지구물리 제 4권 제3호, p. 163-173 (2001년 9월)

스케듈링 기법 연구

송정영 · 백남우§
배재대학교 정보통신공학부, §청주기능대학 생산자동화과

Song, Jeong-Young Back, Nam-Woo, Scheduling Technique for Control Step

ABSTRACT : This paper deals with the scheduling problems which are the most important subtask in High-Level Synthesis. Especially, we have concentrated our attentions on the data-path scheduling which can get the structural informations from the behavioral algorithm as a first step in synthesis procedure.

Suggest Forward scheduling methode is executed the ASAP and ALAP scheduling to use the fifth - order elliptic wave filter of a standard benchmark model, and then it is drawing up T.N matrix table by the number of resource and control-step, using the table extract of the simple than down-limit value of the control-step for the number of given resource to use this table.

All of existing list scheduling techniques determine the priority functions first, and then do the operation scheduling, But, the suggested forward scheduling technique does the schedule first, and determines the priority functions if needed in scheduling process.

Ⅰ. 서 론

자동 설계용 툴을 개발하기 위해 전 설계 과정을 영역별로 표현하면 1) 하위(Physical) 영역의 합성기술 2) 중위(Logical)영역의 합성 기술 3) 상위(Behavioral & Structural)영역의 합성기술로 표현될 수 있으며, 하위영역의 합성기술에서는 공간분할, 배치, 경로등이 다루어지고, 중위영역의 합성기술에서는 RT수준(Register- Transfer Level)의 하드웨어언어의 설계 및 컴파일링, FSM(Finite State Machine)의 설계, 모듈 바인딩(Module Binding)과 이에 따른 논리합성 등이 다루어진다. 그리고 상위영역의 합성기술에서는 고수준 하드웨어 언어의 설계 및 변환(Transformation), 설계공간의 탐색, 설계유형의 선택 및 기능 분할, 제어구간(Control step) 할당(Allocation) 및 하드웨어 할당등이 다루어진다.

각 합성기술의 표현방식은 하위영역과 중위영역보다는 상위영역의 표현방식이 포괄적(Global)이다. 따라서 반도체 설계에서 하위영역의 자동합성기술이 최적의 상태로 운용되기 위해서는 중위영역의 자동합성 기술이 요구되고, 또한 중위영역의 자동합성 기술이 최적의 상태로 운용되기 위해서는 상위영역의 자동합성 기술이 요구된다. 따라서 동일한 반도체 설계라 할지라도 상위영역에서 합성을 하는 것이 하위영역에서 합성을 하는 것보다 이론적이고 체계적이다.

합성기술 수순을 상위영역에서 시작할 경우 전체 설계과정의 체계적인 운영을 할 수 있을 뿐만 아니라 설계시간이 단축되고 설계자원이 절약된다는 이점을 갖게 된다. 따라서 가장 이상적인 자동 설계용 툴을 개발하기 위해서는 상위영역에서 합성을 시작하여 최적화한 정보를 순차적으로 중위영역과 하위영역에서 분석, 최적화시켜 최종적으로 원하는 시스템의 합성을 해 나가는 것이라 하겠다. 결과적으로 상위영역의 합성기술이 선행된 후, 그 결과를 사용하면 중위영역과 하위영역의 설계는 보다 효율적으로 설계를 할 수 있을 것으로 사료된다.

이러한 자동합성기술 중에서 가장 중요한 분야로 여겨지는 상위영역의 합성수순은 1) DFG(Data Flow Graph)의 생성 및 변환 2) 데이터패스(Data-path) 및 제어부 합성으로 이루어 질 수 있으며, 여기서 데이터패스합성은 다시 스케듈링(Scheduling)과 하드웨어 할당으로 세분되는데 스케듈링 과정에서는 동작 알고리듬의 모든 연산을 적합한 제어구간에 할당시키고, 하드웨어 할당 과정에서는 연산자(Operator)나 레지스터(Register) 및 상호연결선(Interconnection)을 데이터패스에 할당시킨다.

