# SoC 설계를 위한 유효 비트 방식의 비동기 FIFO 설계

이용환*

## Design of an Asynchronous FIFO for SoC Designs Using a Valid Bit Scheme

Yong-hwan Lee*


### 요 약

SoC 설계에서는 많은 수의 IP 들이 하나의 칩에 집적되며 이들은 각각 서로 다른 주파수로 동작해야 가장 효율적으로 동작할 수 있다. 이러한 IP들을 연결하기 위해서는 비동기 클럭 동작 사이에 버퍼 역할을 할 수 있는 비동기 FIFO가 필수적이다. 그러나 아직 많은 수의 비동기 FIFO가 잘못 설계되고 있으며 이에 따른 비용이 심각하다. 이에 본 논문에서는 유효 비트 방식의 비동기 FIFO를 설계함으로써 비동기 회로에서 발생하는 metastability를 없애고 비동기 카운터의 오류를 수정함으로써 비동기 클럭들 사이에서 안전하게 데이터를 전송할 수 있는 FIFO 구조를 제안한다. 또한 이 FIFO 구조의 HDL 기술을 바탕으로 합성하여 다른 방식의 FIFO 설계와 비교 평가한다.

### ABSTRACT

SoC design integrates many IPs that operate at different frequencies and the use of the different clock for each IP makes the design the most effective one. An asynchronous FIFO is required as a kind of a buffer to connect IPs that are asynchronous. However, in many cases, asynchronous FIFO is designed improperly and the cost of the wrong design is high. In this paper, an asynchronous FIFO is designed to transfer data across asynchronous clock domains by using a valid bit scheme that eliminates the problem of the metastability and synchronization altogether. This FIFO architecture is described in HDL and synthesized to the gate level to compare with other FIFO scheme. The subject matter of this paper is under patent pending.



키워드

비동기 FIFO, SoC, 유효 비트, HDL, clock scheme


## Ⅰ. 서 론

반도체 공정의 발달에 따라 최근의 SoC (System on a Chip)는 매우 많은 수의 IP (Intellectual Property)를 집적하여 제작하며 각각의 IP들은 공통 버스 (common bus)에 연결되어 동작하게 된다. 버스는 각 IP의 데이터 요청을 중재하여 일정한 순서에 따라 버스의 사용권한을 부여하기 때문에 버스가 현재 사용 중이라면 각 IP는 필요한 시점에 즉시 데이터를 확보할 수가 없다. 따라서 필요한 데이터를 미리 확보하기 위해서 FIFO (First-In First-Out) 방식의 버퍼가 필요하며 이러한 FIFO가 완충작용을 함으로써 IP가 당장 버스를 액세스할 수 없다 하더라도 전에 FIFO에 저장하여 놓은 데이터를 일단 사용하고 나중에 버스의 사용권이 주어

* 금오공과대학교 전자공학부

접수일자 : 2005. 7. 5

질 때 FIFO에 데이터를 보충하는 방식을 사용함으로써 데이터가 없어 동작하지 못하는 것을 방지할 수 있다.

보통 IP들은 서로 다른 주파수에서 동작한다. 그림 1.에서와 같은 예에서 LCD controller는 메모리에 존재하는 frame buffer로부터 LCD에 표시할 화면 데이터를 가져오는데 LCD 화면에 표시하는 주파수와 메모리가 연결되어 동작하는 버스의 주파수는 서로 다르다. 이러한 경우 FIFO에 쓰는 주파수와 읽는 주파수가 서로 다르기 때문에 이러한 경우에도 올바른 동작을 할 수 있는 비동기 FIFO가 필수적이다. 그러나 현재까지 제작된 많은 수의 비동기 FIFO들이 metastability와 카운터 동기화 문제를 제대로 고려하지 않고 설계되었기 때문에 많은 문제가 발생되고 있다.

이러한 두 가지 문제를 고려하지 않고 설계된 비동기 FIFO들은 대부분 99.9%이상의 시간동안 동작이 제대로 되기 때문에 문제가 없는 것으로 생각하기 쉬우며 시뮬레이션 또는 제품이 나온 후에도 오류의 발견이나 원인분석이 매우 어렵다[1]. 본 논문에서는 올바른 방식의 비동기 FIFO를 설계하고 다른 방식의 비동기 FIFO와 성능 비교를 하게 된다.

