Xilinx 버스 매크로를 이용한 동적 부분 재구성 가능한 디자인 설계

유 명 근, 이 재 진, 송 기 용
충북대학교 컴퓨터공학과

Implementation of a Dynamic Partial Reconfigurable Design using Xilinx Bus Macro

Myoung-Keun You, Jae-Jin Lee, Gi-Yong Song
{mkyou77, ceicarus}@chungbuk.ac.kr, gysong@chungbuk.ac.kr
Dept. of Computer Engineering, Chungbuk National University

요 약

동적 부분 재구성은 FPGA 칩에 구현된 디자인에서 변경이 필요한 부분만 재구성하여 줌으로써 실시간적 재구성을 가능하게하는 방법이다. 동적 부분 재구성에 대한 많은 연구를 통하여 게이트 수준의 부분 재구성이 가능하지만, 설계 복잡도가 큰 시스템을 설계시에 게이트 수준의 부분 재구성 방법은 부적절하다. 본 논문에서는 Xilinx에서 제공하는 버스 매크로를 사용하여 모듈 기반의 부분 재구성 기법에 대하여 기술하며, 곱셈기, 레지스터, 그리고 ripple carry adder로 구성된 회로에서 ripple carry adder를 carry lookahead adder로 재구성한다.

I. 서 론

기존의 하드웨어 설계는 칩에 구현할 전체적인 디자인을 설계한 후 생성된 비트 스트림 파일을 칩에 다운로드하여 칩을 구현하는 방법이다. 만약 설계한 디자인의 일부 회로만이 재구성을 필요로 할 때, 기존의 방법은 전체회로를 재설계하여 비트 스트림 파일을 생성한 후 칩에 다운로드한다. 이는 간단한 디자인 설계시 재설계 및 전체 디자인의 비트 스트림 파일을 생성하는데 필요한 시간은 크지 않지만, 복잡한 시스템 레벨의 회로인 경우는 상당한 시간을 필요로 하게 된다. 동적 부분 재구성 방법은 FPGA의 내부의 구조를 칩의 동작 중에도 변경을 할 수 있는 기능을 추가 함으로써 하드웨어 설계의 유용성을 제공한다[1-4].

동적 부분 재구성에 대한 많은 연구가 진행되어 왔으나, 대부분의 동적 부분 재구성 기법들은 게이트 수준의 부분 재구성을 수행한다. 그러나 일반적으로 설계 복잡도가 큰 시스템 수준의 디자인 설계에서는 이를 적용하기가 매우 어렵다. 하지만 최근 Xilinx에서 제공하는 모듈 기반의 부분 재구성 기법은 게이트 수준이 아닌 모듈 단위의 재구성이 가능하기 때문에, 전체 디자인에서 임의의 모듈을 재구성할 시 재구성할 모듈만 수정한 후 비트 스트림 파일을 생성하여 칩에 다운로드하면 된다. 이는 디자인을 구성하고 있는 모듈간의 상호 통신이 Xilinx에서 제공하는 버스 매크로를 통하여 이루어지며 버스 매크로 및 핀 제약 사항을 top 모듈의 .ucf 파일에 정의하기 때문에 가능하다. 이와 같이 상위 레벨에서의 재구성이 가능함으로 복잡도에 상관없이 가변적인 시스템을 FPGA에 구현할 수 있게 된다.

본 논문에서는 버스 매크로를 이용하여 곱셈기, 레지스터, 그리고 ripple carry adder로 구성된 회로에서 ripple carry adder를 carry lookahead adder로 재구성한다.

II. Modular Design

Modular design[1-2]은 다수의 하위 모듈을 병렬적으로 설계한 후, 설계한 모듈들을 통합하여 하나의 FPGA 디자인을 설계하는 방법이다. 이는 하위 모듈들의 병렬적 설계가 이루어져 시스템 구현 시간을 줄일 뿐만 아니라 각각의 하위 모듈들을 단일 프로젝트에서 제작하게 되므로 디버깅에도 매우 효과적이다. Modular design 흐름도를 그림 1에 보인다. Initial budget 단계에서 상위 레벨 설계자 의하여 전체적인 시스템에서의 각 모듈에 대한 영역 지정 및 입출력 포트의 지정 등의 설계 조건이 주어지게 된다. 이 조건을 바탕으로 active module 구현 단계에서 각 모듈의 설계자들은 개별 프로젝트의 모듈 코딩[6-7], 합성, 매핑, P&R을 실행하게 된다. 그리고 각 모듈의 P&R 후 생성된 모듈의 .ncd 파일들을 final assembly 단계에서 통합하여 하나의 시스템을 완성하게 된다.

