최적 모듈 선택 아키텍쳐 합성을 위한 저전력 Force-Directed 스케쥴링에 관한 연구

최 지 영, *김 희 석
제천기능대학 정보통신설비과, *청주대학교 전자공학과
cjy03@kopo.or.kr, *khs8391@chongju.ac.kr

A Study on Low Power Force-Directed scheduling for Optimal module selection Architecture Synthesis

Ji-young Choi, *Hi-seok Kim
Dept. of Information & Communication, Jecheon Polytechnic College
*Dept. of Electronic Engineering, Chongju University
E-mail : cjy03@kopo.or.kr, *khs8391@chongju.ac.kr

Abstract

In this paper, we present a reducing power consumption of a scheduling for module selection under the time constraint.

A a reducing power consumption of a scheduling for module selection under the time constraint execute scheduling and allocation for considering the switching activity. The focus scheduling of this phase adopt Force-Directed Scheduling for low power to existed Force-Directed Scheduling. and it constructs the module selection RT library in account consideration the mutual correlation of parameters in which the power and the area and delay. when it is, in this paper we formulate the module selection method as a multi-objective optimization and propose a branch and bound approach to explore the lager design space of module selection. Therefore, the optimal module selection method proposed to consider power, area, delay parameter at the same time.

The comparison experiment analyzed a point of difference between the existed FDS algorithm and a new FDS_RPC algorithm.

I. 서 론

1)최근에 접어들어 다양한 형태의 디지털 시스템의 휴대화에 대한 수요가 급증하게 되었다. 특히, 휴대용 장치에 있어서 크기나 중량과 더불어 중요한 요소를 차지하는 베터리 용량은 집적 회로가 소모하는 전력에 직접적으로 비례하게 되었다. 그리고 전력 손실이 큰 회로에 대한 packaging/cooling 비용의 상승으로 저 전력 소모를 위한 디지털 시스템 설계의 필요성이 요구되고 있다. 그러나 최근까지는 이러한 추세와는 달리 디지털 시스템의 설계에 있어서 주된 관심은 대상 회로의 속도의 증가와 성능 측면에 중점을 두어 왔으며, 상위 수준에서의 저 전력 소모를 지원하기 위한 연구가 최근 활발히 진행되고 있는 실정이다. 일반적으로 CMOS 회로에서의 전형적인 전력 소모의 요인으로 스위칭 활동(switching activity), 누설전류(leakage current), 폐회로 전류(short-circuit current)등에 의하며, 이들 중 스위칭 동작에 의한 전력 소모가 약 90% 이상으로 가장 큰 비중을 차지한다.[1-4]. CMOS 회로에서는 데이터의 스위칭 동작이 발생하지 않을 경우 전력 손실이 없으므로, 저 전력 회로 설계에 있어서 최소의 스위칭 동작을 허용하는 것이 중요한 관건으로 적용된다. 현재 저 전력 설계를 위해 공급 전압의 감소, 스위칭 동작의 최소화 등을 통한 여러 가지 설계 방식을 제안하고 있다. 앞에서도 언급했듯이, 전력이 공급 전압의 제곱에 비례하는 관계로, 공급 전압의 감소는 큰 전력 감소를 초래할 수 있다. 하지만, 이때 공급 전압의 감소로 인해 지연 시간은 증가한다. 이런 이유로 인해 전력 감소를 위해 여러 변환 기법을 이용하여 회로의 성능을 높인 후, 원래의 성능 제한 조건을 위반하지 않는 범위 한도 내에서 전압을 낮춘다. [5-8] 또한 과거에는 VLSI 설계에서 주 고려사항은 성능 및 비용 신뢰성 면적이었다. 반면 오늘날에는 무선통신 시스템, 음성 및 비디오를 기초로 한 멀티미디어 제품 휴대용 데스크탑과 같은 개인용 컴퓨터의 성장은 휴대용을 요구한다. 모든 휴대용 장치들은 고속의 계산과 복잡한 기능뿐만 아니라 저 전력 소비를 요구한다. 결론적으로, 전력의 고려는 오늘날 VLSI 설계에서 지배적인 것으로 되고 있다.

