論文2003-40SD-5-5

전역적 비동기 지역적 동기 시스템을 위한 고성능 비동기식 접속장치

(A High Performance Asynchronous Interface Unit for Globally-Asynchronous Locally-Synchronous Systems)

吳 明 勳 *, 朴 錫 在 **, 崔 滉 鎔 ***, 李 東 翊 *

(Myeong-Hoon Oh, Seok-Jae Park, Ho-Yong Choi, and Dong-Ik Lee)

요 약

GALS(Globally-Asynchronous Locally-Synchronous) 시스템은 대규모의 칩 설계 시에 설계의 용이성과 신뢰성을 확보할 수 있는 구조로 주목 받고 있다. 본 논문에서는 GALS 시스템에 필수적인 비동기 접속장치를 제안한다. 접속 장치는 크게 센더 모듈과 리시버 모듈로 구성되어 있으며, 센더 모듈에서는 부분적으로 내부 클록과는 무관하게 데이터 전송이 가능하다. 0.25㎛ 공정의 게이트 레벨 표준 셀 라이브러리를 사용하여 설계하였고, 성능 향상 정도를 시뮬레이션을 통하여 예측할 수 있었다. 마지막으로, 접속장치를 장착한 GALS 구조의 예제 회로를 설계하여 올바르게 동작함을 확인하였다.

Abstract

Globally-Asynchronous Locally-Synchronous (GALS) systems are worthy of notice as an adequate architecture for a large scaled chip design with guaranteeing easy designs and functional confidence. In this paper, we suggest an advanced structure of the interface unit which is indispensable for GALS systems by using stoppable clocks. The proposed interface unit is composed of a sender module and a receiver module. The sender module can carry out data transmission partially without the relation to an internal clock. We have designed it with 0.25㎛ standard cell library at the gate level and simulated its operation to show performance improvement. Finally, we constructed an example circuit with the interface unit and proved the correct operation of it.

Keywords : Globally-Asynchronous Locally-Synchronous system, stoppable clocking, synchronization, metastability, asynchronous wrapper

* 正會員, 光州科學技術院 情報通信工學科
(Dept. of Info. & Comm., Kwang-Ju Institute of Science and Technology(K-JIST))
** 正會員, 忠北大學校 半導體工學科
(Dept. of Semiconductor Eng., Chungbuk Nat'l Univ.)
*** 正會員, 忠北大學校 電氣電子컴퓨터工學部
(School of Electrical & Computer Eng., Chungbuk Nat'l Univ.)

※ 본 연구는 BK21, 정보통신부의 정보통신기초기술연구지원사업, IDEC의 지원으로 수행되었음.
接受日字:2002年4月25日, 수정완료일:2003年4月25日

Ⅰ. 서 론

최근, 공정 기술의 발전에 힘입어 소자가 고속화, 소형화되고 이에 따라 칩 내에 집적할 수 있는 트랜지스터의 수도 증가하여 반도체 설계 기술에서는 시스템 온 칩(SoC)의 경향이 보다 가속화되고 있다.

현재, 이러한 SoC과 같은 대규모의 시스템 구성 시에 기존의 전역 클록에 의존하는 동기식 설계 기법을 사용하는 경우, 클록 속도 증가에 따른 클록 스큐와 지터 문제를 해결해야만 하고, 타이밍 가정을 만족시키기 위해 전송 선로의 지연시간을 고려한 설계가 필요하다.

또한, 급격한 시장 변화에 따라, 설계 시간을 단축하려는 요구가 높아지고 있는 상황에서, 대규모의 칩에 대하여 설계 초기 단계에서부터 이러한 문제를 고려하는 것은 어려운 일이다. 더구나 ITRS(International Technology Roadmap for Semiconductor)[1]에서는 향후에 급격히 짧아진 소자 반응 시간에 기인하여 클록 속도가 급격히 증가할 것이지만, 상대적으로 길어진 전송 선로의의 지연시간 때문에 전체의 칩을 전역적 단일 클록으로는 구동시키지 못할 것으로 예측하고 있다. 전력 소모 측면에서도, 칩 전체에 안정된 클록을 배분해야만 하는 동기식 설계 기법은, 클록 배분을 위한 추가적인 회로에서 상당량의 전력을 소모할 수 밖에 없는 문제점을 지니고 있다[2].

비동기식 설계 기법은 단일 클록을 사용하지 않는다는 점에서 이러한 문제점들을 해결할 수 있는 대안으로 제시될 수 있다. 그러나, 비동기 설계 기법을 이용하여 전체 시스템을 설계할 경우, 설계 복잡도가 증가되고, 아직까지는 뒷받침할 수 있는 비동기 CAD 툴이 부족하며, 테스트하기가 어렵다는 단점들을 지니고 있다.

비동기식 설계 기법의 단점을 보완하고 동시에, 동기식 설계 기법의 문제점을 아키텍처 상에서 보다 근본적으로 해결할 수 있는 방안으로 GALS(Globally Asynchronous Locally Synchronous) 시스템이 연구되고 있다. <그림 1> 에서와 같이 GALS 시스템은 기본적으로 전역적 단일 클록을 사용하지 않고, 서로 독립적인 클록에 의해 동작되는 여러 개의 소규모 LS (Locally Synchronous) 모듈로 구성된다. 이 LS 모듈들은 직접적으로 point-to-point방식으로 연결될 수도 있고, 버스를 사용하여 연결될 수 있으며, 두 방식을 혼합하여 연결될 수 있다. 각각의 LS 모듈들은 동기식 설계 기법에 따라, 기존의 동기식 CAD 툴과 검증 방법을 이용하여 설계될 수 있으므로 조립성과 재사용성을 높일 수 있고, LS 모듈만의 지역화된 클록을 사용함으로써 전역 클록에 기인하는 여러 가지 문제점을 해결할 수 있다. 또한, 다른 timing domain을 가진 LS 모듈사이에서 안정된 데이터 전송을 보장하기 위해서, GALS 시스템의 LS 모듈들은 핸드셰이크 프로토콜 제어 신호를 동반하여 비동기적으로 데이터를 전송한다.

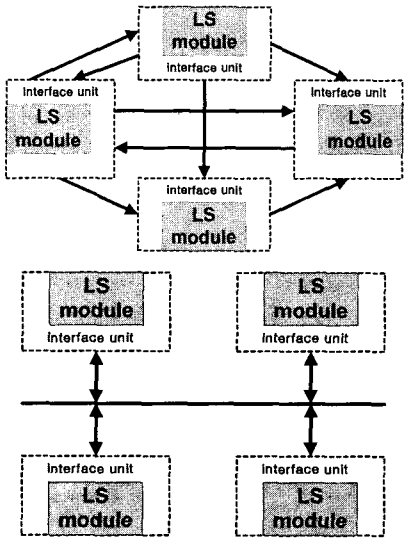


그림 1. GALS 시스템
Fig. 1. GALS systems.

이러한 GALS 시스템에서는 비동기적 데이터 전송에 필요한 특화된 비동기 접속 장치는 필수불가결하며, 모든 데이터 전송이 이를 통해서 수행되므로 비동기 접속 장치의 성능은 전체 시스템의 성능에 직접적인 영향을 미치게 된다.

