|  |
| --- |
| 뉴로모픽 프로세서를 구성하는 빌딩블록의  최적화에 관한 연구  (Optimizing building-blocks of neuromorphic processors)  김 세 반, 황 동 준  (Se-Ban Kim, Dong-Jun Hwang)  **Abstract**  본 연구를 통하여서, 시간 지연 신경망(Time Delay Neural Network, TDNN)의 필수 빌딩블록(Building blocks) 중 하나인 지연회로(Delay circuits)와 곱셈기(Multiplier)의 구현원리를 고찰하고, FPGA에 프로그래밍 할 수 있는 HDL로 기술된 신경망 회로에서, 기존의 회로보다 크기, 소비 전력 측면에서 최적할 수 있는 설계방법을 찾아내었다. 지연회로의 경우, 지연 구간이 크지 않는 회로로는 전통적인 방법인 시프트레지스터로 구현하는 것이 좋으며, 지연 구간이 클 경우에는 메모리(RAM)를 이용한 방식이 앞의 방법보다 효율적일 수 있음을 확인할 수 있었다. 또한 곱셈기 회로의 경우에는 입력 스트림(Stream)을 직렬데이터 처리 방식으로 처리하는 방식인 Bit-Serial곱셈기를 제작함으로써, 회로의 면적을 큰 폭으로 줄일 수 있었다. |
|  |

**Ⅰ. 서 론**

최근 여러 해 동안, 기존의 순차적으로 데이 터를 처리하는 폰 노이만 구조와는 다른, 사람의 신경계(nerve system)를 모방한 '뉴로모픽 프로 세서'의 개념이 대두됨에 따라 현재 뉴로모픽 프로세서에 관한 다양한 연구가 진행 되고 있다. 또한 칩 설계 기술과 컴퓨팅의 기술적인 발전과 저전력 설계가 요구됨에 따라, 사람의 뇌와 유사 하게 데이터를 처리하는 뉴로모픽 프로세서의 각 빌딩블록(building blocks)을 최적화함으로써 회로 크기를 줄이고, 더욱 많은 소자들을 집적할 수 있도록 설계하는 방법에 대한 필요성 또한 증가 하게 되었다. 이 연구에선 주로 FPGA 칩에서의 회로 설계를 다루었으며, 본문의 2장에서, 뉴로 모픽 프로세서를 구성하는 빌딩 블록들 중에 물리계층(physical layer)사이에서 작용 하는 지연 회로(Delay circuits)에 대한 최적화를 다루고, 3장에서 데이터에 각각의 가중치(weight)를 곱함 으로써, 역치(threshold)를 계산하는 역할을 하는 곱셈기(multiplier)에 대한 최적화를 다룸으로써, 회로의 집적도를 높일 수 있는 설계 방법을 제시한다.

**Ⅱ. 지연회로(Delay-circuit)**

1. 배경(Background)

시간 지연 신경망(Time Delay Neural Network, TDNN)은 1988년 Waibel에 의한 기존의 인공 신경망에 시간 지연이라는 시간 개념을 넣어 시간에 따라 변하는 값(time-variance)도 고려할 수 있게 설계한 신경망이다[1]. 입력 스트림(stream) 으로 들어오는 데이터 값을 각 계층들 사이에서 입력 데이터의 프레임(frame)의 시간에 따라 변하 는 값을 고려하여 처리함으로써, 이 신경망을 음성인식, 동적 이미지의 패턴 인식 등에 응용될 수 있다. 이 신경망의 시간 지연(time-delay) 특성 에 의해 신경망 회로 내부에 설계되어야 하는 빌딩블록인 지연회로(delay-circuit)의 회로를 최적 화함으로써, 회로의 집적도를 높일 수 있다.

2. Shift-register

참고가 될 수 있는 지연회로로써, 가장 전통적 인 방법인 시프트 레지스터(shift-register)를 이용하 여 지연회로를 구현하는 방법을 살펴본다. 그림 1에서 살펴볼 수 있듯이 시프트 레지스터는 다수의 D-플립플롭(flip-flop)을 직렬로 연결함으로 설계할 수 있으며, 각각의 플립플롭은 동기화 되어있는 클럭(clock)의 상승 엣지(positive edge)에 맞추어 데이터를 전달하게 된다.

