2002년 한국정보처리학회 춘계학술발표논문집 제9권 제1호

고속 비동기식 FIFO 생성기 개발

임지숙*, 천익재*, 김보관*
*충남대학교 전자공학과
e-mail : jisuk96@pony.cnu.ac.kr

Development of a High Speed Asynchronous FIFO Compiler

Ji-Suk Lim*, Ik-Jae Chun*, and Bo-Gwan Kim*
*Dept. of Electronics Eng., Chungnam National University

요 약

본 논문에서는 single bank 와 multi bank FIFO 를 지원하는 CMOS FIFO memory compiler 를 개발 검증하였다. 이 컴파일러를 사용해서 설계자는 구현하고자 하는 어플리케이션에 적합한 high speed, high density, low power 를 갖는 on-chip memory 를 빠른 시간에 만들어 낼 수 있으므로 설계 시간을 절약할 수 있다. 이와 더불어 설계된 FIFO 의 시뮬레이션을 지원하기위한 Verilog 시뮬레이션 모델을 제공하였다. 현재 FIFO 를 구성하는 단위 셀들은 0.6um 3-metal 공정을 이용하여 설계 하였으며 공정의 변화에 따라 대상 공정에 맞도록 단지 몇 개의 단위 셀만을 재 설계하고 그에 대한 정보를 갱신해 줌으로써 공정의 변화에 대처 할 수 있도록 하였다. 설계된 컴파일러를 이용해 생성된 FIFO 는 표준 셀 라이브러리를 이용한 합성 가능한 FIFO 에 대하여 16bit x 16word FIFO 에서 면적면에서 93%, 속도면에서 70%의 향상을 보였다.

1. 서론

시스템의 규모가 커지고 기능이 다양화 되면서 대부분의 ASIC 디자인이 대용량의 데이터 처리를 위해서 칩 내부에 로직 블록과 메모리를 내장하게 되었다. 이러한 내장 메모리를 이용할 경우 저전력화와 동작속도의 증가를 동시에 만족 시킬 수 있다는 장점을 갖는다. 그러나 표준 셀 라이브러리를 이용한 합성 가능한 메모리의 사용은 작은 면적, 저전력, 고성능의 메모리를 얻을 수 없다. 이는 표준 셀 라이브러리에서 제공하는 셀들이 일반적으로 메모리를 합성하기에 부적합하기 때문이며 부분적으로 built-in Macro 블록 형태로 메모리가 제공되고 있으나 사용자의 요구를 정확하게 반영하지 못하고있다. 이에 따라서 메모리나 기타 큰 블록을 생성해 주는 컴파일러에 대한 연구가 진행되고 있다.

메모리의 종류에는 SRAM, DRAM, ROM, FIFO 등이 있으며 이 중 FIFO 는 각각 다른 데이터 전송률을 가진 서브 시스템에서 데이터를 저장하는 블록으로 다양한 블록에 많이 사용이 되는데 특히 무선 시스템에서 그 활용 빈도가 높다.

본 논문에서는 고집적, 고속의 FIFO 생성을 위한 CMOS FIFO Memory Compiler 를 설계하였다. 이 컴파일러는 링 어드레스 포인터를 갖는 dual port RAM 에 기반 한 FIFO 를 기본 구조로 하여 설계되었다. 현재 0.6um 3-metal 공정에 맞추어 단위 셀을 설계하였으며 단위 셀의 설계변경을 통하여 공정 변화에 대처할 수 있도록 하였다. 설계된 컴파일러는 합성 가능한 기존 표준 셀 라이브러리에 비하여 작은 면적과 속도면에서 이점을 갖고 single bank 와 multi bank FIFO 를 지원한다.

2. FIFO 컴파일러의 구조

컴파일러를 설계할 때 성능과 유연성이라는 두가지 사항을 고려하여야 한다. 성능은 속도와 집적도로 나타내어질 수 있으며 유연성은 확장성과 재구성 여부에 의해서 결정된다.