기본적인 스케듈링 기법은 ASAP(As Soon As Possible) 스케듈링 기법[1]과 ALAP(As Late As Possible) 스케듈링 기법[2]을 들 수 있으며, 자원제약이 가해진 상태에서는 리스트(List) 스케듈링 기법[4][5]이 사용되며, 동작속도의 제약이 가해진 상태에서는 Freedom-based 스케듈링 기법[5][7]과 Force-directed 스케듈링 기법[9]이 사용된다.

제안된 기법의 타당성을 검토하고자 표준 벤치마크 모델인 5-order elliptic 웨이브 필터를 선택하고, ILP(Integer Linear Programming) 스케듈링[11]과 FDS(Force- Direct Scheduling)[10] 및 제안된 스케듈링을 비 파이프라인 데이터 패스 내에서 멀티싸이클링 방법을 수행하여 비교 평가하고자 한다.

Ⅱ. 기본이론

2.1 스케듈링 이론

스케듈링 과정을 거쳐 설계된 회로는 반도체 칩 상에 차지하는 면적에 직접적인 영향을 미치기 때문에 면적을 줄이고자 하는 노력이 하드웨어 할당 수순에서 행해지게 된다. 따라서 효과적인 합성이 수행되기 위해서는 스케듈링 수순과 하드웨어 할당 수순이 상호의존적으로 행해지면서, 동시에 빠른 동작속도와 함께 최대의 자원공유(Resource Sharing)가 가능한 스케듈링이 되어야한다.[12]

2.2 리스트 스케듈링

리스트에 할당시키고자 하는 연산을 우선순위와 임의의 기준에 따라 분류시킨 후 이에 따른 스케듈링을 행하는 기법이다. 이를 사용한 시스템으로는 BUD-DAA시스템[3], BSI(Behavioral Synthesis of Interfaces)시스템[4], SLICER시스템[5]등이 있다.

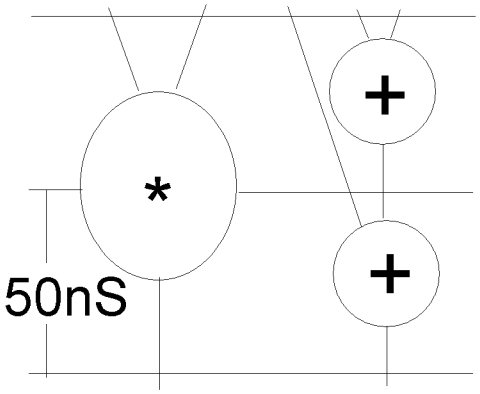
다른 유형의 스케듈링 기법으로서 스케듈링과 하드웨어 할당을 상호의존적으로 고려하는 스케듈링 기법으로, 대표적인 것은 ELF시스템[6], MAHA시스템[7], HAL시스템[8]등이 있다.

이들 시스템들이 사용한 우선 순위는 ELF 시스템은 Urgency라는 개념을 도입해 사용하였으며, MAHA시스템은 이동도(Mobility) 개념을 도입하여 사용하였다.

그중 우선 순위를 Urgency와 이동도 개념을 도입한 시스템을 설명하면,

1) ELF시스템에 사용한 우선 순위는, 임의의 연산결과가 출력으로 이용되기까지에 여러 경로 중에 가장 긴 경로로, Urgency(임계경로를 기준으로 연산처리가 긴급을 요구하는 것부터 우선 순위를 정하는 것)가 높은 순서에 따라 스케듈링을 수행한다.
2) MAHA시스템에 사용한 우선 순위인 이동도는 해당 연산이 ASAP 스케듈링과 ALAP 스케듈링에 의해 스케듈링 되어진 제어구간의 거리를 말하는 것으로, 연산을 임계경로와 비 임계경로로 구분하여 이동도가 적은 순서에 따라 스케듈링을 수행한다.

2.3 멀티싸이클링 연산



[그림 1.] 멀티싸이클링

멀티싸이클링 기법은 제어구간보다 연산자의 지연시간이 더 클 경우 이러한 연산자들은 여러 개의 제어구간에 걸쳐서 할당 해 주는 기법이다.

멀티싸이클링 연산의 장점으로는 지연시간이 짧은 연산자가 서로 다른 제어구간에 할당되므로 이들 간의 자원의 공유가 가능하다는 것이다.

