The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 20, No. 3, pp.181-187, Jun. 30, 2020. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2020.20.3.181
JIIBC 2020-3-25

고성능 차량용 SoC 설계 합성 시스템

A SoC Design Synthesis System for High Performance Vehicles

장정욱[*], 인치호[**]

Jeong-Uk Chang[*], Chi-Ho Lin[**]

요 약 본 논문에서는 고성능 차량용 SoC 설계자동화를 위한 상위수준 합성과정에서의 레지스터 할당 알고리즘과 자원할당 알고리즘을 제안한다. 상위수준 합성에서 가장 중요한 연산자의 특성과 데이터패스의 구조를 분석하고, 멀티사이클 연산의 스케줄링 시 가상연산자 개념을 도입함으로써, 멀티사이클 연산을 구현하는 연산자의 유형에 관계없이 공통으로 적용시킬 수 있는 자원할당 알고리즘을 이용하여 증명하였다. 연산자 간을 연결하는 신호선이 반복적으로 이용되어 연결 신호선 수가 최소가 될 수 있도록 기능연산자를 할당하고, 레지스터 할당 시 연결구조에 따라 가중치를 갖는 구간 그래프를 구성한다. 최소의 클러스터 분할 알고리즘을 이용하여 생성된 최대 크기의 클러스터들에 연결구조가 고려된 레지스터를 할당한다. 연결구조에 대한 멀티플렉서의 중복 입력을 제거하고 연산자에 연결된 멀티플렉서 간의 입력을 교환하는 입력 정렬 과정으로 연결구조를 최소화한다. 기술된 알고리즘의 스케줄링 성능을 평가하기 위하여, 표준벤치마크 모델인 5차 디지털 웨이브필터에 대한 스케줄링을 실행하여 제안한 알고리즘의 효용성을 입증한다.

Abstract In this paper, we proposed a register allocation algorithm and resource allocation algorithm in the high level synthesis process for the SoC design synthesis system of high performance vehicles We have analyzed to the operator characteristics and structure of datapath in the most important high-level synthesis. We also introduced the concept of virtual operator for the scheduling of multi-cycle operations. Thus, we demonstrated the complexity to implement a multi-cycle operation of the operator, regardless of the type of operation that can be applied for commonly use in the resources allocation algorithm. The algorithm assigns the functional operators so that the number of connecting signal lines which are repeatedly used between the operators would be minimum. This algorithm provides regional graphs with priority depending on connected structure when the registers are allocated. The registers with connecting structure are allocated to the maximum cluster which is generated by the minimum cluster partition algorithm. Also, it minimize the connecting structure by removing the duplicate inputs for the multiplexor in connecting structure and arranging the inputs for the multiplexor which is connected to the operators. In order to evaluate the scheduling performance of the described algorithm, we demonstrate the utility of the proposed algorithm by executing scheduling on the fifth digital wave filter, a standard bench mark model.

Key Words : SoC, High-level synthesis, Design automation, Allocation

[*]정회원, 세명대학교 컴퓨터학부
접수일자 2020년 4월 27일, 수정완료 2020년 5월 27일
게재확정일자 2020년 6월 5일

Received: 27 April, 2020 / Revised: 27 May, 2020 /
Accepted: 5 June, 2020
[*]Corresponding Author: ich410@semyung.ac.kr
School of Computer, Semyung University, Korea

Ⅰ. 서 론

최근에 첨단 안전장치와 편의장치 등이 자동차에 이용됨에 따라 다양한 전자장치를 제어하기 위한 차량용 반도체기술이 미래경쟁력을 좌우하는 핵심요소가 되는 추세이다. 현재 우리나라는 대부분의 차량용 SoC(System on Chip)를 수입에 의존하고 있지만, 메모리 반도체와 자동차산업에서 강력한 기반을 갖추고 있어 우리에게 경쟁력을 향상시킬 수 있는 새로운 기회가 되고 있다. 자동차는 안전성 향상과 연비 개선을 주목적으로 하여 25,000여 개의 전자부품이 장착되어 있고, 메커트로닉스 결정체로 자리 잡고 있다. 앞으로는 첨단 장치와 편의장치 등이 자동차에 접목됨에 따라 많은 수의 전자장치를 제어하기 위한 차량용 SoC 기술이 미래경쟁력을 좌우하는 핵심요소가 되고 있고 비메모리 부분의 반도체 시장에서는 '블루오션'으로 두각을 나타내고 있다.