본 논문의 구성은 다음과 같다. 서론에 이어 2장에서는 최적 모듈 선택 아케텍처를 위한 전력감소 FDS_RPC 에 대해 기술하고, 3장에서는 최적의 모듈 선택 방법을 제시하고 4장에서는 기존의 FDS 와 본 논문에서 제안한 FDS_RPC 를 비교 실험하였고, 마지막으로 결론으로 구성되어 있다.

II. 최적 모듈 선택 아키텍쳐 합성을 위한 전력 감소 FORCE-DIRECTED 스케쥴링

최적 모듈 선택을 위한 아키텍처 합성을 위한 전력 감소 스케쥴링은 먼저 첫 번째 단계에서는 전력 감소를 위한 스케쥴링으로 스위칭 활동(switching activity)을 고려하여 스케쥴링 및 할당 바인딩을 수행한다. 이 단계에서의 스케쥴링의 주안점은 기존의 FDS 스케쥴링을 저전력으로 고려한 FDS 방법을 적용한다. 그리고 두 번째 단계에서는 전력, 면적, 지연의 매개 변수의 상호 상관관계를 고려하여 RT 라이브러리를 구축한다. 이때 본 논문에서 제안한 모듈선택 방법은 다중 목적 최적화 방법으로 Branch and Bound 접근 방식을 제안한다. 그러므로 최적의 모듈 선택 방법은 동시에 전력, 면적 지연을 동시에 고려하여 최적화 하는 알고리듬을 제안한다. 또한 전력 부분에서는 전체적인 전력 소모를 스위칭 활동을 기반 하에 두고 다중 전압을 고려하였을 때의 전력도 구현한다. 일반적 모듈 선택 및 전력 라이브러리 및 별도로 취급하여 구축되어왔다. 반면 본 논문에서

1) 본 연구는 과학기술부 · 한국과학재단 지정 청주대학교 정보통신연구센터의 지원에 의한 것입니다.

는 이런 모든 조건을 고려하여 최적의 조건을 만족할 수 있는 RT 라이브러리를 구축하였다. 기존의 전력 라이브러리는 전력 예측 모델을 기반으로 스위칭 활동량이 가장 적은 쪽의 선택하여 구축하였다. 그림1은 RT 라이브러리를 통한 상위 레벨 합성 과정을 나타낸다.

2.1 기존의 FDS 알고리듬

FDS 알고리듬은 ILP based 스케줄링 알고리듬는 달리 heuristic method 로 시간 제약 조건하에서 사용하는 functional unit 수를 최소화 하는 것을 목적으로 한다. 먼저 주어진 DFG에 대해 ASAP 와 ALAP를 수행하여 각 연산에 대해 S_i 와 L_i 를 얻은 후에, 각각의 연산에 대한 타임 프레임과 연산 종류에 따른 분포 그래프를 구성한다. 타임 프레임으로부터는 각각의 연산들이 할당될 수 있는 제어 단계의 범위를 알 수 있는데 이것을 mobility 라고 한다. 그리고 이 분포 그래프를 바탕으로 제어 단계 j에서 각각의 연산의 self-force를 다음과 같이 정의한다.

$$Self-Force(j) = \sum_{i=S_i}^{L_i} Force(i) \quad \text{..... 식(1)}$$

$$Force(i) = DG(i) \cdot x(i) \quad \text{.....식(2)}$$

$$DG(i) = \sum_{O_j \in OpType} Prob(O_j, i) \quad \text{.....식(3)}$$

식(2)에서 DG(i) 는 제어 단계 i에서 해당 연산 종류가 존재할 확률이고, 식(3)의 Prob(O_j, i) 는 연산 종류가 $OpType$ 인 연산이 제어 단계 i 에 존재할 확률이다. 그리고 x(i)는 연산을 제어 단계 i 에 두었을 때 해당 DG(i)의 변화율로서 i 와 j가 같으면 양의 값이고 다르면 음의 값이다. 하나의 연산을 특정 제어 단계에 할당하면 CDFG 상에 연결된 다른 연산들의 타임 프레임을 변화시키므로, CDFG 상에 연결된 다른 연산의 영향을 고려하여 식(1)의 self-force 외에 predecessor-force 와 successor-force를 함께 구하여 total-force를 정의한다. Predecessor-force 는 현재의 연산의 스케줄링에 영향을 받는 상위 제어 단계의 연산들에 대한 force 이고, successor-force 는 반대로 하위 제어 단계의 연산들에 대한 force이다.

