레지스터 기반 비동기 FIFO 구조 설계 기법

이용환*

*금오공과대학교

Design Technique of Register-based Asynchronous FIFO

Yong-hwan Lee*

*Kumoh National Institute of Technology

E-mail : yhlee@kumoh.ac.kr

요 약

현재 SoC 설계에 사용되는 많은 IP들은 대부분 이들이 연결되는 버스 클럭과 주파수가 서로 다른 클럭을 사용하며 이를 위해서는 비동기 FIFO가 필수적이다. 그러나 아직 많은 수의 비동기 FIFO가 잘못 설계되고 있으며 이에 따른 비용이 심각하다. 이에 본 논문에서는 레지스터 기반의 비동기 FIFO를 유효비트를 사용하여 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 설계 방식과 비교 평가한다.

ABSTRACT

In today's SoC design, most of IPs which use the different clock frequency from that of the bus require asynchronous FIFOs. However, in many cases, asynchronous FIFO is designed improperly and the cost of the wrong design is high. In this paper, a register-based asynchronous FIFO is designed to transfer data in asynchronous clock domains by using a valid bits 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.

키워드

FIFO, asynchronous, SoC, HDL, IP

Ⅰ. 서 론

최근의 SoC는 매우 많은 수의 IP를 집적하여 제작하며 각각의 IP들은 공통 버스에 연결되어 동작하게 된다. 버스는 각 IP의 데이터 요청을 중재하여 순서에 따라 버스의 사용 권한을 부여하기 때문에 각 IP는 필요한 시점에 바로 데이터를 확보할 수가 없다. 따라서 필요한 데이터를 미리 확보하기 위해서 FIFO가 필요하며 이러한 FIFO가 완충작용을 하여 IP가 당장 버스를 액세스할 수 없다하더라도 일단 FIFO에 저장되어 있는 데이터를 사용하고 나중에 버스의 사용권이 주어질 때 FIFO에 데이터를 보충하는 방식을 사용할 수 있다.

보통 IP는 서로 다른 주파수에서 동작한다. 그림 1.에서와 같이 LCD controller는 메모리로부터 LCD에 표시할 데이터를 가져오는데 LCD 화면에 표시하는 주파수와 메모리가 동작하는 버스의 주파수는 서로 다르다. 이러한 경우 FIFO에 쓰는 주파수와 읽는 주파수가 서로 다를 경우에도 사용가능한 비동기 FIFO가 필수적이다. 그러나 현재까지 제작된 많은 수의 비동기 FIFO들이 metastability와 카운터 동기화 문제를 제대로 고려하지 않고 설계되었기 때문에 많은 문제가 발

생되고 있다. 이러한 비동기 회로의 고려사항 없이 설계된 비동기 FIFO들은 대부분 99%이상의 시간동안 동작이 제대로 되고 시뮬레이션 또는 제품이 나온 후에도 오류의 발견이나 원인분석이 어렵기 때문에 더큰 문제가 된다[1]. 본 논문에서는 올바른 방식의 비동기 FIFO를 설계하고 다른 비동기 FIFO와 성능 비교를 하게 된다.

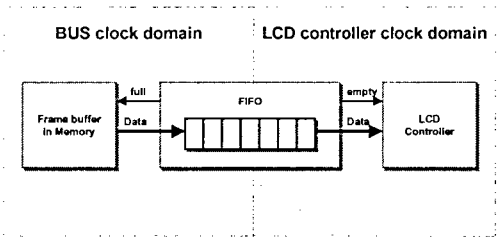

그림 1. FIFO 사용의 예

II. 본 론

대부분의 FIFO는 카운터를 사용하여 구현하며 그 예는 그림 2.와 같다. 4개의 엔트리로 만들어진 FIFO는 3비트의 쓰기 및 읽기 카운터를 사용하며 카운터가 지시하는 엔트리에 값을 쓰거나 또는 그 값을 읽게 된다. 4개의 엔트리는 2비트면 표시가 가능하지만 full과 empty 신호를 만들어 주기 위해서는 그림 3.에서와 같이 3비트가 필요하다. FIFO에 쓰는 입장에서 보면 FIFO가 완전히 차지 않으면 계속 데이터를 FIFO에 저장하면 되므로 full 신호가 필요하고, 읽는 입장에서는 FIFO가 비어있지 않다면 읽어도 되기 때문에 empty 신호가 필요하다. 이러한 원리를 사용하여 만들어지는 동기 FIFO의 구조는 그림 4.와 같다.

