐츠의 복사본을 만드는 것이 좋습니다.

3.5.2 이미지 로드(Windows)

Windows에서 이미지를 복사하려면 USB Image Tool와 같은 특수 응용 프로그램이 필요합니다. 이 도구는 USB 어댑터를 사용하여 microSD 카드에 액세스할 때 적합합니다.

일부 컴퓨터(특히 노트북)에는 SD 슬롯이 내장되어 있으며 Win32 Disk Imager와 같은 다른 도구를 사용해야 할 수도 있습니다. Windows 7을 실행할 때도 마찬가지입니다.

두 도구 모두 웹의 다양한 사이트에서 무료로 다운로드할 수 있습니다. 다음 연습에서는 USB Image Tool을 사용한다고 가정합니다.

그래픽 인터페이스의 경우 “USB Image Tool.exe”를 실행합니다. 기본 창이 나타나면 USB 어댑터를 연결하고 왼쪽 상단에 나타나는 장치 아이콘을 선택합니다. 왼쪽 상단 드롭다운 메뉴에서 “Device Mode”(“Volume Mode”와 반대)에 있는지 확인합니다. Restore를 클릭하고 파일 형식을 “Compressed (gzip) image files”로 설정합니다. 다운로드한 이미지 파일(xillinux-1.1-sockit.img.gz)을 선택합니다. 전체 프로세스는 약 4-5분이 소요됩니다. 완료되면 장치(“safely remove hardware”)를 마운트 해제하고 플러그를 뽑습니다.

일부 시스템에서 GUI는 소프트웨어 초기화 실패라는 오류와 함께 실행에 실패합니다. 이 경우 대체 명령줄을 사용하거나 Microsoft .NET framework 구성 요소를 설치해야 합니다.

또는 명령줄에서 수행할 수 있습니다(GUI 실행 시도가 실패할 경우 빠른 대안). 이것은 두 단계로 수행됩니다. 먼저 장치의 번호를 얻습니다. DOS Window에서 디렉토리를 애플리케이션의 압축을 풀고 이동합니다(일반적인 세션은 다음과 같습니다).

```
C:\usbimage>usbitcmd l
```

```
USB Image Tool 1.57
COPYRIGHT 2006-2010 Alexander Beug
http://www.alexpage.de

  Device    | Friendly Name              | Volume Name | Volume Path | Size
 ------------------------------------------------------------------------------
      2448 | USB Mass  Storage Device  |             | E:\         |   2014 MB
```

("usbitcmd" 뒤의 문자는 숫자 “1”이 아니라 문자 “l”입니다.)

이제 장치 번호가 있으면 실제로 쓰기를 수행할 수 있습니다(“restore”).

```
C:\usbimage>usbitcmd r 2448 \path\to\xillinux-1.1-sockit.img.gz /d /g

USB Image Tool 1.57
COPYRIGHT 2006-2010 Alexander Beug
http://www.alexpage.de

Restoring backup to "USB Mass  Storage Device USB Device" (E:\)...ok
```

다시 말하지만 이 작업은 약 4-5분 정도 소요됩니다. 물론 2448라는 숫자를 첫 번째 단계에서 얻은 장치 번호로 변경하고 `\path\to`는 microSD 카드의 이미지가 컴퓨터에 저장된 경로로 대체됩니다.

3.5.3 이미지 로드(Linux)

중요한:

장치에 원시 복사는 위험한 작업입니다. 사람의 사소한 실수(일반적으로 잘못된 대상 디스크 선택)로 인해 컴퓨터 하드 디스크의 모든 데이터가 복구 불가능한 손실로 이어질 수 있습니다. Enter를 누르기 전에 생각하고 Linux에 익숙하지 않은 경우 Windows에서 이 작업을 수행하는 것이 좋습니다.

방금 언급했듯이 올바른 장치를 microSD 카드로 감지하는 것이 중요합니다. 이것은 USB 커넥터를 연결하여 가장 잘 수행되며 다음과 같은 항목을 찾는 것이 기본 로그 파일(/var/log/messages 또는 /var/log/syslog)입니다.

```
Sep  5 10:30:59 kernel: sd 1:0:0:0: [sdc] 7813120 512-byte logical blocks
Sep  5 10:30:59 kernel: sd 1:0:0:0: [sdc] Write Protect is off
```

```
Sep  5 10:30:59 kernel: sd 1:0:0:0: [sdc] Assuming drive cache: write through
Sep  5 10:30:59 kernel: sd 1:0:0:0: [sdc] Assuming drive cache: write through
Sep  5 10:30:59 kernel:  sdc: sdc1
Sep  5 10:30:59 kernel: sd 1:0:0:0: [sdc] Assuming drive cache: write through
Sep  5 10:30:59 kernel: sd 1:0:0:0: [sdc] Attached SCSI removable disk
Sep  5 10:31:00 kernel: sd 1:0:0:0: Attached scsi generic sg0 type 0
```

출력은 약간 다를 수 있지만 여기서 요점은 kernel이 새 디스크에 어떤 이름을 부여했는지 확인하는 것입니다. 위의 예에서 “sdc”.

image file의 압축을 풉니다.

```
# gunzip xillinux-1.1-sockit.img.gz
```

microSD 카드에 이미지를 복사하는 것은 간단합니다.

```
# dd if=xillinux-1.1-sockit.img of=/dev/sdc bs=512
```

물론 플래시 드라이브로 찾은 디스크를 가리켜야 합니다.

