시분할 FPGA 합성에서 마이크로 레지스터 개수에 대한 하한 추정 기법

(A Lower Bound Estimation on the Number of Micro-Registers in Time-Multiplexed FPGA Synthesis)

엄 성 용 †

(Seong-Yong Ohm)

요 약 시분할 FPGA는 회로가 동작하는 중 회로의 기능을 재구성할 수 있는 동적 재구성 기능을 갖춘 FPGA 칩이다. 따라서 이러한 칩을 위한 회로 합성 기법에서는 주어진 논리 회로를 각각 다른 시간대에 수행할 여러 개의 부분회로로 분할한 후, 동일한 하드웨어 회로를 시간차를 두고 공유하도록 해야 한다. 기존의 연구에서는, 칩의 제한된 용량 문제를 해결하기 위해, 동일 시간대에 필요한 자원으로서 각 세부 함수를 수행하는 LUT(Look-Up Table)의 개수와 LUT의 출력 결과를 다른 시간대에 사용하기 위해 그 결과를 임시 저장하는데 필요한 마이크로 레지스터(micro register)의 개수를 최소화하는 데 중점을 두고 있다.

본 논문에서는 시분할 FPGA 합성용 도구 중의 하나로서 회로 구현에 필요한 메모리 원소, 즉 마이크로 레지스터의 개수에 대한 하한(lower bound)을 추정하는 기법에 대해 설명한다. 이 방법에서는 입력되는 논리 회로를 직접 합성하지 않고서도 그 회로가 필요로 하는 전체 마이크로 레지스터 개수에 대한 하한을 각각 추정함으로써 특정한 합성 기법에 관계없이 회로 구현에 필요한 최소한의 마이크로 레지스터의 개수에 대한 정보를 추출한다. 만일, 기존의 합성 결과가 본 연구에서 추정된 하한과 일치할 경우, 그 결과는 최적의 결과를 의미한다. 반면에, 하한과의 차이가 있는 경우에는 기존의 연구 결과에 비해 더 좋은 합성 결과가 존재하거나, 또는 본 연구에서 추정한 하한보다 더 좋은(큰, 정확한) 하한이 실제 존재함을 의미한다. 따라서 이러한 비교 분석을 통해, 기존 연구는 물론, 향후에 개발할 새로운 합성 방법의 결과가 최적인지, 또는 개선의 여지가 있는지를 판단하는 좋은 지표를 얻을 수 있다.

실험 결과, 추정된 하한은 기존 연구의 합성 결과와 다소 차이가 있었다. 이러한 차이는 우선, 기존의 합성 결과는 LUT 개수를 적절히 유지하는 가운데 마이크로 레지스터를 최소화한 결과인 반면, 본 하한 추정에서는 합성 가능한 모든 결과 중, LUT 개수와는 전혀 무관하게, 마이크로 레지스터 개수를 최대한 작게 사용할 합성 예를 추정하기 때문이라고 판단된다. 또 한편으로는 마이크로 레지스터 개수에 대한 하한 추정 문제 자체가 갖는 거대한 변동성과 복잡성으로 인해 제안한 추정 기법이 정밀도에 한계를 가지는 것으로 해석할 수 있으며, 다른 한편으로는 기존 연구 결과보다 더 좋은 합성 결과가 존재할 가능성이 높음을 의미하는 것으로 해석될 수 있다.

키워드 : DPGA, 재구성, 시분할 FPGA, 하드웨어 합성, 하한 추정, 스케줄링, 마이크로 레지스터

Abstract For a time-multiplexed FPGA, a circuit is partitioned into several subcircuits, so that they temporally share the same physical FPGA device by hardware reconfiguration. In these architectures, all the hardware reconfiguration information called contexts are generated and downloaded into the chip, and then the pre-scheduled context switches occur properly and timely. Typically, the size of the chip required to implement the circuit depends on both the maximum number of the LUT blocks required to implement the function of each subcircuit and the maximum number of micro-registers to store results over context switches in the same time. Therefore, many partitioning or synthesis methods try to minimize these two factors.

In this paper, we present a new estimation technique to find the lower bound on the number of

· 이 논문은 2001년도 한국학술진흥재단의 지원에 의해 연구되었음(KRF-2001-041-E00218)

† 종신회원 : 서울여자대학교 정보통신공학부 교수

osy@swu.ac.kr

논문접수 : 2003년 3월 19일

심사완료 : 2003년 6월 3일

micro-registers which can be obtained by any synthesis methods, respectively, without performing any actual synthesis and/or design space exploration. The lower bound estimation is very important in sense that it greatly helps to evaluate the results of the previous work and even the future work. If the estimated lower bound exactly matches the actual number in the actual design result, we can say that the result is guaranteed to be optimal. In contrast, if they do not match, the following two cases are expected: we might estimate a better (more exact) lower bound or we find a new synthesis result better than those of the previous work.

Our experimental results show that there are some differences between the numbers of micro-registers and our estimated lower bounds. One reason for these differences seems that our estimation tries to estimate the result with the minimum micro-registers among all the possible candidates, regardless of usage of other resources such as LUTs, while the previous work takes into account both LUTs and micro-registers. In addition, it implies that our method may have some limitation on exact estimation due to the complexity of the problem itself in sense that it is much more complicated than LUT estimation and thus needs more improvement, and/or there may exist some other synthesis results better than those of the previous work.

Key words : DPGA, FPGA, Reconfiguration, Time-Multiplexed FPGA, Hardware Synthesis, Micro Register, Lower Bound Estimation, Scheduling

1. 서 론

일반적으로, FPGA(*Field Programmable Gate Array*)는 소프트웨어적인 도구를 통해 해당 회로의 기능을 쉽게 구현, 삭제 또는 변환할 수 있어, 시제품 제작 등 하드웨어 구현이 단기간에 요구되는 분야에서 많이 활용되고 있다[1-2]. 특히, 최근 들어, 하드웨어 회로가 전원이 켜져 동작하는 상태에서 그 회로의 내부를 변경 가능하게 함으로써 구현된 하드웨어의 기능을 수시로 변경할 수 있는 *'동적 재구성이 가능한 FPGA (DRFPGA: Dynamic Reconfigurable FPGA)'* 또는 간략히 *'DPGA'*라고 불리우는 칩이 개발되어 많은 관심을 끌게 되었다[3-9]. DPGA는 주어진 논리 회로를 하드웨어로 구현할 수 있다는 점에서는 일반 FPGA 칩과 동일하지만, 칩이 동작되는 도중에도 회로의 전체 또는 일부를 다른 내용으로 신속히 변경하여 시간대별로 다른 기능을 수행할 수 있는 동적인 특성이 있다.

DPGA는 여러 가지 구조[3-9]가 있는데, DPGA 세부 구조에 따라 합성 기법 및 결과 분석이 달라질 수 있다. 따라서 본 논문에서는 Xilinx의 *'시분할 FPGA (Time-Multiplexed FPGA)'*[3,4]를 중심으로 설명하고자 한다. 시분할 FPGA에서는 주어진 논리 회로를 일의 우선순위 또는 데이타 흐름을 고려하여 시간대별로 수행할 여러 개의 부분 회로들로 분할한 후, 동일한 하드웨어를 시간차를 두고 공유할 수 있게 한다. 이 구조는 내부적으로 진리표 방식으로 회로의 기능을 수행하는 *'LUT (Look-Up Table)'* 블록들과 MUX(Multiplexor)들, 그리고 이들의 입출력 결과들을 저장하는데 사용되는 *'마이크로 레지스타(micro register)'*들로 구성되며, 동적 재구성을 위해 *'context'*라고 불리우는 회로 재구성 정보를 각 시간대별로 생성하여 미리 내부 메모리에 저장해 두고, 시간대에 따라 해당 context를 사용하여 회로를 재구성한다.

