멀티플렉서 기반의 비트 연속 승산기를 이용한 시스톨릭 어레이 띠 행렬 승산기 구현

한영욱, 김진만, 유명근, 송기용
충북대학교 컴퓨터공학과

Implementation of the Systolic Array for Band Matrix Multiplication using Mutiplexer-based Bit-serial Multiplier

Young-Wook Han, Jin-Man Kim, Myoung-Keun You, Gi-Yong Song
{prince06a, kizima, mkyou77}@archi.chungbuk.ac.kr, gysong@chungbuk.ac.kr
Dept. of Computer Engineering, Chungbuk National University

요 약

본 논문에서는 모듈성과 확장성을 갖는 시스톨릭 어레이를 이용한 두 띠 행렬의 비트 연속 승산기 구현에 대하여 기술한다. 띠 폭이 3인 4×4 띠 행렬이 주어질 때 워드 레블 승산기 설계를 위한 3차원 DG로부터 2차원 시스톨릭 어레이를 유도한 후, 워드 레블 PE를 비트 연속 승산기와 가산기를 이용하여 비트 레블 PE로 변환시켜 띠 행렬의 비트 레블 승산기를 설계한다. 구현된 워드 레블 승산기와 비트 레블 승산기는 RT 수준에서 VHDL로 모델링하여 동작을 검증하였다. 검증된 시스톨릭 어레이를 이용한 워드 레블 승산기와 비트 레블 승산기는 Hynix에서 제공하는 0.35㎛ 셀 라이브러리를 사용하여 Synopsys design compiler로 합성되었다.

I. 서론

MAC(multiply-accumulate)의 집약적인 구성으로 이루어진 행렬 연산은 다양한 신호와 영상처리 응용에 널리 이용되고 있다. $m \times n$ 행렬은 m개의 행과 n개의 열로 이루어져 있고 벡터는 하나의 특별한 형태의 행렬로 간주되어질 수 있다.

띠 행렬(band-matrix)은 상삼각행렬과 하삼각행렬의 특정한 원소들의 값이 0인 행렬로 0이 아닌 원소들이 띠 모양으로 대각 원소를 중심으로 이루어진 행렬이다. 따라서 일반 행렬 간의 승산 알고리즘을 통하여 특정한 원소의 값이 0으로 이루어진 띠 행렬 간의 승산을 구하는 것은 비효율적이다. 즉, $N \times N$ 행렬 A, B가 모두 띠 행렬이고 띠폭(band-width)이 각각 P, Q이라고 하면, $C=A \cdot B$를 계산할 때 $N \times N$ 어레이가 아닌 $P \times Q$ 사각형의 어레이로 C를 계산하여 모든 병행성(full parallelism)을 얻는 것이 가능하다.

본 논문에서는 띠 폭이 3인 4×4 띠 행렬이 주어질 때 띠 행렬 간의 승산기 설계를 위하여 동시성, 모듈성, 확장성, 공간적 및 시간적 지역성 특징을 갖는 시스톨릭 어레이(Systolic array)를 사용한다.[1][2][3] 먼저 띠 행렬 간의 비트 레블 승산기 설계를 위하여 워드 레블(word-level) 3차원 DG (dependence-graph)로부터 2차원 시스톨릭 어레이를 유도한 후, 면적과 성능 향상을 위해 비트 연속(bit-seiral) 승산기와 가산기를 이용하여 구현된 PE(process element)를 사용하여 시스톨릭 어레이의 띠 행렬 간 비트 레블 승산기의 구현에 대하여 기술한다. 시스톨릭 어레이를 이용한 띠 행렬의 각각의 승산기는 RT 수준에서 VHDL로 모델링하여 동작을 검증하였다. 검증된 띠 행렬의 각각의 승산기는 Hynix에서 제공되는 0.35㎛ 셀 라이브러리를 사용하여 합성되었으며 면적과 임계지연시간 측면에서 비교 분석되었다.

II. 띠 행렬 승산을 위한 워드 레벨 시스톨릭 어레이

이 장에서는 띠 행렬 간의 워드 레블 승산기를 위해 DG로부터 프로젝션을 통하여 SFG(signal flow graph)를 얻고, 시스톨릭화(systolization) 과정을 통해 워드 레블 시스톨릭 어레이를 유도하는 과정을 기술한다.[3]

행렬의 승산은 간단히 $C=A \cdot B$로 표현되며 식 ?과 같다 .

$$c_{ij} = \sum_{k=1}^{N} a_{ik} b_{kj} \qquad (1)$$

식 ?은 모든 승산이 서로 어떠한 의존도를 가지지 않기 때문에 동시에 계산되어 질 수 있음을 함축하고 있다. 최대한의 병행성을 위해서는 입력 데이터를 곱셈기에 전달해야 한다. 즉, 적어도 $2N^2$개의 입력 데이터 링크가 필요하다.

