고성능 마이크로프로세서를 위한 파이프라인 제어로직

배상태° 김훈국
광주과학기술원 정보통신공학과
{stbae°, hoonkook}@kjist.ac.kr

Fine-Grain Pipeline Control Circuit for High Performance Microprocessors

Sangtae Bae° Hong Kook Kim
Department of Information and Communications Kwangju Institute of Science and Technology

요 약

In a SoC environment, asynchronous design techniques offer solutions for problems of synchronous design techniques. Asynchronous FIFOs have the advantages of easier interconnection methods and higher throughput than synchronous ones. Low latency and high throughput are two important standards in asynchronous FIFOs. We present low latency asynchronous FIFO in the paper, which optimizes GasP[6]. Pre-layout of HSPICE simulations of a 8-stage FIFO on 1-bit datapath using Anam's 0.25μm technology indicates 17% lower latency than GasP.

1. 서 론

최근 공정 기술의 발전에 힘입어 반도체는 고속화, 소형화되고 있다. 이에 따라 칩 내에 집적할 수 있는 트랜지스터의 수도 증가하여 반도체 설계 기술에서는 시스템 온 칩(SoC) 경향이 가속화되고 있다. 미래 SoC 환경에서는 다양한 요구와 time-to-market을 만족시키기 위해 사전에 설계되고 검증된 마이크로 프로세서, DSP등의 IP(Intellectual Property)들을 연결하여 시스템을 구현하는 방법이 주를 이룰 것으로 전망된다.

IP들을 조합하여 새로운 시스템을 구현하는 SoC 환경에서는 기존의 전역 클록에 의존하는 동기식 설계 기법을 사용하는 경우, 클록 스큐에 의한 타이밍 종결(Timing Closure) 문제[1], 고 전력 소비의 문제[2]가 발생할 수 있다. 또한 각각의 동기식 IP들의 경우 다른 클록 속도로 동작하므로 각 IP들의 모듈간의 연결(interconnection)시 많은 설계자의 수정, 보완 노력에 의한 회로의 복잡도가 증가해 전체 시스템의 성능저하를 유발하며 많은 시간이 요구된다.

반면에 전역 클록을 사용하지 않는 비동기식 설계기법을 사용하면 위의 문제들을 해결할 수 있다. 비동기식 설계 기법은 전역 클록을 분배하지 않으므로 클록스큐로 인한 타이밍 종결 문제가 발생하지 않고, 불필요한 클록의 분배가 없으므로 전력 소모도 줄일 수 있다. 또한 비동기 설계기법은 핸드쉐이크 프로토콜(handshake protocol)을 사용하므로 모듈간의 연결이 쉽고, 안정적인 데이터의 전송이 가능하다.[3]

이러한 장점을 지니는 비동기식 설계방법을 이용 SoC를 구현하면 비동기 FIFO를 이용하여 각 모듈간의 연결을 하므로 비동기식 설계 시 필수적인 구성요소이다. 비동기 FIFO의 장점은 데이터가 들어오는 속도와 나가는 속도가 다른 동적인(elastic) 특성을 지니면서 적은 지연시간(low latency), 고 처리속도(high throughput)를 지닌다는데 있다. 이러한 고성능 FIFO는 전체 시스템의 성능 향상을 가져오므로 필수 요구사항이다.

본 논문에서는 기존에 제안되었던 비동기 FIFO인GasP을 개선하여 적은 지연시간(low latency)을 지니는 고성능 비동기 FIFO를 제안한다. 논문은 다음과 같이 구성된다. 제 2절에서는 비동기 설계기법의 간단한 소개와 비동기 FIFO 관련연구를 소개한다. 제 3절에서는 제안하는 비동기 FIFO 에 대해 소개한다. 마지막으로 제 4절에서는 레이아웃전의 실험결과를 보일 것이며, 제 5절에서는 결론과 추후 연구에 대하여 논한다.

2. 관련연구

비동기식 설계기법에서는 전역클록대신 모듈간의 제어신호(request와 acknowledge)에 근간한 핸드쉐이크 프로토콜을 사용한다.[3]

가장 고전적인 형태의 비동기 FIFO는 Sutherland에의하여 제안된 micropipelines[4]이다. Micropipelines는 제어신호의 레벨(Vdd, Gnd)이 모두 의미를 지니는 2-phase 프로토콜을 채택하여 FIFO의 성능을 증가시켰다. 그러나 데이터 회로의 설계 시 복잡한 래치를 사용하여, 성능이 떨어진 단점이 있었다.