그림 1은 Xilinx사의 시분할 FPGA의 간소화된 내부 구조를 보여준다[3]. 그림에서 하나의 *context*란 각 LUT의 기능을 결정하는 진리표 값과 회로간의 연결을 결정하는 MUX 각각에 대한 선택 신호 등으로 이루어진 회로 재구성 정보로서, 해당 시간대에 동작하는 회로의 구체적인 기능을 선택, 결정한다. 이 예의 경우, 해당 논리 함수에 대한 진리표 값과 MUX 선택 신호들은 각각 8개의 다른 값을 가지고 있는데, 이는 회로 전체로 볼 때 8개의 context로 구성되어 있음을 의미한다. 예를 들어, *G1, G2, G3, G4*를 입력으로 하는 논리 함수는 16 비트의 진리표 값에 따라 함수의 기능이 정의되는데, 그 함수는 시간대별로 새로이 구성되어야 하므로, 전체 *context* 수와 일치하는 총 8개의 16비트 진리표 값(전체적으로 16비트×8의 정보)이 그 함수의 시간대별 구성을 위해 미리 결정되어 있음을 알 수 있다.

주어진 논리 회로를 시분할 FPGA로 구현하기 위해서는 *k*개(보통 미리 결정되어 있음)의 *context*를 구성하여야 한다. 예를 들어, 그림 1의 경우, 총 8개의 *context*를 가진다(즉, $k = 8$). 이러한 각 *context*는 시간대별 회로 구성 정보를 의미하기 때문에 *k*개의 시간대 각각을 하나의 *'세부주기(micro cycle)'*이라고 하며, *k*개의 세부주기로 이루어진 전체 주기를 하나의 *'사용자주기(user cycle)'*이라고 한다. 사용자 주기는 논리 회로 설계자 입장에서 볼 때 일반적인 수행 주기, 즉, 하나의 클록(*clock*)을 의미한다. 한편, *context*가 변경되면 이전의 회로는 모두 사라지고 새로운 회로가 구성되기 때문에, 어느 한 시간대에 생성된 결과(조합회로의 출력)가

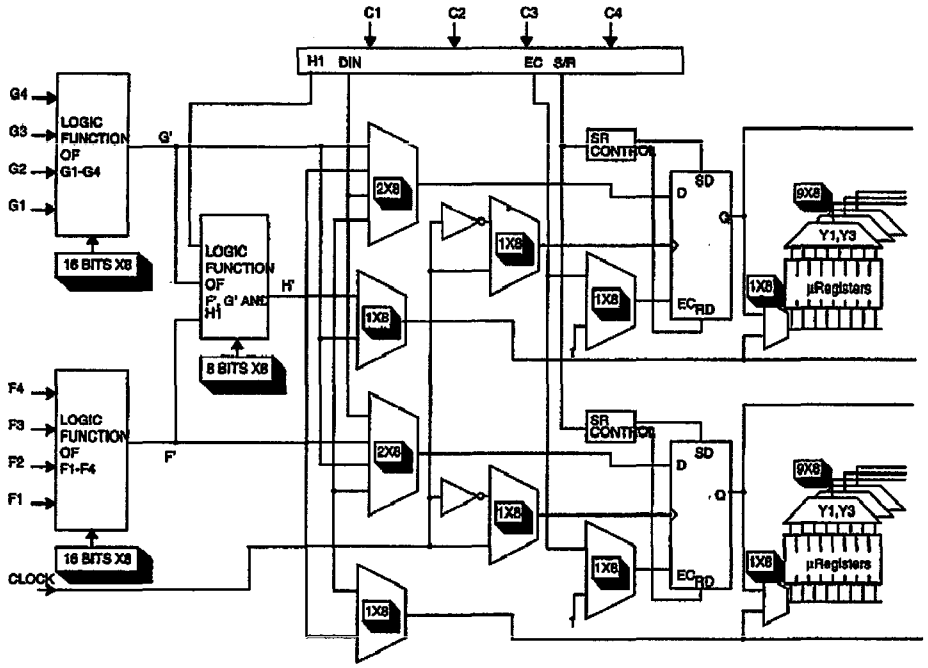

그림 1 시분할 FPGA의 내부 구조

그 이후에 다시 사용되는 경우, 이러한 값들은 '*마이크로 레지스터(micro register)*'라는 별도의 기억장치에 저장하여야 한다. 즉, 다른 *context*간의 정보 전달이 이러한 마이크로 레지스터를 통해 이루어지므로, 각 세부 주기 경계를 지나는 에지(즉 LUT 입출력 신호)들은 마이크로 레지스터에 의해 저장되어야 한다.

그림 2는 시분할 FPGA의 내부 통신 및 저장 구조를 보여준다. 여기서 C_1, C_2, ..., C_k는 시간대별로 수행할 각기 다른 기능의 조합 회로이지만, 시간대를 달리하여 동일한 하드웨어 회로를 시간차를 두고 공유한다. 그림에서 각 시간대별 조합 회로는 외부 입력 신호(PI : *Primary Input*) 및 이전 시간대의 구성에서 출력되어 마이크로 레지스터에 저장된 내부 값들을 입력으로 하여, 해당 기능을 수행한 후, 필요한 값들을 출력한다. 출력된 일부 값은 외부 출력 신호(PO : *Primary Output*)로 직접 전달되기도 하며, 나머지는 다음 재구성 회로에서 사용되기 위해 마이크로 레지스터에 저장되기도 한다. 결과적으로 *k*개의 세부주기를 거친 출력 결과는 주어진 논리 회로에서 하나의 클록(사용자 주기)이 수행된 후의 출력 결과와 동일하다. 여기서 *fanout*들은 하나의 신호로 볼 수 있으며, PI 및 PO 등의 외부 입출력 신호는 사용자 주기 동안 외부의 일반 레지스터에 의해 값을 계속 유지한다고 보기 때문에, 이를 별도의 마이크로 레지스터에 저장하지 않는 것이 일반적이다.

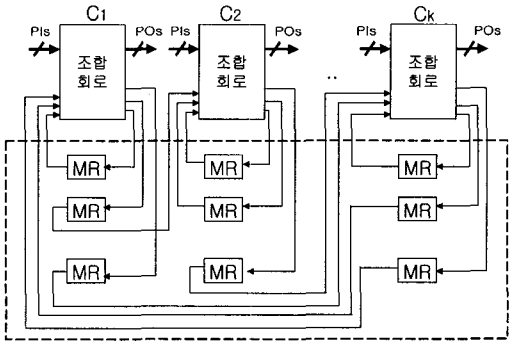

그림 2 시분할된 회로의 내부 통신 구조

그런데, 전체 회로를 시간대별로 수행할 부분 회로로 어떻게 분할하고 재구성하느냐에 따라 회로 구현에 필요한 칩의 용량이 달라지게 된다. 일반적으로 회로 구현에 필요한 칩의 용량은, 그림 1과 그림 2에 나타난 바와 같이, 동일 시간대에 필요한 자원으로서 각 세부 함수를 수행하는 LUT 블록의 개수와 각 LUT 블록의 출력 결과를 다른 시간대에 사용하기 위해 그 결과를 저장하는 데 사용되는 마이크로 레지스터의 개수에 의해 결정된다. 일반적으로 LUT 개수와 마이크로 레지스터 개수, 이 두 가지를 동시에 최소화하는 것이 이상적이지만, 이