그림1과 같이 하나의 승산기 연산이 복수개의 제어구간에 걸쳐 존재하는 것을 말하며, 가산기 연산은 제어구간이 다르게 존재하기 때문에 1개의 가산기만으로 2개의 가산기 연산을 수행할 수 있게 된다. 이런 경우는 제어구간을 50nS로 설정한 경우이고 승산기 연산의 수행을 위해 2개의 시간 간격(Time Slot : TS)이 필요하다는 것을 알 수 있다. 2개의 가산기 연산과 1개의 승산기 연산을 수행하기 위해서는 2개의 제어구간이 필요하고 연산 수행에 걸리는 시간은 100nS로서 체이닝의 경우와 동일하다.

멀티싸이클링을 고려한 스케듈링은 사용되는 기능단위의 유형에 따라 비 파이프라인 스케듈링과 파이프라인 스케듈링으로 구분하고, 어느 경우나 멀티싸이클 연산을 수행할 수 있다는 점은 동일하다.

Ⅲ. 순위적 스케듈링

3.1 스케듈링의 알고리듬

자원제약이 존재하는 경우는 연산이 할당될 수 있는 제어구간의 폭이 필연적으로 늘어나게 되는데, 이때에 발생되는 문제는 그 폭이 얼마만큼 확장되며 확장된 제어구간에 어떤 연산을 어디에 할당시켜야만 주어진 자원제약 조건을 만족할 수 있는가 하는 문제이다. 또한 사용 가능한 기능단위의 개수를 나타내는 자원 수에 따라서 확장값도 달라지고, 주어진 연산자의 연산 시간에 따라서 연산의 하한 제어구간의 확장 값도 달라진다. 즉 DFG상 모든 연산이 할당될 수 있는 최하한 제어구간의 값을 추출하기 위해 Forward

스케듈링과 Backward 스케듈링의 알고리듬을 기술한다.

DFG상의 모든 연산의 이론 전개에 편리하도록 기호를 아래와 같이 정의하여 사용한다.

(1) Ofn : 선행 연산이 없는 연산
(2) Obn : 후행 연산이 없는 연산
(3) Of : 선행 연산이 존재하는 연산
(4) Ob : 후행 연산이 존재하는 연산
(5) Oc : 임계경로에 존재하는 연산
(6) Ocn : 임계경로와 무관한 연산
(7) Ocp : 연산의 출력이 Oc의 입력으로 사용되는 연산 및 이들의 모든 선행(predecessor) 연산
(8) Ocs : 연산의 입력으로 Oc의 출력을 사용하는 연산 및 이들의 모든 후행(successor) 연산

임의의 DFG에 존재하는 연산은 임계경로에 연관된 연산과 임계경로에 무관한 연산으로 구분되고, 임계경로에 연관된 연산은 임계경로(Critical-path)에 있는 연산 Oc 비 임계경로(non-Critical path) 연산 Ocp와 Ocs로 구분된다.

또한 임계경로나 비 임계경로에 있는 연산들 중에 선행연산이 없는 연산은 Ofn, 후행연산이 없는 연산은 Obn으로 구분하며, 임계경로나 비 임계경로에 있는 연산들 중에 연산 Ofn 이외의 연산을 Of, 연산 Obn 이외의 연산을 Ob 로 구분한다.

3.1.1 Forward 스케듈링의 알고리듬

Forward 스케듈링 알고리듬은 DFG상의 모든 연산이 할당될 수 있는 최하한 제어구간의 값을 구하기 위해 Urgency를 우선 순위로 사용한 점은 기존의 것과 거의 동일하다.

Forward스케듈링 알고리듬은 다음과 같다.