이후 비동기 FIFO연구는 최근까지 활발히 진행되어 MOUSETRAP[5], GasP[6], IPCMOS[7] 등이 발표되었다.

콜럼비아 대학의 Singh이 제안한 MOUSETRAP에서는 2-phase 프로토콜을 사용하면서 설계자가 back-end 작업에서의 노력을 줄일 수 있도록 게이트 수준에서의 구현을 하였다. MOUSETRAP의 동작 속도는 0.25μm 공정에서 3.5GHz의 처리속도를 지닌다.

IBM에서 제안한 IPCMOS는 트랜지스터 수준에서의 극도의 세밀한 지연시간 조정(delay matching)이 필요하기 때문에 설계자의 능력에 의존한 회로를 가지게 된다. 사이클 타임(cycle time)도 12게이트를 지니는 반면 빠른 0.18μm IBM 공정을 이용하여 곱셈기를 칩으로 제작, 3.3GHz의 처리속도를 측정하였다.

Sun Micro Systems에서 개발한 GasP에서는 처리속도는 증가시키고 회로를 단순화시키기 위해서 single- track[8] 기법과 self-resetting NAND게이트를 사용하였다. GasP은 0.35μm 공정에서 1.5GHz의 처리속도를 지녔다.

Single-track기법은 제어신호가 request와 acknowledge로 두개의 도선에 나누어져 있던 것을 하나의 도선으로 결합시켜 사용하자는 개념이다.

그림 1은 GasP의 두 단을 보여준다. GasP의 FIFO 한 단은 place와 path의 두 부분으로 구성이 되어 있으며 path는 place의 좌, 우 어떠한 부분과도 한 단을 이룰 수 있다. *Place*는 data의 도착 여부를 알려주는 회로이고, *path*는 데이터가

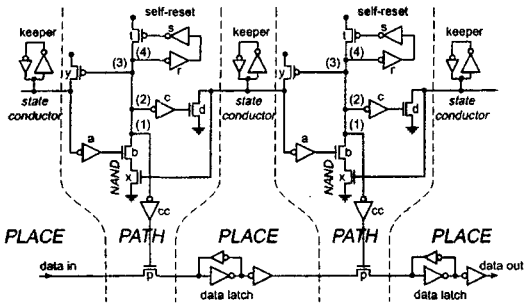

그림1. GasP의 2 Stage

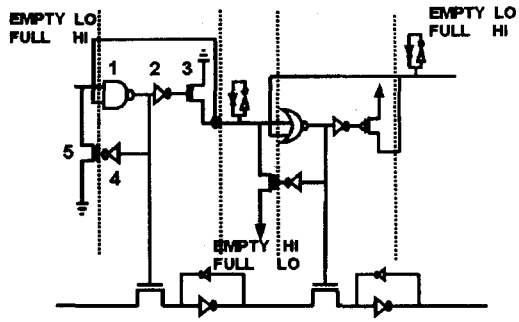

그림2. 제안된 FIFO의 2 stage

전달되기 위한 조건을 검사하는 부분이다. Place의 값이 low일 때, 현재의 단에 유효한 데이터가 저장되었다는 것을 의미한다. 이와 반대로 high값을 가지면 현재의 단에 데이터가 없이 비어있음을 알려준다.

Path회로는 좌측 place와 우측 place가 low와 high*(full과 empty)*의 관계를 지니는지를 NAND 게이트*(b, x)*를 통해 검사하여 다음 단으로의 데이터 전달여부를 결정한다. 좌우 place가 low와 high값을 가진경우, transmission게이트를 열어주는 제어신호를 cc 인버터를 통해 내어준다. 이때 그림 1의 r, s, t 경로의 self- resetting은 입, 출력 캐패시턴스를 줄이기 위해 사용한 것으로서 r, s, t를 통해 path회로의 레벨을 low에서 high 레벨로 변경한다. 그러나 self-resetting은 이전에 새로운 데이터가 준비되었다는 신호가 중복되어 도착하지 않도록 back-end시 설계자의 노력을 요하는 지연시간 가정을 두어 설계하는 단점을 지니게 된다.