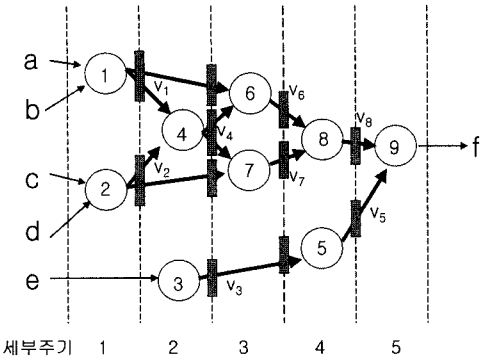

그림 3 LUT 개수를 최소화하는 합성 결과 예

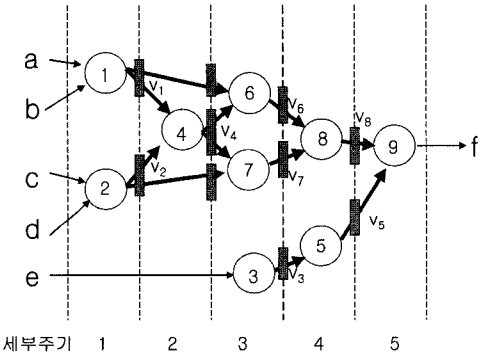

그림 4 마이크로 레지스터 개수를 최소화하는 합성 결과 예

러한 합성 결과를 얻는 것이 쉽지 않다.

예를 들어, 그림 3과 그림 4는 동일한 예제 회로에 대한 두 가지 합성 결과로서, LUT 개수와 마이크로 레지스터 개수 면에서 각각 우수한 결과를 보여 준다. 이 때, 각 LUT(각 연산)의 지연 시간은 설명의 편의상 세부주기 크기와 동일하다고 가정한다. 즉, 3장에서 설명할 압축 또는 체이닝을 허용하지 않는다고 가정한다. 그림 3은 동일 세부주기에서 수행되는 연산이 최대 2개이므로 적어도 2개 이상의 LUT가 필요하며, 또한 세부주기 2에서 산출되어 세부주기 3에서 사용되는 값이 4개이므로 이 값들을 저장하기 위해 적어도 4개의 마이크로 레지스터가 필요하다. 한편, 그림 4의 경우, 그림 3에서 노드 3의 수행시간을 변경하여, LUT 사용 개수는 1개 늘어났으나, 세부주기 경계를 넘는 신호의 최대 개수는 3이므로 3개의 마이크로 레지스터만이 필요하다. 여기서 외부 신호인 *a, b, c, d, e, f*는 마이크로 레지스터에 저장될 필요가 없으므로 무시한다. 한편, 보통의 경우, LUT가 마이크로 레지스터보다는 큰 면적으로 차지하므로 그림 3의 결과가 낫다고 볼 수는 있으나, DPGA 구조에 따라 달라질 가능성이 있으므로 본 논문에서는 이에 대한 *trade-off*는 고려하지 않는다. 다만, 임의의 합성 결과에서 필요한 마이크로 레지스터 개수에 대해서만 관심을 둔다.

따라서 기존의 시분할 FPGA 합성 연구에서는 LUT 개수와 마이크로 레지스터 개수의 두 요소들을 순차적 또는 동시에 최소화하기 위한 방안들을 제안하였다. 하지만, 문제 자체가 복잡하고, 또한 최적 결과에 대한 정보가 부족하거나 산출하기 힘들기 때문에, 산출된 결과가 최적인지 또는 얼마나 개선 가능성이 있는지에 대한 분석 및 판단이 매우 힘든 경우가 많다.

산출된 실제 합성 결과에 대한 최적성 여부 판단은 일반적으로 불가능 하지만, 특정한 조건이 만족되는 경우에는 이에 대한 판단이 가능하다. 즉, 주어진 문제에 대한 알려진(또는 이론적으로 계산한) 하한(*lower bound*)이 있고, 또한 실제 합성한 어떤 결과가 그와 동일한 결과를 낸다면, 그 기존 결과는 분명히 최적의 결과라고 할 수 있다. 하지만, 일반적으로는 문제의 하한을 미리 알 수 없으므로 하한을 추정하게 되는데, 추정된 하한과 기존 결과 사이에 차이가 있는 경우에는 기존의 연구 결과에 비해 더 좋은 합성 결과가 존재하거나, 또는 추정된 하한보다 더 좋은(큰, 정확한) 하한이 존재할 수 있음을 의미한다. 따라서 이러한 비교 분석을 통해, 기존 연구의 결과가 최적인지, 또는 개선의 여지가 있는지를 판단하는 좋은 지표를 제공할 수 있다. 관건은 어떻게 보다 좋은 하한을 빠른 시간내에 추정하느냐의 문제이다.

시분할 FPGA 합성에 대한 하한 추정 연구는 크게 LUT 개수를 위한 하한 추정과 마이크로 레지스터를 위한 하한 추정으로 나눌 수 있는데, 본 연구진은 이미 LUT 개수에 대한 하한 추정 연구를 수행한 바 있으며, 그 내용은 참고문헌 [11]에 자세하게 기술되어 있다. 본 논문에서는 주어진 그래프 정보로부터 해당 회로의 구현에 필요한 전체 마이크로 레지스터 개수에 대해 하한을 추정하는 방법에 대해 설명한다.

이러한 하한 추정의 주요 특징 중의 하나는 추정된 하한이 주어진 문제에 대한 이론적 하한에 비해 항상 같거나 작은 성질을 만족한다는 점과 하한 추정 작업은 여러 가지 복잡한 합성 작업을 전혀 수행하지 않고, 단지 주어진 그래프에 대한 분석만을 통해 이루어진다는 점이다. 따라서 이미 기존의 방법으로 산출된 결과는 물론, 향후에 개발될 합성 기법에 의해 산출될 어떤 최상의 결과보다도 작거나 같은 추정치를 제공한다. 이는 곧 본 추정 기법이 기존에 존재하거나 또는 향후 개발될 새로운 시분할 FPGA용 합성 기법이 산출한 결과의 정확성을 판단하는 보조 지표 도구로 사용될 수 있음을 뜻한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구와 관련된 기존 연구에 대해 설명하며, 3장에서는 마이크로 레지스터 개수에 대한 하한 추정의 기본 원리와 향상 기법, 그리고 확장 기법에 대해 설명한다. 그리고 4장에서는 본 연구의 실험 내용인 추정된 하한과 기존 연구에서 실제 산출한 합성 결과를 비교 분석하며, 5장에서는 연구의 결론을 기술한다.

2. 관련 연구

시분할 FPGA에서 주어진 논리 회로를 시간대별로 수행할 여러 개의 부분 회로들로 나누는 문제는 기존에 상위단계 합성 분야의 주요 단계인 스케쥴링(*scheduling*) 문제[12-15] 또는 회로 분할(*circuit partitioning*) 문제[15-16]와 매우 유사하다[11]. 따라서 시분할 FPGA 합성을 위한 기존의 연구에서는 주로 기존에 상위단계 합성 분야의 연구에서 개발된 각종 스케쥴링 기법 또는 다중 회로 분할 기법들을 수정하여 적용하고 있다. 스케쥴링 기법과 회로 분할 기법에 대한 자세한 설명은 참고문헌[12-16]을 참조하기 바란다.