고성능 차량용 SoC 사용이 증가하는 이유는 반도체기술의 발달과 부품 가격 감소의 영향이 가장 크며 자동차 업체 간 기술 선점에 따른 우위 확보를 위해 개발에 박차를 가하고 있기 때문이다. 특히 자동차 차량용 반도체 분야는 향후 환경친화적 자동차가 활발히 실용화가 되면 해외업체에 대한 의존도가 높아지는 문제가 발생할 소지가 많은 분야이기 때문에 개발이 절실히 요구되고 있고, 대부분 수입에 의존하고 있는 우리나라는 차량용 SoC의 점유율이 거의 전무한 상황이므로 지속적인 관심을 기울여야 한다. 또한, 차량용 SoC 개발과 적용에는 아직 특이한 규제사항이 없는 것으로 알려졌지만, 안전과 중요한 관계를 맺고 있는 부분이므로 신화성과 규격, 업체들의 시험기준에 적합한 반도체의 개발이 아주 중요하다[1].

고성능 차량용 SoC 설계를 위한 상위수준 합성은 알고리즘과 구조적 수준을 고려하여 회로의 동작적 기술로부터 RTL로의 합성을 의미한다. 고성능 차량용 SoC 회로 구현은 여러 설계 수준의 범위를 포함해야 하며 회로 설계 시 초기 단계의 결정은 다음 단계의 큰 영향을 미치므로 상위수준에서의 조기 최적화는 매우 중요하다. 전력 최적화를 위한 할당 방법은 현재까지 많은 연구가 이루어져 있다. 기존의 연구는 공유를 통해 기능 장치나 레지스터의 입력 변수들을 조절하여 시스템 내에 발생하는 스위칭 동작을 줄이고자 하였다.

이전에 제안된 방법으로는 데이터 경로의 병렬화나 파이프라이닝과 같은 구조 변화를 이용하여 공급 전압을 감소시키거나[2-3], 피연산자를 공유하는 시블링 연산(Sibling operation)을 같은 기능 장치에 할당하는 스케줄링과 바인딩 방법을 제시하였다[4-5]. [6]에서는 제어 신호를 필요한 연산자에만 보냄으로써 불필요한 기능 장치의 동작을 줄이는 "Shot down"방법을 제시하였고, [7]에서는 유전자 알고리즘을 사용하여 면적과 평균 전력과 peak 전력, 속도의 최적화를 시도하였다. 또한 [8]에서는 회로의 규칙성에 해당하는 연산들을 같은 기능 장치에 할당함으로써 연결 구조연결 구조상에서 발생하는 전력을 최소화하였다. 그러나 이들의 연구에서는 연결구조의 전력 소비를 고려하지 못해 데이터 패스 일부에 대한 최적화만을 이루었다.

따라서 본 논문에서는 이러한 문제점들을 고려하여, 상위수준 합성과정에서 레지스터 할당과정과 자원할당과정을 나누어 수행하여, 여기서 발생하는 전력 소모를 최소화하는 고성능 차량용 SoC 설계를 위한 상위수준 합성 시스템을 설계 및 개발하고자 한다.

Ⅱ. 고성능 설계자동화를 위한 최소자원 할당 알고리즘

1. 스케줄링의 절차

메모리 할당 문제는 하나 또는 최종 시스템성능이 최적화되도록 모든 제약이 만족할 수 있게 메모리에 지정 데이터 개체의 작업으로 명시할 수 있어야 한다. 임의의 선형 평등과 불평등 제약이 있는 최적화 문제로 메모리 할당 문제를 모델링하기 위해 각각의 솔루션 변수 i와 j에 대한 인덱스를 생성하고 다음과 같이 정의한다.