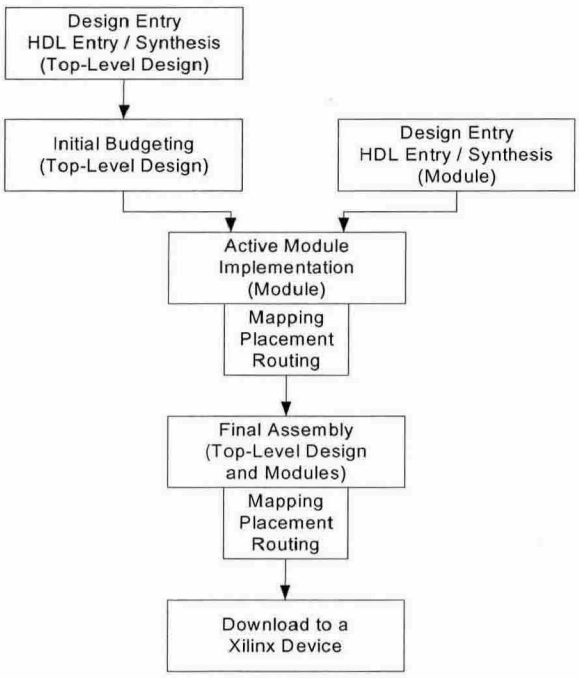

그림 1. Modular Design 흐름도

III. 모듈 기반 동적 재구성

모듈 기반의 동적 부분 재구성 방법[1-2]은 Modular Design에서 각각의 모듈은 자체적인 비트 스트림을 생성할 수 있다는 점에서 착안한 방법으로 그림 2와 같은 흐름도를 가진다.

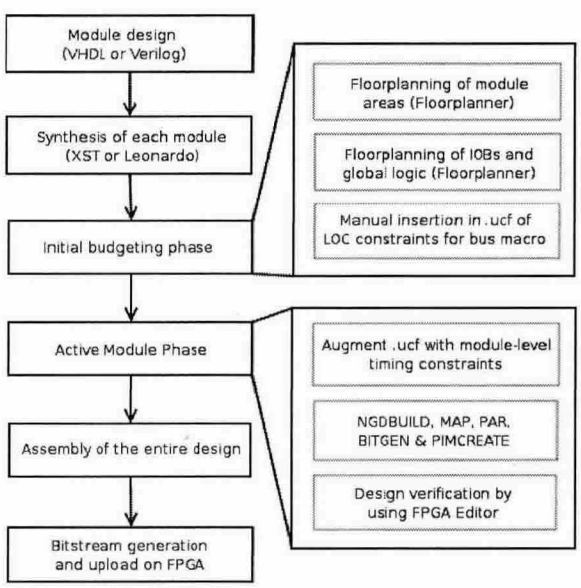

그림 2. 동적 부분 재구성의 흐름도

기본적인 구조는 Modular Design 방법과 동일하지만 다음과 같은 차이점을 보이고 있다.

1. 버스 매크로

모듈 기반의 부분 재구성 방법을 이용할 경우 시스템의 동작 중에 일부 모듈을 교체하는 것이 주목적이기 때문에 각 모듈간의 안정적인 연결을 위한 특수한 연결부분이 필요하다. 버스 매크로는 두개의 모듈 간의 상호 통신을 위해 사용되며, 3-state 버퍼로 구현된다. Xilinx에서 제공되는 버스 매크로는 하드 매크로로서 모듈의 변화에 상관없이 .ucf 파일에서 지정한 위치에 고정적으로 라우팅된다. 버스 매크로는 그림 3에 보이는 것과 같이 5개의 입출력 신호을 가지며 right-to-left 또는 left-to-right 한 방향 통신만을 가능하게 한다. 그러므로 버스 매크로와 각 모듈의 입출력 신호를 P&R 시에 적절하게 연결하여 주게 되면 회로의 재구성시에도 안정적인 연결을 보장 받을 수 있게 된다.