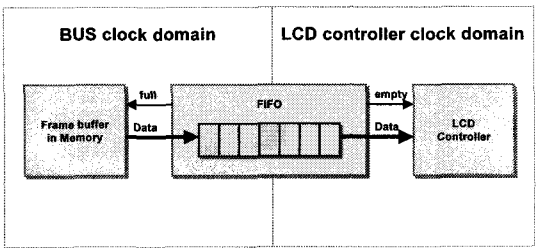



그림 1. LCD controller에서 FIFO 사용의 예
Fig. 1. FIFO usage example in an LCD controller

## Ⅱ. 동기 FIFO 구조

FIFO의 개념은 그림 1.과 같지만 그 구현은 카운터를 사용하여 순환적 개념으로 구현하는 것이 일반적이다. 그림 2.는 4개의 엔트리를 갖는 FIFO의 개념도이다. 여기서 4개의 엔트리가 있으므로 카운터는 2비트만 사용하면 되지만 FIFO 엔트리가 모두 유효하다는 full 신호와 FIFO 엔트리가 모두 비어 있다는 empty 신호를 발생시키기 위해서는 3비트가 필요하다. 카운터는 3비트의 쓰기 카운터 (wptr)와 3비트의 읽기 카운터 (rptr)의 2 가지가 있으며 이들 카운터는 처음에 000의 값을 유지하고 있다가 쓰기 동작이 이루어지면 wptr이 읽기 동작 시에는 rptr이 하나씩 증가하게 되며 그 값이 111이 되면 그 다음 값은 다시 000이 된다.

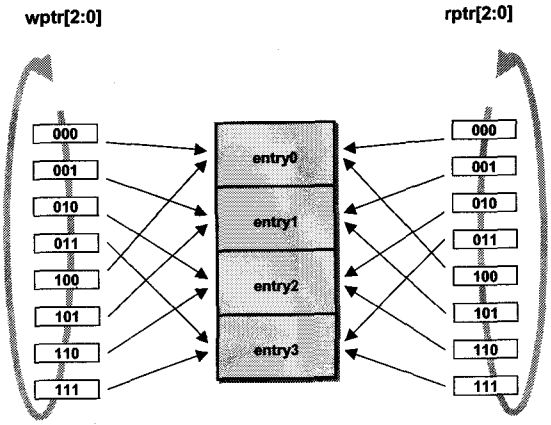
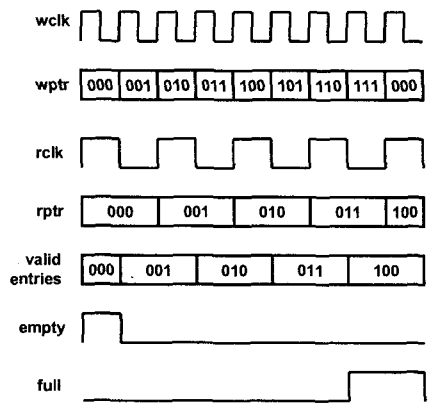



그림 2. FIFO 동작의 개념도
Fig. 2 Conception of the operation of a FIFO

이러한 예의 FIFO 동작은 그림 3.과 같다. 여기서 쓰기용 클럭인 wclk의 상승 에지마다 FIFO에는 값이 새로이 쓰이며 처음 에지에서는 wptr이 000이므로 entry0에 데이터가 쓰여 지고 wptr은 하나 증가하여 001이 된다. 다음 에지에서는 entry1에 값이 쓰여 지고 wptr은 010이 되며 이러한 방식으로 wptr과 FIFO의 엔트리에 값이 채워진다. 읽기 시에는 rclk의 상승 에지에 맞추어 rptr이 가리키는 엔트리의 데이터를 내보내고 이때 rptr을 증가시킨다.