total_force(i) = self_force(i) + Predecessor_force(i)
+ successor_force(i).....식(4)

이와 같이 각각의 연산에 대해 total-force를 구한 후, total-force 가 가장 작은 제어 단계에 연산을 할당한다. 하나의 연산이 제어 단계에 할당되면 새로운 타임 프레임과 분포 그래프가 생성되고 이를 토대로 다시 각각의 연산에 대해 total-force를 구한다. 이와 같은 과정을 마지막 연산이 스케줄링 될 때까지 반복한다. FDS 는 ILP based 스케줄링과는 달리 heuristic 방법이므로 그 수행 속도가 빠르고 각 연산 종류마다 제어 단계상에서 존재하는 확률이 일정하도록 스케줄링하기 때문에 스케줄링 결과도 상당히 우수하다.

2.2 새로운 FDS_RPC 알고리듬

일반적인 스케줄링 방식인 FDS를 저전력 차원에서 FDS 알고리듬을 제안하였다. 먼저 첫 번째 모듈 내부의 switching activity 가 스프링 장력 상수 K와 관련 있다. 기존의 FDS 는 자원(type-distribution)을 위한 다투는 연산 사이의 경쟁과 연관 있다. 두 번째, 자원 공유 선택의 확률을 디스플레이먼트 (displacement) x 와 상관있다. 기존의 방법에서는 연산자 간이동도 변화에 관계했다. 세 번째, 부분적 스케줄링 정보를 이용하기 위해 연산자 사이의 수월한 자원 공유 확률을 결정은 알고리듬 초기에 임계 연산자로 스케줄링한다. 표 1는 force 매개 변수의 분석의 차이점을 설명한다. 그리고 그림 13는 제안한 FDS_RPC 알고리듬을 나타낸다.

표 1. force 매개 변수 분석

알고리듬	스프링 상수 K	변위 X	Force
FDS	개별적 자원의 합	연산자 이동도내의 변화	연산자 병행성
FDS_RPC	모듈의 스위칭 활동	조합 선택이 확률	동적 파워

```
Algorithm FDS_RPC(DFG G, primary input trace VT
 Profile DFG with VT
 latency_constraint ← critical_pathlength(G)
 determine_time_frame(ops ∈ G)
 if for all op ∈ G : OP ∈ ciritcal_ops) then
      op.tstep ← critical_timestep
 while unscheduled ops exist do
   if(op.testp = UNKNOWN) then
   for each t ∈ time_frame(op) do
     evaluate_vaild_combination(op,t)
     foreach c ∈ vaild_combination do
       p ← probability_product(c)
       swact ←switched_activity(c)
       force(op,t,c) ←p * swact
       max_force ← determine_max_force(op,t,c)
       for each op ∈ G do
        for each t ∈ time_frame(op) do
          force_norm(op,t,c)←normalize force(op,t,c) write max_force
compute
          mean(op,t), std_dev(op,t), skew(op,t) of force_norm(op,t)
          total_force(op,t) ← mean(op,t) +std(op,t) + skew(op,t)
          min_total_force(op,t) ←determine_minimum_total_force(op,t)
       op.tstep ← t and update G
  endwhile
  return G
end algorithm
```

그림 2. FDS_RPC 알고리듬

III. 최적의 모듈 선택 방법

물리적 정전 용량을 영향을 주는 상위 수준 합성 단계의 대표적인 것이 모듈 선택 단계이다. 기능연산자, 메모리, 레지스터, 멀티플렉서 그리고 버퍼들의 물리적 정전용량은 모두 모듈 선택 단계에서 라이브러리로부터 선택된 하드웨어 종류에 의존한다. 일반적으로 빠른 하지만 상대적으로 물리적 정전 용량이 큰 그러한 유닛들은 속도가 매우 중요시되는 그러한 회로에서 이용되어진다.