중요한:

/dev/sdc를 예로 들어 설명합니다. 컴퓨터에서 인식된 장치와 일치하지 않는 한 이 장치를 사용하지 마십시오.

그리고 확인

```
# cmp xillinux-1.1-sockit.img /dev/sdc
cmp: EOF on xillinux-1.1-sockit.img
```

응답에 유의하십시오. 이미지 파일에서 EOF에 도달했다는 사실은 다른 모든 항목이 올바르게 비교되었으며 flash drive에 실제로 사용된 것보다 더 많은 공간이 있음을 의미합니다. cmp가 아무 것도 말하지 않으면(일반적으로 좋은 것으로 간주됨) 실제로 뭔가 잘못되었음을 의미합니다. 장치에 쓰지 않고 일반 파일 “/dev/sdc”가 생성되었을 가능성이 큽니다.

3.6 soc_system.rbf 파일을 microSD 카드에 복사

USB 어댑터의 플러그를 뽑았다가 다시 컴퓨터에 연결합니다. 이것은 컴퓨터가 microSD 카드의 partition table로 최신 상태인지 확인하는 데 필요합니다. 필요한 경우 첫 번째 partition(예: /dev/sdb1)를 장착합니다. 대부분의 컴퓨터는 이 작업을 자동으로 수행합니다.

그런 다음 soc_system.rbf(단락 3.4에서 생성됨)를 microSD 카드의 FAT file system에 복사합니다. Windows 시스템에서 이것은 시스템이 표시할 유일한 “disk”입니다. Linux 시스템에서는 최초의(그리고 더 작은) partition입니다. 어느 쪽이든, 올바른 대상은 ’ socfpga.dtb’ 및 ’uImage’라는 두 개의 파일인 기존 콘텐츠로 쉽게 인식됩니다.

완료되면 microSD 카드를 올바르게 마운트 해제하고 컴퓨터에서 플러그를 뽑습니다.

```
> umount /mnt/sd
```

SoCKit 보드가 이미 Xillinux를 실행 중인 경우 보드 자체에서 soc_system.rbf를 업데이트 할 수 있습니다. Xillinux에 FAT 파일 시스템을 마운트하려면 다음으로 이동하십시오.

```
> mkdir /mnt/sd
> mount /dev/mmcblk0p1 /mnt/sd
```

/mnt/sd/에서 file system에 액세스합니다. 새 soc_system.rbf 자체에 문제가 있거나(예: 압축됨) 다른 작업이 부적절하게 수행되면 보드가 다음에 boot를 수행하지 못할 수 있습니다. 따라서 microSD 카드에 액세스할 수 있는 대체 수단을 반드시 준비해 두어야 합니다.

4

boot 수행

4.1 점퍼 및 DIP switch 설정

보드가 microSD 카드에서 boot를 수행하려면 점퍼를 변경할 필요가 없지만 다음 페이지의 첫 번째 이미지에 표시된 설정과 일치하는지 확인하는 것이 가장 좋습니다.

점퍼 J15-J19만 시스템의 boot와 관련이 있습니다. 다른 두 개의 점퍼는 LCD 백라이트 및 HSMC 인터페이스 전압 레벨과 관련이 있습니다.

DIP switches(보드 뒷면)는 일반적으로 다음 페이지의 두 번째 이미지와 같이 변경해야 합니다.

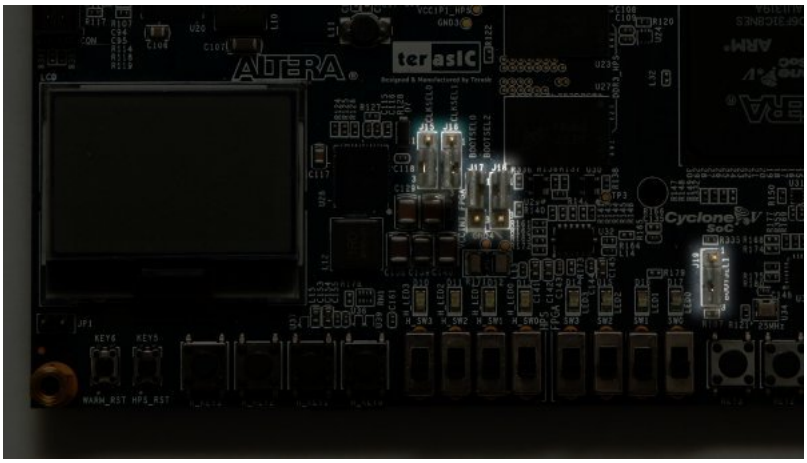
4.2 주변기기 부착

중요한:

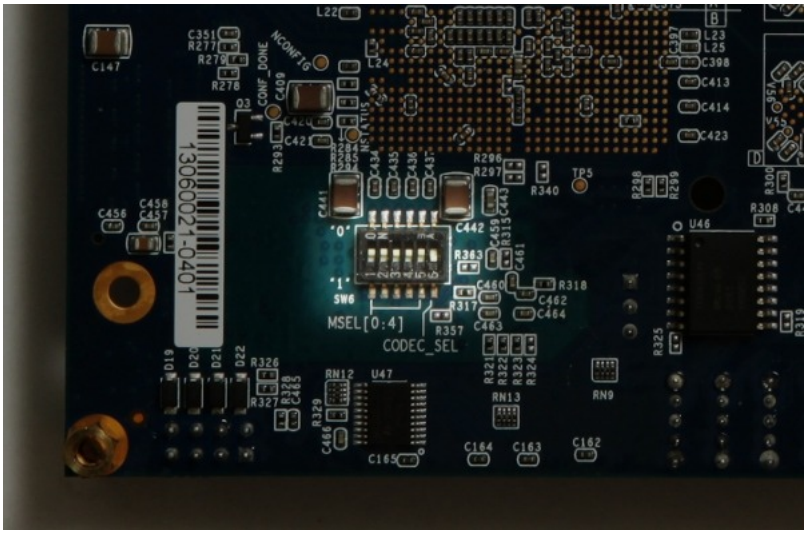
USB 포트가 사용에 필요한 경우(예: 마우스 및 키보드 연결) Linux가 boot를 수행할 때 일부 주변 장치를 OTG USB 포트에 연결해야 합니다(아무것도 연결되지 않은 USB 허브 포함). 이는 Linux가 processor와 이에 연결된 대상 간의 역할을 결정하는 데 필요합니다. 둘 다 OTG 포트의 host일 수 있기 때문입니다.

다음 범용 하드웨어를 보드에 부착해야 합니다.

- 아날로그 VGA 커넥터에 대한 컴퓨터 모니터. Xillinux는 아날로그 VGA 플러그를 통해 VESA 호환 1024x768 @ 60Hz 신호를 생성하기 때문에 어떤 컴퓨터 모니터로도 충분할 것이 거의 확실합니다.
- USB 암 케이블을 통한 USB OTG 커넥터에 대한 마우스 및 키보드(SocKit 하드웨어 키트에는 포함되지 않음). 시스템은 boot가 없을 때 boot를 수행하며 Linux가 boot



보드 앞면, 강조 표시된 점퍼 설정



보드 뒷면, DIP switch가 강조 표시됨. 가장 오른쪽을 제외한 모든 스위치는 위쪽으로 밉니다.

시퀀스를 수행할 때 일부 주변 장치(또는 단지 USB 허브)가 연결되어 있는 한 시스템이 실행되는 동안 키보드와 마우스를 연결 및 연결 해제하는 데 문제가 없습니다. 시스템은 주어진 순간에 연결된 키보드와 마우스를 감지하고 작동합니다.

- Ethernet 포트는 일반적인 네트워크 작업을 위해 선택 사항입니다. 연결된 네트워크에 DHCP 서버가 있는 경우 Linux 시스템은 네트워크를 자동으로 구성합니다.
- UART USB 포트는 선택적으로 PC에 연결되지만 대부분의 경우 중복됩니다. boot 메시지 중 일부가 그곳으로 전송되고 boot가 완료되면 이 인터페이스에서 shell prompt가 발행됩니다.

  PC 모니터나 키보드가 없거나 제대로 작동하지 않을 때 유용합니다.

  이 포트가 컴퓨터에 연결되어 있으면 일부 terminal 소프트웨어가 이 컴퓨터에서 실행되어야 합니다. 그렇지 않으면 Linux가 boot 메시지를 읽을 때까지 기다리는 동안 boot 프로세스가 중지될 수 있습니다. 이 포트가 연결되지 않은 상태에서 boot를 수행하는 것은 괜찮습니다.

4.3 보드 전원 켜기

이 단락에서는 시스템 전원을 켤 때 예상되는 사항에 대해 설명합니다. 아래 설명에서는 앞 페이지의 이미지와 같이 왼쪽 상단에 빨간색 전원 버튼이, 하단에 LEDs 8열이 보이도록 보드를 봤습니다.

중요한:

Xillinux의 root file system은 microSD 카드에 영구적으로 상주하며 시스템이 가동되는 동안 기록됩니다. 따라서 Linux 시스템은 일반적으로 갑작스러운 정전 시 적절한 복구가 관찰되더라도 다른 PC 컴퓨터와 마찬가지로 시스템을 안정적으로 유지하기 위해 보드의 전원을 끄기 전에 적절하게 종료해야 합니다.

microSD 카드를 SoCKit 보드에 연결하고 빨간색 버튼을 눌러 전원을 켭니다. 다음 순서가 예상됩니다.

- 다음 LEDs가 즉시 켜집니다.
  - 전원 버튼 옆에 있는 녹색 LED
  - 약한 조명이 있는 보드 하단의 LEDs 8개 모두
  - LCD 화면의 백라이트 LED(이 표시등을 비활성화하기 위해 점퍼가 설치된 경우 제외)

전원 버튼의 약간 오른쪽에서도 몇 번의 짧은 깜박임이 예상됩니다. UART LEDs입니다. 컴퓨터가 UART USB 포트에 연결되어 있으면 계속 깜박입니다.

- 전원을 켠 후 1초도 채 되지 않아 가장 왼쪽에 있는 4개의 LEDs가 꺼집니다. 이는 initial bootloader가 processor를 초기화했다는 표시입니다. 이런 일이 발생하지 않으면 DIP 스위치 또는 점퍼가 잘못되었거나(섹션 4.1 참조) microSD가 제대로 설치되지 않았거나, 결함이 있거나, Xillinux image와 함께 제대로 로드되지 않았기 때문일 수 있습니다.
- 전원을 켠 후 약 14초 후에 나머지 4개(가장 오른쪽) LEDs도 꺼지고 가장 오른쪽 LED가 1 Hz(대략)에서 깜박이기 시작합니다. “Xillybus” 화면 보호기는 VGA 화면에 1초 미만 동안 나타나고 왼쪽 상단에 두 개의 Linux Penguins 로고가 있는 빈 화면으로 바뀝니다. 이것이 발생하지 않으면 soc_system.rbf 파일이 제자리에 있고 압축되지 않았는지 확인하십시오(5-6 MBytes).
- 전원을 켠 후 약 26초 후에 boot 텍스트가 VGA 화면에 나타납니다.
- login prompt는 전원을 켠 후 35초 이내에 나타나야 합니다. 시스템이 root로 자동 로그인하여 인사말과 shell prompt를 표시합니다. 유사한 shell prompt도 USB UART 링크에 제공되며 대부분 문제 해결을 위해 제공됩니다.