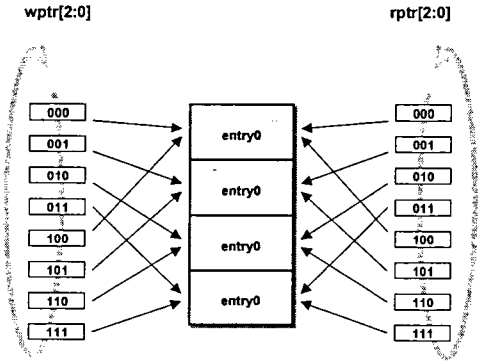

그림 2. FIFO의 동작원리

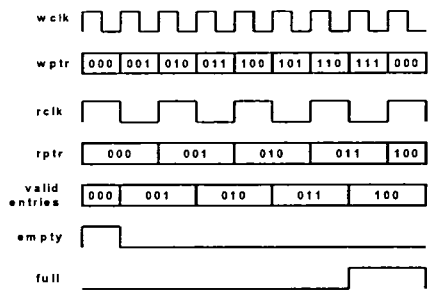

그림 3. FIFO의 동작 파형

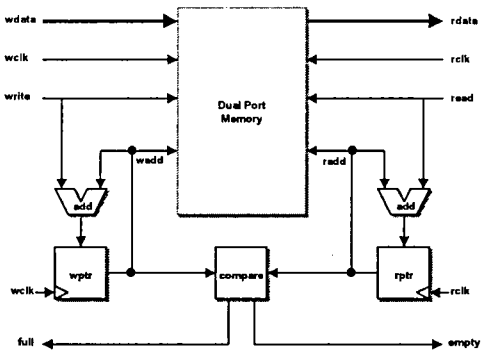

그림 4. 동기 FIFO의 구조

이러한 동기 FIFO를 쓰는 쪽과 읽는 쪽의 주파수가 서로 다른 비동기 회로에서 사용할 수는 없으며 이는 metastability와 카운터의 동기화 문제가 그 원인이다.

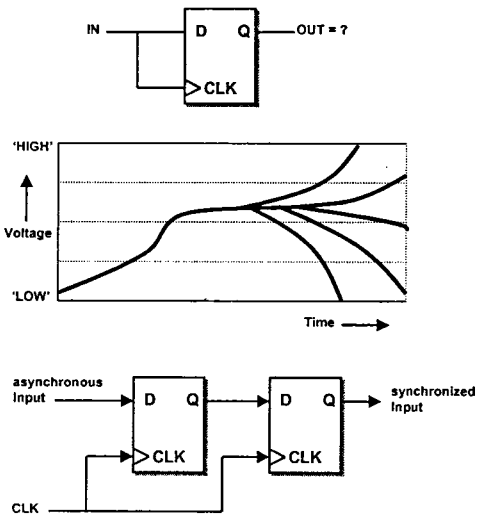

그림 5. Metastability 현상 및 그 해결책

그림 5.의 파형은 DFF의 입력과 클럭 단자를 서로 연결하고 전압을 가한 후 오실로스코프로 측정해 본 결과이다. 여기서 알 수 있듯이 DFF의 출력은 일정한 시간동안 'HIGH'나 'LOW' 값이 아닌 그 중간에 머물러 있다. 따라서 클럭과 데이터가 거의 같은 시점에 DFF에 도달할 수도 있는 비동기 회로에서는, 비록 확률은 매우 적지만 이러한 metastability 현상이 나타날 수 있기 때문에 그림에서와 같이 DFF을 직렬로 연결한 동기화 F/F의 출력을 사용해야만 오류를 최소화할 수 있다. 이러한 동기화 F/F을 사용한 FIFO는 그림 6.과 같다.