저전력 설계의 아키텍처 합성은 전력, 면적, 지연의 매개 변수에 상호 독립적이기 때문에 최적화 문제가 복잡하다. 제안한 모듈선택 방법은 다중 목적 최적화 방법으로 Branch and Bound 접근 방식을 제안한다. 그러므로 최적의 모듈 선택 방법은 동시에 전력, 면적 지연을 동시에 고려하여 최적화 하는 알고리듬을 제안하였다. 일반적인 모듈 선택 및 전력 라이브러리 및 별도로 취급하여 구축되어왔다. 반면 본 논문에서는 이런 모든 조건을 고려하여 최적의 조건을 만족할 수 있는 RT 라이브러리를 구축하였다. 기존의 전력 라이브러리는 전력 예측 모델을 기반으로 스위칭 활동량이 가장 적은 쪽의 선택하여 구축하였다.

3.1 모델 및 표현 (Models and Representations)

1) RT 라이브러리

우리는 라이브러리를 표1과 같이 표시된 것을 채택하였다. 각 아키텍처에 적합한 연산의 전압, 지연, 면적, 전력을 합한 정보를 표현한다. 평균 스위치 활동은 PRESTO [Agr96]을 이용하여 일반화 하였고 컴포너트는 0.6 마이크론 3.3 V VTI표준 셀 CMOS 라이브러리를 이용하였다. 또한 이와 같은 컴포넌트들은 단지 2 입력 NAND 와 NOR 그리고 인버터만으로 합성되었다. 라이브러리 고정 연산 집합을 고정하여 두고, 또한 클럭 사이클은 최적화된 파라메터로서 적용할 수 있게 하였다. 또한 일반적인 규칙으로는, 설계의 클럭 주기가 감소함에 따라 좀 더 많은 사이클의 연산의 컴포넌트들이 느리게 연산되는 원인이 된다.

그리고 재사용 가능한 RT 라이브러리의 예를 들면 그림 4 에

표 2. RT 라이브러리

	Module Type	Voltage (V)	Delay (ns)	area (u_2)	power (㎽)
+	ripple-carry adder(RCA)	3.3	150	1258	5.4
		5	80	1258	22.7
	carry lookahead adder(CLA)	3.3	80	6598	10.5
		5	40	6598	37.3
*	booth multiplier (Booth)	3.3	320	16455	30.7
		5	145	16455	84.0
	array multiplier (Array)	3.3	160	32090	143.1
		5	100	32090	295.6

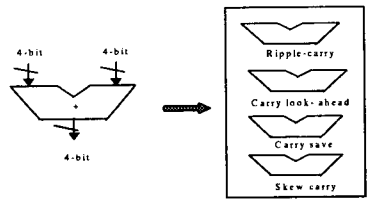
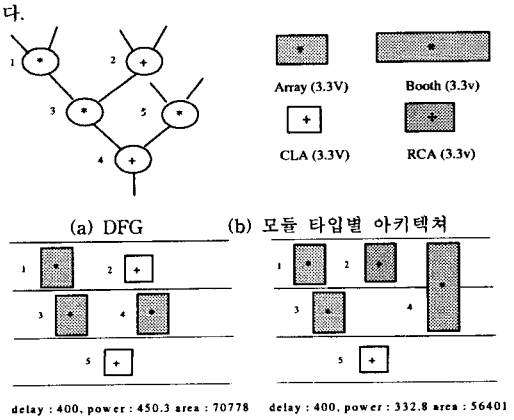


그림 4. RT 컴포넌트 선택 아키텍처

서와 같이 같은 가산 연산이라 하더라도 각 구조에 따라 모듈 선택을 달리 선택하여야 한다. 그림 4는 RT 컴포넌트 선택 아키텍처를 나타낸다.

또한 아키텍처 선택 과정에서 예를 들어 그림 5와 같은 (a) DFG 와 (b) 모듈 타입별 아키텍처의 주어진 제약으로 전력 및 특성이 있을 경우 가산(adder)연산을 리플 캐리 가산(ripple carry adder), 캐리 룩어 헤드 가산(carry-look ahead adder), 스큐 캐리 가산(skew carry adder) 등 여러 아키텍처 구조의 가산 모델링을