가장 왼쪽에 있는 4개의 LEDs가 꺼진 후에도 아무 일도 일어나지 않으면 soc_system.rbf 파일에 문제가 있을 수 있습니다. 4.4 단락에 설명된 대로 UART의 출력을 살펴보는 것이 도움이 될 수 있습니다.

보드의 파워업을 보여주는 짧은 비디오 클립은 다음에서 볼 수 있습니다.

http://youtu.be/mTDaAn4lX3I. UART의 LEDs는 UART USB 포트에 연결된 것이 없기 때문에 클립에서 거의 활동을 보여주지 않습니다.

shell prompt에서 “startx”를 입력하여 Gnome 그래픽 데스크탑을 실행합니다. 바탕 화면을 초기화하는 데 15-30초 정도 걸립니다.

메모:

- root user의 암호는 아무것도 설정되어 있지 않으므로 필요한 경우 root로 로그인할 때 암호가 필요하지 않습니다.
- 흰색 배경의 Xillybus 로고 화면 보호기는 logic fabric이 로드되는 순간부터 Linux kernel이 실행될 때까지 화면에 표시됩니다. 또한 운영 체제가 화면을 “blank” 모드로 설정하는 경우(시스템이 유휴 상태일 때 정상 상태) 또는 X-Windows 시스템이 그래픽 모드 조작을 시도할 때 표시됩니다.

- **파란색** 배경의 Xillybus 화면 보호기 또는 화면의 임의의 파란색 줄무늬는 그래픽 인터페이스가 데이터 부족을 겪고 있음을 나타냅니다. 명백한 이유가 알려져 있지 않는 한 이러한 일이 발생할 것으로 예상되지 않으며 보고해야 합니다.

4.4 boot 중 UART 출력

boot 프로세스가 실패하면 UART의 출력에서 무엇이 잘못되었는지에 대한 힌트를 제공할 수 있습니다. 컴퓨터의 terminal 응용 프로그램은 57600 baud, 8 bits, 1 stop bit용으로 구성되어야 하며 parity 및 flow control(“57600 8N1”)가 없습니다.

일반적으로 다음과 같이 표시됩니다.

```
U-Boot SPL 2012.10 (Dec 30 2013 - 18:03:34)
SDRAM: Initializing MMR registers
SDRAM: Calibrating PHY
SEQ.C: Preparing to start memory calibration
SEQ.C: CALIBRATION PASSED
DESIGNWARE SD/MMC: 0


U-Boot 2012.10 (Dec 30 2013 - 18:03:46)

CPU   : Altera SOCFPGA Platform
BOARD : Altera SOCFPGA Cyclone 5 Board
DRAM:  1 GiB
MMC:   DESIGNWARE SD/MMC: 0
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   mii0
Warning: failed to set MAC address
```

마지막 줄은 몇 초를 카운트다운한 다음 세 개의 파일을 읽으면서 자동으로 계속됩니다. 바이트 수와 표시된 kernel 버전은 다를 수 있습니다.

bootloader가 이러한 파일을 읽는 중 오류를 보고하면 아마도 boot를 수행하지 못한 이유일 것입니다. “bad CRC” 오류는 U-boot가 저장된 환경 변수 세트를 찾지 못했음을 나타냅니다. 따라서 기본값을 사용하는데 문제가 없습니다.

```
reading uImage
```

```
3328400 bytes read
reading socfpga.dtb

15576 bytes read
reading soc_system.rbf

7007184 bytes read
## Booting kernel from Legacy Image at 00007fc0 ...
   Image Name:   Linux-3.8.0-xillinux-1.1
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3328336 Bytes = 3.2 MiB
   Load Address: 00008000
   Entry Point:  00008000
## Flattened Device Tree blob at 00000100
   Booting using the fdt blob at 0x00000100
   XIP Kernel Image ... OK
OK
   Loading Device Tree to 0fff8000, end 0fffecd7 ... OK

Starting kernel ...
```

이 시점에서 boot loader는 제어권을 Linux kernel에 넘깁니다. kernel의 boot 메시지의 시작 부분만 아래에 나와 있습니다. kernel이 boot 시퀀스를 완료한 후 shell prompt가 제공됩니다.