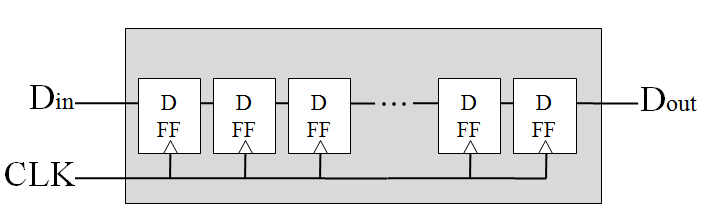


그림1. Shift-register

FPGA칩에서 시프트 레지스터는 논리 셀(logic -cell)이라 불리는 영역의 룩 업 테이블(Look-Up Table, LUT)에서 맵핑(mapping)된다. 그림2에서 확인할 수 있듯이, 1개의 로직 셀의 LUT를 묶어 32 비트 데이터를 처리할 수 있는 시프트 레지 스터를 설계할 수 있다[2].

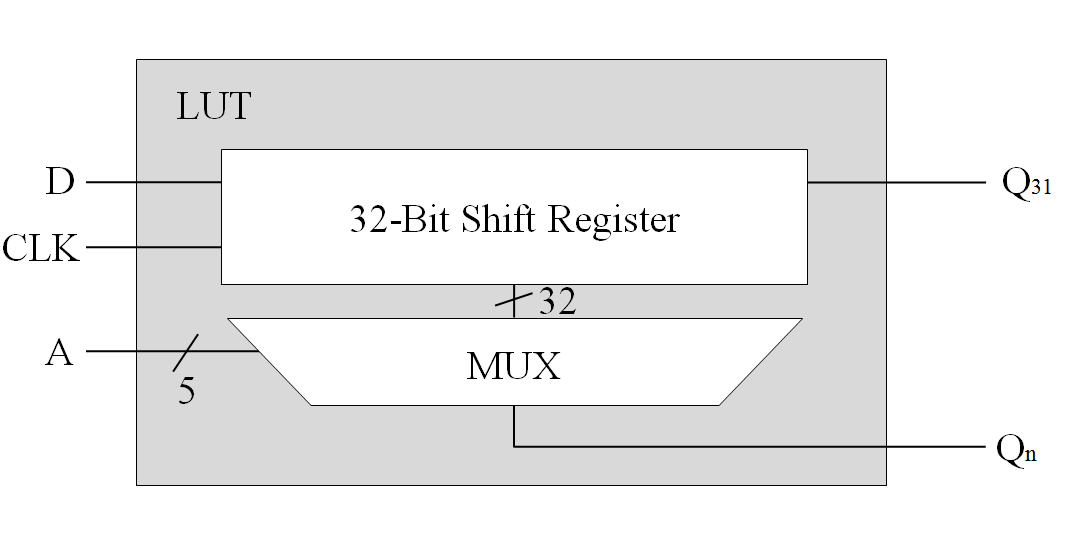


그림2. Mapping shift-register on FPGA

3. RAM(Random Access Memory)

다음으로 램의 동작원리를 이용한 지연회로 모듈을 구현하는 방식을 살펴보면, 일반적인 램에 적절한 관계식을 갖는 쓰기 주소(write-address)와 읽기 주소(read-address)를 지정하여 이를 순차적 으로 증가 시키면서(counting) 데이터를 엑세스 할

수 있도록 하는 방식으로 지연회로를 설계한다. 식 1과 그림 3에, 쓰기 주소에 대한 읽기 주소의 관계를 나타낸다.