변수	자료형	의미
x_i	0-1	: 1인 경우, 변수 i는 X 메모리에 할당
y_i	0-1	: 1인 경우, 변수 i는 X 메모리에 할당
b_i	0-1	: 1인 경우, 변수 i는 X, Y 메모리에 모두 할당
a_{ij}	0-1	: 1인 경우, i와 j를 모두 메모리 X에 할당하고, 그렇지 않으면 0
b_{ij}	0-1	: 1인 경우, i와 j를 모두 메모리 Y에 할당하고, 그렇지 않으면 0

인덱스 세트 의미

N_i "공간 단위"(예를 들어, 바이트, 단어, 등)의 수는 변수 i에 의해 점유. 단위는 가장 작은 데이터 단위와 장치 내 대부분의 제한된 메모리 액세스 모드에 따라 변동

L_i 왼쪽 i를 피연산자로 하는 비 상호 (또는 단항) 연산자의 총 예상 수

R_i 오른쪽 i를 피연산자로 하는 비 상호 (또는 단항) 연산자의 총 예상 수

U_i 해당 업데이트 변수 i의 운영 총 예상 주파수

P_{ij} 이전 피연산자로 피연산자 i와 j를 사용하는 상호 이진 연산의 총 예상 수

M_{ax_x} X 메모리의 용량(바이트)

M_{ax_y} Y 메모리의 용량(바이트)

주어진 솔루션 변수의 정의에 따라, 우리는 X 메모리는 "왼쪽" 메모리를, Y 메모리로 "오른쪽" 메모리를 참조한다. 메모리상의 다른 변수에 a_{ij}와 b_{ij}가 연관하고 모두 같은 시간에 동일하게 1이 지정될 수 있는가를 확인한다. 만약 a_{ij} = 1이라면, 두 변수 i와 j는 같은 X 메모리에 발견되어, 유사한 관찰 Y 메모리에 b_{ij} = 1을 i와 j에 적용한다. 이를 요약하여 정리하면 다음과 같다.

1) 변수 i은 X 또는 Y 메모리, 둘 모두를 기억하기 위해 반드시 하나는 할당됨

$$\forall_{i,j} : x_i + y_i + b_i = 1 \quad \cdots(1)$$

2) 변수 i은 Y 또는 두 개의 메모리에 할당되는 경우, 또 다른 변수 j와 함께 X 메모리에 독점적으로 할당되지 않음

$$\forall_{i,j} : 2(1 - a_{ij}) \geq y_i + b_i + y_j + b_j \quad \cdots(2)$$

3) 변수 i은 X 또는 두 개의 메모리에 할당되는 경우, 그것은 Y 메모리에 할당되지 않음

$$\forall_{i,j} : 2(1 - b_{ij}) \geq x_i + b_i + x_j + b_j \quad \cdots(3)$$

4) 변수 i과 변수 j이 X 메모리에 모두 할당되어 있다면, 강제적으로 $\{A_{ij} = 1 \text{ (in X)}\}$ 명시

$$\forall_{i,j} : i, j \text{ where } i < j : x_i + x_j \leq 1 + a_{ij} \quad \cdots(4)$$

5) 변수 i과 변수 j이 Y 메모리에 모두 할당되어 있다면, 강제적으로 $\{A_{ij} = 1 \text{ (in Y)}\}$ 명시

$$\forall_{i,j} : i, j \text{ where } i < j : y_i + y_j \leq 1 + b_{ij} \quad \cdots(5)$$

6) 메모리에 할당된 변수의 총 크기는 on-chip 데이터 메모리의 크기를 초과할 수 없음

$$\sum_i N_i(x_i + b_i) \ll M_{ax_x}, \sum_i N_i(y_i + b_i) \ll M_{ax_y} \quad \cdots(6)$$

목적함수의 값은 메모리에 추가 지침을 삽입하거나, 또는 데이터 이동 처리를 위해 지연시킬 필요가 있기에 이를 추가 제약 사항으로 반영한다. 보다 구체적으로, 목적함수는 피연산자 가능한 제어스탭(처음 두 조건)에 따라 별도의 제어단계 업데이트 비용을 요구하는 과정을 메모리에 게재되는 것에 대한 비용을 반영하는 가중치 합을 최소화하고자, 피연산자가 메모리에 두 번 추가될 가능성을 별도의 메모리 내 제어단계에 명시해주어야 한다.