식 ?을 재귀적형태로 표현하면 식 ?와 같다

$$c_{ij}^{(k)} = c_{ij}^{(k-1)} + a_{ik}\, b_{kj} \qquad (2)$$

식 ?를 이용하여 얻어진 워드 레블 DG는 그림 2-?에 보인다. 그림 2-?을 i-방향으로 프로젝션하여 얻어진 SFG 즉, 그림 2-?에 보이는 것처럼 행렬 B의 원소들이 각각의 PE에 상수로 저장될 수 있음을 의미한다. 또한 승산과 가산에 의한 충분한 동적인 공간을 유지하기 위해서 추가적인 k-비트의 보호 비트가 필요하게 된다. 식 ?에서 각각의 c_{ij}를 구하는데 필요한 가산기의 수를 N-1이라고 하면 unsigned 데이터인 경우 추가적인 k-비트는 $\lceil \log_2(N-1) \rceil$로 주어지고 signed데이터인 경우는 $\lceil \log_2(N-1)-1 \rceil$로 주어진다. 본 논문에서는 행렬 A, B가 unsigned 데이터로 이루어진 4×4인 행렬이므로 가산기는 4+4+ $\lceil \log_2(4-1) \rceil$ =10비트의 크기를 가져야 한다.[4]

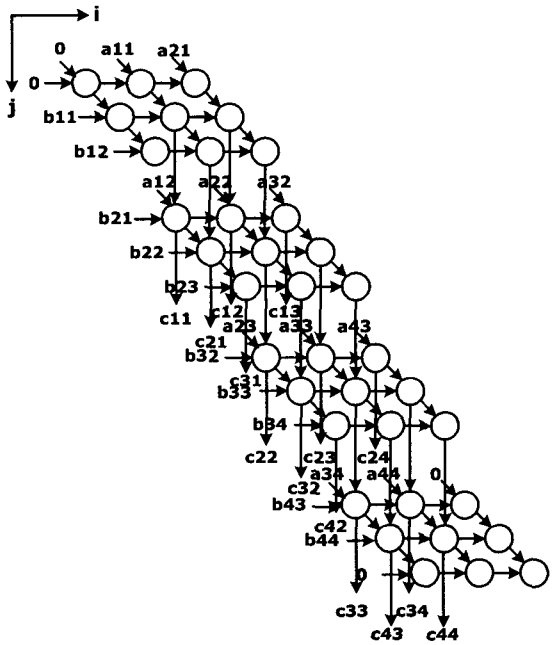

그림 2-3. 띠 행렬 승산의 DG

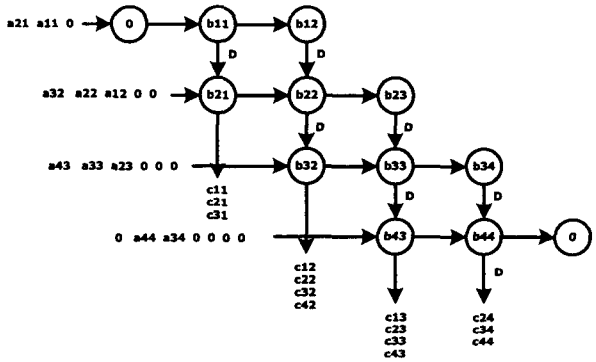

그림 2-4. i-방향으로 프로젝션해서 얻어진 SFG

DG에서 프로젝션해서 얻어진 SFG는 PE들 사이의 데이터 전달 과정에서 단위시간 이상의 지연이 존재해야 한다는 시스톨릭 어레이의 특징인 시간적 지역성을 만족하지 않기 때문에 시스톨릭화(systolization) 과정을 적용해야 한다. 위에서 얻어진 SFG를 시스톨릭 어레이로 만들기 위해 cut-sets으로 나눈 SFG를 그림 2-?보인다.