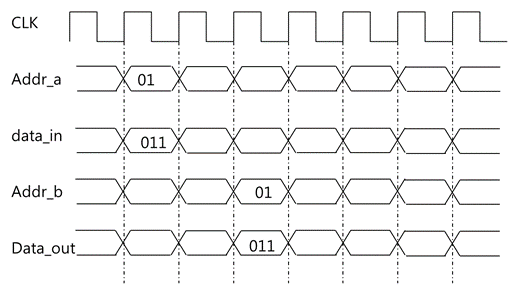
****

그림 3. Timing-diagram

위의 타이밍선도를 보면, 식 (1)의 관계식에 따라 증가하는 주소에 따라, 메모리의 크기에 해당하는 만큼 데이터를 지연시킬 수 있으며, 따라서 이러한 원리를 적용하여 지연회로를 설계한다. 다음으로, RAM을 이용한 지연회로의 설계 방법을 제시한다.

3.1 Distributed RAM

FGPA칩의 내부 게이트(gates)을 이용하여 설계 할 수 있는 Distributed RAM은 그림 4에서 확인 할 수 있듯이, HDL(Hardware Description Language)을 이용하여 데이터를 저장할 수 있는 모듈과 주소를 생성하는 모듈을 결합하여 설계한다.

3.2 Block Memory

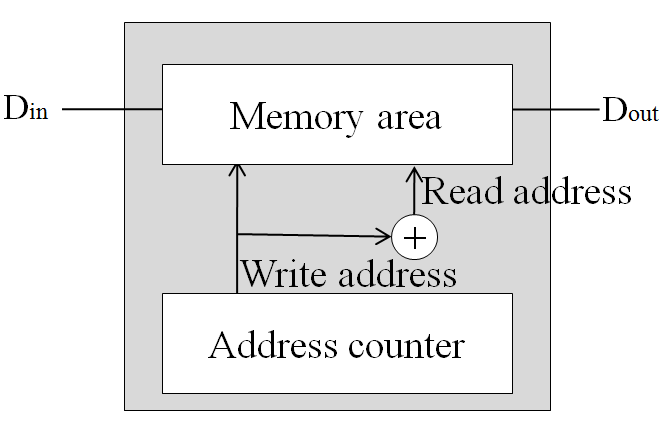
앞선 주제와 마찬가지로, FPGA의 내부 모듈들 

그림4. Block diagram: Distributed RAM

중 블록메모리(block memory)를 사용하여 지연 회로를 설계하는 방법을 알아보자.

그림 5에 Dual-port block memory에 대한 개념 도를 나타내었다[2].

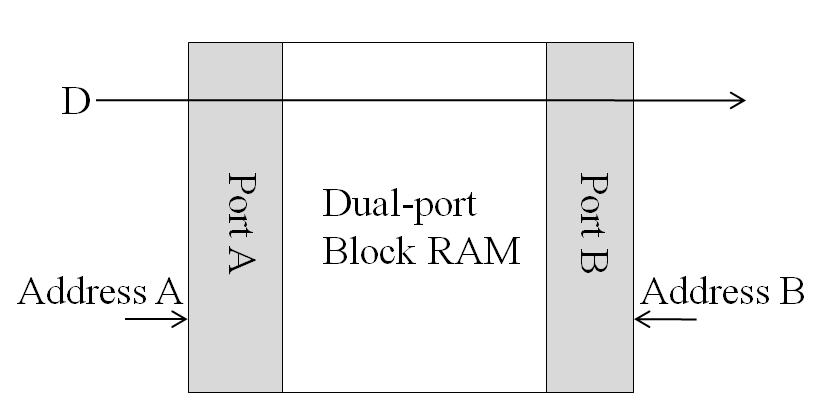


그림5. Dual-port Block RAM

일반적인 FPGA칩이 가지고 있는 블록메모리 모듈은 18Kbyte와 32Kbyte의 데이터를 저장할 수 있으며 이를 이용해서 RAM방식의 지연회로 설계 법을 이용하여 회로를 설계한다.

**Ⅲ. 직렬 곱셈기(Serial-multiplier)**

1. 배경(Background)

곱셈기(multiplier)는 뉴로모픽 프로세서의 신경 망에서 입력 데이터와 가중치를 곱하면서 각각 계산되는 데이터를 누적함으로써, 특정한 크기를 갖는 자극 신호 이상에서만 반응하는 사람의 역치(threshold)에 해당하는 값을 계산하는 역할을 한다. 곱셈기 모듈은 각각의 신경망에서 각각의 시냅스 모듈을 구성하기 때문에, 그 크기를 줄이는 것은 신경망의 집적도를 크게 향상시키는 동시에 전력소모를 크게 줄일 수 있다.