본 논문에서는 비동기 접속 장치인 wrapper 회로의 성능을 효율적으로 향상시키기 위해 데이터의 송수신과 LS모듈의 내부 동작을 분리한 decoupled 구조를 제안하고 이의 동작을 검증하였으며 성능을 평가하였다.

나머지 논문의 구성은 다음과 같다. 먼저, wrapper의 핵심 역할이라고 할 수 있는 데이터의 동기화 실패(Synchronization Failure)를 방지 하기 위한 방법과 지금까지 연구된 wrapper를 Ⅱ장에서 살펴보고, 비동기 프로토콜 상에서 wrapper의 성능을 향상시키기 위한 기본적인 idea를 Ⅲ장에서 제안한다. Ⅳ장과 Ⅴ장에서는 제안된 2가지 형태의 wrapper의 구조와 동작 방식을 설명하고, Ⅵ장에서는 시뮬레이션 환경 및 설계방식을 제시한다. Ⅶ장에는 설계된 wrapper의 성능 측정 및 분석 결과를 요약하고, Ⅷ장에서 제안된 wrapper를 이용한 간단한 GALS 회로 구성 예를 설명한다. 마지막으로 Ⅸ장에서 결론을 맺는다.

Ⅱ. 동기화 실패 (Synchronization Failure) 문제

GALS 시스템에서의 데이터를 받는 리시버 블록의 입장에서는, 시간적으로 무관하게 들어오는 입력 데이터를 내부 클록으로 동기화하여 저장하기 때문에 두 신호간의 준안정 상태(metastability)가 발생할 수 있으

므로, wrapper 회로 설계 시에는 반드시 동기화 실패(synchronization failure) 문제를 고려해야 한다.

이 문제를 해결하기 위한 가장 쉬운 방법은 2개 이상의 래치를 직렬로 연결하는 형태의 동기 장치(synchronizer)를 사용하는 것이다[3]. 구현하기가 쉽고 동기화 실패 확률을 많이 줄일 수는 있지만, 100% 완벽하게 전송 데이터의 동기화를 보장할 수 없고, 동기화의 실패율을 떨어뜨릴수록 데이터의 잠복시간(latency)이 증가된다는 단점을 가지고 있다.

보다 근본적인 해결책으로[4, 5]에서 내부 클록을 제어하는 두 가지 방법을 제시하였다. 첫번째는 준안정 상태를 감지할 수 있는 동기장치(synchronizer)를 사용하여 리시버 블록의 입력단에서 준안정 상태 발생 시에, 이것이 내부 회로에서 해결될 때까지 클록의 발생을 억제하는 방식이다. 구현 시에는 주로 하위 레벨에서 잘 튜닝된 ME(Mutual Exclusion)[6] 블록이나, 준안정 상태를 감지 할 수 있는 회로[7]를 이용한다.

두번째 방법은, 리시버 블록에서 외부의 비동기 입력 신호에 의해서 클록을 발생시키고, 내부 동작이 끝났을 때, 클록을 멈추어서 다음 데이터를 기다리는 stoppable clocking 방식이다. 데이터를 전송하는 동안에는 클록을 휴지상태로 유지시키다가 데이터 전송 프로토콜을 끝마치고 데이터를 저장한 후에 클록을 다시 발생시켜, 준안정 상태를 해결한다기 보다는 이론적으로 준안정 상태를 피할 수 있다. 이 방법에서는 외부와의 핸드세이크 프로토콜과 내부 클록 정지를 통합하여 관리하는 비동기 제어 회로가 필요하다.

동기화 실패 문제를 해결하기 위해 내부 클록을 제어하는 방식 중, 첫번째 방법을 사용한 wrapper로는 [8]의 PCC(Pausible Clocking Control)와 [9]의 P-type wrapper가 있다. PCC는 내부에 ME 블록을 사용하였으며, 핸드세이크 프로토콜의 변화시점마다 클록이 휴지기간을 가지므로, 클록 정지 횟수를 줄이기 위해 외부와의 통신은 2-위상 핸드세이크 프로토콜을 가정하였다. 그러나, 다중 포트를 처리하기 위해서는 wrapper 내부에 사용된 아비터의 부담이 커지고, 여러 개의 데이터 전송을 동시에 요구하는 경우에 한 사이클에 오직 하나의 데이터 전송만을 처리할 수 있다. P-type wrapper 또한 ME 블록을 사용하였는데, 한 사이클에 여러 개의 데이터 전송을 동시에 처리할 수 있는 반면, 안정된 데이터를 저장하기 위해 준안정 상태를 지났음을 알리는 신호를 추가적으로 생성시켜야 한다. 이들 wrapper를 사용한 LS 모듈은 리시버 블록에서 연산이 필요 없는 경우에도 클록이 발생되므로, 불필요한 전력을 소모할 수 있다.

[9]에 제안된 D-type wrapper에 사용된 동기화 기법이 두번째 stoppable clocking 방식에 해당된다. 이 wrapper에서는 리시버 블록에서 연산이 완료되면, 다음 데이터 전송까지 클록은 휴지 상태를 유지하므로 불필요한 전력을 소모하지 않는다. 그러나, 여전히 ME 블록을 사용하였고, 리시버의 응답 시간이 길어지게 되면, 센더에서는 무조건 그 만큼의 시간동안 클록을 정지시켜야 하므로, LS 모듈의 내부 연산과 데이터의 전송이 병렬적으로 수행되지 못하기 때문에 전체 시스템의 성능이 저하될 가능성이 있다는 단점을 지닌다.

위에서 소개한 wrapper에서는 모두 상대적으로 제어하기 쉽다는 점때문에 링오실레이션 방식[6]을 사용하여 내부적으로 클록을 발생시켰다. 이 방법에서는 기본적으로 클록의 주기가 지연소자에 의존하여 결정되므로, 외부 환경에 따라 지연소자의 지연시간이 변하게 되면 클록의 안정성이 문제시 될 수 있다. 이 문제를 해결하기 위해, 지연소자의 지연 시간을 외부 환경에 따라 자체적으로 변화 시킬 수 있는 방법들이 연구되고 있다[10].

본 논문에서 제안하는 wrapper에서는 기본적으로, 동기 모듈과의 데이터 통신을 용이하게 할 수 있고 동기식으로 설계된 래치구조를 변경 없이 그대로 사용할 수 있는 4-위상 핸드세이크 bundled 데이터 방식으로 핸드세이크 프로토콜을 수행한다. 또한, 비동기 회로 설계 시 가장 흔히 사용되는 active-output-passive-input 형태의 푸쉬 채널을 가정한다[11]. 링 오실레이션 방식을 사용하여 내부 클록을 발생시키고, 동기화 실패 문제를 근본적으로 해결하기 위해 내부 클록을 제어하는 방식 중 stoppable clocking 기법을 채택하였으며, ME 블록

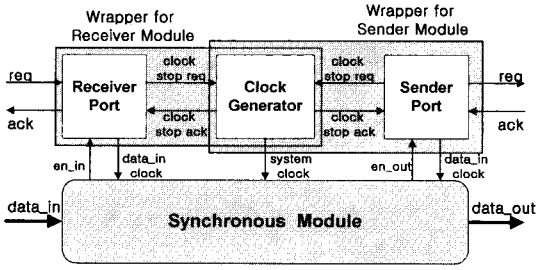


그림 2. wrapper의 구성
Fig. 2. Structure of wrapper.