GasP의 순방향 지연시간(forward latency)은 하나의 place에서 다음 단의 place로 데이터가 전달되었음을 알려주는 데 소요되는 시간으로서 그림 1에서는 s1에서 s2까지 가는데 걸리는 시간이다. 순방향 지연시간의 경우 최초 FIFO가 데이터가 없이 비어 있는 상황에서 입력 단에 처음 도착한 데이터가 얼마나 출력 단에 빨리 도달할 수 있는가를 알 수 있게 한다. 그림 1에서 s1의 low값이 s2의 low값으로 전해지는 구간이고, 4개의 게이트*(a, b, c, d)*를 통과하는 데 걸리는 시간이다. 역방향 지연시간(backward latency)은 FIFO에서 데이터들로 가득 차 있는 상황에서 출력 단의 하나의 데이터가 빠져 나갔을 때 입력 단에서 다음 데이터가 들어 올 수 있는 시간을 말한다. 그림 1에서는 s2의 place 의 high 상태가 s1의 place high로 전달되는 시간으로 2 게이트*(x, y)*이다. 전체 사이클 타임은 순방향 지연시간과 역방향 지연시간의 합으로 6게이트를 지닌다. 그러나 GasP은 트랜지스터 사이징과 시간 가정(timing assumption)과 같은 극도의 회로 기술(aggressive circuit technique)을 이용하였기 때문에 설계자 back-end 작업 시 많은 노력을 요하게 된다.

위의 FIFO들 간의 비교는 쉽지 않다. 첫째로, 게이트 레벨 정도에서의 전반적인 비교는 가능하지만, 실제 적용된 공정이 다르다. 실제 칩으로 나올 때까지 레이아웃이후의 도선의 정보 등의 여러 작업이 추가 될 수 있기 때문이다. 둘째로, 현재 IPCMOS는 칩으로 제작되어 나온 결과이고 MOUSETRAP, GasP은 레이아웃후의 성능측정을 통한 결과이기 때문이다. 따라서 각 FIFO들은 동등한 성능을 지닌다고 판단할 수 있다.

3. 제안된 Pipeline 구조

GasP은 현재까지 알려진 비동기 FIFO중 가장 우수한 성능을 지닌 FIFO의 하나이다. GasP의 단점은 초기 FIFO에 데이터가 비어있는 상황에서는 순방향 지연시간이 4게이트로서 역방향 지연시간에 비해 길기 때문에 새로운 데이터의 전파시간이 느리다. 또한 GasP은 회로 수준에서 설계자의 트랜지스터 사이즈에 따라 성능변화가 크고, 시간 가정을 만족시키기가 어렵다는 단점이 있다. 이러한 단점들을 보완하고 순방향 지연시간을 3게이트로 개선한 그림 2의 비동기 FIFO를 제안한다.

제안하는 FIFO의 좌측 단은 GasP와 유사한 형태이다. 즉, 한 place의 low의 상태는 데이터가 도착하여 있음을 의미하고, high의 상태는 데이터가 비어있음을 뜻한다. FIFO의 우측 단에서는 place의 low와 high의 의미가 반대를 지닌다. 다시 말하면, NAND 게이트가 포함되어 있는 path의 좌측 place에서는 level이 high가 데이터가 준비 되었다는 full의 의미를 지니지만, 우측 place에서는 high 는 데이터를 받을 수 있다는 뜻을 지닌 empty의 의미를 지니게 된다. 순방향 지연 시간이 3게이트의 홀수이기 때문에 한단의 레벨의 low의 개념이 다음단의 high의 개념으로 변화하게 되는 것이다. 따라서 상호 반대적인 의미를 FIFO안에 구현하기 위해서는 2종류의 단이 필요하다. 그림 2에서 볼 수 있듯이 한단에서는 place의 레벨이 high에서 low로 변화될 수 있도록 NAND 형태의 path logic이 필요하고, 다른 단에서는 레벨 low에서 high로 변화할 수 있도록 NOR 형태의 단이 필요하다.