7) 변수 i은 반드시 X 메모리에 연역적으로 할당

$$y_i + b_i = 1 \quad \cdots(7)$$

8) 변수 i은 반드시 Y 메모리에 할당

$$x_i + b_i = 1 \quad \cdots(8)$$

9) 변수 i은 X 메모리에 할당할 수 없는 경우도 발생 가능

$$y_i = 1 \quad \cdots(9)$$

10) 변수 i은 Y 메모리에 할당할 수 없는 경우도 발생 가능

$$x_i = 1 \quad \cdots(10)$$

11) 변수 i은 X / Y 양쪽 메모리로 복제할 수 없음

$$y_i + x_i = 1 \quad \cdots(11)$$

12) 변수 i과 j은 동일한 메모리에 명시될 수 있음

$$b_i + b_j + a_{ij} + b_{ij} \geq 1 \quad \cdots(12)$$

13) 변수 i과 j은 다른쪽 메모리에 나타날 수 있음

$$a_{ij} + b_{ij} = 0 \quad \cdots(13)$$

주어진 정리를 바탕으로, 메모리 또는 데이터를 이동하거나 복사하기 위해 처리가 지연될 경우를 고려하여 최종 목적함수의 추가 지침을 정리하면 다음과 같다.

$$\min\left[\sum_i y_i L_i + \sum_i x_i R_i + \sum_i b_i U_i + \sum_{i,j}(a_{ij} + b_{ij}) P_{ij}\right] \quad \cdots(14)$$

표 1. 변수 크기에 따른 가중치
Table 1. The weight by variable size

변수	N_i (bytes)	L_i	R_i	U_i
A	2	0	2.5	8.5
B	4	20	27.5	21
C	2	2.5	0	20
D	4	40	0	15
E	1	2.5	2.5	10
F	1	0	0	22.5
G	2	0	20	20
H	7	27.5	0	0
I	1	0	0	22.5
J	2	10	0	10
K	4	1	20	12.5
L	1	0	20	0

표 1에서 1열의 이름으로 각 변수를 식별하고, 2열의 바이트 용량으로 각 변수의 크기를 식별한다. 3열과 4열은 왼쪽(X) 또는 오른쪽(Y) 메모리에 대한 접근빈도 수를 나타낸다. 5열은 각 변수들의 상대적인 업데이트 주기를 나타낸다.

표 2. 대응되는 변수의 운영 가중치
Table 2. The Operational weight of corresponding variable

변수 i	변수 j	Pij
E	*F*	20
E	*J*	10
D	*F*	7.5
B	*K*	2.5
G	*L*	7.5

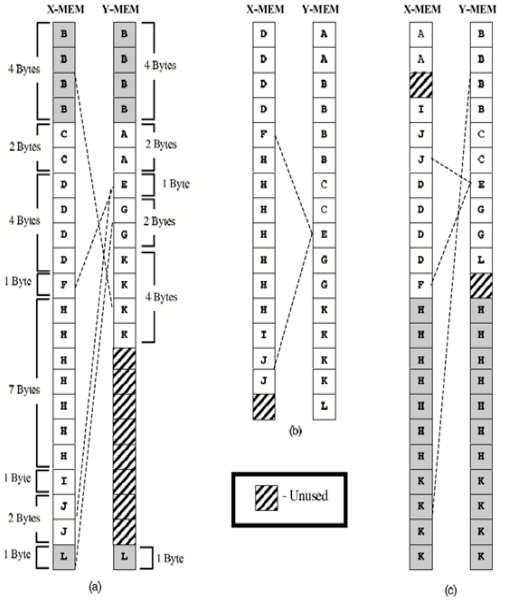


그림 1. 메모리 할당 지정 결과
Fig. 1. Memory Allocation Specified Results

그림 2(a)는 최종 메모리 할당 상태를 보여준다. 회색 음영은 변수가 중복된 인스턴스를 보여준다. 반대 메모리에 공동 피연산자 쌍이 성공적으로 매핑되면 점선으로 표시하였다. 그림 2(b)에서 15바이트의 왼쪽 메모리의 크기를 줄일 수 있었다. 그림 2(c)에서 보듯이 이전 제약조건과 추가제약조건을 처리하여 메모리 공간에 사용자에 의한 모든 데이터 지정이 가능함을 의미한다. 그러나 적은 수의 변수와 메모리에 지정된 배열 및 무차별적인 데이터의 복사에 따른 실험결과에서는 표 3에서 알 수 있듯이 그리 큰 차이점은 없는 것으로 확인된다.