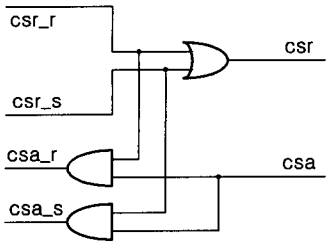


그림 3. 다중 포트를 위한 접속 회로
Fig. 3. Interface circuit for multi-port.

과 같은 특별한 동기장치 없이 표준 셀 라이브러리만을 사용하여 준안정 상태를 피할 수 있는 구조를 가진다. 특히, 데이터 통신 채널사이에 FIFO없이도 내부 연산과 외부 데이터 전송의 병렬성을 높이기 위하여, 각 LS 모듈의 클록과 데이터 전송을 부분적으로 독립적으로 수행할 수 있는 데이터 전송 메커니즘을 제공한다.

Ⅲ. stoppable clocking에 기반한 wrapper의 기본 구조 및 데이터 전송 메커니즘

<그림 2>에서 보는 바와 같이, 기본적으로 제안된 wrapper의 구조는 크게 출력 데이터의 전송을 수행하는 Sender Module과, 들어오는 입력 데이터를 저장하기 위한 Receiver Module로 나뉘어 진다. 다시, 각각의 모듈들은 내부 클록 제어용 신호들을 생성하고 외부와의 통신을 수행하는 Port 블록과, stoppable 클록을 발생시키는 Clock Generator 블록을 가지고 있다. 동기 모듈은 오직 하나의 클록에 의해 작동되므로, <그림 2>와 같이 Sender Port와 Receiver Port를 동시에 지니거나, 다중 포트를 갖는 경우에 각 Port 블록들은 Clock Generator 블록을 공유하게 된다.

동기 모듈에서 데이터를 받아야 하는 경우나, 데이터를 출력해야 하는 시점에서 내부 클록에 동기화된 신호(en_in, en_out)를 wrapper에 보내면, Port 블록에서는 제어 신호(clock stop req)를 생성하여 내부 클록을 정지시킨다. 동시에, 데이터의 저장과 출력을 위해 동기 모듈 맨 끝단의 래치를 구동하는 신호(data_in clock, data_out clock)를 만든 후에, 핸드세이크 프로토콜을 수행한다. 외부와의 통신이 끝나게 되면, 각 Port 블록에서는 클록을 정지시킨 신호와 래치 구동 신호를 해제하여 내부 클록을 다시 작동시키고, 다음 데이터 전송을 준비하게 된다. 다시 말해서, 동기 모듈의 내부 상태에 따라서 데이터 전송 시점이 정해지고, 클록의 정

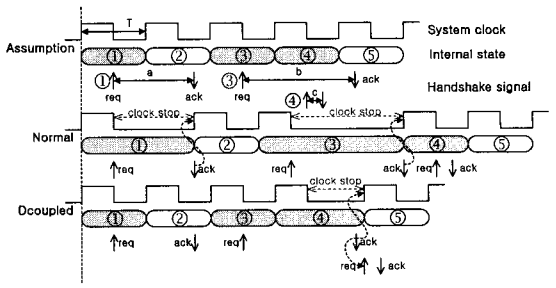


그림 4. stoppable clocking에 기반한 데이터전송 메커니즘
Fig. 4. Data transfer mechanism based on the stopplable clocking.

지 시점도 내부 클록과 동기화 되어 있지만, 다시 클록을 작동시키는 시점은 외부와의 핸드세이크 종료시점에 따라 비동기적으로 달라진다.

<그림 3>은 다중 포트를 갖는 경우에 Clock Generator 블록을 구성하는 내부 접속 회로를 나타내고 있다. 각각, 센더 포트와 리시버 포트에서 발생하는 클록 정지를 요구하는 신호들(csr_s, csr_r)이 동시에 발생할 때에도 모두 클록에 동기화 되어 발생하므로, 이들의 OR 연산을 통해서 한꺼번에 처리 할 수 있고, 내부 신호들의 해저드도 발생하지 않는다. Port 블록과 Clock Generator 블록의 더 자세한 기능은 다음 Ⅳ장과 Ⅴ장에서 설명된다.

<그림 4>는 주기 T를 갖는 시스템 클록을 사용하는 LS 모듈의 센더 모듈에서, stoppable clocking에 기반한 데이터 전송 메커니즘의 한 예를 나타내고 있다. 센더 모듈은 ①, ③, ④의 내부 사이클에서 데이터 전송을 요구하고, ②, ⑤의 사이클은 데이터 전송과는 무관한 내부 동작 상태로 가정한다. 또한, 3개의 데이터 전송에 필요한 핸드세이크 프로토콜 잠복 시간(latency)을 각각 a, b, c로 가정한다. 4-위상 핸드세이크의 시작점인 req(는 시스템 클록의 falling시점으로 동기화되어 있으며, 데이터 전송의 완료를 의미하는 ack(는 시스템 클록과는 무관하게 발생한다.

Normal 형태의 데이터 전송 방식은 전형적인 stoppable clocking 기법을 사용한 동기화 방식으로서, 센더 모듈의 내부 사이클이 전송 상태가 되면 클록을 정지시키고, 전송이 끝날 때까지 휴지 상태를 유지하게 된다. 전송이 완료되었을 때, 클록을 다시 생성하여, 다음 사이클의 동작을 수행한다. 물론, ④ 사이클의 데이터 전송처럼 한 클록 내에 전송이 완료된다면, 클록은

원래의 주기를 유지하게 된다. 그러나, 내부사이클 ②처럼 데이터 전송 상태 이후, 다음 상태가 데이터 전송과 무관하다면, 핸드세이크 프로토콜이 끝날 때까지 클록을 멈출 필요가 없다.

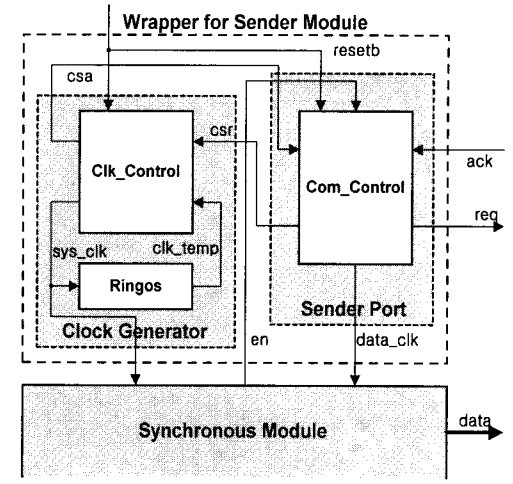
Decoupled 형태에서는 LS 모듈의 ② 상태의 내부 동작과, 외부의 데이터 전송에 필요한 핸드세이크 프로토콜을 병렬적으로 수행할 수 있다. 즉, 데이터 전송을 요구할 때마다 무조건 클록을 정지시키지 않고, Decoupled 형태의 내부사이클 ④처럼 이전의 핸드세이크 프로토콜이 끝나지 않은 상황(내부 사이클 ③에서 생성된 프로토콜 지연시간)에서 다시 데이터 전송 상태가 되면, 그때 비로소 클록을 멈추고 이전의 핸드세이크 프로토콜이 끝난 후, 다시 데이터 전송을 활성화 시키고 클록을 재 생성한다.