1. 비트 연산:곱셈

이진 바이너리코드(binary code)의 비트 연산에 있어, m 비트의 승수와 n 비트의 피승수를 곱하게 되면, m+n 비트의 크기를 같는 곱이 나온다.

(2)

식 (2)를 보면, A(m)은 m 비트 데이터, B(n)는 n 비트 피승수(multiplicand), P(m+n)는 곱(product)을 의미한다. 바이너리 코드에 대한 곱셈과정을 그림6에 도시하였다.



그림6. 비트 연산: 곱셈

3. Signed extension

부호 있는 바이너리 코드의 곱셈연산은 부호 없는 수의 곱셈과는 다르게, m 비트 데이터에 대해 최상위 비트(MSB)를 고 순위의 레지스터에 복사 함으로써, 데이터를 확장(extension)하여 연산 한다 (그림7).

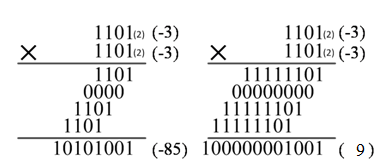


그림7. Signed extension

이러한 원리에 기인하여, 부호 있는 수의 곱셈은 더 많은 레지스터를 필요로 하게 된다. 이러한 문제점을 가지고 있는 기존의 빌딩 블록을 최적화 하기 위한 방법으로, 2의 보수 연산 알고리즘을 사용하여, 곱셈기 회로를 설계함으로써 버퍼의 용량을 줄임으로 회로의 크기를 줄일 수 있다. 그림 8에 나타나있듯이, 데이터의 마지막 비트를 저장하여 부분 곱들의 버퍼의 크기를 줄일 수 있다. 마지막 부분 곱(partial product)에 보수를 취하고, 첫 번째 부분 곱의 MSB에 캐리(carry)를 더함으로써, 곱의 결과는 같게 할 수 있다[4]. 다른 방법으로, 한번에 많은 데이터를 처리하나 회로의 크기가 큰 병렬 데이터 처리 방식보다는 한 비트 데이터를 처리하지만 회로의 크기가 작은 직렬 데이터 처리방식을 이용하여 회로의 크기를 줄인다.

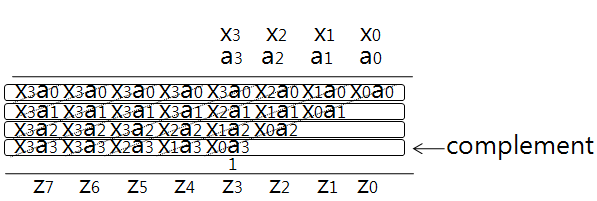


그림 8. Two’s complement algorithm

4. Bit-serial 알고리즘

2의 보수 연산 알고리즘을 사용하면서 직렬 데이터를 처리함으로써 회로 크기를 줄일 수 있는 방법인 Bit-serial알고리즘[3]을 소개한다. 이 알고리즘은 다음 4 단계에 의해 정의된다.

1. 1비트 직렬 데이터를 m비트 피승수와 and 연산한다. MSB는 보수를 취한다.
2. m비트의 연산된 데이터를 전가산기(FA)에 전달한다(매번 초기 연산을 수행할 때 MSB 의 FA의 캐리는 ‘1’로 초기화한다).
3. MSB의 FA로부터 sum은 LSB을 향해 시프 트되면서 입력으로 전달된다.
4. 입력 데이터가 마지막 비트(MSB)일 때 해당 데이터를 저장하면서 나머지 연산을 반복 한다.

다음 절에, 위의 단계를 적용하여 설계한 Bit- serial 곱셈기 모듈을 소개한다.