라이브러리로 선택하고 재사용 하는데 있어 앞에서도 언급했지만 예를 들어, 보통의 공급 전압 하에서의 RCA 는 조금 속도가 느리기는 하지만, 적은 물리적 정전 용량으로 인해 CSA에 비해 에너지 측면에서 효율적이다. 하지만 저 전압 환경으로 감에 따라 그것은 속도 제한을 도저히 만족할 수 없게 된다. 이러한 경우는 물리적 정전 용량이 다소 크더라도 CSA를 사용하는 목적이 여기에 있다. 이렇듯 주어진 전력 감소 제약 특성에 맞게 나타낸 아키텍처 선택을 (c) 와 (d)에서 비교할 수 있다. 일반적으로 스케쥴링 제어스텝(Control Step)은 동일하고 전력 및 면적 면에서 (d)의 아키텍처가 저 전력에 유리한 구조를 나타낸다. 그림 5는 모듈에서의 아키텍처 선택을 다음과 같이 나타낸다.



(c) 아키텍처 1 (d) 아키텍처 2

그림 5. 아키텍처 선택

2) 모듈 선택 용어 정의

(정의 1) A Data Flow Graph (DFG) : 방향성 그래프 G(V, E) 로서 V는 연산의 집합이고 E는 연산간의 의존성을 표현한 에지의 집합이다.

(정의 2) A Type : 연산자 V를 나타낸다. τ(v)는 연산을 수행하는 함수이다. 예를 들어 위의 예제의 타입은 덧셈과 곱셈을 나타낸다.

(정의 3) A Module : 가산기는 덧셈으로 승산기는 곱셈 등으로 연산의 종류를 하나 또 그 이상으로 수행할 수 있는 연산을 나타내며, 반면에 ALU는 모두 수행한다.

(정의 4) A module Library : 모듈의 집합으로 M으로 정의된다. 위의 표는 모듈 라이브러리 예제를 나타낸다.

(정의 5) A Module support set : 연산을 v, 연산종류를 τ(v)로 표현하며, Ψ(v), Ψ(τ)은 연산을 수행할 수 있는 모든 모듈의 집합이다.

예를 들어, 위의 표의 모듈 라이브러리는 Ψ(+) = {RCA, CLA}, Ψ(*) = {ARRAY, BOOTH} 이다.

(정의 6) A module mapping (configuration) : $f : X \rightarrow M$, $X \subseteq V$, 은 X 안의 각 연산들은 수행하고 모듈을 할당한다. 만약 $X \subset V$ 이면 f는 부분적 모듈 매핑이라 불리어진다. 그리고 $X = V$ 이면 완전 모듈 매핑이라 한다. 또한 f^{-1} 는 f에 의해 매핑 되어진 연산자들의 집합을 의미한다. 예를 들어 X = X1 + X2, X1 ∩X2 = ∅,은 두개의 모듈 매핑은 g 와 h 로 병합된 후 X1과 X2로 정의된다. 또한 f = g ⊗ h 로 표현된다. 주어진 모듈 매핑 f 는 비용함수로 정의 할 수 있다.

(정의 7) Area_LB(f), Delay_LB(f), Power_LB(f) 은 상대적으로 f의 면적, 지연, 전력비용의 하한 경계를 나타낸다.

그러므로 모듈 선택은 주어진 최적의 목적을 가진 모듈 라이브러리로 주어진 DFG을 위한 최적의 모듈 매핑을 찾는 문제이다.

