동적 재구성가능 DES의 설계 및 검증
(Design and Verification of Dynamically Reconfigurable DES)

안 민 희 † 양 세 양 †† 윤 재 근 †
(Minhee Ahn) (Seiyang Yang) (Jaegeun Yun)

요 약 최근까지 초고집적 FPGA 혹은 재구성가능 프로세서 들을 이용한 RC(재구성 컴퓨팅) 기술에 대한 많은 연구가 진행되어 왔으며, 최근 들어서는 이와 같은 RC 기술을 응용분야에 실제 적용한 성공적인 상용화 사례들이 보고되고 있다. 본 논문에서는 FPGA의 동적 재구성 기능과 RC 기법을 이용하여 DES 암호화 시스템을 적은 용량의 FPGA에 구현하기 위한 설계와 구현된 DES 암호화 시스템의 시스템 수준 검증 기법을 제안한다. 이를 통하여 동적 재구성 기반의 접근법이 가지는 유용성을 평가할 수 있었는데, 그것은 FPGA의 동적 재구성성을 통하여 임의의 알고리즘의 RC 기법에 의한 하드웨어 구현에 있어서 성능과 가격간의 타협이 매우 효과적으로 이루어 질 수 있다는 것이다.

키워드 : 시스템 설계, 현장프로그래밍가능게이트어레이

Abstract Recently, many researches on RC(Reconfigurable Computing) with highly complex FPGA's and reconfigurable processors have been reported, and even some attempts for commercialization have been successful. In this paper, we introduce the design methodology for implementing DES crypto algorithm on small-capacity FPGA by using its dynamic reconfigurability and a system-level verification technique. Throughout this design project, we could evaluate the effectiveness of this approach, which is the dynamic reconfigurability of FPGAs makes the efficient trade-off between the performance and the cost robustly viable.

Key words : System design, FPGA

1. 서 론

최근의 컴퓨팅 환경에서 수행되는 여러 응용 태스크들은 우수한 가격대성능비를 만족하기 위하여 저렴한 비용으로 고속의 수행 기능을 가져야만 한다. 더구나 대부분의 경우에 있어서 여러 개의 응용 태스크들이 동시에 작업을 진행해야 하므로 제한된 컴퓨팅 파워를 가지고서는 각각의 수행 능력에는 분명한 한계가 존재하게 된다. 따라서 이러한 저렴한 비용을 가지면서도 고속의 결과를 얻기 위해서는 통상의 순수 소프트웨어적인 접근법만으로는 많은 문제점들이 존재한다. 현재 이에 대한 다양한 대안 중에서 제일 일반적인 방법은 ASIC (Application Specific Integrated Circuit)과 같은 고정된 전용 하드웨어로써 구현하는 것인데, 이로서 얻게 되는 이득은 순수 소프트웨어적인 구현에 비해 최대 수십만 배에 달하는 성능 향상이다. 그러나, 이러한 ASIC은 성능 면에서는 절대적으로 유리하지만 초기 개발 과정에서 많은 비용과 시간을 필요로 할 뿐만 아니라, 추가적인 수정 과정에서도 많은 비용과 시간을 필요로 한다.

최근들어 프로그래머블 하드웨어 기술을 이용한 RC(Reconfigurable Computing: 재구성 컴퓨팅) 기법이 소프트웨어와 같은 프로그래밍에 의한 변형 능력과 ASIC과 같은 하드웨어 특유의 미세 수준에서의 대규모 병렬성에 의한 고성능을 동시에 보장 받을 수 있는 잠재성으로 인하여 학계를 중심으로 이에 대한 많은 연구들이 진행되어 왔다[1,2]. 특히 최근의 메모리 집적기술을 이용한 SRAM 기반의 FPGA(Field Programmable Gate Array)는 이미 200MHz 의 속도로 동작이 가능한 수백만 게이트급의 단일 칩 소자가 사용되고 있으며 2003년 초에는 1000만 게이트급 FPGA가 출시될 예정으로 성능 및 집적도 면에서 큰 발전을 지속하고 있다[3]. RC는 이와 같은 최근의 초고집적 FPGA를 이용하거나 혹은 이와 같은 FPGA의 제조를 가능하게 하는

· 본 연구는 IDEC의 설계툴 지원과 부산대학교 연구보조비(4년과제)의 지원으로 이루어졌음
† 비 회 원 : 시스베리 부설연구소 연구원
tigoum@pusan.ac.kr
jaegeun@sysveri.com
†† 정 회 원 : 부산대학교 컴퓨터공학과 교수
syyang@pusan.ac.kr
논문접수 : 2002년 1월 12일
심사완료 : 2003년 6월 23일

초미세 VLSI 제조기술을 이용한 재구성가능 프로세서(reconfigurable processor)[1,2]를 이용하게 된다. 이와 같은 FPGA나 재구성가능 프로세서를 이용하게 되면 동작속도는 최소 수십 MHz에서부터 최대 200 Mhz까지를 가능하게 하면서도, 하드웨어 자체의 변형을 로직 또는 함수적 수준 혹은 인스트럭션 수준에서 가능하게 하여 가격대성능 면에서 특정 응용 태스크에 최적한 구현을 RC 기법으로 달성하는 것이 가능하다. 이와 같은 RC는 최근 들어 학계에서의 연구에 국한되지 않고 산업체에서의 상용 제품에 성공적으로 채용되는 단계에 접어들고 있다[4-7].