본 논문에서는 속도와 집적도에 중점을 두어 컴파일러를 설계하였다. CADENCE 환경에서 회로 설계와 레이아웃 작업이 이루어졌고 컴파일러는 *SKILL*™ Language 를 사용하여 구현하였다. FIFO 컴파일러는 사용자 입력으로 데이터 비트 길이와 데이터 워드 길

이를 받는다. 그리고 입력된 정보에 맞도록 기본 셀들을 데이터 베이스화 한 기본 셀 라이브러리에서 불러들여 합성한다. 이때 컴파일러는 컴파일된 FIFO 의 레이아웃과 회로도, 그리고 Verilog 시뮬레이션 파일을 자동 생성함으로써 응용 어플리케이션에 쉽게 적용하여 테스트를 수행할 수 있도록 하였다.

FIFO 컴파일러는 크게 메인 함수와 user interface generation, layout generation, 그리고 schematic generation 부분으로 구성된다. User interface generation 은 외부 사용자 입력 창을 생성하고 이 입력 창을 통해서 사용자 입력을 받아 들인 후 입력된 값에 따라 생성된 결과를 return 한다. Layout generation 부분은 원하는 크기에 맞도록 기본 셀들을 배치하고 배치된 셀들을 서로 연결해 준 다음 마지막으로 입 출력 핀을 생성한다. Schematic generation 부분은 layout generation 과정과 마찬가지로 각 셀들의 심볼을 배치 하고 서로 연결한 다음 연결된 wire 에 label 을 붙여 준다. 그리고 마지막으로 입/출력 핀을 정의 한다 [그림 1]. 그림 2 는 FIFO 컴파일러의 가장 중요한 부분이라 할 수 있는 layout generation 플로우를 보인다.

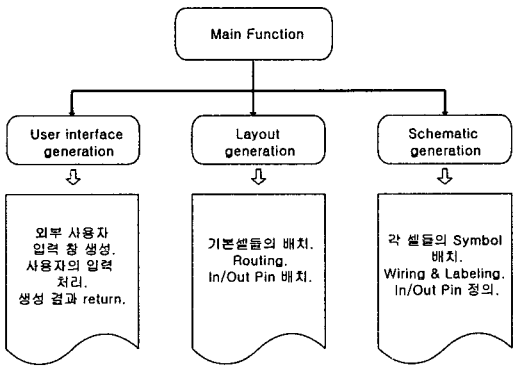

그림 1. FIFO 컴파일러의 구성

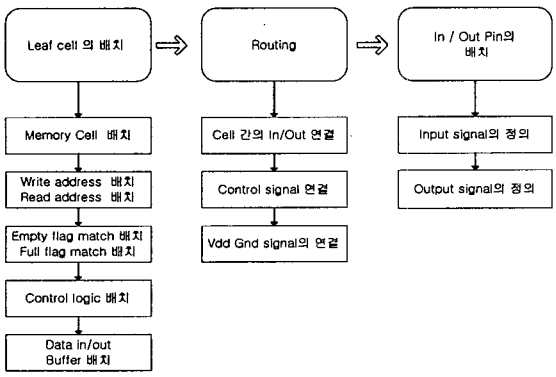

그림 2. 레이아웃 생성 흐름도

3. FIFO 의 구조

본 논문에서 설계한 FIFO 는 메모리 어레이 블록 (memory array block), read/ write address 블록, data input buffer, data output buffer, flag match 블록, 그리고 control 회로 블록으로 구성된다. 메모리 어레이 블록은 사용자의 입력에 따라 그 크기가 변하며 메모리 어레이 블록 좌우에 write address pointer 와 read address pointer 가 배치된다. 이때 어드레스 포인터(address pointer)는 사용자가 정의한 어드레스 워드의 길이에 따라 그 크기가 정해진다 [그림 3].