가. DFG의 모든연산을 Ofn, Obn, Of, Ob, Oc, Ocn, Ocp, Ocs로 구분한다.
나. ASAP 스케듈링과 ALAP스케듈링을 한다.
다. 연산 Oc 와 가장 먼 연산 Ocp를 먼저 표에 할당한다.
 1) 주어진 조건을 연산 Ofn부터 연산 Obn까지 연산들을 위해서 아래로 순차적으로 스케듈링을 한다.
 2) 연산 Ofn 연산들은 첫 제어구간(1-STEP)에 반드시 할당 되야 하고, 마지막 제어구간 전에 할당을 할 수 있다.
 3) 연산 Obn 연산들이 할당할 수 있는 제어구간은 마지막 제어구간에 할당이 되야 하고, 첫 제어구간 이후에 할당을 할 수 있다.
 4) 연산 Of와 연산 Ob는 첫 제어구간과 마지막 제어구간만 할당하지 못하고, 어느 제어구간에도 할당할 수 있다.
 5) 연산 Of와 연산 Ob가 동일한 제어구간에 연산이 여러 개 있을 때에는, 입력까지 제일 긴 경로 길이의 연산을 기준으로 삼아서 Forward 스케듈링을 해야한다.
 6) 스게듈링시 제약된 자원을 넘은 연산이 할당되어 있으면, 이 연산자는 자원의 선후관계가 변하지 않는 범위 내에서 다음 제어구간에 할당시킨다.
라. 스케듈링은 왼쪽에서 오른쪽으로 수행하여 도표에 할당시킨다.
마. 연산 Oc,연산 Ofn와 연산 Obn의 할당이 모두 완료 될 때까지 다. 와 라. 를 반복 수행하다.
바. 연산 Oc, 연산 Ofn와 연산 Obn로 구성되는DFG상의 부분연산을 DFG로부터

제거시킨다.

사. 연산 Oc, 연산 Ofn와 연산 Obn가 적절한 제어구간에 스케듈링이 되어 있지 않으면 다. 부터 바. 까지 다시 수행한다.

아. 자원제약으로 제어구간에 적절히 연산이 할당되었으면 스케듈링을 끝낸다.

3.1.2 Backward 스케듈링의 알고리즘

Backward 스케듈링은 용어가 뜻하는 바와 같이, 도표에 할당시키고자 하는 연산을 우선 순위에 따라 뒤부터 할당시킨 후, 모든 선행연산을 표에 할당 한 다음, 할당된 연산을 기준으로 이들을 해당 제어구간에 할당시키는 방법이다.

Backward 스케듈링 알고리즘은 다음과 같다.

가. DFG의 모든 연산을 Ofn, Obn, Or, Ob, Oc, Odn, Ocp, Ocs로 구분한다.

나. ASAP 스케듈링과 ALAP 스케듈링을 한다.

다. 연산 Oc와 가장 먼 연산 Ocs를 먼저 표에 할당한다.

1) 주어진 조건을 연산을 Obn부터 연산 Ofn까지 연산들을 아래에서 위로, 연산을 역으로 스케듈링을 한다.
2) 연산 Obn 연산들이 할당할 수 있는 제어구간은 첫 제어구간에 할당이 되야 한다.
3) 연산 Ofn 연산들은 첫 제어구간 (1-Step)에 반드시 할당 되야 하고, 스케듈링이 가능한 마지막 제어구간 전에 할당을 할 수 있다.
4) 연산 Of와 연산 Ob는 첫 제어구간과 마지막 제어구간만 할당하지 못하고, 어느 제어구간에도 할당할 수 있다.
5) 연산 Of와 연산 Ob가 동일한 제어구간에 연산이 여러 개 있을 때에는, 출력까지 제일 긴 경로 길이의 연산을 기준으로 삼아서 Backward 스케듈링을 해야한다.
6) 스케듈링시 제약된 자원을 넘은 연산이 할당되어 있으면, 이 연산자는 자원의 선후관계가 변하지 않는 범위 내에서 다음 제어구간에 할당시킨다.

라. 스케듈링은 왼쪽에서 오른쪽으로 수행하여 도표에 할당시킨다.

마. 연산 Oc 연산 Ocp와 연산 Ocs의 할당이 모두 완료 될 때까지 다.와 라. 를 반복 수행한다.