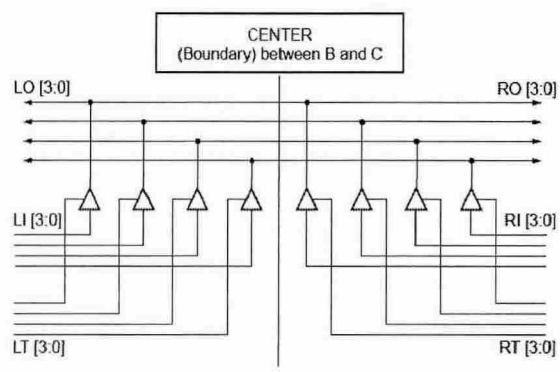

그림 3. 버스 매크로 구조

2. 사용자 지정 제한 파일(.ucf)

모듈 기반 동적 재구성 방법은 각 하위 모듈들이 FPGA 칩 상의 구현될 영역, 모듈의 입출력 핀, 그리고 버스 매크로의 위치을 지정하게 된다. 각 하위 모듈의 영역 지정에서 열은 최대 높이로, 행은 4의 배수로 영역을 지정해 주어야 한다.

하나의 버스 매크로는 4비트의 정보를 전달할 수 있으며, 한 행의 8열 섹션의 3-state 버퍼를 차지한다.

3. 개별 비트 스트림의 생성

기존의 설계 방법이나 modular design은 하나의 전체 시스템에 대한 비트 스트림을 최종적 생성하는 반면, 동적 재구성 방법은 재구성이 필요한 모듈만을 칩에 다운로드하는 방식이기 때문에 개별적으로 비트 스트림을 생성한다. 그러나 처음으로 칩에 시스템을 구현하는 단계에서는 개별적으로 생성된 비트 스트림을 통합하여 전체적인 시스템의 비트 스트림을 생성하여야 한다.

IV. 동적 부분 재구성 가능한 디자인 설계

동적 부분 재구성 방법을 사용하기 위해서는 Xilinx 명령어

를 DOS 창에서 실행시키는 단계를 수행한다. 명령어 중 일부는 Xilinx GUI의 기능을 기초로 한다.

본 절에서는 곱셈기, 레지스터 그리고 ripple carry adder로 구성된 회로를 설계한 후, ripple carry adder 부분을 carry lookahead adder로 Spartan2(xc2s200-pq208) Xilinx FPGA 칩[3-4]에 재구성한다. 설계할 디자인의 구성은 그림 4와 같으며, 디자인을 위한 프로젝트 디렉토리는 그림 5와 같은 구조를 가진다.

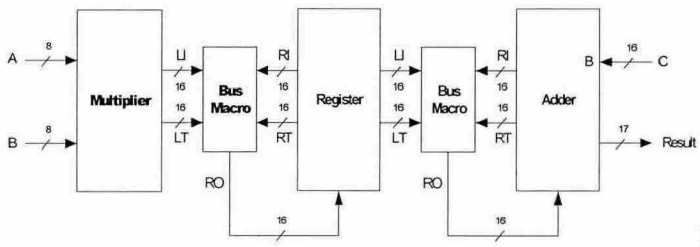

그림 4. 설계할 디자인 블록 다이어그램

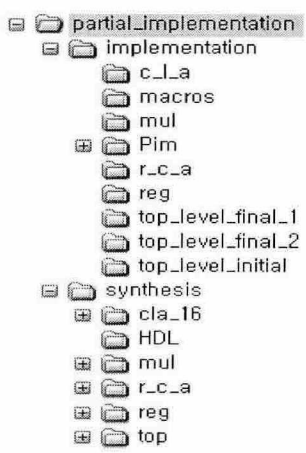

그림 5. 프로젝트 디렉토리 구조

프로젝트 디렉토리는 크게 implementation과 synthesis 디렉토리를 가진다. Synthesis 디렉토리의 하위 디렉토리는 active module 단계에서 각 모듈에 대하여 개별적인 프로젝트를 Xilinx ISE를 이용하여 생성한 것이다. Implementation 디렉토리의 하위 디렉토리는 각 모듈의 네트 리스트 정보를 포함한 .ngc 파일과 top 모듈의 .ucf 파일을 사용하여 Xilinx 명령어를 이용한 매핑과 P&R 단계를 수행한 결과파일을 포함하고 있다. 각 모듈에 대하여 합성시 top 모듈을 제외한 나머지 모듈에 대하여서는 합성 옵션에서 '*Add I/O Buffers*' 옵션을 제거한 후 합성을 한다.