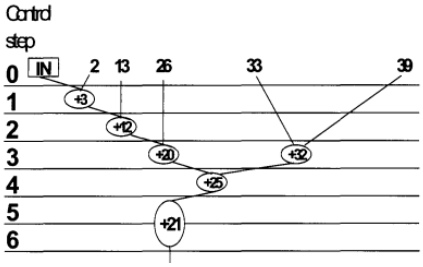
2. 기능연산자 할당

기능연산자 할당과정은 연산을 기능연산자에 할당하는 과정으로써 기능연산자의 수는 스케줄링 단계에서 결정된 것으로 한다. 할당에 사용되는 연산자는 다기능 연산자가 아닌 단일 기능을 갖는 연산자형 별(예, +, *, 카운터 등)로 할당을 수행한다. 연산을 기능연산자에 할당하는 구조를 최소화할 수 있도록 연산을 기능연산자에 할당하여야 한다. 본 논문에서 제안하는 기능연산자 할당은 두 개의 연산 종류에 대해 따로 수행한다. 즉, 1개의 연산만이 존재하는 경우와 2개 이상의 연산이 존재하는 경우이다.

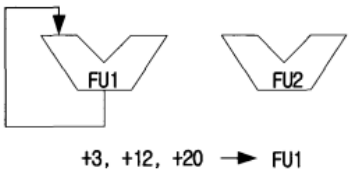
특정 형(type)의 연산은 그림 2(a)에서 제어스탭 5에 있는 +21과 같이 하나의 제어스탭에 올 수 있는 같은 형의 연산 수가 1개인 연산으로 이 연산을 수행하는 기능연산자를 우선으로 할당한다. 남은 형의 연산은 그림 2(a)에서 제어 스텝 3에 있는 +20과 같이 하나의 제어스텝에 올 수 있는 같은 형의 연산 최대수가 2 이상인 연산으로 이와 같은 연산이 할당될 수 있는 기능연산자 수가 N이고 하나의 제어 스텝에 오는 이 연산의 수가 M인 경우에 연산을 기능연산자에 할당하는 방법은 N순열 M의 가지 수가 있고 할당하는 방법에 따라 연결 구조가 변하게 된다. 그림 2(b)와 그림2(c)는 기능연산자의 할당 형태에 따라 연결 구조가 변하는 예를 보여준다.

그림 2(b)는 연산 +3과 +20을 기능연산자 1에, 연산 +12와 +32를 기능연산자 2에 할당하는 경우이고, 그림 3(c)는 +3, +12와 +20을 기능연산자 1에 +32를 기능연산자 2에 할당하는 경우이다. 그림 3(c)와 같이 할당할 때 연결 구조가 최소가 될 수 있다. 그림 3(c)의 할당은 그림 3(a)에서 보면 한 연산이 할당된 기능연산자와 그 연산의 입력과 출력이 되는 연산이 기능연산자가 같은 경우로서 기능연산자 할당과정에서 연결구조를 최소로 하기 위해 비용을 식 (15)와 같이 설정하고, 하나의 제어스탭에서 연산이 기능연산자에 할당될 수 있는 경우에 대하여 각각의 비용을 계산한 후 비용이 가장 큰 경우로 연산을 할당한다. 스케줄링된 결과의 첫 번째 제어 스텝에서 마지막 제어스텝까지의 식 (15)를 이용하여 모든 연산을 기능연산자에 할당한다.