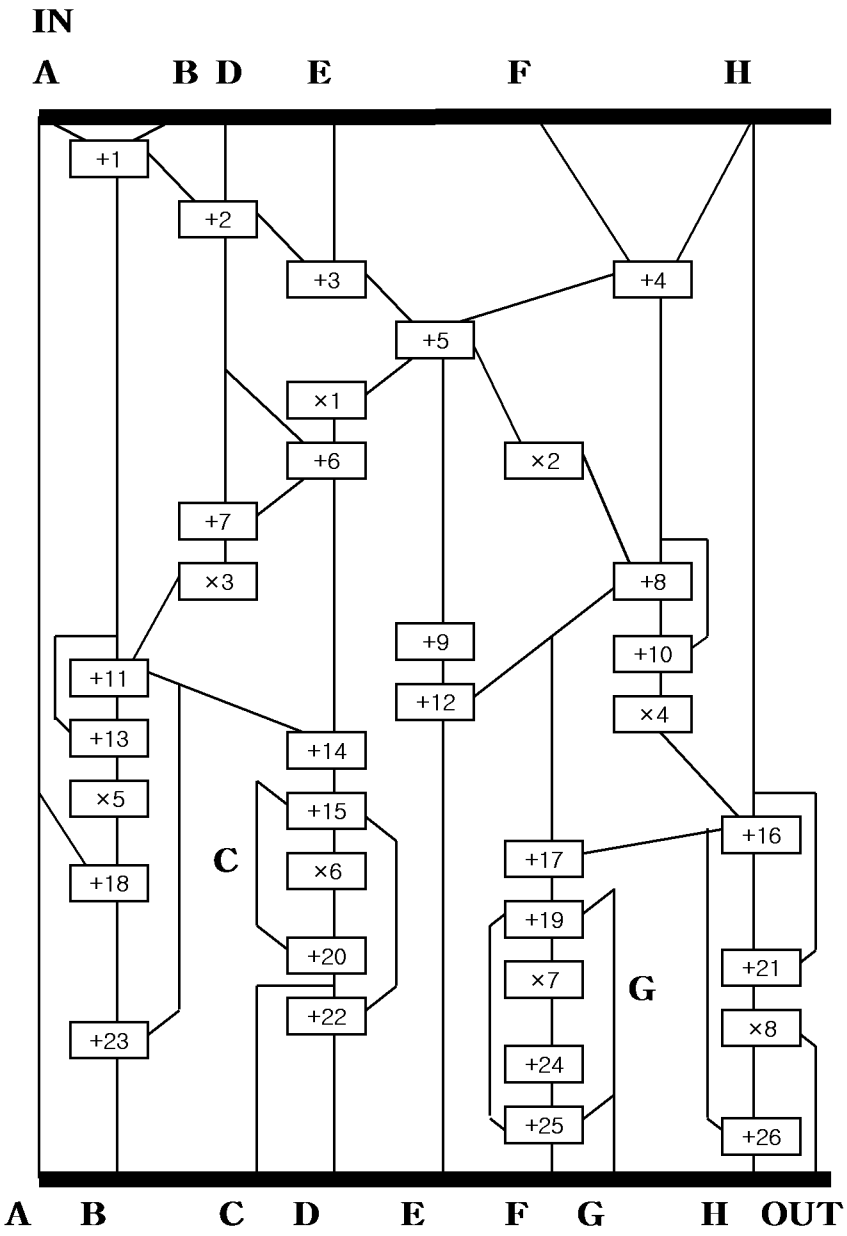
바. 연산 Oc, 연산 Ocp와 연산 Ocs로 구성되는 DFG상의 부분연산을 DFG로부터 제거시킨다.

사. 연산 Oc, 연산 Ocp와 연산 Ocs가 적절한 제어구간에 스케듈링이 되어 있지 않으면 다. 부터 바. 까지 다시 수행한다.

아. 자원제약으로 제어구간에 적절히 연산이 할당되었으면 스케듈링을 끝낸다.

이러한 알고리즘으로 하여, 주어지는 자원에 따라 확장되는 제어 스텦 수를 최소화시키는 방향으로 스케듈링을 한다. 이때 할당시키고자 하는 연산의 수가 사용 가능한 자원의 수를 초과할 때에는, 초과분에 대한 할당을 다음 제어구간으로 미룬다.

그림2의 5-order elliptic wave filter 모델로 예를 들어 Forward 스케듈링을 적용시킨다.



[그림 2.] 5-order elliptic 웨이브 필터

먼저 임계경로 연산 Oc 중에 가장 앞쪽의 연산-임계경로 연산 중에 입력 포트로부터 입력신호를 부여받은 연산을 스케듈링 도표에 할당한다.