참고문헌 [4], [5], [6] 등의 연구에서는 기존의 스케쥴링 기법을 변형한 방법을 사용한다. [4]에서는 기존의 리스트 스케쥴링(*List Scheduling*) 기법을 합성 문제에 알맞게 수정 적용하여 일단 초기 결과를 얻은 후, 스케쥴링 압축 및 스케쥴링 확장 기법을 이용하여 그 결과를 반복적으로 향상시킨다. [5]에서는 회로의 임계경로(*Critical Path*)를 고려하는 PCS 스케쥴링(*Precedence Constraint Scheduling*) 기법을 사용하는데, 초기 결과를 토대로 반복적인 재스케쥴링(*rescheduling*), 즉 부분적인 스케쥴링 결과 조정을 통해 보다 좋은 결과를 찾는다. 특히, 이 방법에서는 마이크로 레지스터 및 IO의 최소화를 위해, 오히려 LUT의 개수가 증가되는 것을 의도적으로 허용하기도 한다. [6]에서는 가장 빠른 회로 구현을 위해, FDS 스케쥴링(*Force-Directed Scheduling*)[14]을 변형한 알고리즘을 사용하며, LUT 개수 이외에 마이크로 레지스터 개수의 최소화도 고려한다.

반면에 [7], [8], [9], [17], [18] 등의 연구에서는 기존의 회로 분할 기법을 변형한 방법을 적용한다. [7]에서 제안한 FBP-m은 마이크로 레지스터의 최소화에 중점을 두고 있으며, 이를 위해 다중 회로 분할이 가능한 '네트워크 흐름 방식의 회로 분할(*network flow based circuit partitioning*)' 기법을 활용한다. 이 방법에서는 세부적으로 최대흐름-최소분할(*maximum flow & minimum cut*) 알고리즘을 사용하는데, 리스트 스케쥴링을 이용한 기법보다 약 20% 내외의 성능 개선이 있는 것으로 보고되고 있다. [8]과 [9]는 [7]의 후속 연구로서 [8]에서는 [7]의 FBP-m 기법을 Dharma[6] 구조에 적용할 수 있도록 수정한 연구이며, [9]에서는 [7]의 결과를 향상시키기 위한 최적의 스케쥴링 압축 기법을 제안하고 있다. [17]에서는 2단계에 걸친 회로 분할 기법을 이용한 PAT 시스템을 제안한다. PAT의 첫번째 단계는 인접도가 높은 노드들을 묶어 클러스터로 나누는 단계로서 문제 자체의 크기를 줄이는 역할을 하며, 두번째 단계는 앞서 구한 클러스터 각각을 하나의 노드로 하는 그래프를 구성하여 이에 대한 이분 분할을 반복적으로 수행하여 최종 다분할 결과를 구한다. 분할할 노드를 결정할 때는 확률에 기초한 이득(*gain*)을 계산하여 통신비용, 즉 마이크로 레지스터의 개수를 최소화한다. 이 방식은 [7]의 네트워크 흐름 방식에 비해서 12.4% 적은 마이크로레지스터 개수를 요구하는 것으로 알려져 있다. [18]은 ILP(*Integer Linear Programming*) 기법에 의해 최적에 가까운 해를 구한다. 이 ILP 방식에 의한 실험 결과는 [17]의 결과에 비해서 마이크로 레지스터의 개수 면에서 21.5% 향상되었으나, 문제의 복잡도를 줄이기 위한 회로의 사전 클러스터링 결과에 따라서 분할 결과가 크게 영향을 받을 수 있다는 점과, 실행 시간이 매우 오래 걸린다는 단점이 있다. 한편, 본 논문에서 적용할 하한 추정 기법에 대한 기존 연구로는 [10]과 [11]이 있다. [10]은 회로 구성에 필요한 *Functional Unit*, 레지스터, 버스 등의 용량을 상위 단계에서 각각 추정한다. [11]은 [10]의 기본 방법을 확장하여 시분할 FPGA에서의 LUT 개수에 대한 하한 추정에 적용하였으며, 마이크로 레지스터의 개수에 대한 연구는 이루어지지 않았다.

3. 마이크로 레지스터 개수에 대한 하한 추정

3.1 문제 및 용어 정의

본 논문에서는 단일 함수 기능을 의미하는 연산 노드들의 집합 V와 노드간의 선행 관계 내지 입출력 연결 신호를 나타내는 에지들의 집합 E로 구성된 그래프 G=(V, E)를 입력으로 한다. 여기서 그래프 상의 각 연산 노드는 LUT에 수행되며, 각 에지는, 필요할 경우, 마이크로 레지스터에 의해 임시 저장된다. 단, *fanout* 신호들은 동일한 값을 가지므로 하나의 신호로 취급될 수 있으며, 또한 PI 및 PO 등의 신호는 사용자 주기(여러 개의 세부주기) 동안 칩 외부의 일반 레지스터에 저장되어 그 값을 계속 유지한다고 보는 것이 일반적이기 때문에, 이를 별도의 마이크로 레지스터에 저장하지 않는다.

본 논문에서는 여러 가지 복잡한 합성 작업을 수행하지 않고, 단지 주어진 그래프에 대한 분석만을 통해, 해당 회로의 구현에 필요한 전체 마이크로 레지스터 개수에 대해 하한을 추정한다. 이 때, 문제 자체의 특성상,

어느 특정 신호가 구체적으로 어느 마이크로 레지스터에 저장되는지는 전혀 고려하지 않으며, 다만 구현에 필요한 레지스터 개수에만 관심을 갖는다. 또한 본 논문에서 제안한 기법에서 추정한 하한은 주어진 문제에 대한 이론적 하한에 비해 항상 같거나 작은 성질을 만족한다. 따라서 이미 기존의 방법으로 산출된 결과는 물론, 향후에 개발될 합성 기법에 의해 산출될 어떤 최상의 결과보다도 반드시 작거나 같은 추정치를 제공한다.

본 논문에서는, 설명의 편의를 위해 다음의 몇 가지 용어 및 표기를 사용한다. 주어진 전체 그래프에 대해 각 노드를, 노드간의 선행 관계를 유지하는 가운데, 최대한 빠른 세부주기에 배정하는 ASAP(*As Soon As Possible*) 스케쥴링[12,14]과 가장 늦은 세부주기에 배정하는 ALAP(*As Late As Possible*) 스케쥴링[12,14]을 각각 실행하였을 때, 노드 v가 스케쥴링되는 세부주기를 각각 노드 v의 ASAP 세부주기 및 ALAP 세부주기라고 하고, S_v, L_v라고 표기한다. 또한 주어진 그래프의 각 에지로 표현되는 하나의 입출력 신호가 마이크로 레지스터에 반드시 저장되어야 할 세부주기 기간을 그 에지(신호)의 *'생명주기(life time)'*라고 하며, 그 에지(신호)는 자신의 생명주기 동안 *'활동적(active)'*이라고 표현한다. 회로가 안정하게 동작하기 위해서는 하나의 신호는 자신의 생명주기 동안 반드시 마이크로 레지스터에 저장되어야 한다.

그러나, 노드들이 스케쥴링 되지 않은 경우, 에지들의 생명주기가 가변적일 수 밖에 없다. 이에 각 신호 $e_{i,j}$의 가변적인 생명주기 가운데 특정 스케쥴링 결과에 관계없이 반드시 활동적이라고 보장되는 최소 생명주기의 크기를 *'최소수명(Minimun Life)'*이라고 하고 $W_{i,j}$라고 표기한다.

3.2 기본적인 하한 추정 기법

그래프 상의 각 연산 노드에 대한 스케쥴링이 수행된 경우에는 각 에지의 생명주기를 분석하면, 회로 구현에 필요한 최소의 마이크로 레지스터의 개수를 쉽게 추출할 수 있다. 하지만, 본 하한 추정 기법에서는 특정 합성 기법에 관계없이 산출될 수 있는 최적의 결과를 추정하는 도구로서, 여러 가지 복잡한 합성 작업을 거치지 않고, 단지 주어진 그래프에 대한 분석만을 통해서 빠른 시간내에 하한을 추정하여야 한다. 그래서 신호의 생명주기가 가변적일 수 밖에 없고, 이에 고려할 사항이 매우 많아지는 문제 자체의 복잡성을 내포하고 있다. 따라서 가변적인 생명주기를 어떻게 처리하느냐가 본 기법의 핵심이다.