"Starting kernel..." 이후에 아무 것도 나타나지 않으면 보드에서 파란색 LED가 깜박이고 있는지 확인하십시오. 이는 FPGA 부품이 성공적으로 로드되었음을 나타냅니다. soc_system.rbf 파일 로드 실패가 가장 큰 이유입니다.

```
Booting Linux on physical CPU 0x0
Initializing cgroup subsys cpuset
Linux version 3.8.0-00116-gffe44a8 (eli@ocho.localdomain) (gcc version 4.6.3 (S3
CPU: ARMv7 Processor [413fc090] revision 0 (ARMv7), cr=10c5387d
CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine: Altera SOCFPGA, model: Altera SOCFPGA Cyclone V
Memory policy: ECC disabled, Data cache writealloc
PERCPU: Embedded 8 pages/cpu @80e77000 s10880 r8192 d13696 u32768
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 260096
Kernel command line: console=ttyS0,57600 root=/dev/mmcblk0p3 rw rootwait
```

4.5 첫 번째 boot 직후 수행할 작업

4.5.1 file system 크기 조정

root file system 이미지는 장치에 최대한 빨리 쓰기 위해 작게 유지됩니다. 반면 microSD 카드의 전체 용량을 사용하지 않을 이유가 없습니다.

중요한:

file system의 크기를 조정하려고 시도하는 동안 전체 microSD 카드의 내용을 지울 상당한 위험이 있습니다. 따라서 이러한 사고의 비용은 microSD 카드 초기화(이미지 및 soc_system.rbf 쓰기)를 반복하는 것뿐이지만 가능한 한 빨리 이 작업을 수행하는 것이 좋습니다.

시작점은 일반적으로 다음과 같습니다.

```
# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       2.3G  1.9G  304M  87% /
devtmpfs        505M  4.0K  505M   1% /dev
none            101M  736K  101M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            505M     0  505M   0% /run/shm
```

따라서 root filesystem은 2.3 GB이고 304 MB는 무료입니다.

첫 번째 단계는 microSD 카드를 다시 분할하는 것입니다. shell prompt에서 다음을 입력합니다.

```
# fdisk /dev/mmcblk0
```

그런 다음 다음과 같이 입력합니다(아래 세션 기록 참조).

- d [ENTER]– partition 삭제
- 3 [ENTER]– partition 번호 3 선택
- n [ENTER]– 새 partition 만들기

ENTER 를 4번 눌러 기본값인 primary partition, 숫자 3을 수락합니다. 가능한 가장 낮은 sector에서 시작하여 가장 높은 sector에서 끝납니다.

- w [ENTER]– 저장하고 종료합니다.

이 시퀀스 도중에 문제가 발생하면 CTRL-C(또는 q [ENTER])를 눌러 변경 사항을 저장하지 않고 fdisk을 종료하십시오. 마지막 단계까지 microSD 카드에서 아무 것도 변경되지 않습니다.

일반적인 세션은 다음과 같습니다. sector 번호는 다를 수 있습니다.

```
root@localhost:~# fdisk /dev/mmcblk0

Command (m for help): d
Partition number (1-4): 3

Command (m for help): n
Partition type:
   p   primary (2 primary, 0 extended, 2 free)
   e   extended
Select (default p):
Using default response p
Partition number (1-4, default 3):
Using default value 3
First sector (112455-15523839, default 112455):
Using default value 112455
Last sector, +sectors or +size{K,M,G} (112455-15523839, default 15523839):
Using default value 15523839

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16:
         Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
```

시스템에 표시된 첫 번째 sector의 기본값이 위의 것과 다른 경우 여기에 표시된 것이 아니라 시스템의 기본값을 선택하십시오.

이 시퀀스에서 fdisk의 기본값에서 전환하는 것이 합리적일 수 있는 유일한 위치는 file system을 가능한 최대값보다 작게 만들기 위해 마지막 sector입니다.

하단의 경고에서 알 수 있듯이 root partition이 사용 중이기 때문에 partition table의 Linux 보기를 업데이트할 수 없습니다. 따라서 재부팅해야 합니다.

```
# shutdown -h now
```

UART console에 “System halted.”라는 메시지가 표시되면(또는 VGA 화면에서 커서가 깜박임을 멈춤) 보드의 전원을 껐다가 다시 켭니다. 시스템은 이전과 마찬가지로 boot를 수행해야 하지만, 다시 분할하는 동안 무언가 잘못 수행된 경우 boot는 어떤 단계에서든 실패할 수 있습니다.

file system은 아직 크기가 조정되지 않았습니다. 크기를 조정할 수 있는 공간만 주어졌습니다. 따라서 shell prompt에서 다음을 입력합니다.

다음 응답이 예상되는

```
# resize2fs /dev/mmcblk0p3
```

:

```
resize2fs 1.42 (29-Nov-2011)
Filesystem at /dev/mmcblk0p3 is mounted on /; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/mmcblk0p3 is now 1926423 blocks long.
```

block count는 partition의 크기에 따라 달라지므로 다를 수 있습니다.

유틸리티에서 알 수 있듯이 크기 조정은 활발하게 사용되는 file system에서 발생합니다. 중간에 전원이 손실되지 않는 한 안전합니다.

결과는 즉시 적용됩니다. 재부팅할 필요가 없습니다.

8 GB microSD 카드를 사용하는 일반적인 세션:

```
# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.1G  1.9G  5.0G  28% /
devtmpfs        505M  4.0K  505M   1% /dev
none            101M  736K  101M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            505M     0  505M   0% /run/shm
```

“df -h” 유틸리티가 제공하는 크기는 1 GiB = 2^{30} bytes로, 10^9 bytes의 Gigabyte보다 7.3%가 더 큽니다. 이것이 8 GB 카드가 위의 7.1 GiB로 나타나는 이유입니다.

4.5.2 원격 SSH 액세스 허용

보드에 ssh 서버를 설치하려면 보드를 인터넷에 연결하고 shell prompt에

```
# apt-get install ssh-server
```

를 입력합니다. root 암호는 기본적으로 없음이며 ssh는 암호가 없는 사람의 로그인을 정당하게 거부합니다.