본 논문에서는 FPGA의 재구성 능력을 이용하여 동적재구성 기능을 실현할 수 있음을 논의하고, 이를 기반으로 임의의 알고리즘을 RC기법으로 구현하기 위한 일반적인 동적 재구성가능 시스템의 구조를 제시하고, 동적 재구성 방식의 장단점을 분석한다. 마지막으로 동적 재구성의 예로써 DES 암호 알고리즘을 동적 재구성(dynamically reconfigurable) 방식으로 구현하여 실제 성능을 검증해 본다.

2. 본 론

2.1 동적재구성

임의의 알고리즘은 일반적으로 다수의 응용 태스크의 집합이며, 이들 특정 응용 태스크들을 최상의 성능으로 수행하기 위해서는 특정 응용 태스크 각각을 개별적인 하드웨어 로직으로 구현하면 된다. 그러나, 이 경우 응용 태스크를 구현할 FPGA 등의 하드웨어 장비가 전체 응용 태스크의 개수 만큼 준비되어야 하므로 비용이 높아지며 회로의 검증등 수정과정에도 시간이 많이 들게 된다. 더욱이 전체 로직의 규모가 커지게 되면 해당 응용 태스크를 구현하는 FPGA도 더 큰 용량의 모델로 교체가 되어야 하기 때문에 비용의 증가가 비례적으로 발생하게 된다. 따라서 이와 같은 기존의 시스템으로는 구현 비용의 감소, 검증시간의 단축, 향후 개선비용의 절감과 같은 핵심 요건을 충족시킬 수 없음이 자명하고, 이들 핵심 요구사항을 만족시키기 위해 어떠한 시스템을 고려해야 할지 분석해보자.

구현비용의 감소는 하드웨어 장비의 규모를 축소하여 획득할 수 있다. 예를 들어, FPGA의 개수가 2개였다면, 이를 1개로 줄여서 구현하는 것이다. FPGA의 개수를 줄이기 위해서는 원래의 하드웨어 디자인을 재디자인(최적화)하여 크기를 줄이거나, 이후에 논의할 동적재구성을 위한 디자인분할을 수행하는 방법이 있다. 검증시간의 단축은 전체 하드웨어 디자인을 분할하여 모듈로 나누어 관리함으로써 실현된다. 예를 든다면, 대용량의 FPGA 한 개에 구현되는 디자인에서 오류가 발견되어 이것을 수정하여 컴파일(Compile)하고 PNR(Place and Routing)할 경우에 소요되는 총 시간보다는, 용량이 작은 여러 FPGA에 각각 구현되도록 분할된 디자인 중에서 오류가 발견된 모듈만 교정한다면 시간이 훨씬 줄어들 것이다. 개선비용의 절감은 하드웨어 시스템을 재구성가능 시스템으로 대체하는 것이다. 여기서 재구성가능 장비란 정적 재구성가능 시스템을 말하며, 시스템의 동작초기(전원인가 직후시점)에 구성이 완료되어 작동이 종료되는 시점까지 구성이 지속적으로 유지되는 체계이다. 만약 ASIC유형의 고정된 하드웨어 시스템을 이후에 개선하려고 한다면, 반드시 새로운 ASIC을 제작해야만 한다. 명백하게도 새로운 ASIC의 제작비용은 "개선"이라고 해서 저렴해지는 것이 아니므로 아예 새로운 시스템을 제작하는 것과 동일한 부담을 짊어져야 하는 것이다. 이에 정적 재구성가능 시스템을 도입하게 되면, 하드웨어 디자인이 바뀔지라도 하드웨어 시스템의 특별한 변경 없이 곧바로 적용하는 것이 가능하다. 따라서, 개선비용은 ASIC 제작비용에서 영(Zero)으로 감소하게 된다. 앞의 세 가지 요구를 모두 확보하려면, (정적 재구성가능 장비의 특성을 확장하여) 실시간으로 재구성을 운영할 수 있는 동적 재구성가능 시스템을 구축하면 된다. 동적 재구성은 실시간으로 특정 응용 태스크들을 하드웨어상의 구성을 교체해가며 수행하는 것이 가능한 시스템을 말한다. 실시간으로 응용 태스크들을 하드웨어상에서 구성을 바꿀 수 있기 때문에, FPGA의 개수를 초기의 요구량보다 적게 사용하더라도 모든 태스크들을 수행하는 것이 가능하고(구현비용 감소요인), 규모가 줄어든 FPGA 상에서 전체 태스크를 구동하기 위해 태스크를 분할하여 구동할 것이므로, 각 분할단위로 검증이 이루어지게 된다(검증시간 단축요인). 또, FPGA를 기반으로 하는 하드웨어 시스템이므로 차후에 태스크중 일부가 변경, 개선되더라도 하드웨어 시스템의 하드와이어를 재수정할 필요가 없게 된다(개선비용 절감요인).