Initial budget 단계에서 floorplanner 명령을 실행한 후 GUI를 이용하여 디자인을 구성하는 모듈의 영역 지정과 top 모듈에 대한 입출력 핀을 지정하여 .ucf 파일을 생성한다[2-3,5]. 생성된 .ucf 파일에 에디터를 이용하여 버스 매크로 위치를 다음과 같은 형태로 추가한 후 floorplanner를 이용하여 최종적으로 작성한 .ucf 파일을 보면 그림 6와 같다.

INST "busMulToReg_bus1" LOC="TBUF_R11C12.0";

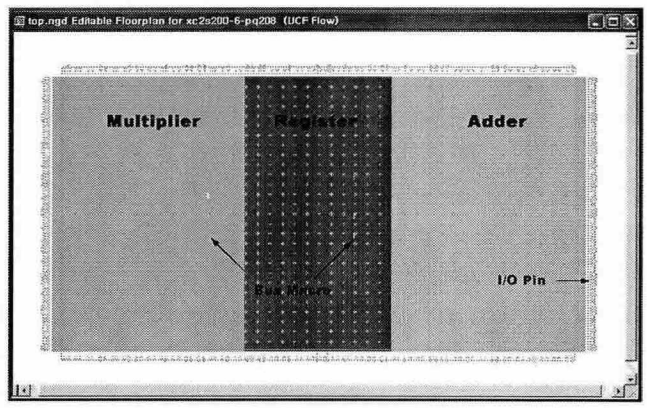

그림 6. floorplanner으로 .ucf 파일 보기

Active module 단계에서는 앞서 언급한 것과 같이 설계할 디자인의 하위 모듈들에 대하여 합성단계에서 생성한 .ngc 파일과 initial budget 단계에서 생성한 .ucf 파일을 사용하여 Xilinx 명령어를 이용한 매핑과 P&R 단계를 수행한다. 이 단계를 거치면 각 하위 모듈들은 각 모듈에 대한 부분적인 비트스트림을 생성할 수 있다. 본 논문에서 설계한 디자인의 하위 모듈인 곱셈기, 레지스터 그리고 ripple carry adder의 P&R 후의 결과를 FPGA Editor로 본 스케메틱을 그림 7, 그림 8, 그림 9에 각각 보인다. 각각의 모듈에 대해 P&R한 정보를 통합하여 디자인의 전체적인 P&R 후의 결과를 그림 10에 보인다.