<그림 4>의 예에서 5개의 사이클을 수행하는 동안 소요되는 시간을 살펴보면, Normal 형태에서는 4T+a+b, Decoupled 형태에서는 3.5T+b로, Normal 형태에서는 각 전송 데이터의 잠복 시간이 모두 포함되나, Decoupled 형태에서는 각각의 전송 데이터가 내부 클록 즉, 내부 동작과 병렬적으로 수행됨으로써 더 빠른 종료 시간을 가진다.

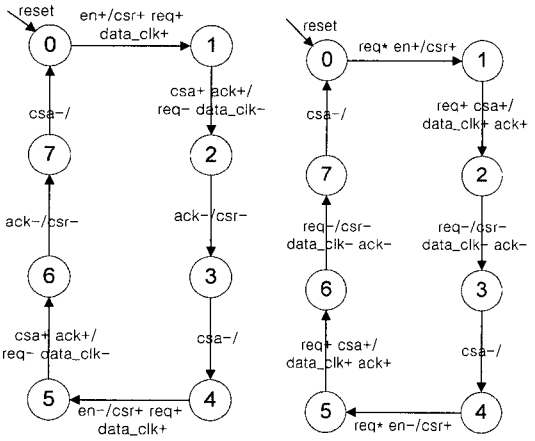
본 논문에서는 이 두 가지 형태의 데이터 전송 메커니즘을 지원하는 wrapper를 설계하고 성능 측면에서 비교하였다.

Ⅳ. Normal 모델

<그림 5(a)>는 Ⅲ장에서 언급한 Normal 형태의 wrapper 모델을 나타낸다. 홀수개의 인버터로 구성된 Ringos 블록과 아비터 역할을 수행하는 Clk_Control 블록은 동기 모듈 내부에서 사용될 클록을 생성한다. Port 블록을 구성하고 있는 Com_Control 블록은 클록을 제어하기 위한 신호를 발생하고, 외부와의 핸드세이크 프로토콜을 수행한다. 동기 모듈에서 발생하는 en 신호는 동기 모듈이 데이터 전송을 요구하는 경우 발생하며, csr(Clock Stop Request)신호와 csa(Clock Stop Acknowledge)신호는 데이터 전송 시에 클록을 멈추기 위해 Com_Control 블록과 Clk_Control 블록 사이에서 필요한 제어 신호이다. data_clk 신호는 전송할 데이터를 최종적으로 발생시키거나(센더의 경우), 입력 데이터를 저장하기 위해(리시버의 경우) Com_Control 블록에서 생성되는 신호이다.



(a)



(b) (c)

그림 5. (a) Normal 모델 wrapper의 블록도
(b) Com_Control 블록의 AFSM(센더의 경우)
(c) Com_Control 블록의 AFSM(리시버의 경우)

Fig. 5. (a) Block diagram of normal model wrapper
(b) AFSM of Com_control block(sender)
(c) AFSM of Com_control block(receiver).

이 Normal 모델에서는 센더 모듈과 리시버 모듈의 구조는 유사하며 단지, Com_Control 블록내의 AFSM (Asynchronous Finite State Machine)의 수행 방식이 달라진다. <그림 5(b)>와 <그림 5(c)>는 센더와 리시버 모듈의 AFSM을 보여준다. 본 논문의 AFSM은 burst-mode 기법[12]을 사용하고 있으며, Mealy 방식의 동기식 순서회로와 동작이 유사하다. 즉, 주어진 상태에서 모든 입력 신호의 변화가 만족되면, 출력신호를 발생시키고 내부 신호가 안정된 후에 다음 상태로 천이

하게 된다.

동기 모듈의 데이터 전송 요청 신호인 en 신호의 "return to zero"에 의한 추가적인 성능저하 요소를 제거하기 위하여, rising과 falling에서 모두 의미있는 동작을 부여하였다.

센더 모듈의 경우, 동기 모듈이 데이터 전송을 요청하게 되면(en+ 또는 en-), 클록을 정지시키기 위한 요청 신호(csr+)를 Clk_Control 블록에 전달하고, 동시에 데이터를 출력하며(data_clk+), 외부와의 핸드세이크 프로토콜을 생성한다(req+). 리시버로부터 데이터 저장을 완료했음을 의미하는 ack+ 신호가 도착하면, 출력단의 래치를 닫고(data_clk-), 4-위상 핸드세이크 신호의 'return to zero"를 위해 req- 신호를 발생시켜 통신이 끝나기를 기다린다. 최종적으로 ack- 를 인지하면, 클록 정지를 해제한다(csr-).

리시버 모듈에서는 기본적으로 핸드세이크 프로토콜의 시작 유무과 관계없이, 동기 모듈에서 받아야 할 데이터를 요구하는 경우에 클록을 정지한다. 즉, <그림 5(c)>의 초기 상태 ⓞ에서 req 신호 변화 유무에 상관없이, 리시버 모듈이 데이터를 받아야 되는 시점을 알리는 en 신호가 발생하기만 하면, csr+ 신호로 클록을 멈춘다. 그 이후에 req+ 신호가 미리 와있으면 바로 핸드세이크 프로토콜을 수행하며, req+ 신호가 오지 않았을 경우에는, 기다렸다가 도착 시에 ack+ 신호를 센더 모듈로 보내면서 핸드세이크 프로토콜을 수행한다. req- 신호가 도착했을 때, csr- 신호로 클록을 다시 발

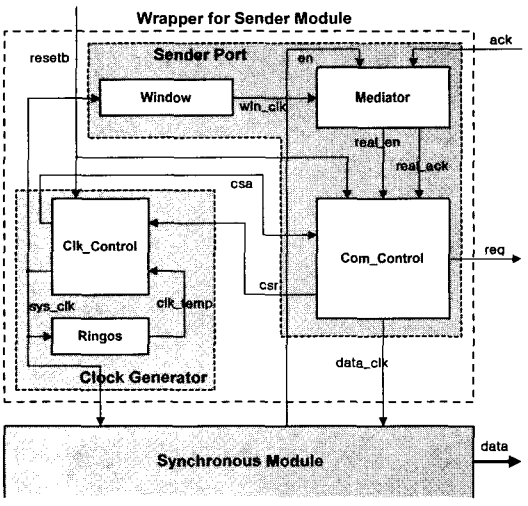


그림 6. Decoupled 모델 wrapper의 센더 모듈 블록도
Fig. 6. Block diagram of decoupled model wrapper.