따라서, 동적 재구성가능 시스템은 앞서 논의된 시스템의 성능, 비용, 시간 등의 제반 요구사항과 제약조건을 만족하는 가장 우수한 해법이 될 수 있다.

2.2 동적재구성을 위한 시스템의 요구사항

동적 재구성가능 시스템의 목표는 하드웨어 시스템의 제작비용은 최소화하면서도 응용 태스크의 수행성능을 가능한 최대로 높이는데 있다. 따라서, 적은 용량의 재구성 가능 FPGA를 이용하여 전체 응용 태스크들을 얼마나 많은 분할로 나눌 것인지 결정함에 있어서, 성능을 높이기 위해 FPGA를 몇 개나 쓸 것인가와 동적 재구성가능 시스템과 Host 시스템과의 연동에 대한 것을 함께 고려해야 한다.

동적 재구성가능 시스템의 운용에 있어서 가장 성능을 저해하는 요인은 FPGA를 재구성하는데 드는 시간이다. FPGA의 용량이 커짐에 따라 재구성(Configuration: Download)을 위해 요구되는 시간도 비례하여 커지며 용량이 적은 FPGA의 경우에도 재구성 방법에 따라 최악의 경우 수초에 달하는 시간이 소비되기도 한다. 따라서, 동적 재구성가능 시스템의 성능감소를 줄이기 위해서는 동적 재구성 횟수를 줄여야 한다. 하지만, 동적 재구성 횟수가 너무 줄게 되면 동적 재구성이 가지는 장점을 잃게 되므로, 소프트웨어적 구현과 정적 재구성가능 구현 사이에서 성능과 비용절감의 중재가 요구된다.

동적 재구성가능 시스템에서 동적 재구성을 수행하는 마스터가 반드시 필요한데, 동적 재구성을 어느 시점에 개시할 것인지를 판단할 능력을 겸비해야 한다. 또 동적 재구성을 위해 FPGA에 다운로드할 다수의 비트스트림(Bit-Stream)들을 유지해야 한다. 따라서, 마스터가 비트스트림을 유지하며, 동시에 동적 재구성가능 시스템의 재구성 및 구동을 관리해야 하는데, 기본적으로 여기에 적합한 것은 Host 시스템(PC 컴퓨터)를 사용하는 것이다. 물론, 최종적인 동적 재구성가능 시스템의 개발, 즉 독립기동을 위해서는 마스터를 동적 재구성가능 시스템상에 같이 장착해야 할 것이다. 앞서 언급한 바대로 동적 재구성가능 시스템의 성능에 가장 큰 영향을 미치는 요소는 FPGA를 재구성하는데 소비되는 시간이다. 재구성 시간은 FPGA가 가진 재구성 모드에 따라 상당한 차이를 나타내는데, 마스터를 사용하는 재구성 방식(이때 FPGA는 슬레이브(Slave))이 가장 느리며, FPGA가 인접한 메모리에서 비트스트림을 직접 읽어와서 스스로 재구성이 되는 방식이 가장 빠르다. 별도의 마스터를 두는 방식의 경우, 마스터가 빠르게 재구성 비트스트림을 FPGA에 공급할 수만 있다면, FPGA가 스스로 재구성되는 방식에 근접한 재구성속도를 실현할 수 있다. Host 시스템에서 비트스트림을 고속으로 전송할 수 있는 통신 프로토콜을 사용하고(예: PCI, USB 등), 동적 재구성가능 시스템상에서 이 프로토콜을 신속하게 처리하는 통신 모듈을 구성할 수 있게 되면, 재구성으로 인한 성능저하를 최소화 할 수 있다. 이외에 마스터가 추가로 수행해야 할 작업에는 각 FPGA 재구성에 대해 공급할 데이타의 스케쥴링, 실제 수행에 시간을 할당(예: Clock 개수), 동적 재구성가능 시스템의 상태점검과 고장진단 등이 있다. 전체 디자인을 동적 재구성가능 시스템의 FPGA에 다운로드 가능하도록 분할하였을 때, 각 분할된 모듈이 전체 디자인의 입력단(Primary Input)에서 출력단(Primary Output)까지의 데이타패쓰(Data-path)상의 어디에 해당하는 부분인가에 따라, FPGA에 다운로드되는 순서가 결정된다(입력단에 가까울수록 수행순서가 앞서게 된다). 수행이 완료된 모듈의 출력값은 다음 FPGA에 다운로드될 모듈의 입력으로 공급되어야 한다. 따라서 출력값은 다음 수행될 모듈에 입력으로 공급이 완료될 때까지 별도로 저장될 필요가 있고, 이것은 동적 재구성가능 시스템상에 중간결과값 저장용 메모리가 요구된다는 의미이다. 또 출력값을 샘플링하여 메모리에 저장하거나 다음 모듈의 입력으로 공급하기 위해 메모리에서 읽는 기능을 수행할 메모리 관리자도 필요하다. 메모리 관리자는 마스터(FPGA에 비트스트림을 다운로드하고, 실제 수행을 관리하는)에 의해 동작이 제어되는 슬레이브 요소이다.