그러나 읽기 동작과 쓰기 동작의 속도가 같지 않기 때문에 유효한 데이터가 없지만 읽어가려 할 경우와 FIFO에 유효한 데이터로 가득 찼는데도 쓰려고 할 경우가 발생할 수 있다. 따라서 이때에는 읽는 쪽과 쓰는 쪽에 더 이상 읽거나 쓰지 못하도록 신호를 발생시켜 알려주어야 한다. 더 이상 읽을 유효한 데이터가 없다는 것은 empty라는 신호로 알려주어 읽는 쪽에서 더 이상 데이터를 읽어가지 못하도록 해주며, 유효한 데이터를 더 이상 쓸 공간이 없을 때는 full 신호로 쓰는 쪽에 알려주어 더 이상 데이터를 쓰지 못하도록 한다. Empty 신호는 두 카운터의 값이 모두 같을 때 발생하며 full 신호는 최상위 비트의 값이 틀리고 나머지 비트들의 값이 같을 때 발생시킨다.




```
full = (wptr[3] != rptr[3]) & (wptr[2] == rptr[2]) &
       (wptr[1] == rptr[1]) & (wptr[0] == rptr[0]);

empty = (wptr[3] == rptr[3]) & (wptr[2] == rptr[2]) &
        (wptr[1] == rptr[1]) & (wptr[0] == rptr[0]);
```

그림 3. FIFO 동작 파형의 예
Fig. 3 Waveform of FIFO operation example

그림 4.는 동기 FIFO의 구조를 나타낸 것이다. 이 구조에서는 FIFO 엔트리들을 저장하기 위해 이중 포트의 메모리를 사용하였고 wptr과 rptr을 증가시키기 위해 adder를 사용하였다. 그러나 이러한 동기 FIFO는 쓰는 클럭과 읽는 클럭의 주파수가 같거나 배수의 관계가 되는 동기 클럭 사이에서만 사용할 수 있으며 두 클럭의 주파수가 서로 상관관계가 없는 비동기 클럭 사이에서는 쓰일 수 없다.

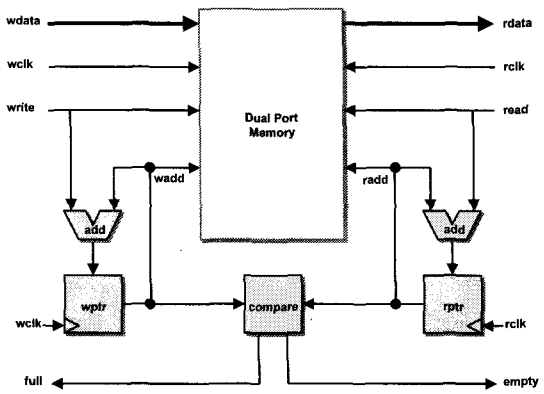



그림 4. 동기 FIFO의 구조
Fig. 4 Architecture of synchronous FIFO

## Ⅲ. 비동기 FIFO 구성 시 metastability 문제

동기 FIFO를 비동기 클럭 도메인에서 사용할 때 99.9% 이상의 시간동안은 문제가 없으나 두 클럭의 에지가 서로 비슷한 시기에 상승할 때에는 심각한 문제가 발생한다. 이러한 원인은 metastability의 발생과 카운터의 동기화 문제가 때문이다.

그림 5.(a)는 metastability 현상을 확인하기 위하여 D-type FF(Flip/flop)의 입력과 클럭 단자를 서로 연결한 후 입력단에 전압을 가하기 위한 회로이다. 이 회로에 전압을 가한 후 출력단을 오실로스코프로 측정해 본 결과가 그림 (b)에 나와 있다. 여기서 알 수 있듯이 DFF의 출력은 일정한 시간동안 '0'이나 '1' 값이 아닌 그 중간에 머물러 있다. 따라서 클럭과 데이터가 거의 같은 시점에 DFF에 도달할 수도 있는 비동기 회로에서는 비록 확률은 매우 작지만 이러한 metastability 현상이 나타날 수 있다. 이러한 중간의 값