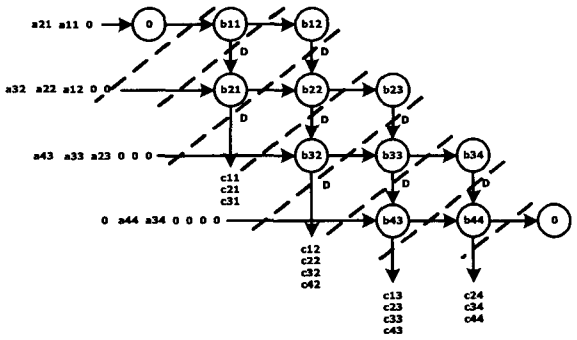

그림 2-5. SFG의 cut-sets

cut-sets을 통해 각 PE의 입출력 방향 모두 다 지연(delay)을 할당할 수 있다. 위 과정을 통해 그림 2-?와 같은 띠 행렬의 워드 레블 승산기를 위한 시스톨릭 어레이를 얻을 수 있다. PE의 내부 구조는 그림 2-?에 보인다.

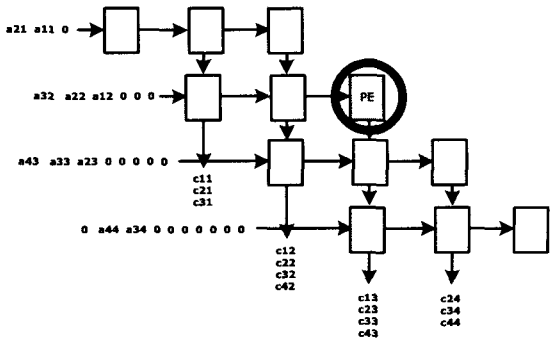

그림 2-6. 띠 행렬 승산의 시스톨릭 어레이

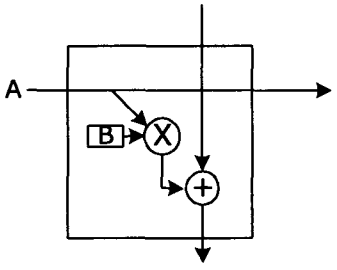

그림 2-7. PE의 구조

III. 띠 행렬 승산을 위한 비트 연속 시스톨릭 어레이

이 장에서는 Ⅱ장에서 소개한 띠 행렬 승산을 위한 워드 레블 시스톨릭 어레이의 PE를 면적과 임계 지연 시간측면에서 더 나은 결과를 얻기 위해 비트 연속 승산기를 사용한 PE로 전환하여 비트 레블 시스톨릭 어레이를 구현하는 과정을 기술한다.

1. 비트 연속 승산기

비트 연속 승산기는 피승수와 승수를 입력 데이터로 받을 때 승수의 각 비트를 멀티플렉서(multiplexer)의 선택 인자로 하고 0과 피승수가 멀티플렉서의 입력으로 전달되면 피승수가 승수에 의해 결정되도록 하는 구조를 가진다. 즉, 피승수의 비트수는 비트 연속 승산기 구조에 무관하고 필요한 멀티플렉서는 승수의 비트 수에 비례한다. 또한 승수가 N비트일 때 사용되는 전가산기의 개수는 N-1이며 전가산기 사이에 지연 소자를 이용하여 클럭 주기를 줄이는 파이프라인 구조를 가진다. 계산 값은 $\lfloor \log_2 N \rfloor$ -1 클럭 후부터 출력된다.

본 논문에서 사용된 승수가 4비트인 비트 연속 승산기는 4개의 멀티플렉서, 3개의 전가산기를 가지고 2단 파이프라인을 가지며 구조는 그림 3-?에 보인다.

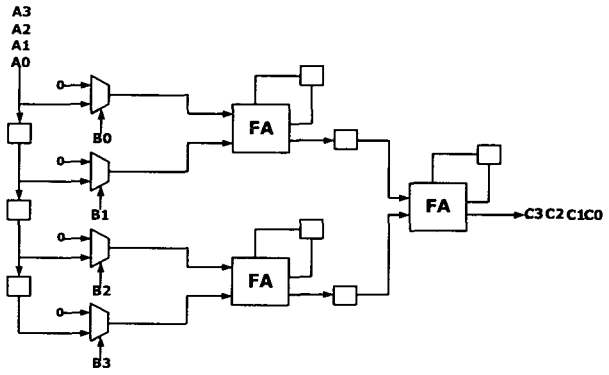

그림 3-1. 비트 연속 승산기

2. 비트 레블 시스톨릭 어레이 PE의 구조

그림 2-?에서 PE의 워드 레블 연산을 비트 연속 승산기와 추가적인 지연소자 및 전가산기 각 1개씩을 사용하여 구현된 비트 레블 시스톨릭 어레이 PE의 구조를 그림 3-?에 보인다.