2.3 동적재구성을 위한 Design 분석(Analysis)과 분할(Partition)

동적재구성가능 시스템상에서 구동될 디자인은 일정한 몇 개의 모듈로 분할되어야 한다. 전체 하드웨어 디자인을 분할함에 있어서 반드시 고려되어야 할 제약사항이 몇 가지가 있다. 분할된 모듈의 크기제한, 분할된 모듈 각각의 상대적인 크기, 분할된 모듈 각각의 상대적인 수행속도에 대한 고려가 그것이다.

첫째, 분할된 각 모듈의 크기들은 모두 시스템상의 재구성가능 FPGA의 최대 수용 용량을 넘어설 수 없다. 당연하게도 다운로드를 할 FPGA의 수용 용량보다 더 큰 모듈은 FPGA상에서 재구성을 할 수 없을 것이다. 둘째, 해당 디자인이 면적최소화라면, 분할된 각 모듈의 크기가 서로 최대한 비슷하도록 고려해야 한다. 만약 크기가 작은 모듈을 만들게 되면, 결국은 분할된 모듈의 개수가 증가하게 될 것이기 때문이다. 앞서도 논의한 바와 같이, 동적 재구성가능 시스템의 성능향상에 가장 큰 장애요인은 재구성가능 FPGA에 비트스트림을 다운로드하는 시간이며, 동적 재구성시 다운로드해야 할 비트스트림의 개수가 많아지면 다운로드의 횟수도 증가하게 되기 때문에 전체 성능을 그만큼 느리게 만들 우려가 있는 것이다. 셋째, 해당 디자인이 타이밍최적화라면, 분할된 각 모듈사이의 수행시간(예: Clock 개수) 차이가 전체 수행성능에 어떤 영향을 미치는지 분석해보아야 할 필요가 있다. 모듈의 개수가 n개라고 했을 때, 모든 모듈의 재구성가능 FPGA에 비트스트림을 다운로드 하는 시간을 제외한 순수 수행시간(예: Clock 개수)의 합은 분할하기 이전의 원래 디자인의 수행시간과 동일하다. 따라서, 총 순수 수행시간은 모듈의 개수나 모듈 개개의 수행속도 차이와 무관하고, 모듈의 분할에서 모듈 각각의 수행시간의 차이에 대한 고려는 제외되어도 됨을 알 수 있다. 그리고, 앞의 두 번째 고려 사항이었던 면적최소화형에 대한 모듈간 크기차이는 원래 디자인이 면적최소화형인가 타이밍최적화형인가와 상관없이 항상

고려해야 할 사항이다. 타이밍최적화형이라고 하더라도, 분할된 모듈의 개수가 증가하게 되면 그만큼 동적 재구성을 위해 FPGA에 다운로드해야 할 횟수가 증가하기 때문이다. 요약하면, 원래 디자인을 분할함에 있어서 담당 FPGA의 수용 용량에 대비한 분할된 모듈의 크기와 분할된 각 모듈간의 상대적인 크기차이 최소화라는 두 가지 제약사항을 고려하면 된다.