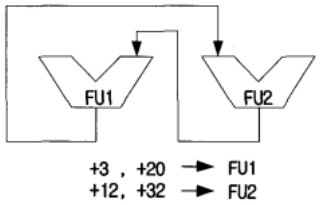
$$cost = \sum_{x=1}^{N} \sum_{y=1}^{M} ((\text{같은입력FU수}+\text{같은출력FU수})-(\text{다른입력FU수}+\text{다른출력FU수})) \quad \cdots(15)$$



(a) CDFG의 예
(a) An ezample of CDFG



(b) 기능연산자의 할당
(b) Functional unit allocation



(c) 연결 구조를 고려한 기능연산자의 할당
(c) Functional unit allocation for the interconnection

그림 2. 기능연산자 할당의 예
Fig. 2. An example of Functional unit allocation

3. 레지스터 할당

본 알고리즘에서는 메모리 소자로서 레지스터를 이용한다. 따라서 메모리 할당과정은 레지스터의 수를 결정하고 연산의 출력인 변수를 레지스터에 할당하는 과정으로서 변수가 가지는 일반적인 특성과 제약 조건에 의해 크게 좌우된다.

변수의 일반적인 특성은 하나의 입력이 되는 연산을 두고 하나 이상의 출력되는 연산을 가지며 생존시간을 가진다. 생존시간은 변수의 입력 연산과 출력 연산 사이의 제어 스텝 구간이며 변수가 레지스터에 할당되는 제어 스텝 구간이다. 그러나 VHDL 기술문에서 보면, 메모리에 저장될 변수는 크게 전역 변수와 내부 변수로 구분할 수 있다. 전역 변수는 VHDL 기술문에서 2개 이상의 프로세스 문에서 읽기/쓰기를 하는 변수이고 내부 변수는 하나의 프로세스문 또는 블록문 내에서만 사용되는 변수이다. 전역 변수의 경우 시스템 레지스터로 사용되는 경우가 많고 또한 프로세스별 또는 블록별로 합성을 수행하기 때문에 각 프로세스에서 전역 변수의 읽기/쓰기를 예측하기 어렵다. 따라서 전역 변수는 변수 간의 메모리 공유를 하지 않고 별도의 메모리를 할당한다. 본 논문에서 제안하는 메모리 할당 알고리즘은 내부 변수에 대해서만 수행한다.

본 논문에서는 같은 변수명을 갖는 변수들을 같은 레지스터에 할당한다. 그러나 조건 분기 등에 의해 같은 변수명을 갖는 변수들이 다른 조건에 나타나기 때문에 처음부터 하나의 변수로 처리할 경우, 레지스터를 공유할 변수를 찾기가 쉽지 않다. 따라서 같은 변수명을 갖더라도 다른 그래프 경로에 나타나는 변수들은 새로운 변수로 분리한다.

```
변수들을 register 입력 별로 분리.
각 변수의 life time을 구성.
Life time의 중첩 관계에 의해 구간 그래프 생성.
for(각 변수의 구간 그래프에 대하여)
    최소의 cluster partitioning 알고리듬.
for(각 minimal cluster partitions 에 대하여)
    각 register를 할당.
```

그림 3. 레지스터 할당 알고리즘
Fig. 3. Register allocation algorithm

그림 3과 같이 변수의 생존주기를 구성한다. 그림 3과 같이 변수의 생존주기가 구해지면 1) 같은 변수명, 2) 생존주기의 중첩, 3) 동작 조건 등을 검색하여 변수명이 같거나, 동작 조건이 중첩되지 않거나, 또는 변수의 생존주기가 중복되는 기간 중 동작 조건이 다른 변수들은 레지스터 공유가 가능한 변수라 하고 각 변수 간의 레지스터 공유 가능 여부를 나타내는 구간 그래프를 구성한다. 구성된 구간 그래프를 최소의 클러스터 분할 알고리즘을 적용하여 최소의 클러스터 분할 수를 구성하는 최대 크기의 클러스터들을 구하며 알고리즘은 그림 3과 같다.

각 클러스터의 변수들에 대하여 각각의 레지스터를 결정하고 분리된 변수들을 각 결정된 수만큼의 레지스터에 할당한다. 따라서 하나의 레지스터에 할당되는 모든 변수는 같은 기능연산자의 입력을 두게 된다. 이는 기능연산자의 출력과 레지스터의 입력 사이에 멀티플렉서를 지원하지 않고 레지스터 출력과 기능연산자의 입력 사이에만 멀티플렉서를 지원함으로써 멀티플렉서의 입력이 공유되는 경우를 증가시키게 되고 따라서 연결 신호선 수가 줄어드는 결과를 가져온다.