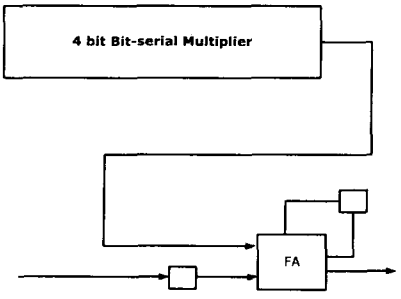

그림 3-2. 비트 레블 시스톨릭 어레이 PE

워드 레블 PE를 비트 레블 PE로 대체한 후, 행렬 A의 입력을 비트열로 주면 띠 행렬 승산의 비트 레블 시스톨릭 어레이의 구조를 가진다.

Ⅳ. 시뮬레이션, 합성 및 구현

본 논문에서는 띠 폭이 3인 4×4 띠 행렬을 사용하여 띠 행렬 간의 승산을 워드 레블 시스톨릭 어레이와 비트 레블 시스톨릭 어레이를 이용한 각 방법으로 구현하여 면적과 임계 지연 시간을 비교 분석하였다.

식 ?에서 A, B를 아래와 같이 주었을 때, 행렬 간의 승산 결과를 보인다.

$$\begin{bmatrix} 17 & 25 & 18 & 0 \\ 19 & 72 & 37 & 14 \\ 12 & 38 & 68 & 22 \\ 0 & 25 & 26 & 19 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 0 & 0 \\ 1 & 5 & 7 & 0 \\ 0 & 4 & 2 & 6 \\ 0 & 0 & 5 & 3 \end{bmatrix} \begin{bmatrix} 4 & 2 & 0 & 0 \\ 3 & 7 & 6 & 0 \\ 0 & 5 & 1 & 2 \\ 0 & 0 & 7 & 3 \end{bmatrix}$$

위의 행렬 A, B를 입력으로 주었을 때 VHDL[5]로 구현한 워드 레블 시스톨릭 어레이를 이용한 시뮬레이션 결과를 그림 4-?에 보인다.

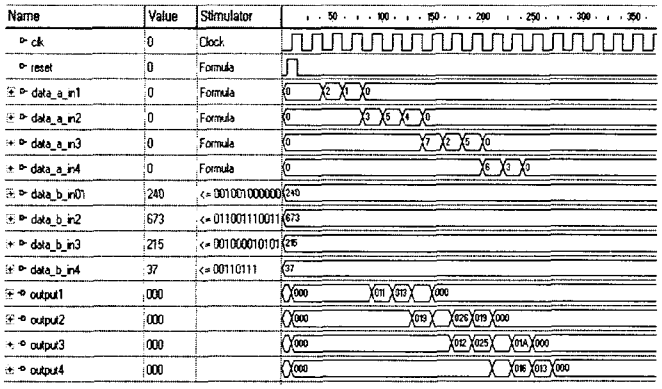

그림 4-1. 워드 레블 시스톨릭 어레이의 시뮬레이션 결과

그리고, 비트 레블 시스톨릭 어레이를 이용한 시뮬레이션 결과를 그림 4-?에 보인다.

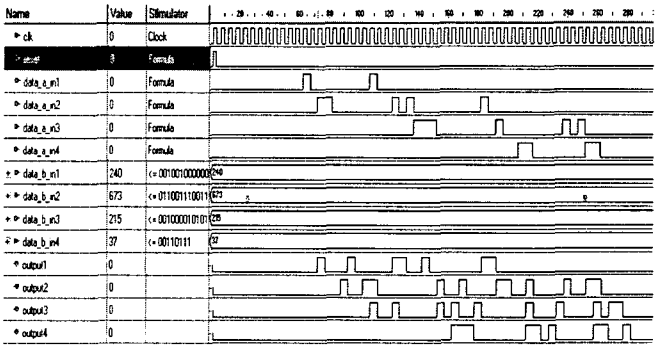

그림 4-2. 비트 레블 시스톨릭 어레이의 시뮬레이션 결과

검증된 각각의 시스톨릭 어레이는 Hynix에서 제공하는 0.35 ㎛ 셀 라이브러리를 사용하여 Synopsys design compiler [6][7]로 합성되었다. 각각의 시스톨릭 어레이와 비트 레블 시스톨릭 어레이 PE에 대한 합성 Schematic은 그림 4-?, 4-?, 4-?에 보인다. 그리고, PE 내에서 사용된 비트 연속 승산기의 합성 Schematic은 그림 4-? 보인다. 또한 두 디자인의 면적 및 성능 분석은 표 1에 보인다.