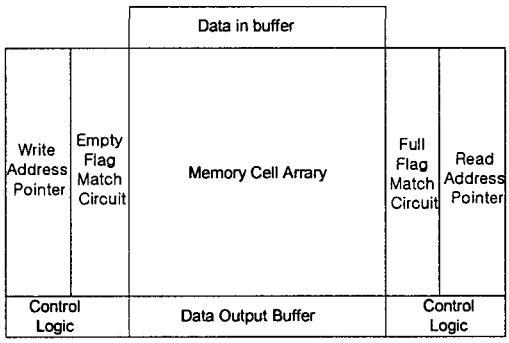

그림 3. FIFO 의 레이아웃 구조

FIFO 의 설계에 있어서는 기존의 메모리에서 사용하던 어드레스 디코더 대신에 쉬프트 링 카운터를 어드레스 포인터로 사용하고 read pointer shift register 와 write pointer shift register 의 각각의 비트가 해당하는 워드라인을 직접 구동하도록 설계하였다. 이와 같이 쉬프트 레지스터를 사용하게 되면 FIFO 의 동작속도가 빨라지고 전력소모를 줄일 수 있으며 FIFO 의 전체 구조가 단순화 된다. Empty 와 full flag 신호의 생성은 read 와 write word 라인을 직접 비교하는 방식을 사용하도록 설계하였다. 이것은 비교기의 카운터를 없앰으로써 FIFO 의 면적을 줄이고 구조를 더욱 단순화 시켰다 [그림 4].

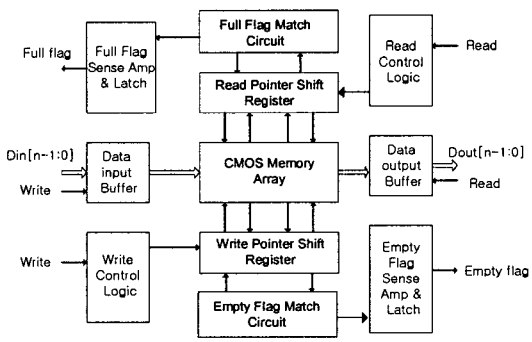

그림 4. FIFO 의 블록 다이어그램

본 논문에서 설계한 컴파일러는 멀티 뱅크 구조를 지원하도록 설계되었다. 따라서 각각의 메모리 뱅크가 데이터 입 출력 버퍼를 공유하는 형태가 아니라 하나

의 완전한 FIFO 블록이 반복되는 형태로 구현된다. 이러한 구조는 각각의 뱅크에 입 출력 버퍼가 존재하고 이들 버퍼가 서로 버스를 공유하는 형태로 되어 있으므로 뱅크의 증가에 따른 입출력 버퍼의 부하량이 커지지 않는 장점을 갖는다. 뱅크간 배선을 할 때에는 어드레스 버스가 정확히 연결되도록 하고 각 뱅크의 flag out 신호를 OR 해서 최종 출력 신호를 만들어 주는 회로를 추가로 설계하였다.

3.1 메모리 셀

메모리 셀은 8T 셀을 사용하였다. 기존의 6T 셀을 사용할 경우 메모리 셀에 데이터 1 을 적을 경우 값이 제대로 전달 되도록 하기 위해서는 별도의 회로가 필요하게 된다. 따라서 구조를 간단히 하고 값을 제대로 전달하기 위해서 N-mos 와 P-mos 의 쌍으로 구성된 패스(pass) 트랜지스터를 사용하였다 [그림 5].

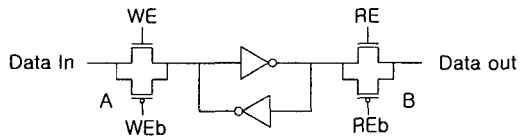

그림 5. 메모리 셀

3.2 어드레스 포인터(Address pointer)

어드레스 포인터는 어드레스 카운터를 사용하는 대신에 쉬프트 링 카운터를 사용하였다. 링 카운터를 사용하면 어드레스 디코딩 회로가 필요하지 않으므로 구조가 간단해지고 전력소모와 속도가 개선된다. 링 카운터는 연속된 m 개의 플립플롭으로 구성되며 각각의 비트가 하나의 워드 라인을 구동한다.

3.3 제어 회로

Control 회로는 FIFO 내부의 여러 가지 동작을 조정하기 위해 외부에서 인가되는 신호를 받아서 FIFO 내부의 컨트롤 신호들을 발생 시키는 회로이다. Read 신호와 write 신호를 받아들여 flag 신호 생성을 위한 제어 신호를 생성한다. 그리고 생성된 flag out 신호를 랫치(latch)하여 출력한다.