그림2의 모델을 설명하면, 임계경로 연산인 +1, +2, +3, +4, +5, +6, +7, +11, +14, +15, +20, +22, *1, *3, *6은 연산 Oc 해당하며, +4는 Ocp에 해당하고 나머지 연산자들은 Ocs에 해당된다. 그리고 승산기 1개와 가산기 2개로 예를 들면, 사용가능한 가산기가 2개이므로 +1과 +4를 첫 제어구간에 할당시킨다. 이때 +4는 첫 제어구간에 할당을 하지만 스케듈링이 되는 순서는 4번째이며 이후 이들을 DFG에서 제거시킨다. 다음의 할당 대상인 +2와 그 다음의 할당 대상인 +3에 대해서도 과정을 반복한다. *1의 할당에서는 승산기 연산자에 제약이 1개 있으므로 *2를 다음제어구간에 미룬다.

이러한 과정을 모든 연산에 대해 적용시킨 결과가 표 3-1이며, 각 연산자위에 표기된 숫자는 스케듈링이 되는 순서를 나타낸 것이다.

표1은 주어진 자원의 수(N)와 제어구간의 수(T)로 구성하는 T·N matrix 표를 작성하여, 이 표를 이용하여 제어구간의 하한 값을 추출한다. 이때 사용하는 기능 단위는 승산기(*) 1개와 가산기(+) 2개로 하고, 승산기 지연시간은 80nS, 가산기 지연시간은 40nS, 제어구간 시간 간격은 100nS로 가정한다.

표1에서 알 수 있듯이, 1개의 승산기와 2개의 가산기로는, 수행시간이 1700nS인 5-order elliptic wave filter를 구현할 수 있다.

[표 1.] Forward 스케듈링

구간수	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A1	1 +1	2 +2	3 +3	5 +5		7 +6	9 +7	12 +9	14 +11	17 +13	20 +14	22 +15	24 +18	27 +23	30 +20	32 +22	34 +25
A2	4 +4						10 +8	13 +10	15 +12	18 +16	21 +17	23 +21	25 +19	29 +26		33 +24	
M1					6 *1	8 *2		11 *3	16 *4		19 *5		26 *8	28 *6	31 *7		

3.2 멀티싸이클 연산의 Forward 스케듈링(비파이프라인)

멀티싸이클 연산은 비파이프라인 기능단위(non-pipeline functional unit) 또는 파이프라인 기능단위(pipeline functional unit)에 의해 수행될 수 있다.

어느 경우나 멀티싸이클링 연산을 수행할 수 있다는 점은 동일하나 자원공유(resource sharing)의 관점에서 볼 때 그 방법상 차이가 있다. 즉, 비파이프라인 기능단위는 일단 연산이 수행하면 해당 연산을 마칠 때까지 다른 연산의 수행이 불가능한 반면, 파이프

라인 기능단위는 연산의 수행 도중에도 다른 연산을 수행할 수 있기 때문에 보다 많은 자원공유를 행할 수 있다.

어느 경우나 멀티싸이클링 연산을 수행할 수 있다는 점은 동일하나 자원공유(resource sharing)의 관점에서 볼 때 그 방법상 차이가 있다. 즉, 비파이프라인 기능단위는 일단 연산이 수행하면 해당 연산을 마칠 때까지 다른 연산의 수행이 불가능한 반면, 파이프라인 기능단위는 연산의 수행 도중에도 다른 연산을 수행할 수 있기 때문에 보다 많은 자원공유를 행할 수 있다.

지연시간이 d-제어구간인 비파이프라인 기능단위가 임의의 연산 Oi를 k번째 제어구간부터 수행하기 시작한다고 가정할 경우, 이 기능단위가 다른 연산의 수행에 할당될 수 있는 제어구간은 (k + ℓ)번째 이후의 제어구간부터이다.

따라서 이 기능단위는 k번째 제어구간부터 (k + ℓ - 1)번째 제어구간 사이에 걸쳐, 계속적으로 사용되기 때문에 이 구간에서는 재사용이 불가능하다는 의미가 된다.

비파이프라인 기능단위는 일단 임의의 연산을 수행하게 되면, 해당 연산이 종료되기 이전에는 다른 연산의 수행에 할당될 수 없다. 이와 같이 비파이프라인 기능단위에 의한 멀티싸이클링 연산의 스케듈링을 하는 경우 자원의 제약조건이 존재한다.