5. Bit-serial 곱셈기

앞 절에서 제안한 Bit-serial알고리즘의 설계 방법 에 따라 고안된 회로는, 회로의 크기 면에서 중요한 이점을 보인다. 레지스터를 주로 사용하는 기존의 병렬 데이터 처리 방식의 회로가 아닌, 게이트 연산이 주를 이루기 때문에, 이를 구현 하기 위한 소자가 현저히 줄게 된다. 개괄적인 블록선도를 그림 9에 도시하였다.

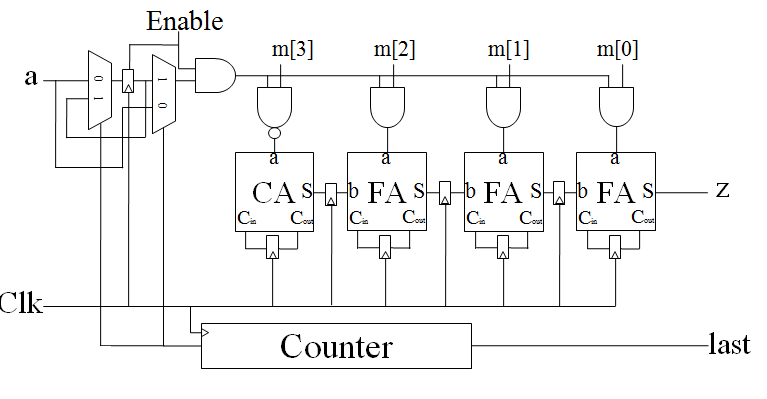


그림 9. 4 bits bit-serial multiplier

위의 그림을 보면, 입력 부분에서 카운터가 MSB 를 가리킬 때, 입력 데이터를 D-플립플롭이 계 속해서 저장하면서 피승수(m)과 and연산을 시킴 으로써 sign-extension을 구현하고, 각각의 FA 가 덧셈을 수행하면서 곱(product)를 계산한다.

**Ⅳ. 결과 비교**

1. 시스템 설명(System description)

본 연구에서, 회로를 검증하는데 사용한 시스 템은 Xilinx.Co의 ARTIX7모델의 FPGA칩을 사용 한다.

ARTIX7은 각각의 모듈로서, 101,440개의 논리 셀(Logic cell), 15,850개의 슬라이스(Slices)과 그 외 여러 모듈들로 이루어져 있다. 그 중 논리 셀과 슬라이스는 HDL로 기술되는 논리회로를 구현하는 중요한 모듈이다. 또한, Xilinx.co에서 개발되어 있는 CAD툴인 VIVADO를 사용하여 Verilog-HDL로 설 계한 회로 들을 FPGA에 프로그래밍함으로써 칩에 매핑되는 회로의 크기, 설계 전력을 비교 하였다.

2.1 시프트 레지스터

VIVADO를 사용하여, 직렬로 연결되어 있는 D-플립플롭의 개수를 늘림에 따라 FPGA칩에 매핑되는 회로의 크기를 비교한 데이터를 표 1에, 각각의 실제 칩에 매핑되는 회로의 모습을 그림 10, 11에 도시하였다.

표 1. 플립플롭 개수에 따른 회로 크기

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Delays** | **8** | **16** | **32** | **64** | **128** | **256** | **1000** | **2000** |
| **FF** | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **LUT** | 1 | 1 | 1 | 2 | 4 | 8 | 32 | 63 |
| **Power** | 0.36 | 0.36 | 0.36 | 0.36 | 0.36 | 0.37 | 0.43 | 0.5 |