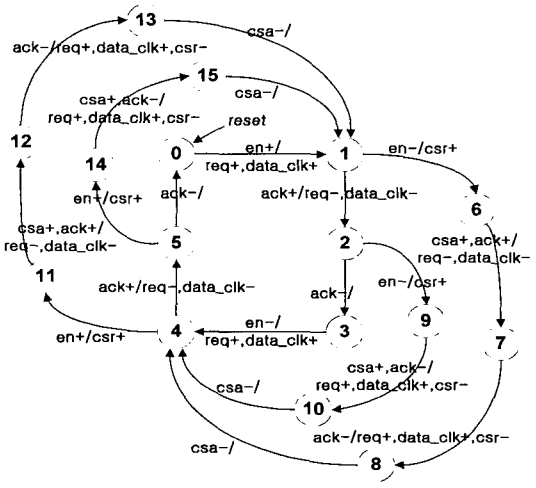


그림 7. Com_Control 블록의 AFSM
Fig. 7. AFSM of Com_control block.

생시키며, 최종적으로 상태 ④에서 센더로부터 한 개의 데이터 전송을 완료한 상태가 된다.

V. Decoupled 모델

Decoupled 모델의 센더 모듈은 기본적으로 Normal 모델의 센더 모듈 구조와 유사하다. 즉, Normal 모델에서 설명된 Clock Generator 블록을 사용하고, 동기 모듈과 wrapper 사이의 신호들도 모두 Normal 모델의 그것과 동일한 의미를 지닌다. 단지, Ⅲ장에서 설명한 데이터 전송 메커니즘을 구현하기 위한 설계 제약조건 때문에, en 신호와 ack 신호를 바로 쓰지 않고 Window 블록과 Mediator 블록에 의해 재생성된 real_en 신호와 real_ack 신호를 사용한다.

새로 추가된 Mediator 블록과 Window 블록과 함께, 외부와의 핸드세이크 프로토콜에 필요한 신호와 클록 정지 신호를 생성하는 Com_Control 블록은 <그림 6>의 Sender Port 블록을 구성한다.

Decoupled 모델의 센더 모듈은 Normal 모델과 달리, 핸드세이크 프로토콜을 시작할 때마다 내부 클록을 정지시키지 않으므로, 처음 req+ 신호를 보낸 후에, 프로토콜 종료를 의미하는 ack- 신호를 인지하기 전까지 다시 en 신호 발생을 허용한다. 이렇게 핸드세이크 프로토콜 종료 전에 또 다른 en 신호가 발생되었을 때, 비로소 Com_Control 블록에서는 csr+ 신호를 발생시켜 시스템 클록을 정지시키고, 핸드세이크 프로토콜이 끝

날 때까지 기다린다. ack- 신호를 확인하면, 나중에 발생한 en 신호에 대한 req+ 신호를 내보내어 핸드세이크 프로토콜을 시작하고, csr- 신호를 발생시켜 다시 시스템 클록을 활성화하여 동기 모듈의 동작을 시작시킨다.

1. Com_Control 블록

상대 리시버 모듈, 그리고 Clk_Control 블록과 각각 비동기적으로 통신함으로써 외부와의 데이터 전송과 내부 클록의 멈춤 여부를 동기화 실패없게 조절하는 기능을 담당하고, 특히, Decoupled 모델에서 필요한 외부 데이터 전송과 내부 클록과의 부분 병렬성을 구현하기 위한 Com_Control 블록의 AFSM 기술은 <그림 7>과 같다. 설계된 Com_Control 블록의 입력은 real_en 신호와 real_ack 신호이나, 편의상 입력 신호를 en과 ack로 설명한다.

초기 상태 ⓞ에서 동기 모듈의 en+ 신호를 받게 되면, 외부 리시버 모듈에 req+ 신호를 발생시키고 동시에, data_clk+ 신호를 동기 모듈에 보내어 데이터를 출력시킨다. 상태 ⓞ에서 ③까지는 외부와의 4-위상 핸드세이크 프로토콜을 나타내고 있으며, 마찬가지로 상태 ③에서 ⓞ까지는 en- 신호에 대한 수행과정을 나타내고 있다.

그러나, 상태 ①에서 ⑥으로의 천이와 ②에서 ⑨로의 천이처럼, 아직 핸드세이크 프로토콜이 완전히 끝나지 않은 상황에서 en+ 신호가 발생되면, csr+ 신호로 클록을 정지 시키고, 전체 회로가 휴지상태에서 아직 끝나지 않은 프로토콜을 수행한다. 수행이 끝남을 의미하는 ack- 신호에 의해서 두 번째로 발생한 en- 신호에 대한 req+ 신호를 발생한다. 마찬가지로, 상태 ④와 ⑤에서도 상태 ⑪, ⑭로 천이 할 때 위와 같은 일을 수행한다.

Decoupled 모델에서는 en과 ack 신호가 서로 독립적으로 발생함을 허용하기 때문에, 근본적으로 이들 신호 사이의 순서 관계는 정해지지 않는다. 그러므로, 위에서 설명한 <그림 7>과 같이 AFSM에 기반하여 설계할 때, 일반적으로 AFSM을 올바르게 동작시키기 위해서는, 다음과 같은 두 개의 설계 제약조건을 만족시켜야 한다[12].

fundamental-mode environmental constraint : 두 신호 중 하나의 신호에 의해 발생하는 출력 신호가 안정화되기 전에 다른 나머지 신호가 발생되지 않아야 한다.

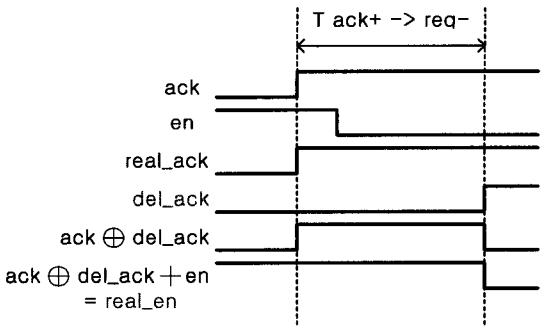


그림 8. fundamental-mode environmental constraint 의 해결방법

Fig. 8. Solution for fundamental-mode environmental constraint

distinguishability constraint : 주어진 한 상태에서 두 개 이상의 상태로 분기되는 곳에서의 입력 신호들은 그 중 오직 한 개의 상태만으로 천이될 수 있도록 입력 신호들이 발생되어야 한다.

2. Mediator 블록

Mediator 블록은 이 두 가지 제약 조건 중 fundamental-mode environmental constraint를 만족시키기 위해서 서로 독립적으로 발생하는 en 신호와 ack 신호를 시간적으로 차이를 두어 발생시킨 신호들(real_en, real_ack)을 생성시킨다. <그림 8>은 예로써, <그림 7>의 상태 ①에서 ②로 천이 시에 ack+와 en- 신호의 이벤트 발생 상황에 대한 기본적인 해결 방법을 보여주고 있다.

<그림 8>처럼 en-와 ack+중에서 ack+가 먼저 발생한 경우, ack+에 대한 출력 값인 req-까지의 시간(Tack+->req-) 동안에 en-이 발생하면, AFSM이 오동작을 할 가능성이 있다. 이때에는 ack+ 신호를 Tack+->req- 만큼 지연시킨 del_ack 신호와 ack 신호를 사용하여, Tack+->req- 구간에 일종의 필터 신호(ack(del_ack)를 만들고, 이 신호 구간에 발생하는 en 신호는 req- 이후로 지연시킨다(real_en). 결과적으로, ack+가 먼저 발생되면, 상태 ①에서 상태 ②로 천이되고, 이 때의 출력 값인 req- 신호가 안정화되기 전에 발생한 en 신호는 그 이후로 지연되며, AFSM은 이 지연된 신호를 사용함으로써 fundamental-mode environmental constraint를 만족시킬 수 있다.