연산을 할당함에 있어, 할당시키고자 하는 연산이 멀티싸이클링 연산인 경우, 해당연산이 할당되는 제어구간을 연산의 지연시간만큼 확장시켜 다른 연산의 할당을 금지시키면 된다.

[표 2.] Forward 스케듈링(비파이프라인 멀티싸이클링)

구간수	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
A1	+1	+2	+3	+5			+6	+7	+8	+10	+11	+13	+15	+17	+18	+23	+20	+22	+24	+25	+26
A2	+4							+9		+12		+14	+16	+21	+19						
M1					*1	*1	*2	*2	*3	*3	*4	*4	*5	*5	*6	*6	*7	*7	*8	*8	

멀티싸이클링 연산에 의한 비파이프라인 스케듈링을 하기 위해, 멀티싸이클링 연산을 고려한 Forward 스케듈링을 수행한다. 그리고 Forward 스케듈링의 결과로 작성된 스케듈링 표는 제어구간 수를 의미하며 그 결과는 표2에 나타내었다.

5-order elliptic wave filter의 DFG 모델을 1개의 승산기와 2개의 가산기에 대해 멀티싸이클링을 고려한 비파이프라인 스케듈링을 행한다. 승산기 지연시간은 80nS, 가산기의 지연시간은 40nS, 제어구간의 시간간격은 50nS 인 것으로 가정한다.

또한 CPL은 임계경로의 제어구간 수를 나타내며 L은 확장 될 제어구간의 수를 나타내며 Smin = 21 이고 CPL = 14 이므로 L = Smin - CPL = 7 만큼 제어구간의 확장이 요구되며, 1개의 승산기와 2개의 가산기로써는 수행시간이 1050nS인 5-order elliptic wave filter가 구성됨을 알 수 있다. 이상과 같이 승산기 1개와 가산기 2개로 예를 들어 Forward 스케듈링의 결과를 표3에 요약하면 다음과 같다.

[표 3.] 스케듈링 결과 (자원의 수 : *1, +2)

스케듈링	수행시간
normal	1700ns
체이닝	1100ns
멀티싸이클링(비파이프라인)	1050ns

Ⅳ. 검증 및 결과

본 연구에서 제안한 스케듈링 기법을 사용하여, 표준 benchmark 모델인 fifth-order elliptic wave filter[2]에 적용시킨 결과는 표 4-1와 같다.

5-order elliptic wave filter는 " + " 연산자의 지연시간은 40nS, " * " 연산자의 지연시간은 80nS, 제어구간의 시간간격은 멀티싸이클링 연산시 50nS으로 택하여 실험모델을 선택하였다.

표4는 비파이프라인 데이터 패스에서 5-order elliptic wave filter를 Forward 스케듈링으로 수행하여 나온 결과를 나타낸 것으로써, T·N matrix 표를 작성하여 주어진 자원의 수에 따라 확장되는 제어구간의 하한 값을 추출한 것이다.

표4를 비파이프라인 승산기에서 자원의 수가 가산기 2개, 승산기 2개의 경우에 예를 들어 설명하면, FDS 기법에 의한 스케듈링의 하한 값은 19 제어구간이고 제안한 기법에 의한 스케듈링의 하한 값은 18 제어구간수가 되므로 동일하게 주어진 자원의 수를 가져도 제안한 스케듈링 기법에서는 제어구간수가 좋아짐을 알 수 있었다.

표4는 비파이프라인 데이터 패스에서 16-point FIR 필터를 제안한 기법으로 수행하여 나온 결과를 나타낸 것으로써, T·N matrix 표를 작성하여 주어진 자원의 수에 따라 확장되는 제어구간의 하한 값을 추출한 것이다.

표5의 내용을 설명하면 9 step (+3, *2)은 16-point FIR filter를 ASAP와 ALAP 스케듈링을 수행하여 상하한 값 9 step과 자원의 수인 가산기 3개와 승산기 2개를 추출한 것이다.

추출한 자원의 수(+3, *2)를 기준으로 하여 순차적으로 자원의 수를 감소시켜 가면서 확장되는 제어구간의 하한 값을 추출하였다. 제안한 스케듈링 기법으로 멀티싸이클링 연산에서는 제어구간의 시간간격을 50nS로 가정한다.