3.4 Flag Signal Generation

Flag 신호를 생성할 때 Write Address Pointer 와 Read Address Pointer 를 직접 비교하는 방법(Direct Word Line Comparison Technique)을 사용했다. 먼저 empty flag 생성 과정을 보면 address pointer 는 리셋이 되면 base position 에 있게 되고 다음에 read 나 write 신호가 들어오면 다음에 읽어가거나 쓸 번지를 가리키게 된다. 따라서 read operation 시에 현재 읽어가는 read address pointer 보다 하나 앞선 pointer 와 데이터를 마지막으로 쓴 번지를 가리키는 write address pointer 를 비교해서 그 값이 같을 경우 empty flag 가 발생하도록 하였다. Full flag generation 은 현재 쓰고있는 write address pointer 보다 하나 앞선 pointer 와 마지막에 데이터를 읽어간 위치를 가리키는 read address pointer 를 비교해서 full flag 가 발생하도록 하였다 [그림 6]. 또한 empty flag 와 full flag 를 만들기 위해 current sensing 방법을 사용하였다. Current sensing 방법은 조합논리 회로를 사용할 경우에 비하여 고속, 저전력, 그리고 작은 면적의 회로를 설계할 수 있다는 장점을 갖는다 [그림 7].

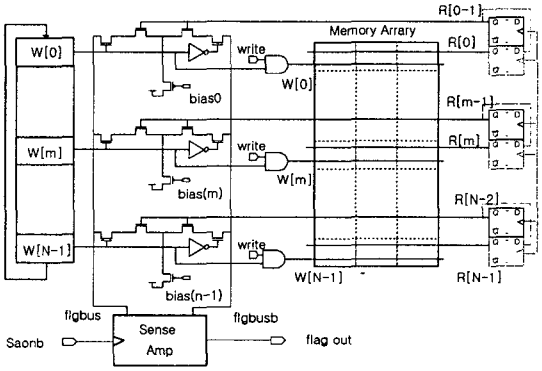

그림 6. Empty flag 생성 블록도

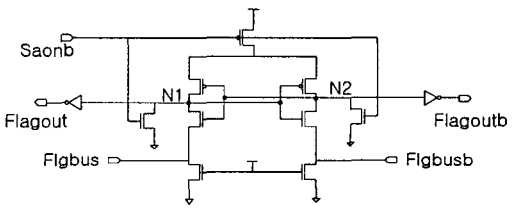

그림 7. Flag 신호 생성을 위한 current sense amplifier

4. Data In/Out Buffer

이 버퍼는 메모리 셀에 저장될 데이터를 전달해 주는 역할을 한다. 현재 설계된 입력 버퍼는 메모리 셀의 입력 커패시턴스 값을 계산하여 지연시간이 2nsec 를 만족하도록 설계되었고 이 입력 버퍼를 사용하면 8kbyte 까지 메모리 합성이 가능하다. 입력 버퍼의 크기를 구동 능력별로 설계하면 더 큰 사이즈의 메모리도 합성이 가능하다. 출력 버퍼는 read 동작일 때 turn on 되어 데이터 버스로 데이터를 출력하고 write 동작일 때는 데이터 출력 버퍼를 tri-state 로 만들어준다.

5. 실험 결과

다음의 그림 8 은 본 논문에서 설계한 FIFO 컴파일러에 의해 생성된 비동기식 FIFO 의 레이아웃을 보이고 있다. 이는 앞에서 보인 그림 1 의 FIFO 의 레이아웃 구조를 따라서 배치 되었음을 보인다. 멀티 뱅크 FIFO 의 경우 같은 크기의 싱글 뱅크 구조에 비하여 입출력 버퍼의 부하를 줄일 수 있으며 보다 빠른 FIFO 를 설계할 수 있다는 장점을 갖는다.