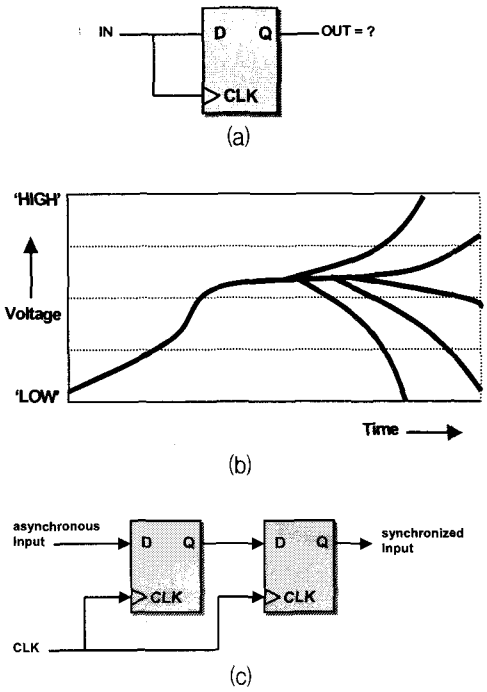



그림 5. Metastability의 (a) 실험과 (b) 그 결과 파형 및 (c) 해결책
Fig. 5 (a) Experiment, (b) waveform result and (c) solution of metastability

을 갖는 출력값이 전파된다면 '0'과 '1'만을 사용하는 디지털 로직회로에서는 큰 문제가 된다. 이러한 metastability 문제를 해결하기 위해서는 그림 5.(c) 에서와 같이 DFF을 직렬로 연결한 동기화 FF의 출력을 사용하여야 한다. 중간의 값도 일정 시간이 흐르면 '0' 이나 '1' 값으로 바뀌게 되므로 한 클럭 주기 만큼 시간이 흐른 후 그 값을 사용할 수 있도록 하기 위함이다.

따라서 비동기 클럭을 사용하는 비동기 FIFO에서는 이러한 동기화 F/F을 사용해야 하며 그 그림은 그림 6.과 같다. 여기서 wptr은 wclk 도메인에 의해 증가하며 wptr을 rclk 도메인에서 사용하기 위해서는 wptr을 rclk으로 두 번 래치한 후 그 값을 rptr과 비교하여 empty 신호 발생을 위해 사용한다. rptr은 wclk로 두 번 래치한 후 wptr과 비교하여 full 신호를 발생시킨다.

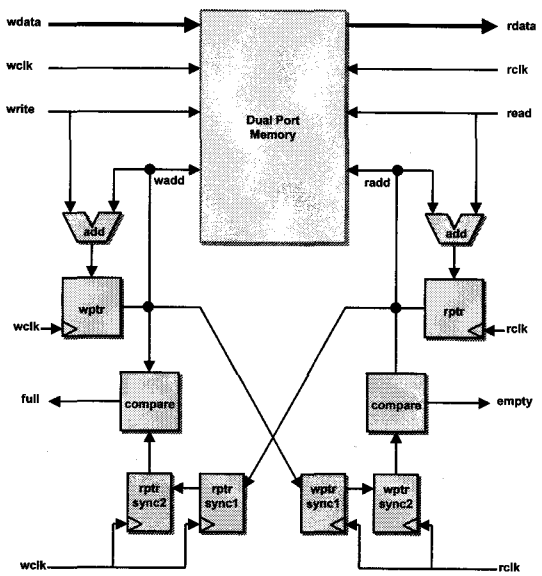



그림 6. 동기화 FF을 적용한 FIFO
Fig. 6 FIFO with synchronizing FF

## Ⅳ. 비동기 FIFO 구성 시 카운터 동기화 문제

FIFO에 동기화 FF을 적용하였다 하더라도 비동기 클럭에서 사용할 수 없는 다른 문제가 있는데 그것은 카운터의 동기화 문제이다. 그림 7.(a)와 같은 회로에서 wclk와 rclk이 비슷한 시간에 상승 에지가 발생할 때 문제가 된다. 그림 7.(b)에서는 이러한 문제를 파형으로 보여주고 있다. 동기화 FF wptr sync1이 wptr의 값을 받으려할 때 wptr 값의 변화가 rclk의 에지에서 일어나므로 wptr sync1은 000 또는 111의 값을 받게 되고 이때에는 문제가 발생하지 않는다. 그러나 만약 3비트의 FF으로 구성되어 있는 wptr sync1에서 아주 조금이라도 각 비트에 도달하는 rclk이 차이가 난다면 000 또는 111이외의 값을 갖게 될 것이고 이때에는 치명적인 오류가 된다. 왜냐하면 순차적으로 111 → 000 → 001 과 같이 값이 변하지 않고 갑자기 다른 값으로 바뀌게 되기 때문이다.