[표 4.] 비파이프라인 데이터 패스 (5-order elliptic wave filter)

17 step (+ 3, * 3)								
멀티싸이클링연산 (비파이프라인)	소요 자원	+	3	3	2	3	2	1
		*	3	2	2	1	1	1
	구간수	FDS[10]	9	9	10			15
		ILP[11]	6	8	10			12
		Forward					15	18

[표 5.] 비 파이프라인 데이터 패스의 16-point FIR filter

9 STEP (+ 3, * 2)						
FORWARD	소요 자원	+	3	2	2	1
		*	2	2	1	1
	구간수	normal	9	9	10	15
		체이닝	6	8	10	12
		normal	9	9	10	15
		멀티싸이클링 (비파이프라인)	10	11	18	18

Ⅴ. 결론

본 논문은 순위적 스케듈링 문제에 대한 접근방식으로는 스케듈링 기법을 사용하고, 제한된 Forward와 Backwad도 사용해 우선 순위를 사용한 점에서 기존의 방법과 동일하나, 본 연구는 스케듈링 과정에서 자원의 충돌(resource conflicts)이 발생할 경우 충돌이 일어난 연산의 우선 순위 함수를 산출하여 이를 이용하였다. 따라서, 기존의 스케듈링 방식보다 복잡도(complexity)가 대폭 개선할 수 있었다.

성능평가 방법은 비 파이프라인 데이터 패스 내에서 멀티싸이클링 연산에서는 50nS로 제어구간을 분할하여 평가되었다. 그리고, 제안된 스케듈링 기법을 16-point FIR filter 모델에 적용해 멀티싸이클링 연산을 수행하여 제어구간의 수를 추출하였으며, 그 결과 확장되는 제어구간의 수를 쉽게 추출할 수 있었다.

2002年度 培材大學校 工學硏究所 校內學術硏究費 支援에 의하여 遂行된 硏究의 一部로 이에 感謝 드립니다.

참고문헌

1] C.H.Gevotys, M.I.Elmasry, "A VLSI Methodology with testability constraints", in Proc. 1987 Canadian Conf. VLSI, Winnipeg, Oct 1987.

2] S.Y.Kung, H.J.Whitehouse, T.Kailath, "VLSI and Modern Signal Processing", Englewood Cliffs, NJ : Prentice Hall, pp.258-264. 1985.

3] M.C.McFarland, "BUD:Bottom-Up Design of Digital Systems", Proc. of the ACM/IEEE 23rd DAC., 1986, pp.474-479

4] J.Nester, dD.E.Thomas, "Behavioral Synthesis of Interfaces", Proc. of the IEEE ICCAD-86, pp.112-115, 1986.

5] B.M.Pangrle, D.D.Gajski, "State synthesis and connectivity binding for microarchitecture compilation", Proc. of ICCAD-86, pp.210-213, nov. 1986.

6] E.F.Girczyc, "An ADA to standard cell hardware compiler based on graph grammers and scheduling", Proc. of ICCAD-84, pp.726-731, Oct. 1984.

7] A.C.Parker et al, "MAHA:Aprogram for datapath synthesis", Proc. of 23rd DAC., pp.461-466, 1986.

8] P.G.Paulin, J.P.Knight, "Force-directed scheduling for the vehavioral synthesis of ASIC's", IEEE Tr.CAD, Vol.8, pp.661-679, June. 1989.

9] K.S.Hwang, A.E.Casavant, et al., "Scheduling and hardware sharing in pipeline datapath", Proc. of ICCAD-89, pp.24-27, 1989.

10] P.G.Pauin and J.P.Knight, "Force-directed Scheduling in automated Data Path Synthesis", Proc. of 24th DAC., pp.195-202, jun. 1987.

11] C.T.Hwang, Y.C.Hsu, and Y.L.Lin, "Optimum and heuristic data path scheduling under resource constraints", Proc. of 27th DAC., pp.65-70, 1990.

12] N.W.Beak. “A study on Forward algorithm using a Matrix", 청주기능대학 논문집, 제 2권,pp.359-377, July, 1997.