디자인을 분할할 때, 가능한한 레지스터를 경계로 하여 모듈을 나누는 방법이 이후의 동적 재구성 가능 시스템상에서의 관리에 유리하다. 레지스터를 경계로 분할이 이루어지게 되면, 모듈의 입력단과 출력단이 레지스터에 직접 연결되게 되므로 수행결과를 동적 재구성가능 시스템상의 메모리와 연계시키는 과정이 단순명료해지는데, 이유는 클럭에 맞추어 메모리에 읽고/쓰는 작업이 가능해지기 때문이다. 하지만, FPGA 용량의 한계로 레지스터를 경계로 분할을 할 수 없는 경우도 있다. 이때는 리타이밍 방법을 사용하여 모듈내부의 레지스터를 모듈의 경계쪽으로 뽑아내면 된다. 만약 리타이밍 방법을 쓸 수 없는 경우에는 모듈의 경계쪽에 레지스터를 추가하여 클럭을 1개 더 사용하면 된다.

2.4 동적재구성의 장점과 단점

동적재구성가능 시스템의 장점은 최소의 비용으로 하드웨어 시스템을 구성할 수 있다는 것이다. 이것은 높은 가격대성능비를 발생시키므로 고성능과 저비용의 잇점을 동시에 취하는 것이 가능해진다. 본 시스템의 특성인 재구성 능력을 활용할 수 있는 상황이 다양하다. 특히 개발완료 이후 단계인 유지, 보수 그리고 추후 확장을 할 때도 하드웨어 시스템을 변경없이(또는 변경비용을 최소로 하면서) 그대로 사용할 수 있으므로 생산성이 매우 높아지게 된다.

단점으로는 디자인의 분할은 컴파일 이전 단계인 소스수준에서 수행되므로 FPGA에 수용 가능한 크기로 분할이 되었는지 확인하기 위해서 컴파일과 PNR을 반드시 진행해 보아야 한다는 것이다. 만약 FPGA의 용량을 넘어섰다면, 소스단계에서 다시 재분할을 시행하고 컴파일과 PNR단계를 진행하는 작업을 반복해야 하므로 많은 시간을 소비해야 한다. 응용 태스크를 한꺼번에 모두 하드웨어 시스템상에 구현하는 것이 아니기 때문에 성능상의 감소요인이 있음을 감안해야 한다. 분할된 여러 모듈을 순차적으로 FPGA에 다운로드하면서 수행을 진행해야 하기 때문에 중간 결과를 계속 유지해야 한다는 것, 그리고 다운로드 시간이 전체 수행과정동안에 지속적으로 요구된다는 점이 성능저하의 주된 요인이다.

3. 동적재구성의 검증에: DES의 동적재구성 구현

DES 암호 알고리즘을 구현하여 소프트웨어적 구현에 비하여 큰 성능 향상을 얻으면서도 비용 부담을 최소화하고자 하는 본 설계 방법에서 가장 중요한 것은 동적 재구성을 이용하여 DES 암호 알고리즘의 하드웨어 설계를 등가의 디자인으로 분할(partition)하는 것이다. 또한, 현재의 FPGA기술로는 구성(configuration) 속도가 DES 알고리즘 수행 속도에 비해 매우 느리므로 최대한 재구성 횟수를 줄이는 것도 중요한 과제이다.

분할을 하기위해 DES의 16단계를 살펴 보면 첫 단계에는 데이타와 키를 교환(permutatation)하는 IP(initial permutater)와 PC1(permutated choice) 가 있고, 마지막 단계에는 IP^{-1}(invert permutation)가 있다. 나머지 단계는 동일한 형태로 되어 있고, 각 16 단계를 파이프라인하는 형식으로 동작한다. DES의 각 단계는 그림 1과 같이 IP, PC1, IP^{-1} 등이 모두 있다. 이러한 블록을 16개 연결시켜서 파이프라인하면 가장 좋은 성능을 낼 수 있는 시스템이 되겠지만 구현 시에 사용되는 하드웨어 자원을 많이 사용하게 되는 문제가 있다. DES 하나의 블록조차도 용량 부족으로 Altera FLEX 10K10 FPGA에 넣을 수 없는데 이것을 해결하기 위해 본 설계에서는 각 단계의 블록들의 특징만을 남기고 필요 없는 부분은 줄이는 형태로 각 단계마다 따로 설계 하였다. 그림 2는 첫 단계 블록이고, 그림 3은 중간 단계 블록, 그림 4는 마지막 단계 블록의 블록도이다.