본 논문에서는 마이크로 레지스터 개수에 대한 하한을 추정하기 위해, 주어진 각 에지에 대해, 최소수명을 되도록 크게 추출하고 이를 토대로 마이크로 레지스터 개수에 대한 하한을 추정한다.

각 에지 $e_{i,j}$의 최소수명 $W_{i,j}$는 다음과 같이 구할 수 있다. 만일, 노드 i에서 노드 j로 연결된 에지 $e_{i,j}$의 경우, 만일 노드 i가 세부주기 a에 스케쥴링했다고 가정할 때, 노드간의 의존 관계에 의해, 노드 j가 세부주기 b 또는 그 이후의 세부주기에 스케쥴링되면, $e_{i,j}$의 최소수명 $W_{i,j}$는 $b-a+1$ 이라고 할 수 있다. 즉, 이 에지의 값은 적어도 $b-a+1$ 세부주기 만큼은 활동적이며, 적어도 그 동안은 하나의 마이크로 레지스터를 필요로 한다고 볼 수 있다. 따라서, 노드 j가 자신이 스케쥴링될 수 있는 가장 빠른 세부주기인 S_j에 스케쥴링된다고 가정할 때, 주어진 그래프상의 데이타 의존관계가 유지되는 가운데, 노드 i가 스케쥴링될 수 있는 가장 늦은 세부주기 c를 추출하여 S_j-c+1를 $W_{i,j}$라고 한다. 한편, 노드 i가 자신이 스케쥴링될 수 있는 가장 늦은 세부주기인 L_i에 스케쥴링된다고 가정할 때, 주어진 그래프상의 데이타 의존 관계가 유지되는 가운데, 노드 j가 스케쥴링될 수 있는 가장 빠른 세부주기 d를 추출하여 $d-L_i+1$를 $W_{i,j}$로 선택할 수도 있으나, 이 두 값은 항상 동일하므로 한 가지만을 계산하여 그 에지의 최소수명으로 채택한다.

각 에지 $e_{i,j}$에 대해 $W_{i,j}$가 구해지면, 보다 큰 값을 하한으로 추정하기 위해, 각 시간 범위 Z에 대해 다음의 과정을 반복한다. 즉, 각 에지 $e_{i,j}$에 대해, $P_{i,j,Z} = [S_j - W_{i,j} + 1, S_j] \cap Z$와 $Q_{i,j,Z} = [L_i + 1, L_i + W_{i,j}] \cap Z$를 각각 구한다. 여기서 $P_{i,j,Z}$는 노드 j를 가장 빨리 스케쥴링함과 동시에 노드 i는 가장 늦게 스케쥴링할 경우에 그 에지가 활동적이라고 보장된 세부주기 범위와 Z와의 공통부분을 의미하며, $Q_{i,j,Z}$는 노드 i를 가장 늦게 스케쥴링하면서 노드 j를 가장 빨리 스케쥴링하는 경우에 그 에지가 활동적이라고 보장된 세부주기 범위와 Z와의 공통부분을 의미한다. 따라서 각 범위 Z에 대해 생각할 때, 이 두 값 중 작은 값, 즉 적어도 $M_{i,j,Z} = min\ (|P_{i,j,Z}|,|Q_{i,j,Z}|)$만큼의 세부주기 기간 동안은 에지 $e_{i,j}$가 어느 한 마이크로 레지스터에 저장되어야 함을 뜻한다. 제안된 방법에서는 모든 가능한 시간 범위 Z에 대해 위의 과정을 반복하여 가장 큰 값을 구하고, 이를 마이크로 레지스터에 대한 하한 값으로 최종 선택한다.

예를 들어, 그림 3과 그림 4에서 설명한 예제 회로에서 세부주기 범위 Z = [1,3]에 대해 생각할 경우, 적어도 한 세부주기 이상 반드시 저장되어야 하는 신호들은 임계경로상에 있는 노드 1, 2, 4, 6, 7의 출력값들이며, 범위 Z에 대한 이들 출력 신호들의 최소수명은 각각 2, 2, 1, 1, 1이다. 따라서 Z = [1, 3]에서의 마이크로 레지스터 개수에 대한 하한은 $\lceil (2+2+1+1+1) / 3 \rceil = 3$으로 추정된다. 다른 범위에 대해서도 동일한 방법이 적용되어, 그 최대값을 하한으로 추정한다. 이 경우, 하한은

1. 주어진 그래프 $G = (V, E)$와 *context* 수 k를 입력으로 받는다.
2. G에 대한 ASAP 스케쥴링과 ALAP 스케쥴링을 각각 수행하여,
   각 노드 v 대해, 그 노드가 스케쥴링될 수 있는 세부주기 범위
   $[S_v, L_v]$를 구한다.
3. <*fanout* 제거> 및 <에지 통합> 기법을 적용, 새로운 에지 집합 E'을 구한다.
4. 각 에지 $e_{i,j} \in E'$에 대해, 최소수명 $W_{i,j}$를 구한다.
5. 설정 가능한 모든 범위 $Z = [from, to]$ $(1 \le from \le to \le k+1)$에 대해,
   모든 에지 $e_{i,j} \in E'$에 대해,
   $$P_{i,j,Z} = [S_j - W_{i,j} + 1, S_j] \cap Z;$$
   $$Q_{i,j,Z} = [L_i + 1, L_i + W_{i,j}] \cap Z;$$
   $$M_{i,j,Z} = min\ (|P_{i,j,Z}|, |Q_{i,j,Z}|);$$
   $$LB = max\ (LB, \lceil \Sigma_{i,j} M_{i,j,Z} / |Z| \rceil);$$
6. **LB**를 마이크로 레지스터 개수에 대한 추정된 하한 값으로 반환/출력한다.

그림 5 마이크로 레지스터 개수에 대한 하한 추정을 위한 간소화된 알고리즘

3으로 추정되는데, 이는 그림 4와 일치하는 결과로서, 이를 통해 그림 4의 결과는 마이크로 레지스터 합성 면에서 최적임을 확인할 수 있으며, 또한 이 경우 하한 추정 결과가 매우 정확함을 재확인할 수 있다.

그림 5는 본 논문에서 제안하는 마이크로 레지스터 개수에 대한 하한 추정 알고리즘을 간략하게 요약한 것이다. 이 알고리즘의 시간 복잡도는 $O(|E| \cdot (|E|+|V|-+k^2))$이다[10].

다시 강조하고자 하는 것은 이렇게 추정된 하한은 주어진 문제에 대한 이론적 하한에 비해 항상 같거나 작은 성질을 만족하기 때문에 기존의 방법으로 산출된 결과는 물론, 향후에 개발될 합성 기법에 의해 산출될 최상의 결과보다도 작거나 같은 추정치를 제공하는 것을 보장한다.

3.3 추정 향상 기법

하지만, 많은 경우, 각 에지에 대해 큰 최소수명을 찾는 것이 매우 힘들다. 더욱이 다음 절에서 설명할 연산 노드간의 체이닝을 허용할 경우, 이러한 값을 찾는 것은 매우 어려워진다. 따라서 본 논문에서는 이러한 문제를 보완하기 위해, 불필요한 *fanout*을 제거하는 방법과 여러 개의 에지를 하나로 통합하는 방법 등 2가지 추정 향상 기법을 추가 적용한다. 이 향상 기법에서는 기존의 에지 집합 E로부터 하한 추정에 보다 유리한 새로운 에지 집합 E'를 생성한다. 이 기법들은 하한 추정 결과를 향상시킬 뿐만 아니라, 문제의 크기를 줄여, 오히려 하한 추정에 걸리는 수행 시간도 단축하는 효과도 있다.