이를 수정하려면 shell prompt에서 다음 명령으로 root password를 설정하십시오.

```
# passwd
```

SSH 서버를 설치한 후 /etc/ssh/sshd_config 하단에 다음 줄을 추가하는 것이 좋습니다.

```
UseDNS no
```

이렇게 하면 SSH 서버에서 수행하는 다소 의미 없는 역 DNS 검사가 해제되어 세션 시작을 시도할 때 지연이 발생합니다.

4.6 데스크탑 사용

Xillinux 데스크탑은 Ubuntu 데스크탑과 같습니다. microSD 카드의 상대적으로 낮은 데이터 대역폭으로 인해 애플리케이션은 다소 느리게 로드되지만 데스크탑 자체는 상당히 응답합니다.

데스크탑 환경에서 응용 프로그램을 실행하려면 데스크탑의 왼쪽 상단 Ubuntu 아이콘(“Dash home”)을 클릭하고 원하는 응용 프로그램의 이름을 입력합니다(예: shell prompt terminal 창의 경우 “terminal”, gedit 텍스트 편집기의 경우 “edit”).

Ubuntu 배포판과 마찬가지로 추가 패키지를 “apt-get”와 함께 설치할 수 있습니다.

4.7 종료

시스템 전원을 끄려면 바탕 화면의 오른쪽 상단 아이콘을 선택하고 “Shut Down...”을 클릭합니다. 또는 shell prompt에

```
# shutdown -h now
```

“System Halted”라는 텍스트 메시지가 UART console에 나타나면 보드의 전원을 끄는 것이 안전합니다. 또는 cursor가 VGA 화면의 텍스트 console에서 깜박임을 멈추면 Linux가 종료되었다는 신호이기도 합니다.

4.8 여기에서 무엇을해야합니까

SoCKit 보드는 이제 모든 용도로 Linux를 실행하는 컴퓨터가 되었습니다. Xillybus IP core를 통해 logic fabric와 상호 작용하는 기본 단계는 Getting started with Xillybus on a Linux host에서 찾을 수 있습니다. Xillybus용 driver는 이미 Xillinux 배포판에 설치되어 있으므로 가이드에서 설치를 다루는 부분은 건너뛸 수 있습니다.

5.1 단락은 응용 프로그램별 logic을 Linux 운영 체제와 통합하는 것을 나타냅니다.

Xillinux에는 gcc compiler 및 GNU make이 포함되어 있으므로 host 애플리케이션은 보드의 processors에서 직접 compiled가 될 수 있습니다. apt-get을 사용하여 배포판에 추가 패키지를 추가할 수도 있습니다.

5

수정

5.1 맞춤형 logic와 통합

중요한:

FPGA design이 demo bundle와 함께 제공된 파일이 아닌 다른 QSF 파일로 빌드된 경우 자세한 내용은 5.3 단락을 참조하십시오.

Xillinux 배포판은 application logic와 쉽게 통합되도록 설정되었습니다. 데이터 소스와 sinks를 연결하기 위한 프런트 엔드는 xillydemo.v 또는 xillydemo.vhd 파일입니다(선호하는 언어에 따라 다름). boot image 키트의 다른 모든 HDL 파일은 Linux host와 logic fabric 간의 데이터 전송으로 Xillybus IP core를 사용하기 위해 무시할 수 있습니다.

사용자 지정 logic designs가 있는 추가 HDL 파일을 3.4 단락에서 처리된 프로젝트에 추가한 다음 처음에 수행한 것과 동일한 방식으로 다시 빌드할 수 있습니다. 업데이트된 logic이 있는 시스템의 boot의 경우 soc_system.rbf를 3.4 단락에 설명된 대로 재생성하고 3.6 단락에 설명된 대로 microSD 카드에 기록해야 합니다. 초기 배포 배포의 다른 단계를 반복할 필요가 없으므로 logic의 개발 주기는 상당히 빠르고 간단합니다.

Xillybus IP core를 사용자 지정 application logic에 연결할 때 FIFOs를 통해서만 Xillybus IP core와 상호 작용하고 logic에서 FIFO의 동작을 모방하려고 시도하지 않는 것이 좋습니다. 최소한 첫 번째 단계에서는 아닙니다.

이에 대한 예외는 메모리 또는 register 어레이를 Xillybus에 연결할 때이며, 이 경우 xillydemo 모듈에 표시된 방법을 따라야 합니다.

xillydemo 모듈에서 FIFOs는 host에서 다시 도착하는 데이터를 루프백하는 데 사용됩니다. FIFOs의 양쪽 측면은 Xillybus IP core에 연결되어 있어 core가 자체 데이터 소스 및 sink로 작동합니다.

더 유용한 시나리오에서는 FIFO의 한쪽 끝만 Xillybus IP core에 연결되고 다른 쪽 끝은 애

플리케이션 데이터 소스 또는 sink에 연결됩니다.

xillydemo 모듈에 사용되는 FIFOs는 양쪽 모두 Xillybus의 메인 clock로 구동되기 때문에 양쪽에 대해 하나의 공통 clock만 수용합니다. 실제 응용 프로그램에서는 읽기 및 쓰기를 위한 별도의 clocks가 있는 FIFOs로 교체하는 것이 바람직할 수 있으므로 데이터 소스와 sinks가 bus clock이 아닌 clock에 의해 구동될 수 있습니다. 이렇게 함으로써 FIFOs는 중재자 역할을 할 뿐만 아니라 적절한 clock domain crossing을 위한 역할도 합니다.