마찬가지로 상태 ①에서 ⑥, 상태 ②에서 ③과 ⑨, 상태 ④에서 ⑤와 ⑪, 상태 ⑤에서 ⓞ와 ⑭로의 천이도 같은 방법으로 적용될 수 있다. 모든 경우를 적용하여

생각할 수 있는 Mediator 블록의 스키메틱은 <그림 9>와 같다.

fundamental-mode environmental constraint를 만족시켜야할 총 8가지의 경우를 분석했을 때, en+, en-, ack+, ack- 다음에 오는 신호들을 지연시켜야 하므로, 필요한 지연 소자의 수를 4개로 구분하였다. ack와 win_clk 신호의 입력 신호로 이루어진 Asymmetric C-element는 다음 Window 블록에서 설명한다.

실제로 Mediator 블록의 설계 시에 중요하게 생각되어야 할 것은, 회로의 올바른 동작을 보장하면서 동시에 성능을 최대화할 수 있도록 4개의 최적 지연 소자의 조건을 구하는 작업이다. 이를 위하여, <그림 8>에서 든 예를 <그림 9>의 게이트 지연시간까지 고려하면, <그림 10>의 타이밍도로 표현할 수 있다.

case 1 에서, real_en- 신호가 발생되어야 하는 시점은 <그림 6>의 Mediator 블록에서 Tdel_ack 시간을 지연시키는 지연소자 del_ack1를 거친 del_ack 신호가 내부 XOR 게이트, OR 게이트, C-element를 거친 이후

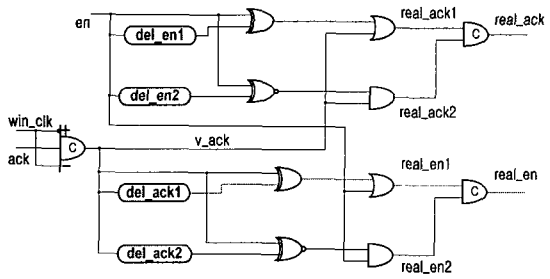


그림 9. Mediator 블록의 게이트 레벨 스키메틱
Fig. 9. Gate level schematic of Mediator block.

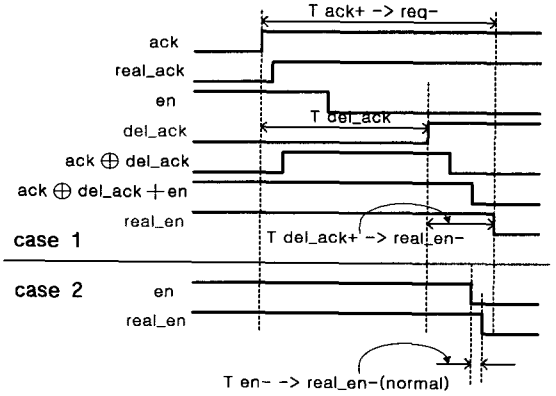


그림 10. 지연 소자 최적 조건 계산을 위한 타이밍도(1)
Fig. 10. Timing diagram for optimal conditions for delay cell (1).

여야하며, 이 Tdel_ack+→real_en-시간은 내부적으로 고정되어 있으므로, 구하고자 하는 Tdel_ack는 식 (1)에서 유도될 수 있다.

$$Tdel\_ack > Tack+ \rightarrow req- - Tdel\_ack+ \rightarrow real\_en- \quad (1)$$

또한, case 2 경우처럼 en-의 변화가 ack ⊕ del_ack 이후에 발생할 때, en-에 대한 real_en- 신호의 변화는 req- 이전에 발생할 수도 있으므로, Tdel_ack은 (1)을 만족시키면서 동시에 식 (2)을 만족시켜야 한다.

$$Tdel\_ack > Tack+ \rightarrow req- - Ten-(real\_en- - Tdel\_ack+ \rightarrow xor\_ack- \quad (2)$$

식 (1)과 식 (2)에서 ack 신호에 대한 지연 조건을 만족시켜 req-가 안정화 된 후에 real_en-을 발생시켰다고 할지라도, <그림 7>의 상태 ①에서 ②로의 천이 후, 상태 ⑨로 천이 시, real_en-이 변화하여 출력 값인 csr+가 안정화되기 전에, req- 신호에 대한 응답인 ack- 신호가 발생될 수 있다. 다시 말해서, 지연소자 del_ack1의 upper bound가 필요하게 된다. <그림 11>은 상태 ① → 상태 ② → 상태 ⑨의 천이 시에 만족되어야 하는 Tdel_ack 조건을 표현하고 있다.

Tack+→req-는 Mediator 블록과 Com_Control 블록에서의 고정된 시간이며, Tack-→real_ack-과 Tdel_ack+→real_en-는 Mediator 블록에서, Treal_en-→csr+는 Com_Control 블록에서 고정된 시간이다. 또한, 리시버 모듈의 Treq-→ack-의 최소 시간을 구할 수 있으므로, Tdel_ack 최대 조건을 구할 수가 있다. ack+가 발생하여 외부 리시버 모듈과 핸드세이크 프로토콜을 수행하여 다시 real_ack-가 발생하는 시간보다, ack+ 신

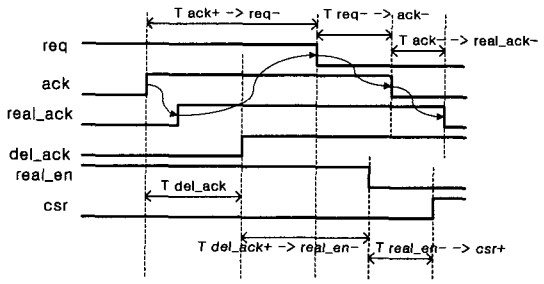


그림 11. 지연 소자 최적 조건 계산을 위한 타이밍도(2)
Fig. 11. Timing diagram for optimal conditions for delay cell (2).

호를 지연시킨 후 발생되는 real_en- 신호가 csr+을 발생하는 시간이 더 적어야 한다. 즉,

$$Tack+ \rightarrow req- + Treq- \rightarrow ack- + Tack- \rightarrow real\_ack- > Tdel\_ack- + Tdel\_ack+ \rightarrow real\_en- + Treal\_en- \rightarrow csr+ \quad (3)$$

의 식을 만족해야하고 다시 쓰면 식 (4)와 같다.

$$Tdel\_ack < Tack+ \rightarrow req- + Treq- \rightarrow ack- + Tack+ \rightarrow real\_ack- - Tdel\_ack+ \rightarrow real\_en- - Treal\_en- \rightarrow csr+ \quad (4)$$

지금까지의 V장 2절의 설명을 요약하면, <그림 7>의 Com_Control 블록을 구성하는 AFSM에서, 상태 ①에서 ②로의 천이 과정에서 발생하는 fundamental-mode environmental constraint를 해결하고 성능 저하를 최소화하기 위해서는, <그림 9>의 Mediator 블록의 최적 지연 소자의 조건을 식 (1), (2), (4)에 의해 결정하면 된다.