설계초기에 튜닝 없이 하나의 단계를 FPGA에 구현하기 위한 컴파일을 수행한 결과, 필요한 FPGA 하드웨어 자원 용량이 795가 나왔는데, FLEX 10K10의 용량은 최대 573이므로 이를 수용할 수 없었다. 따라서 DES 알고리즘의 복잡도는 그대로 유지하면서 용량을 최소화 하기 위해서 데이타 처리 블록에서는 SBOX, Pure Permutator, Extended Permutator의 값을, 키 생

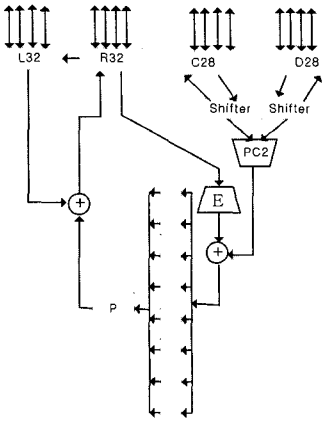

그림 1 통상의 DES 시스템

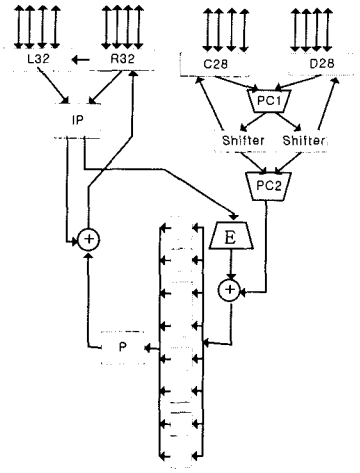

그림 2 재구성 가능한 DES part 1 : stage 0

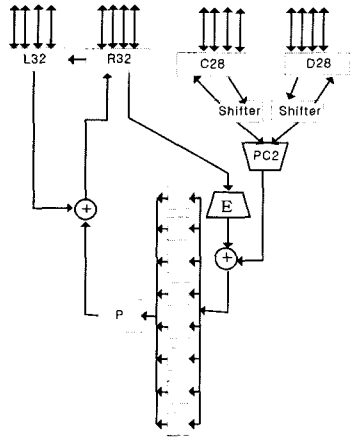

그림 3 재구성 가능한 DES part 2 : stage 1-14

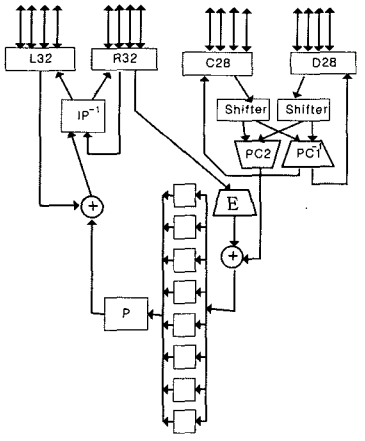

그림 4 재구성 가능한 DES part 3 : stage 15

성 블록에서는 C shifter, D shifter의 shift 값을 튜닝하였는데 Shifter의 경우에는 키값이 복호화에 사용될 것을 고려하여 조정하여야 한다. 튜닝의 요점은 분리된 각 단계의 처리동작에 멀티플렉스 또는 베럴쉬프터가 요구되지 않도록 만들자는 것이다. 데이타 처리 블록상에서 볼 때 SBOX, Permutator의 순서 바꾸기 동작에는 멀티플렉스가 요구되며, 키 생성 블록상의 C shifter, D shifter의 shift 동작에는 베럴쉬프터가 필요하다. 이런 성분들은 하드웨어 자원을 많이 소비하므로 SBOX, Permutater, Shifter의 각 단계별 값들을 조절하면 멀티플렉스, 베럴쉬프터를 사용하지 않고도 동일한 처리 결과를 얻을 수 있다. 튜닝한 결과를 적용하여 한 단계의 용량을 230까지 줄일 수 있었다. 만약 795에서 230까지 줄이는 것이 불가능한 경우가 발생한다면, 디자인을 더 분할하면 된다. 즉, 795의 디자인을 300+395나 350+345처럼 FLEX 10K10의 용량 573에 들어갈 수 있는 크기로 나누면 된다. 앞에서 튜닝하여 만들어진 한 단계에다 이전단계에서 데이타를 받아오는 로직과 다음단계로 데이타를 넘겨주는 로직(제어부와 버퍼)이 추가로 필요하여 이를 합하여 컴파일한 결과, 한 단계에서 필요한 FPGA 하드웨어 자원 용량이 470이 되었다(그림 5). 또한 용량을 줄일 수 있는 다른 방법으로 키를 받아 들일 때 rotate를 선택해주는 멀티플렉스 로직이 크기 때문에 암호화하는 블록과 복호화하는 블록을 분리하여 이 멀티플렉스 로직을 제거할 수 있었다. 지금까지의 튜닝작업을 통해 만들어진 하나의 블록으로 파이프라이닝을 사용하는 대신 반복수행을 해서 중간 14 단계를 커버할 수 있다. 설계과정은 우선 사용되는 FPGA의 하드웨어 자원 용량에 맞게 DES를 분할하여 VHDL로 설계하고 이것을 VHDL 시뮬레이터(ModelSim)를 이용하여 소프트웨어적으로 검증을 마친 후, Synopsys FPGA Compiler로 합성하여 전위 설계과정을 마친다. 이 과정이 끝난 후 FPGA 프로토타이핑 보드를 이용하여 디버깅 준비를 수행하고 Altera MAX+PLUS II로 후위 설계과정을 통하여 생성된 비트스트림을 FPGA 프로토타이핑 보드의 FPGA에 다운로딩하고 검증하였다.