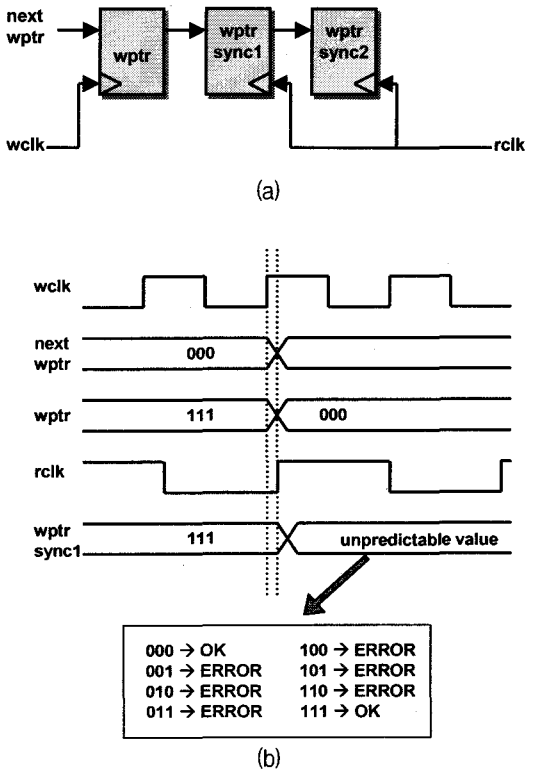



그림 7. 카운터 동기화 문제의 (a) 회로 (b) 파형 예
Fig. 7 (a) Logic circuit and (b) waveform example of the error for the synchronizing counter

따라서 이를 방지하기 위한 방법이 본 논문에서 제안하는 valid 비트를 사용하는 방식이며 이는 그림 8.(a)에 나타나 있다. 이 방식은 FIFO의 엔트리 수만큼 유효 비트 (wvalid)를 만들어 이를 사용한다. 카운터

자체를 동기화 FF에 연결하지 않고 카운터가 가리키는 FIFO 엔트리에 해당되는 유효비트를 쓰기 또는 읽기가 실행될 때마다 토글하는 방식을 사용한다. 이 방식을 사용하면 쓰기 또는 읽기를 할 때마다 오직 하나의 비트만이 바뀌게 되며, 바뀌는 비트의 값을 동기화 FF (wvalid sync1)에 저장할 때 바뀌기 전의 값을 저장하거나 또는 바뀐 값을 저장하거나에 상관없이 정상 동작이 이루어진다. 그림 8.(b)에서와 같이 wvalid가 1101에서 1111로 바뀔 때 바뀌는 값이 오직 하나의 비트이기 때문에 wvalid sync1의 각 비트에 도달하는 rclk가 다소 차이가 있다고 해도 wvalid sync1은 1101 또는 1111 둘 중의 하나의 값만을 받게 되며 이때 어떤한 값을 받아도 문제가 되지 않는다.