나머지 상태 천이에 대해서도 fundamental-mode environmental constraint를 해결하기 위한 각 지연 소자의 조건식은 <표 1>에 요약되어 있다.

표 1. 지연소자 조건
Table 1. Conditions of delay cell

상태천이	지연소자	조건
⑤→⑭	Del_en1	Tdel_en > Ten+ + csr+ + Tdel_en+ + real_ack Tdel_en > Ten+ + csr+ + Tack + real_ack + Tdel_en+ + xor_en
②→⑨		Tdel_en > Ten + csr+ + Tdel_en + real_ack Tdel_en > Ten + csr+ + Tack- + real_ack + Tdel_en + xor_en-
④→⑪	Del_en2	Tdel_en > Ten+ + csr+ + Tdel_en+ + real_ack+ T_en > Ten+ + csr+ + Tack+ + real_ack+ + Tdel_en+ + xnor_en-
⑦→⑥		T_en > Ten + csr+ + Tdel_en + real_ack+ Tdel_en > Ten + csr+ + Tack+ + real_ack+ + Tdel_en + xnor_en+
①→②	Del_ack1	Tdel_ack > Tack+ + req + Tdel_ack+ + real_en Tdel_ack > Tack+ + req + Ten + real_en + Tdel_ack+ + xor_ack Tdel_ack < Tack+ + req + Treq + ack + Tack + real_ack Tdel_ack+ + real_en + Treal_en + csr+
②→③		Tdel_ack > Tack + stable + Tdel_ack- + real_en Tdel_ack > Tack- + stable + Ten + real_en + Tdel_ack + xor_ack-
④→⑤	Del_ack2	Tdel_ack > Tack+ + req + Tdel_ack+ + real_en+ Tdel_ack > Tack+ + req + Ten+ + real_en+ + Tdel_ack+ + xnor_ack+ Tdel_ack < Tack+ + req + Treq + ack + Tack- + real_ack- Tdel_ack+ + real_en+ + Treal_en+ + csr+
⑤→⑥		Tdel_ack > Tack + stable + Tdel_ack + real_en+ Tdel_ack > Tack + stable + Ten+ + real_en+ + Tdel_ack + xnor_ack+

3. Window 블록

<그림 9>의 Mediator 블록에서 en 신호가 회로에 인가되었을 때, XOR 게이트나 XNOR 게이트의 출력 값이 안정화되지 않은 상태에서 ack 신호가 인가되면, 내부 신호들에서 글리치가 생길 가능성이 발생하고, 이는 Mediator 블록의 최종 출력 값에도 영향을 미치게 된다. 마찬가지로, ack 신호가 XOR, XNOR 게이트를 지나고 있는 도중에, en 신호가 인가되어도 Mediator 블록은 오동작을 할 수 있다. 다시 말해서 distinguishability constraint를 만족시키지 못하므로 Com_Control 블록의 AFSM에서 분기 시에 상태에서 천이가 올바르게 진행되지 못한다.

이 문제를 해결하기 위해, en 신호가 시스템 클록과 동기화 되어 발생한다는 점에 착안하여, Window 블록은 en 신호의 발생가능 시점의 전후에서 일종의 필터 역할을 할 수 있는 신호를 발생시킨다. 이 신호의 필터 구간에 ack 신호가 발생하는 경우, Asymmetric C-element를 사용하여 이 ack 신호를 필터 구간이 끝나는 시점과 동기화하여 출력시킬 수 있으므로 distinguishability constraint를 만족시킬 수 있다.

<그림 12>에서와 같이 제안된 wrapper에서는 시스

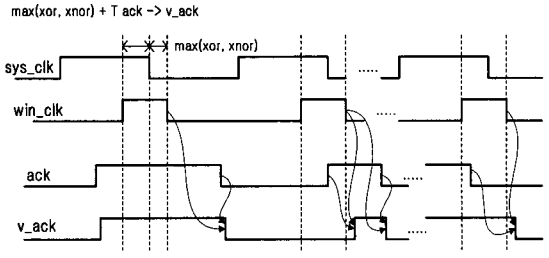


그림 12. distinguishability constraint를 위한 타이밍도
Fig. 12. Timing diagram for distinguishability constraint.

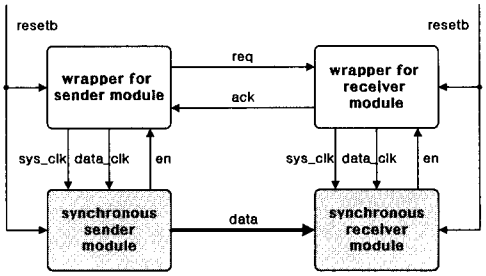


그림 13. 시뮬레이션 환경
Fig. 13. Simulation environment.

템 클록의 falling 시점에서 en 신호가 발생하므로, 그 falling 시점에서 뒤로 XOR와 XNOR 게이트의 최대 지연시간만큼과, 앞으로 동기화에 필요한 Asymmetric C-element와 두 게이트의 최대 지연시간의 합만큼의 필터 구간을 만드는 win_clk 신호를 발생시키고, 이 신호의 falling 시점과 외부의 ack 신호를 동기화하게 되면 Mediator 블록의 오동작을 막을 수 있다.

VI. 설계 및 시뮬레이션 환경

stoppable 클록에 기반한 전통적인 방식인 Normal 형태의 wrapper와, 외부 데이터 전송과 내부 동작의 병렬성을 높이기 위해 센더 모듈을 개량한 Decoupled 형태의 wrapper 사이에서 성능 개선 정도를 확인하기 위하여, <그림 13>과 같은 one-to-one구조를 시뮬레이션 환경으로 구성하였다. 즉, 동기 센더 모듈 1개와 동기 리시버 모듈 1개가 존재하고, 이들 사이의 데이터 전송을 위한 wrapper를 Normal 모델과 Decoupled 모델의 2개의 시뮬레이션 그룹으로 구성하였다. Decoupled 모델의 리시버 모듈은 Normal 모델의 것을 동일하게 채택하였다.

두 모델 wrapper내의 센더와 리시버 모듈에 존재하는 Com_Control 블록의 AFSM은 3D 툴[12]을 사용하여 논리식 추출 후, 2-level AND-OR 로직으로 구성하였다. 여러 개의 인버터 체인 형식으로 구성된 Ringos 블록에서는 인버터의 개수를 변경하여 원하는 시스템 클록의 주기를 조절하였다. 기본적으로 verilog-HDL로 모델링하였고, Com_Control 블록 내의 AFSM과 같이 타이밍이 중요한 부분은 게이트 맵핑하였으며, Decoupled 모델 wrapper의 센더 모듈에서 사용된 C-element는 표준 셀 라이브러리를 사용하여 RS 래치에 조건식을 연결하는 방식으로 설계하였다. Ⅳ장에서 구한 Decoupled 모델의 지연소자 조건에 의해, 지연 소자 삽입 시에는 50% 마진율을 적용하였다.