4. 실험결과

성능을 단일 FPGA 칩(Altera FLEX10K10PLCC84)으로 구현한 DES(OC DES), 소프트웨어(펜티엄 III 800MHz, 256Mbyte memory를 가진 PC)로 구현한 DES 그리고 FPGA의 동적 재구성성을 이용한 DES(DR DES) 각각을 비교해 보면, 속도는 그림 6에서와 같이 OC DES보다는 3배 정도 느리지만 PC 상에서 수행된 소프트웨어에 비해서는 4.7배 이상의 향상을 볼 수 있었고, 용량은 그림 7에서 보는 바와 같이 OC

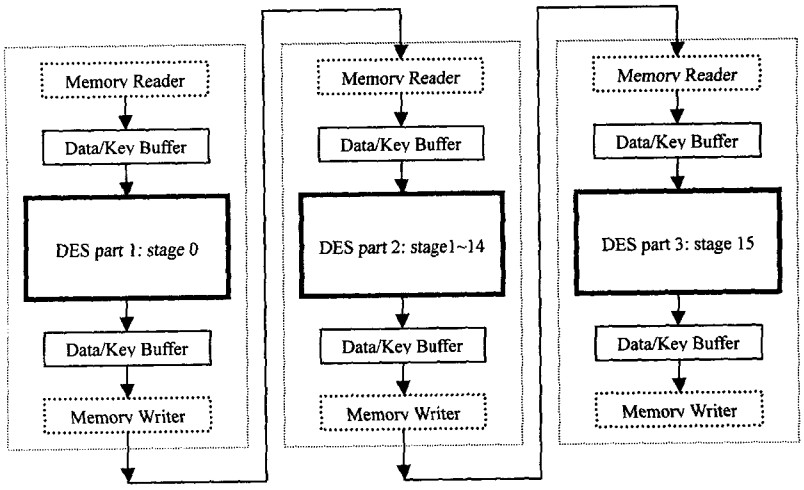

그림 5 단계사이의 데이타 및 키의 전달

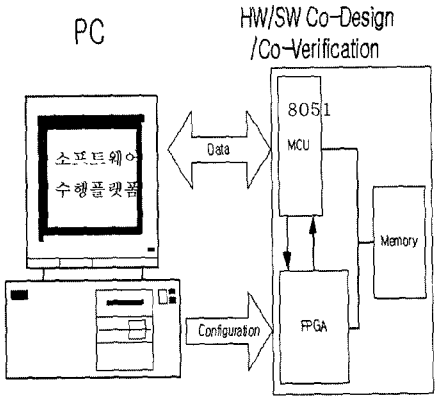

그림 6 동적 DES를 위한 FPGA 프로토타입 설계/검증 환경 구성도

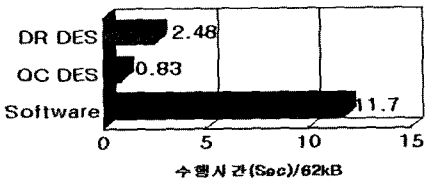

그림 7 성능 비교

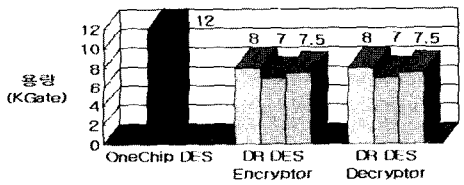

그림 8 용량 비교

DES는 12K 게이트 정도 이지만 DR DES는 8K 게이트의 용량으로 구현 되어 비용면에서는 34%의 개선이 있었다.

특히, 수행속도의 상당 부분(2.48초*85%=2.11초)은 세 개의 블록으로 나누어져서 3번에 걸쳐서(맨처음 구성되는 것 포함) 동적 구성되는 과정에서, OC DES보다 추가적인 2번의 재구성 시간이 차지하고 있다. 이는 사용된 FPGA 칩이 부분 재구성(partially reconfigurable)이 불가능한 디바이스여서 재구성 시간을 줄이는 것이 불가능한 것에 원인을 찾을 수 있는데, 최근의 디바이스들은 부분 재구성이 가능한 것들이 있음으로 이를 사용하는 경우에는 재구성 시간을 단축시키는 것도 가능할 것이다[3]. 또, Host 시스템에서 프로토타입 보드로 패러렐포트를 통해 다운로드를 하기 때문에 비트스트림의 전송 속도면에서 고속화가 어렵다는 문제점도 있는데, 이것을 PCI나 USB를 통해 통신하게 만들면 재구성 시간을 최소로 줄일 수 있을 것이다. 전체적으로는 본 실험결과를 통하여 동적 재구성에 의한 설계기법이 성능/비용간의 매우 효과적인 타협점을 제공하는 구현 방법임을 확인할 수 있었다.