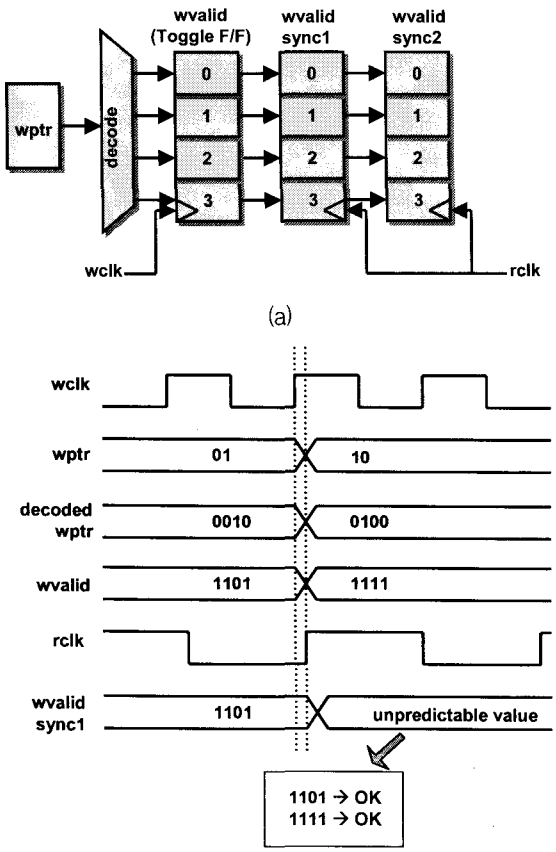



그림 8. 유효 비트 방식의 (a) 회로 (b) 파형 예
Fig. 8 (a) Logic circuit and (b) waveform example of the valid bit scheme

## V. 결과 고찰

동기식 FIFO의 두 가지 문제점을 해결하여 비동기 클럭들 간에 사용할 수 있는 본 논문의 비동기 FIFO 구조는 그림 9.와 같으며 full 및 empty 신호를 만드는 블럭이 그림 10.에 나타나 있다.

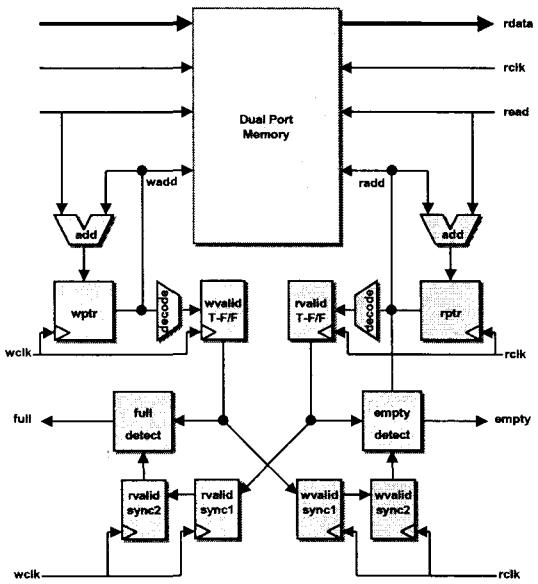



그림 9. 유효 비트 방식 비동기 FIFO 구조
Fig. 9 Proposed asynchronous FIFO

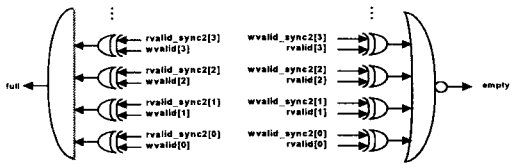



그림 10. Full 및 empty 신호를 위한 로직
Fig. 10 Generation of full and empty signals

유효 비트 방식은 기존의 구조에 비해 다소 많은 수의 F/F이 사용된다. 이는 기존의 읽기 및 쓰기 카운터가 각 1비트씩 적지만 FIFO 엔트리 수만큼의 FF이 사용되기 때문이다. 그러나 이 방식은 이해가 쉽고 속도가 매우 빠르다는 장점이 있다.

본 논문의 방식 이외에 Grey code를 사용하여 카운터를[2][3] 구성함으로써 FIFO의 쓰기 또는 읽기 시 카운터가 한 비트씩만이 변하도록 하는 방식이 있다. 이

방식의 장점은 F/F의 수나 이에 사용되는 combinational 게이트 수가 동기 FIFO에 비해 그리 크지 않다는 것이다. 그러나 이 Grey code 카운터 방식의 최대 약점은 속도가 매우 느리다는 것이다. Full 또는 empty 신호를 만들기 위해서는 Grey code에서 일반 카운터의 값으로의 변환이 필요하며 이를 위해서는 가산기를 거쳐야 하기 때문이다. 이에 비해 본 논문의 유효 비트 비동기 FIFO는 이러한 연산할 필요가 없고 단지 두 단계의 게이트만 거치면 full 또는 empty 신호를 만들 수 있기 때문에 매우 빠르다. 또한 가산기 등의 복잡한 회로를 거치지 않고 신호를 만들 수 있기 때문에 전력 면에서도 매우 유리하다.