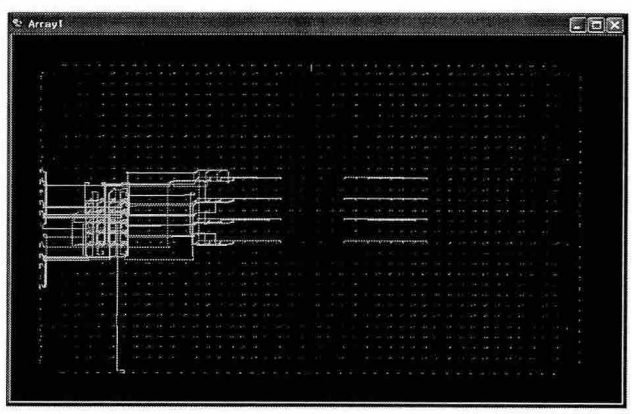

그림 7. 곱셈기의 구현 스케메틱

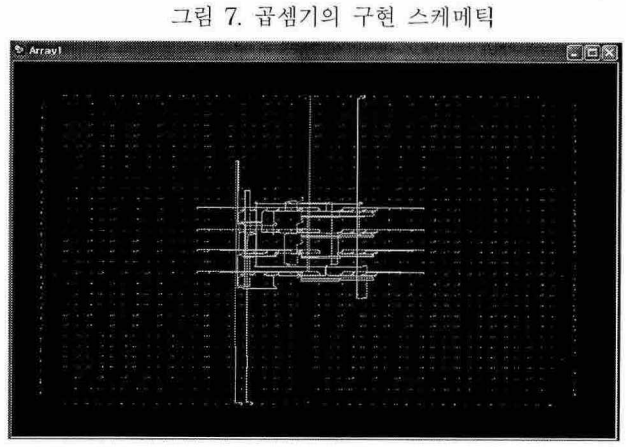

그림 8. 레지스터 구현 스케메틱

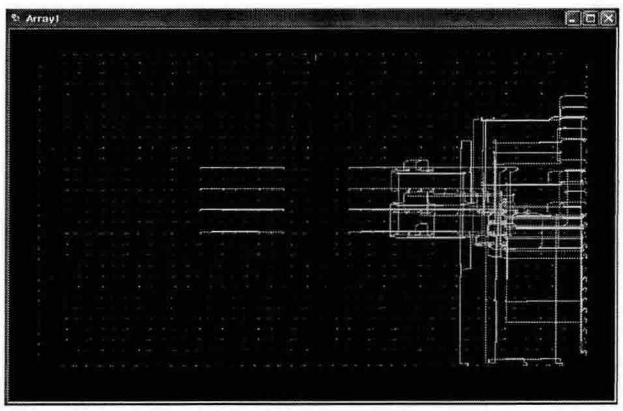

그림 9. ripple carry adder의 구현 스케메틱

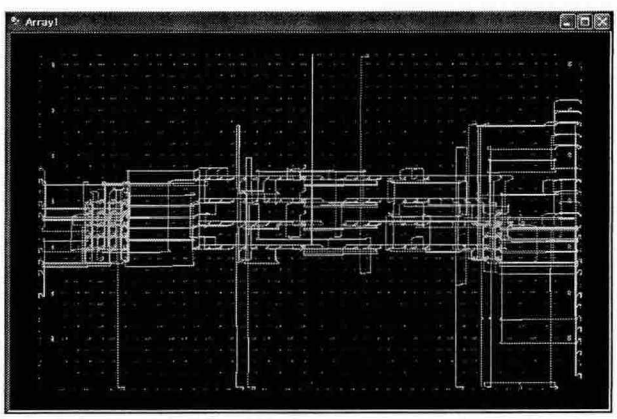

그림 10. 설계한 디자인의 구현 스케메틱

그림 10은 그림 4의 디자인 블록 다이어그램에서 덧셈기 부분을 ripple carry adder로 설계한 구현 스케메틱이다. 덧셈기를 carry lookahead adder로 부분 재구성한 구현 스케메틱을 그림 11에 보인다. 입출력 핀, 곱셈기, 그리고 레지스터 부분은 그림 10과 동일하며 덧셈기 부분만이 달라졌음을 확인할 수 있다.

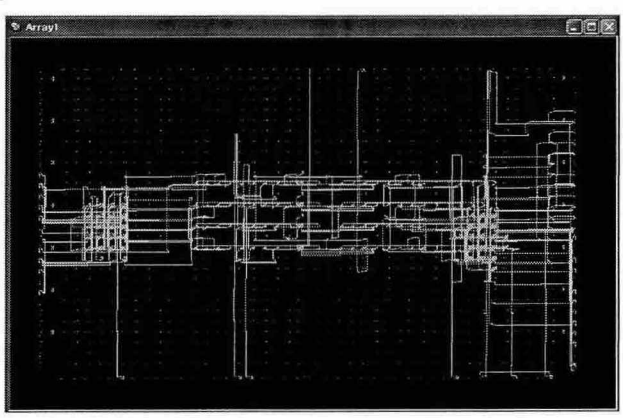

그림 11. carry lookahead adder로 재구성한 스케메틱

V. 결론

본 논문에서는 Xilinx Spartan2(xc2s200-pq208) FPGA 칩을 사용한 동적 부분 재구성에 대하여 기술하였다. 디자인은 곱셈기, 레지스터 그리고 덧셈기로 모듈화하였고 각각 모듈은 Xilinx에서 제공하는 버스 매크로를 이용하여 상호통신한다.

동적 부분 재구성을 위해 덧셈기 부분을 ripple carry adder로 설계한 후, carry lookahead adder로 재구성하였다. 그림 10과 그림 11에서 디자인의 입출력 핀, 곱셈기, 그리고 레지스터 부분은 동일하지만 덧셈기 부분만이 달라졌음을통하여 재구성됨을 확인할 수 있다.

참 고 문 헌

[1] *Development System Reference Guide*, Xilinx Inc., 2005

[2] *Two Flow for Partial Reconfiguartion : Module Based or Differnce Based*, Xilinx XAPP290, 2004

[3] *The Programmable Logic Data Book*, Xilinx, Inc., 1999

[4] Xilinx Inc., http://www.xilinx.com

[5] 김혁, *Real Xilinx FPGA World*, 엔트미디어, 2004

[6] K.C Chang, *Digital Systems Design with VHDL and Synthesis*, IEEE Computer Society Press, 1999.

[7] Ben Cohen, *VHDL Coding Styles and Methodologies*, Kluwer Academic Publishers, 1999