우선, *fanout* 제거 기법에서는 여러 *fanout* 신호들 중 특정 *fanout*의 생명주기가 다른 *fanout*의 생명주기의 부분집합임이 분명한 경우, 이를 삭제한다. 이는 *fanout* 신호들은 동일한 값을 가지므로 하나의 신호만이 마이크로 레지스터에 저장되면 되기 때문이다. 이를 통해, 문제를 간소화시킴은 물론, 후속 작업인 에지 통합 기법 적용시 보다 나은 결과를 산출하도록 도와준다. 예를 들어, 그림 6에서 $e_{1,3}$과 $e_{1,4}$는 동일한 값을 갖는 *fanout* 신호들인데, 노드간의 선행관계상 $e_{1,3}$의 생명주기가 항상 $e_{1,4}$의 생명주기를 포함한다. 따라서 하한 추정시 내부적으로 에지 $e_{1,4}$는 무시하고, $e_{1,3}$만을 고려한다. 즉, $E' = E - \{e_{1,4}\}$가 된다. 그림 6에서 *fanout* 신호 가운데 제거 가능한 *fanout*은 점선으로 표시하였다. 이 예의 경우, 2개의 *fanout*들이 이와 같은 방법으로 제거된다.

한편, 에지 통합 기법에서는 두 개의 에지를 따로 고려할 경우, 각각에 대한 최소수명이 적거나 없지만, 추정 목적상 두 개의 에지를 하나로 통합하여 생각할 경우, 통합한 에지에 대해서는 각각의 최소수명을 더한 것보다도 많은 최소수명을 얻어, 추정 결과를 향상시키는 기법이다. 예를 들어, 그림 6에서 에지 $e_{1,3}$과 $e_{3,6}$에 대해 생각해 보자. 세부주기 2와 세부주기 3의 경계에 대

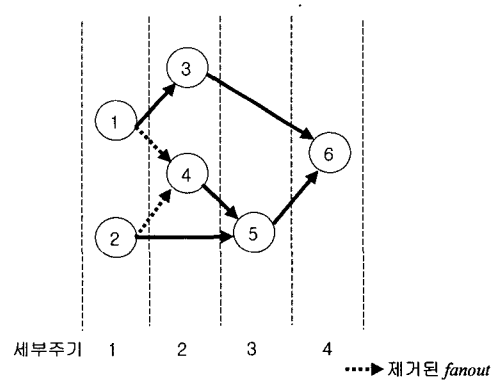

그림 6 향상 기법 적용 예제

해 고려할 때, 어느 에지도 그 경계에서 활동적이라고 말할 수 없다. 하지만, 분명한 것은 두 에지 중의 하나는 반드시 그 경계에서 활동적이라고 할 수 있다. 따라서 이러한 경우, 이 두 에지($e_{1,3}$과 $e_{3,6}$)를 무시하고, 이 두 에지를 통합한 새로운 에지(이를 $e_{1,3,6}$이라고 표현하자) 이용하여 하한을 추정함으로써 추정의 정밀성을 높인다. 즉, $E' = E - \{e_{1,3}, e_{3,6}\} \cup \{e_{1,3,6}\}$이 된다.

3.4 추정 기법의 확장

본 논문에서 다루는 시분할 FPGA에서는 *context*의 최대 개수가 8개로 제한되어 있다. 따라서 9개 이상의 연산이 순차적으로 수행되어야 하는 경우, 즉 그래프상의 임계경로의 길이가 8보다 큰 경우에는 이를 수용할 수 없다. 실험된 많은 벤치마크 예에서 이러한 현상이 나타난다('4장 실험 결과 및 분석' 참조). 이 경우에는 순차적인 2개 이상의 연산들을 하나의 마이크로 주기에 실행하도록 조정해야 한다. 이렇게 2개 이상의 순차적인 연산이 하나의 주기에 수행하는 것을 *'스케쥴링 압축(scheduling compression)'*[9] 또는 *'체이닝(chaining)'*[10,12]이라고 한다.

예를 들어, 그림 7은 전체 *context*의 수가 4로 제한되어 있다고 가정할 경우, 임계경로의 길이가 5인 그림 3과 그림 4의 예제 회로에 대해 체이닝 기법을 적용한 예이다. 이 예에서는 각 연산을 수행하는 LUT의 지연시간은 LUT 2개가 순차적으로 수행되더라도 세부주기 크기를 넘지 않는다고 가정한다. 그림에서 두 개의 순차적인 연산 4와 6, 4와 7, 그리고 3과 5가 각각 동일한 세부주기에 스케쥴링되어 있음을 알 수 있다. 이 예의 경우, 세부주기 2와 3에서 각각 3개의 연산이 수행되므로, 구현에 필요한 LUT의 개수는 3이 된다.

근본적인 차이점은 체이닝을 허용할 경우, 체이닝된 연산 사이의 에지는 세부주기 경계를 지나지 않으므로 마이크로 레지스터를 필요로 하지 않는다는 점이다. 따라서 하한 추정시 이러한 점을 고려하여야 한다. 일반적으로 임계경로가 C이고 전체 *context*의 수가 k인 경우, 적어도 $\lceil C/k \rceil$개의 순차적인 연산 또는 LUT가 동일한 주기에 수행될 수 있다. 따라서 각 연산 노드의 ASAP 스케쥴링 및 ALAP 스케쥴링시 이를 세밀히 고려하여 각 연산 노드 v 대한 세부주기 범위 $[S_v, L_v]$를 적절히 변경하고, 이 값들에 대해 동일한 하한 추정 기법을 적용한다.

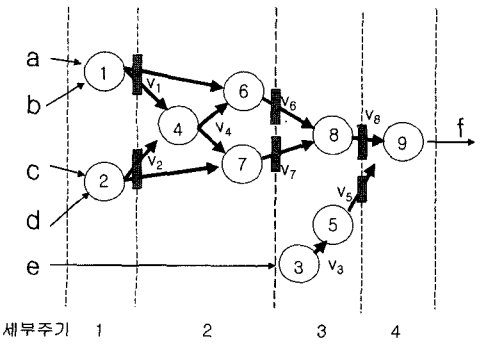

그림 7 임계경로 길이가 *context* 수보다 많아, 체이닝 기법을 적용한 예

이러한 체이닝의 허용은 그래프 상의 각 노드의 스케쥴링에 대해 보다 많은 가변성을 제공하기 때문에, 노드의 스케쥴링 결과에 따라 달라지는 에지들의 생명주기를 토대로 하한을 추정하는 마이크로 레지스터 추정을 한층 어렵게 만든다. 예를 들어, 그림 7의 경우에서 두 개의 순차적인 연산이 동일 세부주기에 스케쥴링 가능하므로, 모든 스케쥴링 가능성을 고려할 때, 세부주기 범위 $Z = [1, 3]$에서 한 세부주기 이상 반드시 저장되어야 하는 에지(신호)들은 전혀 없다.