Ⅲ. 실험 및 결과

본 논문에서 제안한 스케줄링 알고리즘의 타당성을 입증하기 위하여, 다음과 같은 벤치마크 모델에 적용했다. 실험의 대상인 벤치마크 모델로서는 High-level Synthesis Workshop에서 표준벤치마크 모델로 채택된 5차 디지털 웨이브 필터[9]를 택하였으며, 실험결과는 공개된 스케줄링 성능이 우수한 것으로 판정된 ALPS 시스템의 스케줄링[10] 결과와 비교하였다. 5차 디지털 웨이브 필터와 6차 대역 통과 필터를 실험모델로 선택한 대부분 논문에서와 마찬가지로, '+' 연산의 지연시간은 $40ns$, '*' 연산의 지연시간은 $80ns$, 제어스텝의 시간 간격은 50 ns으로 설정하였다.

표 3과 표 4는 fifth-order elliptic filter 와 sixth-order bandpass filter에 대한 실험결과를 보여준다. 하드웨어 할당에서는 제어 스텝의 수 와 연산자의 수는 기존의 다른 시스템과 같으나, 연결구조의 비용을 고려한 멀티플렉서 입력 수의 측면에서 표 3의 MAP과 비교 할 때 약 67%만을 필요로 하는 결과를 보여주고 있다. 그러나 본 하드웨어 할당이 주로 멀티플렉서를 연결구조로 사용했기 때문에 버스 측면에서는 유리한 결과를 얻지 못했다.

하드웨어 할당에서는 제어 스텝의 수와 연산자의 수는 기존의 다른 시스템과 같으나, 연결구조의 비용을 고려한 멀티플렉서 입력 수의 측면에서 표 4의 HAL과 비교 할 때, 약 30%만을 필요로 하는 결과를 보여주고 있다. 그러나 본 논문의 하드웨어 할당 결과는 작은 수의 논리 블록에 의해 연결 구조가 구현됨을 보여주며, 하드웨어 할당의 경우 칩 면적을 감소시킬 수 있었고, 멀티플렉서 입력 항목에서의 결과값을 보면 본 논문에서 제안한 알고리즘을 적용한 실험결과가 다른 시스템보다 감소되었음을 확인할 수 있었다.

표 3. 5차 디지털 웨이브에 대한 실험 결과
Table 3. Experimental resunls for the fifth-order elliptic filter

	본 논문	Map	ADPS	PUBSS
C_steps	11	11	11	11
#ALUs	2	2	2	2
#Multipliers	1	1	1	1
#MUX_inputs	8 / (4xMUX21)	12	27	10
#Buses	5	5	N/A	5
#Registers	11	11	14	11

표 4. 6차 대역 통과 필터에 대한 실험 결과
Table 4. Experimental results for the sixth-order bandpass filter

	본 논문	HAL	Map	SPAID
C_steps	19	19	19	19
#ALUs	2	2	2	2
#Multipliers	1	1	1	1
#MUX_inputs	9 / (4×MUX21)	26	10	17
#Buses	5	6	5	N/A
#Registers	12	12	14	19

Ⅳ. 결 론

본 논문에서는 고성능 차량용 SoC 설계자동화를 위한 상위수준 합성과정에서의 레지스터 할당 알고리즘과 자원할당 알고리즘을 제안한다. 상위수준 합성에서 가장 중요한 연산자의 특성과 데이터패스의 구조를 분석하고, 멀티사이클 연산의 스케줄링 시 가상연산자 개념을 도입함으로써, 멀티사이클 연산을 구현하는 연산자의 유형과 관계없이 공통으로 적용할 수 있는 자원할당 알고리즘을 이용하여 증명하였다. 연산자 간을 연결하는 신호선이 반복적으로 이용되어 연결 신호선 수가 최소가 될 수 있도록 기능연산자를 할당하고, 레지스터 할당 시 연결구조에 따라 가중치를 갖는 구간 그래프를 구성한다. 최소의 클러스터 칸막이 알고리즘을 이용하여 생성된 최대 크기의 클러스터들에 연결 구조가 고려된 레지스터를 할당한다. 연결구조에 대한 멀티플렉서의 중복 입력을 제거하고 연산자에 연결된 멀티플렉서 간의 입력을 교환하는 입력 정렬 과정으로 연결구조를 최소화한다.