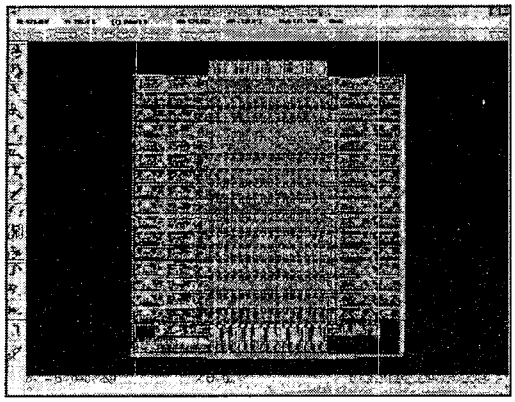

그림 8. Single bank FIFO 레이아웃(16x16)

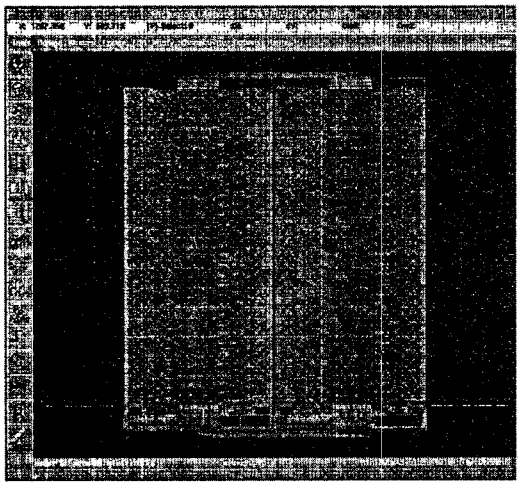

그림 8. Two bank FIFO 레이아웃(8x32)

이렇게 합성된 레이아웃은 합성 가능한 표준 셀 라이브러리를 이용했을 때 보다 작은 면적과 고속의 특성을 갖는다. 표 1 에서 보이는 것처럼 8 x 8 FIFO 의 경우 설계된 컴파일러를 사용했을 때 기존의 표준 셀 라이브러리로 합성한 경우보다 면적이 약 9 배 정도 작음을 알 수 있고 16 x 16 인 경우 약 13 배 정도 작은 크기의 회로를 설계할 수 있었다. 또한 그 동작 속도에 있어서 합성된 FIFO 에 비하여 access time 에 있어서 약 3 배의 성능 향상을 보임을 확인하였다 [표 2]. 이는 FIFO 의 크기가 커질수록 면적 면에서 더 큰 장점을 가짐을 보여준다.

표 1 생성된 FIFO 의 면적 비교

Size	Synthesized FIFO	Compiled FIFO
8x8	1843.4 x 10^3	205.75 x 10^3
16x16	5944.7 x 10^3	445.2 x 10^3

표 2 생성된 FIFO 의 동작 속도 비교

Access time	Synthesized FIFO	Compiled FIFO
High read	6.7nsec	2nsec
Low read	7.05nsec	2.02nsec

6. 결론

본 논문에서는 고속, 고집적의 비동기식 FIFO 컴파일러를 개발하고 관련 셀들을 설계하였다. 설계된 FIFO 는 read/write 어드레스 포인터로 링 카운터를 사용함으로써 면적을 줄였다. 2048 word x 32 bit 까지 합성이 가능하며 평균 data access time 이 2nsec 로 빠른 동작 특성을 갖는다. 또한 delay 정보를 포함한 Verilog simulation model 을 제공하였다. 이 컴파일러와 timing model 을 사용하여 사용자는 빠르고 쉽게 원하는 FIFO 를 응용프로그램에 적용시킬 수 있을 것이다.

Acknowledgement

본 연구는 반도체 설계 교육 센터(IDEC) 그리고 BK21 충남대학교 정보통신 인력 양성사업단의 지원에 의하여 수행되었습니다.

참고문헌

[1] Larry R.Fenstermaker, Kevin J.O'conor "A low-power generator-based FIFO using ring pointers and current-mode sensing," ISSCC, 1993.

[2] Masaharu Kawauchi "A 1.0um compilable FIFO buffer for standard cell," IEEE custom integrated circuits conference, 1989

[3] Charles E. Molnar, Ian W,Jones, William S.Coates, and Jon K.Lexau "A FIFO ring performance experiment," International Symposium on Advanced Research in Asynchronous Circuits and Systems, 1977.

[4] Giao N. Pham, P.E Kenneth C. Schmitt, P.E "A high throughput, asynchronous, dual port FIFO memory implemented in ASIC technology," IEEE ASIC Seminar and Exhibit, 1989.

[5] Yervant Zorian, Ad J. Van de Goor, Ivo Schanstra "An effective BIST scheme for ring address type FIFOs," IEEE International Test Conference, 1994.