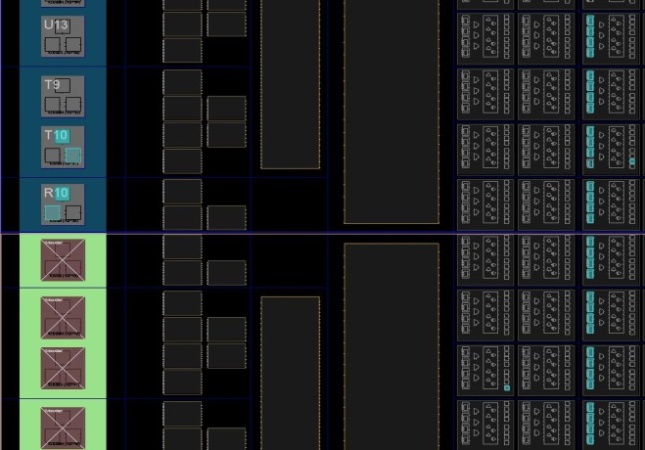


그림 10. Feature of circuit for 1000-delays

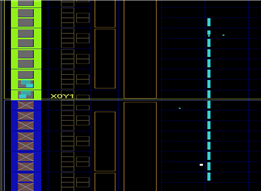


그림 11. Feature of circuit for 2000-delays

그림 2를 통하여 이미 알려진 바대로 32번 지연시키는데 LUT가 1개가 사용되고, 지연회로의 크기를 늘림에 따라, 선형으로 증가하다 1000번을 지연시키는 회로부턴 LUT가 크게 증가하는 것을 알 수 있다.

2.2 Distributed RAM

Verilog-HDL로 기술한 메모리 회로의 메모리 영역(memory area)의 크기를 늘림에 따른 회로의 크기를 표 2에 나타내었다.

표 2. 메모리 영역의 크기에 따른 회로 크기

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Delays** | **8** | **16** | **32** | **64** | **128** | **256** | **1000** | **2000** |
| **FF** | 3 | 4 | 5 | 6 | 7 | 8 | 10 | 11 |
| **LUT** | 7 | 8 | 9 | 10 | 11 | 18 | 69 | 109 |
| **Power** | 0.47 | 0.48 | 0.5 | 0.5 | 0.54 | 0.65 | 2.1 | 2.16 |

VIVADO툴을 이용하여 칩에 매핑되는 회로의 모습을 그림 12에 시뮬레이션 결과를 13에 나타 내었다.

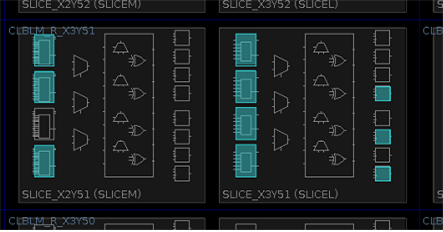


그림 12. Feature of circuit for 8-delays(D-RAM)

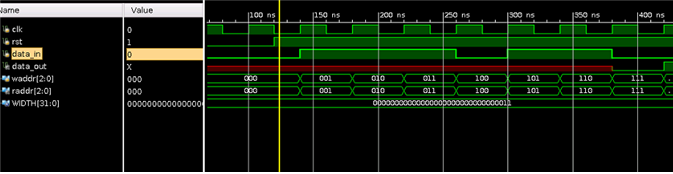


그림 13. Simulation: Timing diagram

2.2.1 시프트 레지스터 vs. Distributed RAM

표 3를 참고하면, 시프트 레지스터 방식에 의해 FPGA에 매핑된 회로는 RAM 방식보다 더 적은 플립플롭, LUT, 소비 전력을 가짐을 나타 낸다. 즉, 시프트 레지스터 방식이 Distributed RAM방식보다 효과적임을 알 수 있다.

표3. 시프트 레지스터와 Distributed RAM방식을 2000-delay에서 비교

|  |  |  |
| --- | --- | --- |
|  | **Shift register** | **Distributed RAM** |
| **FF** | 2 | 11 |
| **LUT** | 63 | 109 |
| **I/O** | 63 | 64 |
| **BUFG** | 1 | 1 |
| **POWER(W)** | 0.5 | 2.2 |

시프트 레지스터방식이 Distributed RAM방식 보다 적은 LUT를 사용하는 이유는 그림 2를 통하여 설명하였듯이 ARTIX7의 Memory Slice는 4개의 LUT를 묶어 432 bit의 시프트 레지스터로 사용될 수 있기 때문이다. 반면, Distributed RAM 방식 지연 회로에서는 Delay 인터벌(interval) 이 길어질수록 회로 크기의 증가율이 기하급수 적으 로 증가한다. 이는 메모리 영역뿐만 아니라 주소 영역의 카운터 회로와 내부적인 조합 논리회로의 크기가 커짐에 기인한다.