본 논문에서 제안한 알고리즘의 성능을 평가하기 위하여, 표준벤치마크 모델 5차 디지털 웨이브 필터에 대한 스케줄링을 한 결과, 기존의 데이터패스 스케줄링 결과와 일치함으로써, 제시된 모든 수식이 정확하게 기술되었음을 알 수 있었고, 알고리즘의 효용성을 입증하였다.

References

[1] B-J. Kang, J-W. Kim, "Decision Support System of Obstacle Avoidance for Mobile Vehicles", Journal of the Korea Academia-Industrial cooperation Society(JKAIS), Vol.19, No.6, pp. 639-645, 2018.
DOI: http://dx.doi.org/10.5762/KAIS.2018.19.6.639

[2] C-H. Lin, "A New Register Transfer Level Synthesis Methodology for Efficient SOC Design", The Journal of The Institute of Internet, Broadcasting and Communication(JIIBC), VOL.11 No.2, April 2011.
DOI: https://doi.org/10.7236/JIWIT.2011.11.2.161

[3] Gebotys, Catherine H., and Mohamed I. Elmasry. "Optimal VLSI architectural synthesis: area, performance and testability." Vol. 158. Springer Science & Business Media, 2012.
DOI: http://dx.doi.org/10.1007/978-1-4615-4018-2

[4] Gajski, Daniel D., et al. "High—Level Synthesis: Introduction to Chip and System Design." Springer Science & Business Media, 2012.

[5] C.Tseng, D.P.Siewiorek, "Automated Synthesis of Datapath in Digital System", IEEE Tr.CAD, Vol.CAD-5, pp.379-385, July. 1986.
DOI: http://doi.org/10.1109/TCAD.1986.1270207

[6] S.Y.Kung, H.J.Whitehouse, T.Kailath, "VLSI and Modern Signal Processing", Englewood Cliffs, NJ : Prentice Hall, 1985.

[7] H.Tricky, "Flamel : A High-Level Hardware Compiler", IEEE Tr. on CAD, pp.259-269, March. 1987.
DOI: http://doi.org/10.1109/TCAD.1987.1270270

[8] P.G.Paulin, et al, "HAL : A Multi-Paradigm Approach to Automatic Data Path Synthesis", Proc.of 23rd DAC, pp.263-270, June. 1986.
DOI: http://doi.org/10.1109/DAC.1986.1586099

[9] Lars Wanhammar, Ya Jun Yu, "Digital Filter Structures and Their Implementation", Academic Press Library in Signal Processing, Vol.1, pp.245-338, 2014.
DOI: http://doi.org/10.1016/B978-0-12-396502-8.00006-1

[10] Jothi Komal, Akkary Haitham, "Tuning the continual flow pipeline architecture." In: Proceedings of the 27th international ACM conference on International conference on supercomputing. ACM, pp.243-252. 2013.
DOI: http://doi.org/10.1145/2464996.2465011

저 자 소 개

장 정 욱(준회원)

- 현재 : 세명대학교 일반대학원 박사과정 (전산정보학 전공)
- e-mail : nagajang@naver.com
- 주요관심분야 : 스마트응용, IoT, 빅데이터, 모바일컴퓨팅, 임베디드 시스템, Cad 알고리즘, SoC 설계, 자율주행시스템

인 치 호(정회원)

- 현재 : 세명대학교 컴퓨터학부 교수
- e-mail : ich410@semyung.ac.kr
- 주요관심분야 : 스마트응용, IoT, 빅데이터, 모바일컴퓨팅, 임베디드 시스템, Cad 알고리즘, SoC 설계, 자율주행시스템

※ 본 논문은 2019학년도 세명대학교 연구년 지원에 의한 연구임.