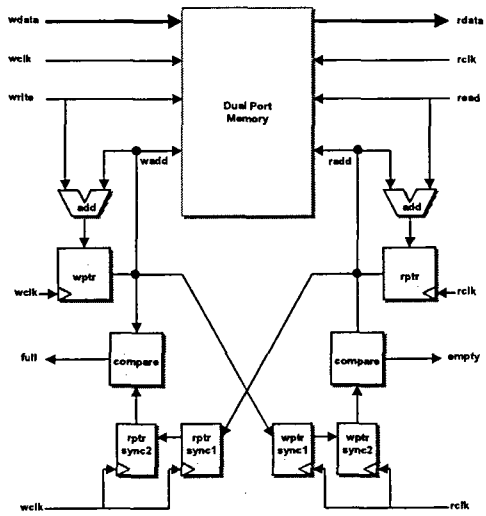

그림 6. 동기화 F/F을 사용한 비동기 FIFO

그러나 이러한 비동기 FIFO에 또 다른 문제가 있다. 이것은 카운터의 동기화 문제로써 이는 그림 7.에서와 같이 두 개의 비동기 클럭이 서로 비슷한 시간에 toggle될 때 발생한다. 동기화 F/F wptr sync1은 wptr의 값을 받으려하지만 wptr 값의 변화가 rclk의 에지에서 일어나므로 wptr sync1은 000 또는 111의 값을 받게 되고 이때에는 문제가 발생하지 않은다. 그러나 만약 3비트의 F/F으로 구성되어 있는 wptr sync1에서 아주 조금이라도 각 비트에 도달하는 rclk이 차이가 난다면 000 또는 111이외의 값을 갖게 될 것이고 이때에는 치명적인 오류가 된다.

따라서 이를 방지하기 위한 방법이 본 논문에서 제안하는 valid 비트를 사용하는 방식이며 이는 그림 8.에 나타나 있다. 이 방식은 카운터 자체를 동기화 F/F에 연결하지 않고 카운터가 가리키는 FIFO 엔트리에 해당되는 유효비트를 쓰기 또는 읽기가 실행될 때마다 toggle하는 방식을 사용한다. 이 방식을 사용하면 쓰기 또는 읽기를 할 때마다 오직 하나의 비트만이 바뀌게 되며 바뀌는 비트의 값을 동기화 F/F이 저장할 때 바뀌기 전의 값을 저장하거나 또는 바뀐 값을 저장하거나에 상관없이 정상동작이 이루어진다. 이러한 방식의 FIFO의 구조는 그림 9.와 같으며 full 및 empty 신호를 만드는 블럭이 그림 10.에 나타나 있다.