2.3 Block memory

블록 메모리를 사용한 지연회로를 VIVADO를 사용하여 FPGA칩에 매핑하였을 때의 회로 크기 를 표 4에, 회로의 모습을 그림 14, 15에 도시 하였다.

|  |  |  |
| --- | --- | --- |
|  | **1000 delays** | **2000 delays** |
| **FF** | 10 | 11 |
| **LUT** | 20 | 20 |
| **I/O** | 4 | 4 |
| **BRAM** | 0.5 | 0.5 |
| **BUFG** | 1 | 1 |
| **POWER(w)** | 1.06 | 1.15 |

표 4. The size of delay-circuits

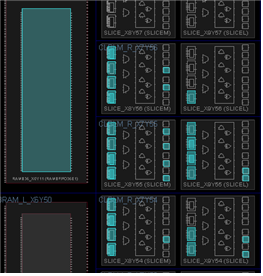


그림 14. Feature of circuit for 1000-delays

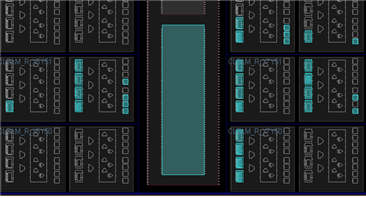


그림 15. Feature of circuit for 2000-delays

2.3.1 시프트 레지스터 vs. Block memory

블록 메모리 방식으로 고안된 지연회로의 경우 기본적으로 시프트 레지스터 방식의 회로에 비해 많은 수의 플립플롭을 사용한다. 그 이유는 지연 회로의 주소 영역에서의 카운터 회로가 추가되기 때문이다. 하지만, 데이터를 1000번 이상 지연시키는 회로의 경우를 보면, ARTIX7의 1개의 Block RAM가 저장할 수 있는 데이터는 18, 32Kbytes에 해당하기 때문에 Block RAM의 크기는 크게 증가하지 않고, 카운터회로의 크기만 증가 하는 경향을 보인다. 이에 따른 결과를 표 5에 나타내었다.

표 5. 시프트 레지스터 vs. Block Memory at 2000 delays

|  |  |  |
| --- | --- | --- |
|  | **Shift register** | **Block memory** |
| **FF** | 2 | 11 |
| **LUT** | 63 | 20 |
| **I/O** | 63 | 4 |
| **BRAM** | 0 | 0.5 |
| **BUFG** | 1 | 1 |
| **POWER(W)** | 0.502 | 1.15 |

2.4 Sub-conclusion: delay-circuits

256번 이하의 지연 구간을 갖는 회로의 경우에는 시프트 레지스터 방식으로 회로를 설계하게 되면, ARTIX7의 시프트 레지스터 매핑(mapping) 방식에 따라(그림 2) RAM방식보다 효율적인 회로로 FPGA 칩에 프로그래밍할 수 있다.

하지만, 1000번 이상의 지연 구간을 갖는 회로의 경우에는 두 가지 RAM방식 지연회로 들 중 Block RAM에 의한 회로 설계방식이 전체적인 회로의 크기에서는 유리할 수 있다. 또한, 소비 전력의 측면에서 두 방식을 비교해 볼 때, 시프트 레지스터 방식이 절대적으로 유리함을 알 수 있으며, 이는 Distributed RAM에서는 많은 레지 스터 수에 의한 전력소모, Block memory RAM 에서는 Block memory 소자에서 자체적으로 소모 하는 전력이 시프트 레지스터보다 상대적으로 많다는 사실에 기인한다.

3.1 Serial multiplier

Finite State Machine(FSM)방식의 컨트롤러에 의해 제어되는 기존의 SHIFT-ADD방식의 곱셈기 는 정확 한 계산을 위해, FSM의 레지스터를 통해 마지막 부분 곱을 전체 합에 의해 뺄셈을 수행하도록 설계되어 있다. 따라서, 이 회로는 2 개의 카운터와 1개의 모듈 컨트롤러, 각각의 부분 곱에 대한 버퍼로 구성된다. 위의 방식으로 설계 된 회로가 FPGA칩에 매핑된 크기를 표 6에 도시 하고, 매핑된 모습을 그림 16에 나타내었다.