제안된 FIFO의 성능을 분석하기 위해 지연 시간과 사이클 타임을 게이트 수준에서 분석해보면 주어진 회로의 유용성을 확인할 수 있다. 새로운 데이터의 최소 반복시간인 사이클 타임은 GasP과 마찬가지로 $t_{nand}(1) + t_{inv}(2) + t_{nfet}(3) + t_{nand}(1) + t_{inv}(4) + t_{nfet}(3)$의 6게이트를 지닌다. 이는 GasP 에서의 사이클 타임과 동일하기 때문에 구현시 동등한 처리속도를 지닐 수 있음을 뜻한다. 반면에 순방향 지연시간과 역방향 지연시간 모두가 3게이트로서 균일하다. 즉, GasP의 순방향 지연 시간인 4게이트보다 게이트 하나의 delay를 줄였음을 뜻한다. 순방향 지연시간은 $t_{nand}(1) + t_{inv}(2) + t_{nfet}(3)$의 3게이트로 짧아진다.

순방향 지연시간의 중요한 의미는 FIFO에 데이터가 비어 있는 상황에서 입력 단으로 입력된 데이터를 FIFO의 출력 단으로 출력하는 데까지의 시간이다. 따라서 비동기 FIFO를 사용하는 SoC환경에서는 여러 IP들의 연결 시에 이러한 적은 순방향 지연시간의 의미는 중요하다. 결과적으로 제안되는 FIFO는 외부 IP 입력 단에서의 처리속도가 FIFO의 출력 단보의 속도보다 느린 경우나 빠른 경우 모두 사용할 수 있는 범용성을 지닌다.

제안된 FIFO가 가지는 부가적 장점은 GasP에 비해 게이트 레벨에서의 디자인을 하기 때문에 back-end 작업시 트랜지스터의 사이징 노력을 줄일 수 있고, 시간가정이 존재하지 않는다는 것이다.

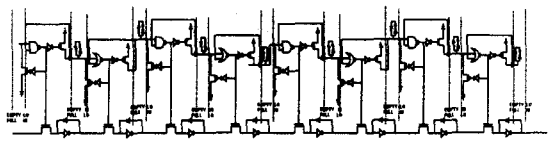

그림 3. 8 stage FIFO

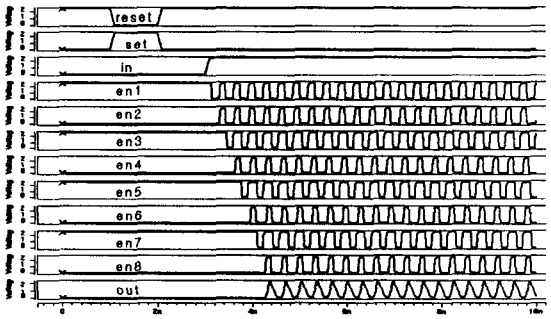

그림 4. Simulation 결과

4. Simulation 결과

본 절에서는 GasP과 개선된 비동기 FIFO의 HSPICE 레이아웃 이전의 성능측정 결과를 소개한다. 그림 3과같이 8단의 비동기 FIFO의 제어 로직을 구성하고 데이터 사이즈는 1비트로 설정하였다. GasP을 동일한 환경으로 구현해 제안된 FIFO와 성능을 비교하였다.

그림 4는 제안된 FIFO의 실험 결과 파형을 나타낸 것이다. 신호 중에 *reset*은 NAND 형태의 FIFO 한단을 초기화하는 신호이고, *set*은 NOR 형태의 FIFO 한단을 초기화하는 신호이다. *en1*~*en8*은 데이터를 통과시키기 위해 path에서 래치에 인가하는 제어 신호이다. *in*신호는 FIFO의 입력 단에 가해주는 request 신호이며, *out*신호는 FIFO의 외부로 나가는 request 신호의 파형이며 사이클 타임의 역수를 취하면 FIFO의 처리속도를 계산할 수 있다.

FIFO의 입, 출력단의 환경은 폐쇄 회로(closed loop)를 구성해 입, 출력단의 인가하는 신호가 반복 형성될 수 있도록 구성했다.