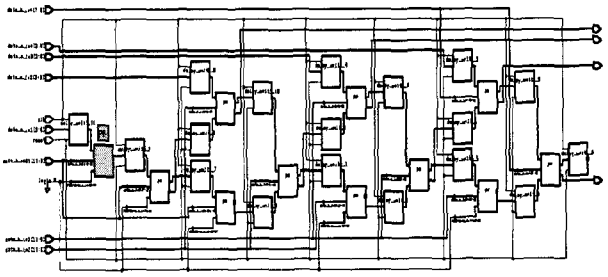

그림 4-3 워드 레블 시스톨릭 어레이의 합성 Schematic

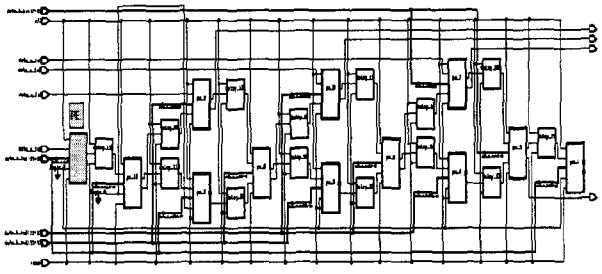

그림 4-4 비트 레블 시스톨릭 어레이의 합성 Schematic

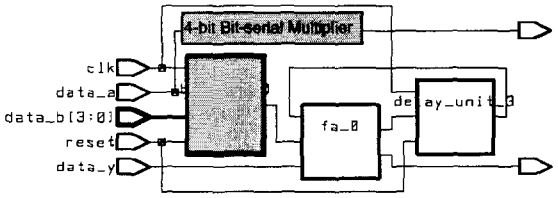

그림 4-5 PE 합성 Schematic

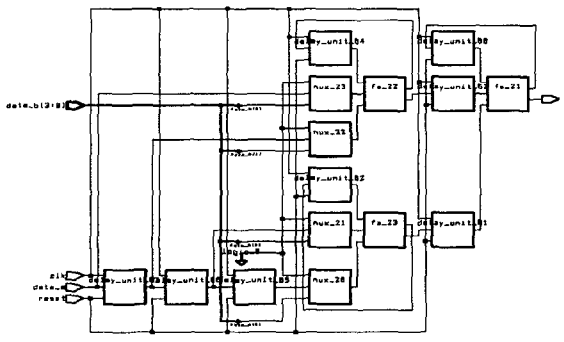

그림 4-6 4-비트 연속 승산기의 합성 Schematic

표 2. 합성 결과

합성 정보 \ 디자인	워드 레블	비트 레블
Combinational Area	2090	595
Noncombinational Area	644	938
Net Interconnect Area	6.32	4.07
Total Area	2740.32	1537.07
Critical Delay	13.88 ns	4.88 ns

V. 결론

본 논문에서는 두 개의 띠 행렬이 주어질 때 워드 레블 시스톨릭 어레이와 비트 레블 시스톨릭 어레이를 이용한 승산기 구현에 대하여 기술하였으며, 워드 레블 승산기와 비트 연속 승산기는 RT 수준에서 VHDL로 모델링하고 시뮬레이션을 통해 동작을 검증하였다. 검증된 시스톨릭 어레이를 이용한 각각의 승산기는 Hynix에서 제공하는 0.35㎛ 셀 라이브러리를 사용하여 Synopsys design compiler로 합성되었다. 표 ?에 보이는 것처럼 비트 레블 디자인은 워드 레블 디자인에 비해 면적에서는 약 1/2로 줄어들었으며 임계 지연시간 측면에서는 약 1/3로 줄어들어 성능이 향상 되었다. MAC 집약적인 구성인 행렬 승산에서 데이터의 비트수가 커지면 커질수록 비트 레블 시스톨릭 어레이를 이용한 디자인은 면적과 임계 지연시간 측면에서 워드 레블 시스톨릭 어레이를 이용한 디자인에 비해 우수할 것으로 기대된다.

참고문헌

[1] H. T. Kung and C. E. Leiserson, "Systolic array (for VLSI)," In *sparse Matrix Symposium*, pp. 256-282, SIAM, 1978.

[2] H.T.Kung, "Why Systolic Architectures?," *Computer* Vol.15, No.1, pp.37-46, January 1982.

[3] S.Y.Kung, *VLSI Array Processors,* Prentice Hall, 1988.

[4] U.Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Springer, 2001.

[5] Y.C.Hsu, K.F.Tsai, J.T.Liu and E.S.Lin, *VHDL Modeling for Digital Design Synthesis,* Kluwer Academic Publishers, 1995.

[6] K.C. Chang, *Digital Systems Design with VHDL and Synthesis*, IEEE Computer Society Press, 1999.

[7] Weng Fook Lee, *VHDL Coding and Logic Synthesis with Synopsys*, Academic Press, 2000.