표 6. The size of SHIFT-ADD multiplier

|  |  |
| --- | --- |
| **Site Type** |  |
| **Slice LUTs as Logic** | 22 |
| **Slice Register as Flip-flop** | 12 |

실험에 사용한 Bit-serial곱셈기는 구성 소자로 서, 2개의 MUX와 1개의 입력(input) D-플립플롭, 데이터 길이(data length)만큼의 AND게이트와 전가 산기, 2개의 플립플롭, 그리고 입력 데이터의 LSB와 MSB를 구별하기 위한 1개의 카운터 가 사용된다.



그림 16. Feature of SHIFT-ADD multiplier on FPGA

3.2 Bit-serial multiplier

하지만, 곱셈기 모듈 바깥 쪽에서의 적절한 인풋 데이터의 LSB, MSB구분에 대한 프로토콜 (protocol)을 지정함으로써, 카운터 회로에 대한 최적화가 가능 하다. FPGA칩에 매핑한 결과를 표 7에 나타내고, 매핑된 모습을 그림 17에 나타 내었다. 또한, 회로를 검증하기 위한 연산 을 수행하도록 하였을 때의 타이밍 선도를 그림 18에 도시하였다.

표 7. The size of bit-serial multiplier

|  |  |
| --- | --- |
| **Site Type** |  |
| **Slice LUTs as Logic** | 9 |
| **Slice Register as Flip-flop** | 7 |

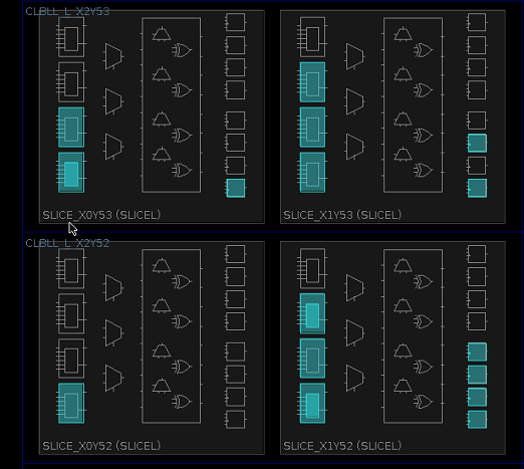


그림 17. Feature of Bit-serial multiplier on FPGA

3.3 Sub-conclusion: multipliers

SHIFT-ADD방식의 곱셈기와 비교했을 때, bit-serial 곱셈기의 회로 크기가 크게 감소함을 알 수 있다. 하지만, 병렬(parallel) 데이터 처리방식이 아닌 직렬(serial)데이터 처리 방식이기 때문에, 이 를 구동하기 위해 데이터 길이만큼의 빠른 클럭 (fast clock)신호가 필요하게 된다.

**Ⅴ. 결 론**

본 논문에서는 FPGA에 프로그래밍하는 뉴로 모픽 프로세서의 빌딩블록들(building blocks) 중 지연회로(delay circuits)와 곱셈기회로(multiplier)의 최적화 방법에 대해 제안하였다.

여러 가지 가능할 수 있는 지연 구간에 따라 전통적인 방법(시프트 레지스터)과 RAM의 구동 방식에 따른 설계 방법을 정해 회로를 설계할 수 있으며, 곱셈기의 경우 직렬 데이터 처리방식을 갖는 이미 알려져 있는 bit-serial 알고리즘을 따라 bit-serial곱셈기를 설계함으로써, 회로 크기를 줄여 모듈의 최적화함에 있어 적절한 설계방식으로 사용될 수 있다.

**참 고 문 헌**

[1] Dong-Sik Jang, Man-Hee Lee, Jong-Won Lee, Motion Analysis with Time Delay Neural Network,p3,1999

[2] Xilinx.co, 7-Series Architecture Overview

[3] Behrooz Parhami,Computer Arithmetic: Algorithms and Hardware Design

[4] Jaeyong Chung,Woochul Kang, Exploiting Sparse Activation for Low-power Design of Synchronous Neuro- morphic systems,p4,2015