5. 결 론

본 논문에서는 임의의 알고리즘을 위한 응용 태스크를 처리하는 시스템으로서 FPGA의 재구성 능력을 이용하여 실시간으로 재구성이 가능한 동적 재구성가능 시스템을 도입하여 기존의 전용 하드웨어 시스템에 비해 성능저하를 최소화 하면서도 비용을 획기적으로 감소시킬 수 있음을 논의하였다. 그리고 이를 확인하기 위해서 DES 암호 알고리즘을 동적 재구성(dynamically

reconfigurable) 방식으로 구현하였고, 이 과정을 통하여 동적 재구성 방식이 성능과 비용 간의 매우 효과적인 타협점을 제공할 수 있음을 확인할 수 있었다.

FPGA의 동적 재구성 기능을 이용하여 DES 알고리즘을 등가의 여러 개의 서브 블럭들로 나누어서 하드웨어 자원 용량이 적은 FPGA에 구현하였는데, 실험 결과에서도 확인할 수 있듯이 이러한 동적 재구성의 성능저하에 가장 큰 요인은 FPGA에 비트스트림을 다운로드하는데 걸리는 시간에 의한 것임이 확인되었다. 이것은 본 논문이 제안했던 Host 시스템의 마스터가 고속으로 FPGA를 재구성하는 방법을 프로토타입 보드상에서 구현하지 않았기 때문인데, 이에 대한 구현과 검증에 대한 추가적인 연구가 더 필요하다고 생각되며, 동적 재구성을 위해서는 원래 설계를 신속하고 효과적으로 분할해야 하는데 본 논문에서는 수작업으로 하였지만 이것을 자동화하기 위한 소프트웨어의 개발도 추후 연구 과제이다.

참 고 문 헌

[1] M.J. Wirthlin and B.L. Hutchings "A Dynamic Instruction Set Computer," IEEE Symposium on FPGAs For Custom Computing Machines, pages 99-107, 1995.

[2] P.C. French and R.W. Taylor "A Self-Reconfiguring Processor," IEEE Symposium on FPGAs For Custom Computing Machines, pages 50-59, 1993.

[3] Xilinx. "Virtex-II 1.5V Filed-Programmable Gate Arrays," Datasheet document part number DS03 1-1(v1.7), October 2, 2001.

[4] J.R. Hauser and J. Wawrzynek, "GARP: A MIPS Processor with a Reconfigurable Coprocessor," in Proc. IEEE Symp. on FPGAs for Custom Computing Machines, Napa Valley, California, 1997, pp. 12-21.

[5] SPIE Int. Symp. "Reconfigurable Processors for Handhelds and Wearables: Application Analysis," on Convergence of IT and Communications (ITCom'01), Denver, CO, USA, August 19-24, 2001.

[6] Altera, "ARM-Based Embedded Processor Device Overview," Datasheet document part number A-DS-EXCARM-01.1, February 2001.

[7] Altera, "Nios Soft Core Embedded Processor," Datasheet document part number M-DS-EXCNIOS-01, June 2000.

[8] Man Young Rhee, Cryptography and Secure Communications, McGraw-Hill Series on Computer Communications, 1994.

[9] 지용의, 프로텍트의 모든 것, 정보문화사, 1993.

안 민 희

1996년 동서대학교 컴퓨터공학과 졸업(공학사). 1998년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2000년 부산대학교 대학원 컴퓨터공학과 박사과정 수료. 2000년~2001년 ㈜서두로직 부설연구소 선임연구원. 2001년~2003년 ㈜시스베리 부설연구소 선임연구원. 관심분야는 HW/SW 동시설계/동시검증, Reconfigurable computing

양 세 양

1981년 고려대학교 전자공학과 졸업(공학사). 1985년 고려대학교 대학원 전자컴퓨터공학과 졸업(공학석사). 1990년 메사추세츠대학교 대학원 전기컴퓨터공학과 졸업(공학박사). 1990년~1991년 Microelectronics Center of Northe Carolina 선임연구원. 1991년~현재 부산대학교 컴퓨터공학과 재직(현 교수). 관심분야는 하드웨어기반 설계검증, HW/SW 동시검증, 논리합성, 테스팅

윤 재 근

1999년 부산대학교 컴퓨터공학과 졸업(공학사). 2001년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2001년~현재 (주)시스베리 선임연구원 재직. 관심분야는 하드웨어기반 설계검증