FIFO 메모리를 제외한 Grey 카운터 FIFO와 유효비트 FIFO의 비교는 표 1.에 나타나 있다. 이 표에서 괄호안의 값은 유효비트 FIFO를 1로 가정했을 때의 비교 값이다. 이 표에서 유효비트 FIFO는 엔트리 개수가 늘어나면서 소요되는 FF과 게이트 수가 늘어나는 것을 볼 수 있다. 그러나 FIFO 엔트리 당 비트 수가 32 비트 이상이라면 FIFO 메모리의 크기가 매우 크기 때문에 FIFO 전체의 크기는 유효비트 FIFO가 Grey code FIFO에 비해 그리 크지 않다.

표 1. Grey 카운터 FIFO와의 비교
Table. 1 Comparison with grey counter FIFO

| | Grey counter FIFO | | | Valid bit FIFO | | |
|---|---|---|---|---|---|---|
| FIFO entries | 4 | 8 | 12 | 4 | 8 | 12 |
| Number of F/F | 23 (0.8) | 28 (0.6) | 34 (0.3) | 28 (1) | 54 (1) | 110 (1) |
| Number of combinational gates | 176 (1.4) | 273 (1.1) | 394 (1.0) | 130 (1) | 245 (1) | 405 (1) |
| Delay of critical path (ns) | 3.6 (1.2) | 5.6 (1.6) | 7.4 (1.8) | 3.0 (1) | 3.6 (1) | 4.1 (1) |

유효비트 FIFO가 Grey code FIFO에 비해서 유리한 점은 지연 시간에서 알 수 있다. FIFO entry 수가 4 일 때는 약 20% 빠르며 FIFO entry 수가 12 일 때는 80% 빠르다. 따라서 FIFO 엔트리 수와 비트 수가 커질수록 칩 면적은 전체 면적에 비해 그리 크지 않은 반면 속도는 매우 빠르다는 장점을 지닌다. 최근의 SoC 추세는 고성능 및 저전력화를 목표로 하고 있으며 반도체 공정의 발달에 따라 매우 많은 수의 트랜지스터를 집적할 수 있으므로 약간의 칩 면적 증가는 그리 중요한 요소가 아니다. 따라서 본 논문의 유효비트 FIFO 구조는 빠른 속도와 연산량의 감소에 의한 저전력화의 이점이 크므로 많은 SoC 설계에 적용될 수 있다.

## VI. 결 론

본 논문에서는 유효 비트 방식의 비동기 FIFO를 설계함으로써 metastability와 카운터의 동기화를 해결하는 구조를 제안하였고 Grey 카운터 방식의 비동기 FIFO와 비교 결과 20%~80%의 빠른 속도를 낼 수 있음을 알았다. 최근의 SoC는 반도체 공정의 발달로 인하여 적은 수의 게이트의 추가 보다는 동작 속도나 저전력 소모가 주된 관점이다. 따라서 본 논문의 비동기 FIFO는 고속 또는 저전력 IP 제작시 유용하게 사용될 수 있을 것이다.

## 참고문헌

[1] Edward Paluch, "Synthesis Optimized Universal Synchronous/Asynchronous Generic FIFO Design", SNUG San Jose 2003 paper

[2] Clifford Cummings, "Simulation and Synthesis Techniques for Asynchronous FIFO Design", SNUG San Jose paper 2002

[3] Clifford Cummings, Peter Alfke, "Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparisons", SNUG San Jose paper, 2002

### 저자소개

**이용환(Yong-hwan Lee)**

1993. 2. 연세대학교 전자공학과 (공학사)
1999. 2. 연세대학교 전자공학과 (Ph.D.)
1999~2002 하이닉스반도체
2003~2004 삼성전자
2004~ 금오공과대학교 전자공학부
※관심분야 : SoC, 임베디드 시스템 및 소프트웨어, 마이크로프로세서 구조