Xillybus IP core는 FPGA에서 host까지의 streams에 대해 *일반* FIFO 인터페이스(First Word Fall Through와 반대)를 예상합니다.

다음 문서는 맞춤형 logic 통합과 관련이 있습니다.

- logic design용 API: Xillybus FPGA designer's guide
- Linux host의 기본 개념: Getting started with Xillybus on a Linux host
- 프로그래밍 애플리케이션: Xillybus host application programming guide for Linux
- 맞춤형 Xillybus IP core 요청: The guide to defining a custom Xillybus IP core

5.2 다른 보드 사용

SoCKit 보드 이외의 보드에서 Xillinux를 실행하기 전에 특정 수정이 필요할 수 있습니다.

무엇보다도 여기에는 핀 및 clocks 교체가 포함됩니다.

- xillybus.v에 정의된 clock PLL의 속성은 50 MHz(및 그에 따라 SDC 파일이 업데이트됨)가 아닌 경우 clk_bot1에 연결된 자유 실행 clock와 일치하도록 설정해야 합니다.
- VGA 출력은 의도한 보드와 일치해야 합니다.
- HPS' 멀티플렉스 핀: ARM core에는 고정 배치로 칩의 물리적 핀으로 라우팅되는 I/O 핀이 있습니다. ARM core는 Qsys에서 이러한 핀에 특정 역할을 할당하도록 구성되어 있습니다(예: USB 인터페이스, Ethernet 등). 이는 보드에서 이러한 핀이 연결된 것과 일치해야 합니다.

후자의 문제는 중요한 하드웨어 기능이 실패할 수 있을 뿐만 아니라 보드의 잘못된 물리적 신호 조건으로 인해 하드웨어가 손상될 수 있기 때문에 중요합니다(실제 손상은 매우 드물지만).

HPS 핀 할당의 불일치는 세 단계로 수정됩니다.

- Qsys에서 이러한 할당을 수정합니다.

- 전체 프로젝트(FPGA 부분 포함)를 빌드하고 새 handoff 파일을 기반으로 업데이트된 boot loader를 생성합니다(processor에서 관련 registers를 설정하는 U-boot의 SPL 부분).
- 장치 할당과 일치하도록 DTS 파일을 업데이트하고 이 파일의 compilation을 수행하여 시스템의 boot를 수행할 DTB 파일을 생성합니다.

5.3 사용자 정의 빌드 프로젝트 및 preflow.tcl

demo bundle의 소스가 다른 Quartus II 프로젝트에 채택된 경우 QSF 파일 간에 정보를 전송하는 일반적인 방법이 적용됩니다. demo bundle의 soc_system 하위 디렉토리에 있는 비표준 script, preflow.tcl을 처리하려면 각별한 주의가 필요합니다.

script의 주요 목적은 상호 연결을 구현하는 버그가 있는 AXI bus logic을 우회하기 위해 Qsys에 의해 생성된 SoC 인프라의 toplevel module을 다시 배선하는 것입니다. 이 재배선이 없으면 AXI bus를 통한 데이터 전송이 제대로 작동하는 것처럼 보일 수 있지만 산발적인 데이터 손상이 관찰됩니다.

script는 compilation이 발생하기 전에 Quartus II에 의해 자동으로 실행됩니다. 이것은 xillydemo.qsf에서 다음 행의 결과입니다.

```
set_global_assignment -name PRE_FLOW_SCRIPT_FILE \
                  quartus_sh:../soc_system/preflow.tcl
```

Xillybus 기반 FPGA design이 다시 배선되지 않은 인프라를 실수로 사용하지 않도록 하기 위해 preflow.tcl은 일부 top-level 포트의 이름도 수정하여 원본과 호환되지 않도록 합니다. 해당 포트는 xillybus_0_conduit_*로 명명된 포트입니다.

모듈이 script에 의해 수정되었는지 쉽게 알 수 있습니다. 수정된 파일의 첫 번째 줄인 soc_system/soc_system/synthesis/soc_system.v에는

```
// soc_system.v (mangled by preflow.tcl)
```

파일이 실제로 재연결된 경우.

사용자 정의 프로젝트에 SoC 인프라를 포함하려면 이미 완전히 구축된 demo bundle에서 파일을 복사하여 전체 SoC 모듈 트리를 채택하고 QSF 파일에 다음 행을 추가하십시오.

```
set_global_assignment -name QIP_FILE path/to/soc_system.qip
```

5.4 Qsys 프로젝트의 변경 사항

Qsys에 의해 생성된 Verilog 파일 중 하나의 자동 수정으로 인해 Qsys 프로젝트를 수정하지 않는 것이 좋습니다. 특히 프로젝트의 토폴로지가 변경된 경우 수정된 Qsys 프로젝트가 script에 의해 올바르게 수정된다는 보장은 없습니다.

processor의 핀 다중화를 수정하는 것과 같이 요소의 속성에 대한 변경은 아마도 괜찮을 것입니다. 그러나 HPS processor의 속성이 변경되면 logic이 아니라 프리로더에 포함된 C 파일의 변경을 통해 적용됩니다.

6

문제 해결

6.1 포트 " xillybus_0_conduit_..."가 존재하지 않습니다.