4. 실험 결과 및 분석

본 논문에서 제안한 하한 추정 기법은 SUN 워크스테이션(Sun Blade-1000, 750MHz UltraSPARC III, 2,048MB)의 UNIX 운영체제(Solaris 2.8)에서 C 프로그래밍 언어로 구현되었다. 개발된 하한 추정 시스템의 성능을 검증하기 위해, MCNC 벤치마크를 대상으로 실험하였다. 단, 현재로선 본 시스템이 순차 회로에 대해서는 적용할 수 없는 한계성을 갖고 있기 때문에, 벤치마크 중 조합 회로만을 대상으로 실험하였다. 한편, 본 논문에서는 *context*의 수가 8인 시분할 FPGA를 주 고려 대상으로 삼았기 때문에, 최대 세부주기 수를 8(즉 k = 8)로 잡았으며, 임계경로의 길이가 8보다 큰 경우에는 자동적으로 최대 $\lceil$ (임계경로 길이) / 8) $\rceil$개의 연속적인 노드가 동일 세부주기에 체이닝된다고 가정하였다.

표 1은 기존의 연구 결과와 본 논문에서 제안한 하한 추정 결과 및 추정에 걸린 수행 시간을 비교하여 보여준다. 표에서 기존 합성 결과란 문헌 [17], [18]에서 발췌한 것으로서 기존 연구 FBP-m[7], PAT[17], ILP[18]의 합성 결과 및 합성에 소요된 수행 시간이다. 참고로, 기존 연구 결과의 CPU 소요 시간은 Pentium II PC(300MHz, 512MB)에서 측정되었으며, ILP 기법의 경우 Lindo 패키지가 사용되었다. 표 1에는 하한 추정에 대한 두 가지 결과가 제시되었는데, 하나는 *fanout* 제거나 에지 통합 등의 추가 향상 기법을 적용하기 전의 결과이며, 다른 하나는 이 향상 기법을 적용한 후의 결과이다.

실험 결과, 마이크로 레지스터의 개수에 대한 하한 추정 실험에서는 기존 연구의 합성 결과와 다소 큰 차이가 발생하였다. 이러한 결과는 기존의 LUT 개수에 대

한 하한 추정이 기존 합성 결과와 거의 모든 예에서 정확히 일치하는 것과는 대조적이다[11]. 표 1의 경우, 추정된 하한은 문헌상에 제시된 합성 결과 중 최상의 결과와 대비할 때 평균 34.5%에 해당하는 수치였다. 이러한 차이는 우선, 기존의 합성 결과는 LUT 개수를 적절히 유지하는 가운데 마이크로 레지스터를 최소화한 합성 결과인 반면, 본 하한 추정에서는, LUT 개수와는 무관하게, 마이크로 레지스터 개수를 가능한 작게 사용하는 경우를 추정하기 때문이라고 판단된다. 또한 한편으로는 LUT 개수에 대한 하한 추정에 비해 마이크로 레지스터 개수에 대한 하한 추정 문제 자체가 갖는 거대한 변동성과 복잡성으로 인해 제안한 추정 기법이 정밀도에 한계를 가지는 것으로 해석할 수 있으며, 다른 한편으로는 기존 연구 결과보다 더 좋은 합성 결과가 존재할 가능성이 높음을 의미하는 것으로 해석될 수 있다. 따라서 한편으로는 하한 추정의 정밀성을 높이기 위한 연구가 진행되어야 하며, 또 다른 한편으로는 기존 방법보다 성능이 개선된 새로운 합성 방법의 개발이 기대된다고 해석할 수 있다.

본 실험에서 한가지 흥미로운 사실은 두 가지 향상 기법을 추가 적용한 경우, 추정 결과가 향상되었을 뿐만 아니라, 대부분의 경우, 전체 수행 시간이 오히려 줄어들었다는 것이다. 표 1의 경우, 향상 기법 적용 전에 비해 적용 후의 수행시간이 평균 8.78초에서 5.74초로 약 38.7%정도 오히려 감소하였는데, 이는 두 가지 향상 기법이 추가 적용되어, 일정 CPU 시간을 소비하였음에도 불구하고, 문제의 크기를 줄여 오히려 전체 CPU 시간을 줄이는 효과를 냈다고 판단된다. 한편, 그림 9는 표 1의 결과 분석에 대한 이해를 돕기 위해 기존의 여러

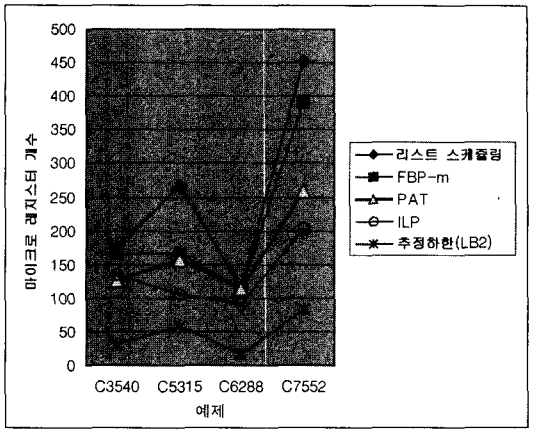

그림 8 최상의 실제 결과와 하한 추정 결과간의 연관성을 보여주는 그래프

합성 결과들과 추정 하한간의 상호 연관성을 그래프로 표시한 것이다.

한편, 실험 대상 벤치마크 회로 예들 중 표 1에 제시된 결과 이외의 다른 예들의 경우, 기존의 합성 결과가 문헌상에 제시되어 있지 않아 직접 비교하기 힘들다. 하지만, 향후 연구를 위해, 이들 예제에 대한 하한 추정 실험 결과를 표 2에 요약 정리하였다. 표 2의 예들의 경우, 하한 추정에 소요된 수행시간은 평균 1.20초이다.

5. 결 론

시분할 FPGA 합성 문제는 회로의 구체적인 분할 방법, 시간대별 회로 재구성 방법, 그리고 구체적인 회로 할당 방법에 따라 서로 다른 결과를 산출할 수 있다. 본

표 1 마이크로 레지스터 개수에 대한 하한 추정과 기존 합성 결과 비교표

예제	노드수 (|N|)	PI/PO 개수 (PIO)	저장할 신호수 (|E|-PIO)	임계 경로 길이 (C)	마이크로 레지스터 개수(개, 초), k = 8								하한추정 대비 최상결과 (LB2/B) (%)
					기존 합성 결과					하한 추정 결과			
					리스트 [4,7]	FBP-m [7]	PAT [17]	ILP [18]	최상 결과 (B)	향상 기법 적용전 (LB1)	향상 기법 적용후 (LB2)	향상기법 적용 전후 차이 (LB2-LB1)	
C3540	1,038	72	1,016	38	177	166 (0.31)	126 (1)	135 (3558)	126	28 (2.06)	31 (1.78)	3 (-0.28)	24.60
C5315	1,778	301	1,655	30	265	265 (0.78)	157 (7)	106 (4856)	106	55 (5.57)	58 (4.91)	3 (-0.66)	54.72
C6288	2,856	64	2,824	145	117	114 (0.13)	114 (2)	92 (6114)	92	13 (18.98)	17 (8.74)	4 (-10.24)	18.48
C7552	2,247	313	2,140	25	453	392 (0.54)	260 (17)	204 (5648)	204	78 (8.51)	82 (7.52)	4 (-0.99)	40.20
합계 (평균)	7,919	750	7,635	238	1,012	937 (0.44)	657 (6.75)	537 (5044)	528	174 (8.78)	188 (5.74)	14 (-3.04)	34.50