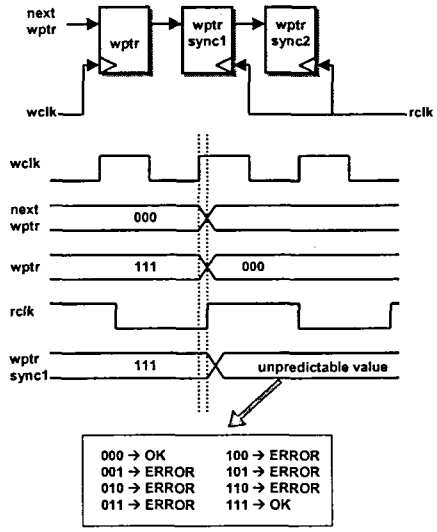

그림 7. 카운터 동기화 문제의 파형 예

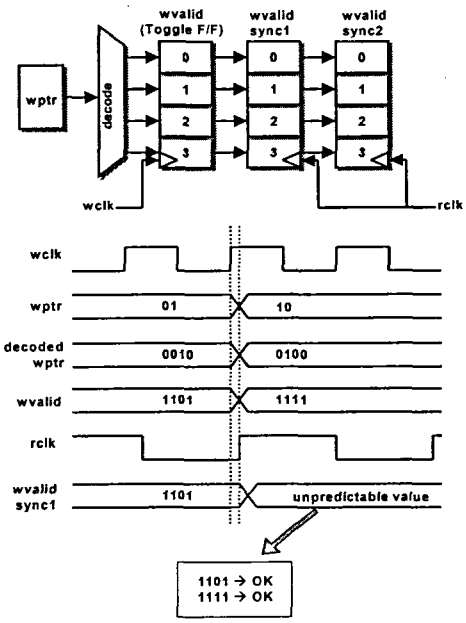

그림 8. Valid 비트 사용방식

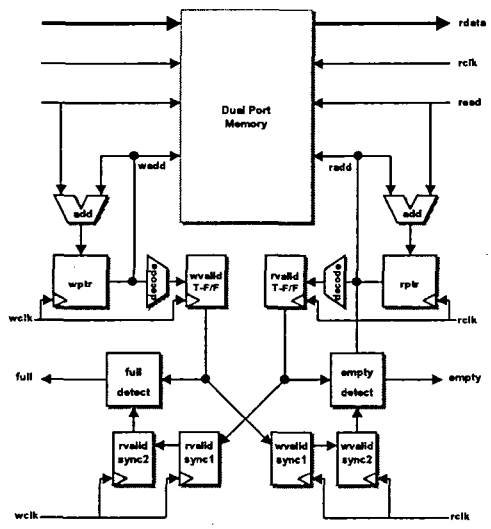

그림 9. Valid bit 방식의 비동기 FIFO 구조

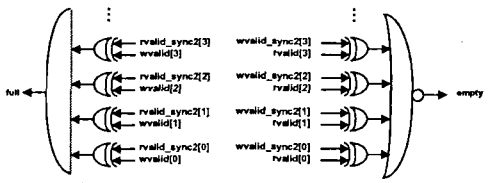

그림 10. Full 및 empty 신호를 위한 회로

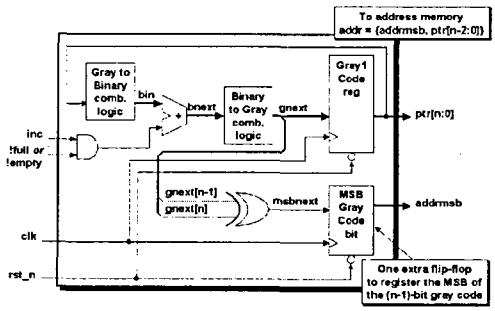

그림 11. Grey code 카운터 방식의 FIFO

Valid bit 방식은 기존의 구조에 비해 많은 수의 F/F이 사용된다. 이는 읽기 및 쓰기 카운터에 더하여 FIFO 엔트리 수만큼의 F/F이 사용되기 때문이다. 그러나 이 방식은 이해가 쉽고 속도가 매우 빠르다는 장점이 있다.

그림 11.은 Grey code를 사용하여 카운터를[2] 구성함으로써 FIFO의 쓰기 또는 읽기 시 카운터가 한 비트씩만이 변하도록 함으로써 카운터의 동기화 문제를 해결할 수 있도록 한 것이다. 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 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. Grey 카운터 FIFO와 유효비트 FIFO의 비교는 표 1.에 나타나 있다.

표 1. Grey 카운터 FIFO와의 비교

	Grey counter FIFO			Valid bit FIFO		
FIFO entries	4	8	12	4	8	12
Number of F/F	23	28	34	28	54	110
Number of combinational gates	176	273	374	130	245	405
Delay of critical path (ns)	3.6	5.6	7.4	3.0	3.6	4.1

III. 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

※ 반도체설계교육센터(IDEC)의 CAD Tool 지원에 감사드립니다.