동작을 실험한 결과 제안한 FIFO는 지연시간이 6.3%~17%의 향상되었음을 알 수 있었다. 이는 최대처리속도의 감소가 GasP을 트랜지스터 사이징을 통해 최적화 할 경우에도 최대 5.7% 정도의 감소뿐이라는 사실을 고려하면 주목할만한 결과이다.

표 1에 실험 결과를 요약하였다. 표 1에서 GasP의 경우는 트랜지스터의 사이징을 최소로 가지는 기본 트랜지스터를 사용하였을 때의 결과이고, $GasP_{opt}$의 경우는 처리속도의 최대 값을 유도할 수 있도록 트랜지스터의 사이징을 조정 후, 실험을 반복한 후 구해진 최적의 실험 결과이다. 제안된 FIFO에서도 동일한 조건을 적용하여 실험을 수행하였다. 표 1에서 볼 수 있듯이 GasP은 트랜지스터의 사이징의 변화에 따라 12.23%, 25.92%의 성능 변화폭을 지니는 반면, 제안된 FIFO에서는 0.8%, 6.45%라는 성능의 변화만을 가져, 설계자의 노력을 줄이면서 고속의 회로를 구현할 수 있음을 확인하였다. 이는 설계자가 게이트 수준에서 FIFO를 디자인 할 수 있는 장점을 제공하기 때문이다.

표1. 제안된 FIFO의 Simulation 결과(0.25μm 아남)

	지연시간 (latency)	성능변화	처리속도 (throughput)	성능변화
GasP	1.455ns	12.23%	2.8GHz	25.92%
$GasP_{opt}$	1.277ns		3.5GHz	
제안된FIFO	1.207ns	0.8%	3.1GHz	6.45%
제안된$FIFO_{opt}$	1.197ns		3.3GHz	

5. 결론 및 추후연구

본 논문에서는 GasP보다 적은 지연시간 특성을 지니는 비동기 FIFO를 제안하였다. 실험의 결과 제안된 비동기 FIFO는 기존의 GasP과 비교해 최대 17%의 latency의 향상을 시킬 수 있음을 보였다. 또한 제안된 FIFO는 GasP보다 SoC 환경에서 설계자에게 게이트 수준에서의 간편한 설계환경을 제공할 수 있는 매력적인 회로로서 충분한 가치가 있을 것이다.

향후 연구로서 주어진 linear FIFO를 확장하여 parallel FIFO를 제안하고 실제적인 레이아웃후 좀더 정확한 성능 측정을 수행할 예정이다.

6. 참고문헌

[1] International Technology Roadmap for Semiconductors, Semiconductor Industry Association, 2001

[2] V.Tiwari et. al., "Reducing Power in High Performance Microprocessors," 35th DAC, June 1998

[3] A. Davis and S. M. Nowick, Asynchronous circuit design : Motivation, background, and methods. In G. Birtwistle and A. Davis, editors, Asynchronous Digital Circuit Design, Workshops in computing, pages 1-49. Springer-Verlag, 1995

[4] I. E. Sutherland, "Micropipelines," Communications of the ACM, Volume 32, No.6, pp. 720-738, June 1989

[5] Montek Singh and Steven Norwick, MOUSETRAP : Ultra-High-Speed-Transition Signalling Asynchronous Pipelines. In Proc. International Conference on ICCD 2001, pages 9-17

[6] I. Sutherland and S. Fairbanks. GasP : A minimal FIFO control, In Proc. Intl. Symp. Adv. Res. Async. Circ. Syst.(ASYNC), pages 46-53. IEEE Computer Society Presss. Mar. 2001.

[7] S. Schuster, W. Reohr, P. Cook, D. Heidel, M. Immeditato, and K. Jenkins, Asynchronous interlocked pipelined CMOS circuits operating 3.3-4.5GHz. In Proc. ISSCC, Feb. 2000

[8] K. van Berkel and Bink, "Single-Track Handshaking Signaling with Application to Micropipelines and Handshake Circuits, "Proc. of the Second International Sympsium on Advanced Research in Asynchronous Circuits and Systems, 1996.

[9] I. E. Sutherland, B. Sproull, and D. Harris, Logical Effort : Designing Fast CMOS Circuits, Morgan Kaufmann Publishers, Inc, 1999