Quartus II가 있는 Xillydemo 프로젝트의 compilation 중에 다음과 같은 오류 메시지가 나타날 수 있습니다.

```
Error (12002): Port "xillybus_0_conduit_M_AXI_ARADDR" does not exist in macrofunction "u0" File: xillybus.v Line: 442
Error (12002): Port "xillybus_0_conduit_M_AXI_ARBURST" does not exist in macrofunction "u0" File: xillybus.v Line: 442
Error (12002): Port "xillybus_0_conduit_M_AXI_ARCACHE" does not exist in macrofunction "u0" File: xillybus.v Line: 442
Error (12002): Port "xillybus_0_conduit_M_AXI_ARID" does not exist in macrofunction "u0" File: xillybus.v Line: 442
Error (12002): Port "xillybus_0_conduit_M_AXI_ARLEN" does not exist in macrofunction "u0" File: xillybus.v Line: 442
Error (12002): Port "xillybus_0_conduit_M_AXI_ARLOCK" does not exist in macrofunction "u0" File: xillybus.v Line: 442
Error (12002): Port "xillybus_0_conduit_M_AXI_ARPROT" does not exist in macrofunction "u0" File: xillybus.v Line: 442
Error (12002): Port "xillybus_0_conduit_M_AXI_ARREADY" does not exist in macrofunction "u0" File: xillybus.v Line: 442
Error (12002): Port "xillybus_0_conduit_M_AXI_ARSIZE" does not exist in macrofunction "u0" File: xillybus.v Line: 442
...
```

포트를 찾지 못한 것은 아마도 compilation 이전에 실행되어야 했던 script가 실행되지 않았음을 나타냅니다. 이는 QSF 파일의 잘못된 변경 또는 사용자 지정 Quartus II 프로젝트에 소스를 부적절하게 채택한 결과일 수 있습니다. 자세한 내용은 5.3 단락을 참조하십시오.

6.2 USB 키보드 및 마우스 문제

거의 모든 USB 키보드와 마우스는 호환 동작에 대한 표준 사양을 충족하므로 인식되지 않는 장치에서 문제가 발생할 가능성은 거의 없습니다. 문제가 발생하면 가장 먼저 확인해야 할 사항은 다음과 같습니다.

- Linux가 boot를 수행할 때 장치가 연결되었습니까? 그렇지 않은 경우 연결된 마우스 및/또는 키보드로 Linux를 재부팅하십시오.
- 올바른 USB 플러그를 사용하고 있습니까? 중간에 “HPS USB”로 표시된 것이어야 합니다.

- USB hub을 사용하는 경우 키보드나 마우스만 SoCKit 보드의 OTG 포트로 가는 USB 케이블에 직접 연결해 보십시오.

일반 시스템 로그 파일 /var/log/syslog에 유용한 정보가 있을 수 있습니다. “less /var/log/syslog”로 콘텐츠를 보는 것은 때때로 도움이 될 수 있습니다. 더욱이 “tail -f /var/log/syslog”을 입력하면 새 메시지가 도착할 때 console에 새 메시지가 덤프됩니다. 이는 특히 USB bus의 이벤트가 감지된 내용과 이벤트 처리 방법에 대한 자세한 설명을 포함하여 이 로그에 항상 기록되기 때문에 유용합니다.

shell prompt는 USB UART를 통해서도 액세스할 수 있으므로 키보드 연결에 실패하면 serial terminal로 로그를 볼 수 있습니다. UART 링크를 설정하는 방법은 SoCKit 보드 설명서를 참조하십시오.

6.3 File system 마운트 문제

경험에 따르면 적절한 microSD 카드를 사용하고 보드의 전원을 끄기 전에 시스템을 제대로 종료하면 영구 저장 장치에 전혀 문제가 없습니다.

ext4 file system이 다음 mount에서 journal로 자체 수리하기 때문에 root file system을 마운트 해제하지 않고 보드 전원을 끄면 file system 자체에서 영구적인 불일치가 발생할 가능성이 없습니다. 그러나 전원이 꺼졌을 때 쓰기 위해 열린 파일이 잘못된 내용으로 남거나 완전히 삭제될 수 있기 때문에 운영 체제의 기능에 누적 손상이 있습니다. 이것은 갑자기 전원이 꺼진 컴퓨터의 경우에도 마찬가지입니다.

root file system이 마운트되지 않거나(boot 동안 kernel panic 발생) mount 읽기 전용을 수행하는 경우 가장 가능성이 높은 원인은 낮은 품질의 microSD 카드입니다. 이러한 저장 소가 잠시 동안 제대로 작동한 후 임의의 오류 메시지가 나타나기 시작하는 것은 매우 일반적입니다. /var/log/syslog에 다음과 같은 메시지가 포함되어 있으면 (Micro)SD 카드가 원인일 가능성이 큽니다.

```
EXT4-fs (mmcblk0p2): warning: mounting fs with errors, running ec2fsck
   is recommended
```

이러한 문제를 방지하려면 Sandisk 장치를 고집하십시오.

6.4 “startx” 실패(그래픽 데스크탑이 시작되지 않음)

직접적인 관련은 없지만 microSD 카드가 Sandisk에서 제작되지 않은 경우 이 문제가 자주 보고됩니다. 그래픽 소프트웨어는 시작할 때 카드에서 많은 양의 데이터를 읽으므로 읽기 오류를 생성하는 microSD 카드의 주목할만한 희생자가 될 수 있습니다.

확실한 해결책은 Sandisk microSD 카드를 사용하는 것입니다.