표 2 기존 문헌상에 비교 대상이 없는 벤치마크 예들에 대한 하한 추정 실험 결과

예제	노드수 (|N|)	PI/PO 개수 (PIO)	저장할 신호수 (|E|-PIO)	임계 경로 길이 (C)	마이크로 레지스터 개수에 대한 하한 추정 결과(개) 및 수행 시간(초), k = 8		
					향상기법 적용전 (LB1)	향상기법 적용후 (LB2)	향상기법 적용 전후 차이 (LB2-LB1)
C2670	924	221	860	33	20 (1.35)	21 (1.17)	1 (-0.18)
C880	390	86	364	24	14 (0.18)	17 (0.15)	3 (-0.03)
alu4	590	22	584	11	36 (0.65)	40 (0.54)	4 (-0.11)
dalu	605	91	589	6	94 (0.62)	94 (0.52)	0 (-0.10)
des	1,437	501	1,192	7	176 (4.21)	177 (3.87)	1 (-0.34)
i10	1,508	481	1,322	16	83 (4.23)	111 (3.79)	28 (-0.44)
i9	212	151	149	5	30 (0.13)	30 (0.14)	0 (+0.01)
i8	632	214	551	6	86 (0.91)	87 (0.88)	1 (-0.03)
k2	1,177	88	1,144	6	179 (2.83)	179 (2.82)	0 (-0.01)
t481	408	17	407	8	144 (0.18)	153 (0.10)	9 (-0.08)
vda	301	56	275	6	40 (0.20)	40 (0.19)	0 (-0.01)
x3	419	234	322	5	47 (0.24)	47 (0.24)	0 (0.00)
합계 (평균)	8,603	2,162	7,759	133	949 (1.31)	996 (1.20)	47 (-0.11)

연구에서는 기존에 본 연구진이 개발한 바 있는 상위 단계 합성용 하한 추정 알고리즘을 수정하여 시분할 FPGA 합성 문제에 적용하였다. 이러한 하한 추정 도구를 통해, 칩의 주요 자원인 LUT는 물론, 마이크로 레지스터의 개수에 대해서도 기존의 합성 결과에 대한 최적성 분석 및 결과 향상 가능성 분석 등이 어느 정도 가능하게 되었다.

실험 결과, 몇가지 향상 기법을 적용하였음에도 불구하고, 기존의 실제 합성 결과와 다소 큰 차이를 보인다. 이러한 차이는 기존의 합성 기법에서는 LUT 개수를 적절히 유지하는 가운데 마이크로 레지스터를 최소화한 반면, 본 하한 추정 기법에서는 LUT 개수와는 무관하게 마이크로 레지스터 개수를 가능한 작게 사용하는 합성 예를 추정하는데 기인한 것으로 판단된다. 또한 LUT 개수에 대한 추정 기법과는 달리, 마이크로 레지스터에 대한 하한 추정 기법이 그 문제 자체가 갖는 엄청난 가변성 및 복잡성으로 인해 추정의 정밀성에 한계를 가짐을 의미한다. 특히, 시분할 FPGA의 경우, *context* 수의 제한에 따라 체이닝을 고려하여야 하므로, 문제의 복잡성이 한층 높아진 것으로 추정된다. 그러나, 다른 한편으로는 기존 연구 결과보다 더 좋은 합성 결과가 존재할 가능성이 매우 높은 것으로 해석될 수 있다.

본 연구는 앞으로 해결해야 할 여러 과제들을 안고 있다. 우선, 마이크로 레지스터에 대한 하한 추정의 정밀도를 높이기 위한 연구가 필요하다. 또한 시분할 FPGA 합성 연구에서는 조합 회로는 물론 순차 회로도 고려해야 하는 반면, 본 논문에서는 하나의 클록 내에서 동작하는 조합 회로만을 대상으로 하였기 때문에, 순차 회로에 대해 실험한 기존 연구 결과에 대한 비교 분석이 제대로 이루어질 수 없었다. 향후, 조합 회로는 물론 순차 회로의 경우에도 적용할 수 있도록 본 논문에서 제안한 하한 추정 기법을 확장하는 연구가 필요하다. 한편, LUT 개수의 최소화와 마이크로 레지스터 개수의 최소화는 경우에 따라서는 동시에 만족될 수 없는 상호 배타성이 있는 바, 하한 추정시 이에 대한 *trade-off*를 고려하는 방법의 개발도 필요하다.

참 고 문 헌

[1] R. Murgai, K. Brayton, and A. Sangiovanni-Vincentelli, "Logic Synthesis for Field Programmable Gate Arrays," Kluwer Academic Publisher, 1995.

[2] Kang Yi, Seong Y. Ohm, and Chu S. Jhon, "An Efficient FPGA Technology Mapping Tightly Coupled with Logic Minimization," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E80-A, pp. 1807-1812, Oct. 1997.

[3] S. Timberger, "A Time-Multiplexed FPGA," Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines, 1997.

[4] S. Trimberger, "Scheduling Designs into a Time-Multiplexed FPGA," Proceedings of International Symposium on the Field Programming Gate Array, pp. 153-160, 1998.

[5] D. Chang and M. Marek-Sadowska, "Buffer Minimization and Time-Multiplexed I/O on Dynamically Reconfigurable FPGAs," ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 142-148, 1997.

[6] D. Chang and M. Marek-Sadowska, "Partitioning Sequential Circuits on Dynamically Reconfigurable FPGAs," ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 161-167, 1998.
[7] H. Liu and D.F. Wong, "Network Flow Based Circuit Partitioning for Time-Multiplexed FPGAs," Proceedings of International Conference on the Computer Aided Design, pp. 497-504, 1998.
[8] H. Liu and D.F. Wong, "Circuit Partitioning for Dynamically Reconfigurable FPGAs," ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 187-194, 1999.
[9] H. Liu and D. F. Wong, "A Graph Theoretic Optimal Algorithm for Scheduling Compression in Time-Multiplexed FPGA Partitioning," Proceedings of International Conference on the Computer Aided Design, pp. 400-405, 1999.
[10] Seong Y. Ohm, Fadi J. Kurdahi, Nikil Dutt, "A Unified Lower Bound Estimation Technique for High-Level Synthesis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 16, No. 5, pp. 458-472, May 1997.
[11] 엄성용, "시분할 FPGA 합성에서 LUT 개수에 대한 하한 추정 기법", 정보과학회 논문지, 제29권 제7/8호, pp. 422-430, 2002년 8월.
[12] Seong Y. Ohm, Chu S. Jhon, and Fadi J. Kurdahi, "An Optimal Scheduling Approach using Lower Bound in High-Level Synthesis," IEICE Transactions on Information and Systems, Vol. E78-D, No.3, pp.231-236, March 1995.
[13] C. T. Hwang, J. H. Lee, and Y. C. Chu, "A Formal Approach to the Scheduling Problem in High Level Synthesis," IEEE Transactions on Computer-Aided Design, pp. 464-475, April 1991.
[14] P. G. Paulin and J. P. Knight, "Force-directed Scheduling for the Behavioral Synthesis of ASIC's," IEEE Transactions on Computer-Aided Design, pp. 661-679, June 1989.
[15] M. C. McFarland, A. C. Parker, and R. Composano, "Tutorial on High Level Synthesis", Proceedings of the 25th Design Automation Conference, pp. 330-336, June 1988.
[16] Hyun-Chul Shin and Chung-Hee Kim, "A Simple Yet Efficient Techniques for Partitioning," IEEE Transactions on VLSI Systems, Vol. 1, No. 3, pp. 380-386, 1993.
[17] M. C. Chao, G-M. Wu, I. U.-R. Jiang, and Y-W. Chang, "A Clustering- and Probability-based Approach for Time-multiplexed FPGA Partitioning," Proceedings of International Conference on the Computer Aided Design, pp. 364-368, 1999.
[18] Guang-Ming Wu, Jai-Ming Lin, and Yao-Wen Chang, "Generic ILP-based approaches for time-multiplexed FPGA partitioning," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 20 No. 10, pp. 1266-1274, Oct. 2001.

엄 성 용
정보과학회논문지 : 시스템 및 이론
제 30 권 제 3 호 참조