3) 모듈 선택 알고리듬 방법

이전에서도 언급했듯이 전체적인 최적의 다중 목적을 고려한 아키텍처에 도달하기 위해서는 모듈 선택의 전체 설계 영역을 조사해야 한다. 위에서 정의한 모듈 선택의 조건에 따라 다중 목적 Branch and Bound 알고리듬을 이용하여 최적의 모듈 선택을 휴리스틱 방법으로 위 조건의 모듈을 선택해서 최적의 결과를 얻은 모듈 선택을 찾는다. 그림 6은 모듈 선택 및 합성 알고리듬을 나타낸다.

```
Main()
{   Bound ← A vector of Maximum numbers
    v ← An operation which has no predecessor
    X ←(v) ; f ← ∅
    Call ModuleSelection (X, f, v, Bound)
    For each f in the feasible configuration list
    { If LB(f) - Bound > δ Delete f /* Re-pruning */
    For each f in the feasible configuration list
       {  Doing Scheduling and Allocation on f
          Find the optimal architecture
       }
       target architecture ← the optimal architecture
 }

 Proc ModuleSelection(X, f, v, Bound)
 { If X = ∅ Stop
   Repeat
     If all the value of Ψ(v) have been tried
      { X ← X| {v} ; Return /* Backtrack */ }
      Else {
             g(v) ← next un-tried value of Ψ(v)
             f ← f ⊗ g(v)
             IF LB(f) - Bound > δ Continue /* Pruning the branch
*/
             Else { X ← X ∪ {v}
                IF f is a full mapping
                { feasible configuration list ← f
                  If LB(f) < Bound  Bound ← LB(f) }
                  Else { v ← next un-visited operation in V
```

```
        Call ModuleSelection( X, f, v, Bound)
          /* Depth First Search */ )
      )
    )
  Until( all the value of Ψ(v) hace been tried)
)
```

그림 6. 모듈 선택 및 합성 알고리듬

Ⅳ. 실 험 결 과

제안한 최적 모듈 선택 아키텍쳐 합성을 위한 전력 감소 알고리듬은 C 언어로 SUN SPARC 워크스테이션에서 구현되었다. 본 논문에서 비교 실험한 벤치마트는 첫 번째 Testexample으로 테스트 스케쥴링 알고리듬을 위한 전형 모형 DFG로 5개의 덧셈과 2개의 뺄셈, 15개 에지로 구성되어 있는 예제이고, 두 번째는 FIR(Second Order Finite Impluse Response filter)필터로 5개의 곱셈과 4개의 덧셈, 19개 에지로 구성된다. 세 번째는 IIR(Second Order Infinite Impluse Reponse filter)필터로 5개의 곱셈과 4개의 덧셈, 19에지로 구성되어 있다. 마지막으로 Paulin는 6개의 곱셈과 3개의 덧셈, 15개 2개의 뺄셈, 25에지로 구성되어 있다. 본 알고리듬은 DFG로 입력을 받아 기존의 FDS 방식과 본 논문에서 제안한 FDS_RPC 알고리듬을 비교 평가하였다. 표3는 시간 제약과 사용된 자원의 수를 나타내는 표이다. 여기에서는 결과는 기존의 FDS 나 제안한 FDS_RPC 알고리듬은 동일한 제어 스텝의 결과를 얻었다. 기능 연산자(functional unit)와 멜티플렉서(multiplexer)간의 절충(tradeoff)을 제외하고는 레지스터(register) 개수는 동일한 결과 값을 보였다.

표 3. 시간제약에 따른 제어 스텝과 사용된 자원의 수 비교

벤치마크	제어스텝		사용된 자원					
	FDS	FDS_RPC	FDS			FDS_RPC		
			Reg	FUs	Mux	Reg	FUs	Mux
Test	4	4	8	3	7	8	4	5
FIR	5	5	10	3	6	10	4	8
IIR	5	5	10	5	10	10	6	9
Paulin	4	4	14	4	11	14	5	10

또한 데이터 환경은 ARMA (Autoregressive moving average) 신호를 사용해서 모델링되었다. ARMA 모델은 일반화된 데이터 스트림(data stream)안의 특정한 정확성을 설명하는데 사용되어지며 널리 응용되어지고 있다. [17][18] 본 논문의 비교실험에서는 2개의 ARMA 방정식 SIG 1 : Υ(n) - 0.5x(n-1)과 SIG 2 : Υ(n) + 0.1x(n-1)을 데이터 환경으로 채택하였다. 이런 ARMA 방정식은 오디오와 스피치 신호에 사용된다. [18] 표4과 표5는 각각의 방정식에 따른 파워 세이빙(Power saving)의 양을 나타내는 표이다. 동일한 환경 안에서의 실험에도 불구하고 벡터 집합의 길이가 다양한 결과 값을 나타냈다. 여기에서는 FDS의 전제하에서 실험하였다.

표6는 FDS_RPC 기반으로 한 부분적인 설계 파워 세이빙으로 설계 단계의 최대 파워 세이빙은 23.9 %를 얻을 수 있었고, 반면에 최소 10.9% 얻었다. 또한 모든 벤치마크 평균 수치는 16.4 %가 되었다.

표4. SIG 1 환경으로부터의 데이터 흐름을 위한 파워 세이빙

벤치마크	파워세이빙(%), 길이(Length)		
Test	22.5% (44)	23.8% (67)	23.9% (106)
FIR	11.1% (31)	11.5% (52)	11.0% (86)
IIR	9.3% (34)	10.9% (50)	9.3% (81)
Paulin	18.9% (15)	19.3% (49)	19.3% (103)

표5. SIG 2 환경으로부터의 데이터 흐름을 위한 파워 세이빙

벤치마크	파워세이빙(%), 길이(Length)		
Test	22.5% (44)	23.8% (67)	23.9% (106)
FIR	11.1% (31)	11.5% (52)	11.0% (86)
IIR	9.3% (34)	10.9% (50)	9.3% (81)
Paulin	18.9% (15)	19.3% (49)	19.3% (103)

표 6. 설계레벨, 모듈 레벨 그리고 컨트롤 레벨의 파워 세이빙

벤치마크	Design	Reg	FUs	MUX	Control
Test	23.9%	5.7%	59.3%	61.4%	15.2%
FIR	11.5%	8.3%	7.4%	26.3%	8.5%
IIR	10.9%	12.2%	8.0%	13.8%	8.1%
Paulin	19.3%	2.0%	39.0%	44.2%	12.3%
평균 세이빙	16.4%	7.1%	28.5%	36.7%	11.1%

Ⅴ. 결 론

본 논문은 시간 제약 조건하에서의 모듈 선택을 이용한 전력 감소 스케쥴링을 제안하였다.

시간 제약 조건하에서의 모듈 선택을 고려한 전력 감소 스케쥴링은 먼저 첫 번째 단계에서는 전력 감소를 위한 스케쥴링으로 스위칭 활동을 고려하여 스케쥴링 및 할당 바인딩을 수행한다. 이 단계에서의 스케쥴링의 주안점은 기존의 FDS 스케쥴링을 저 전력으로 고려한 FDS 방법을 적용한다. 그리고 두 번째 단계에서는 전력, 면적, 지연의 매개 변수의 상호 상관관계를 고려하여 RT 라이브러리를 구축한다. 이때 본 논문에서 제안한 모듈선택 방법은 다중 목적 최적화 방법으로 Branch and Bound 접근 방식을 제안한다. 그러므로 최적의 모듈 선택 방법은 동시에 전력, 면적 지연을 동시에 고려하여 최적화 하는 알고리듬을 제안했다. 비교 실험에서와 모듈 선택을 고려한 새로운 FDS_RPC 알고리듬과 기존의 FDS 알고리듬의 간의 차이점을 비교 분석하였다.

향후 연구과제로는 모듈 선택을 전력 감소 스케쥴링의 알고리듬을 토대로 레지스터 전송 단계까지의 확장된 다양한 벤치마크와 수행이 선행되어야 하겠다.

참 고 문 헌

[1] R. Hartley, "Behavioral to Structural Translation in a Bit-Serial Silicon Compiler," IEEE Trans. CAD, vol. 7. no. 8, Aug. 1988, pp.877-886

[2] A. Chandrakasan, R. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits," IEEE Proceedings, vol. 83, no. 4, April 1996, pp.498-523

[3] A. Chandrakasan et al., "Low-Power CMOS Digital Design," J. Solid-State Circuits, vol.27, no.4, April 1992, pp.473-484

[4] A. Ghosh, " Estimation of Average Switching Activity in Combination and Sequential Circuits", in Proc. 29th DAC, June 1992, pp.253-259

[5] P. Landman, " Power Estimation of High-Level Synthesis", in Proc. European DAC, Feb. 1993, pp.361-366

[6] A. Chandarksan et al., "HYPER-LP: A System fo Power Minimization Using Architecture Transformation," in Proc. ICCAD, Nov. 1992, pp.300-303

[7] R. Martin, "Power-Profiler : Optimizing ASICs Power Consumption at the Behavioral Level," in Proc. 32nd DAC, June 1995, pp.42-47

[8] J. Chang, "Register Allocation and Binding for Low Power", in Proc. 32nd DAC, June 1995, pp.29-35