아비터 역할을 수행하는 <그림 5>와 <그림 6>의 Clk_Control 블록은 두 입력 신호인 csr 신호와 clk_temp 신호의 발생 시점이 모두 예측 가능하므로, ME 블록을 사용하지 않고 간단하게 구현할 수 있었다. 다시 말해서, csr 신호는 클록의 falling 시점에서 발생하는 en(real_en) 신호에 의해 Com_Control 블록 내의 고유한 지연시간 이후에 발생하며, clk_temp 신호도 Ringos 블록의 인버터 체인의 지연시간 후에 발생하므로, en(real_en) 신호 발생 시에 clk_temp 신호 발생전에만 csr 신호가 발생할 수 있다면, <그림 14>와 같이 Clk_Control 블록을 간단하게 구성할 수 있다. 대신 최소 클록 주기가 <그림 15>와 같이 결정되어야 한다.

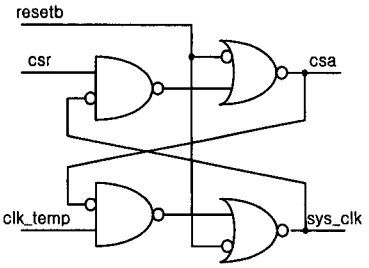


그림 14. Clk_Control 블록
Fig. 14. Clk_Control block.

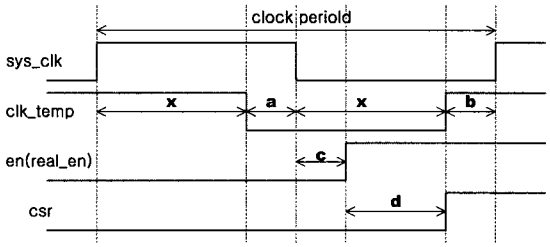


그림 15. 최소 클록 주기 발생 조건
Fig. 15. Condition for minimal clock period.

x 는 인버터 체인의 rising 혹은 falling 지연시간이고, a와 b는 각각 Clk_Control 블록의 지연시간이며, c는 클록 falling 시점에서 en 신호가 발생할 때까지의 시간(Decoupled 모델의 경우는 real_en 신호 발생시간), d는 Com_Control 블록에서 csr 신호의 발생 시간이다. 여기에서, clk_temp 신호 발생 이전에 csr 신호가 발생해야하므로, x>c+d 의 조건을 만족해야하고 최대 클록 주기를 결정하는 2x+a+b의 값을 구할 수 있다. 설계된 wrapper의 최대 클록 주파수는 Nomal 모델의 경우는 대략 800Mhz, Decoupled 모델의 경우는 대략 400Mhz 였고, 올바른 동작을 시뮬레이션을 통해 확인하였다.

회로의 합성은 SYNOPSYS 툴과 0.25㎛ IDEC-C221 라이브러리[13]를 사용하였으며, CADENCE 툴을 이용하여 시뮬레이션을 수행하였다. 또한, Mediator 블록의 동작을 SPICE 레벨에서 시뮬레이션하여 회로의 안정성을 검증하였다.

Ⅶ. 실험 결과 및 분석

제안된 Decoupled 모델과 Normal 모델의 throughput 측정을 위해서 공통적으로 데이터 전송 수와, 센더 모듈과 리시버 모듈 사이에 Micropipeline 형태[11]로 설계된 FIFO의 단 수를 변화시키면서 최종적으로 5000 클록 사이클이 끝나는 시간을 측정하였다. 또한, 센더와 리시버의 클록 주기도 변화시킴으로써 성능에 어떤 영향을 주는 지도 관찰하였다.

동기 모듈 전송 시점 상태 즉, 센더 모듈에서는 데이터를 보내야 되는 상태와, 리시버 모듈에서는 데이터를 받아야 되는 상태는, 각 모듈의 클록 진행 상황에 따른 내부 상태에서 균일 분포 확률로 각각 같은 수를 추출하였다. 다시 말해서, 데이터 전송 수를 0에서 최대 5000개로 변화시켰고, 각각에 대해서 모두 60개의 무작위적으로 추출된 전송 시점 상태 데이터 정보로 시뮬레이션 한 다음, 최종 5000 사이클의 종료 시점을 평균하여 계산하였다. 동시에, FIFO 단 수를 변경하여 Decoupled 모델의 병렬성 정도를 비교하였다. 또한, 클록 주파수도 33㎒ 부터 250㎒까지 변경하여 시뮬레이션을 수행하였다.

<그림 16>에는 센더 모듈의 클록을 80㎒, 리시버 모듈의 클록을 48㎒로 작동시키고, 각 모델에서 FIFO의 단 수를 0에서 2개까지 변화 시켰을 때, 데이터 전송 수에 따른 리시버 모듈의 5000 클록 사이클의 종료 시

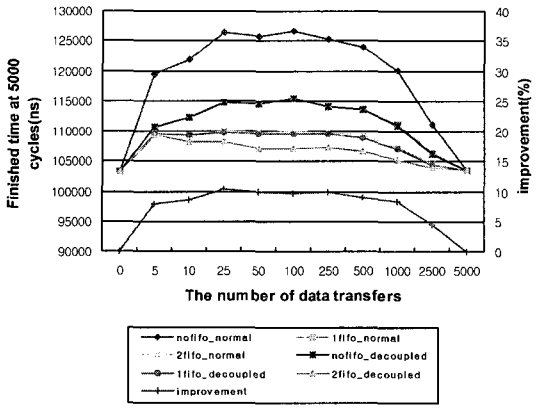


그림 16. 데이터 전송 수와 FIFO 단 수에 따른 리시버 모듈의 종료 시간

Fig. 16. Completion time of receiver module according to the number of data transfer and FIFO stage.

표 2. data 크기에 따른 게이트 수

Table 2. The number of gate according to data size.

Data size(bit)	32	64	128
Decoupled	511.6	511.6	511.6
Normal + 1 FIFO	476.4	620.4	908.4

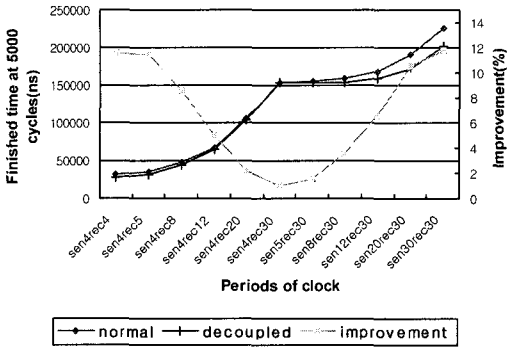


그림 17. 다른 클록 주파수에 따른 리시버 모듈의 종료 시간

Fig. 17. Completion time of receiver module according to different clock frequencies.

간을 나타내고 있다. Decoupled 모델이 Normal 모델에 비해서 얻는 성능 향상 정도는 전송 데이터가 없거나 (데이터 전송 수가 0), 매 사이클 데이터를 전송하는 경우(데이터 전송 수가 5000)를 제외하면, 약 5%에서 12%까지 증가 함을 알 수 있었다. FIFO의 단 수를 늘림으로써 관찰된 결과로는, Decoupled 모델의 성능이 FIFO 한 단을 더 추가한 Normal 모델의 성능과 비슷함을 알 수 있었다.

<표 2>는 비슷한 성능을 가지는 Decoupled 모델과, FIFO 1단이 추가된 Normal 모델 구조에서, 전송 데이터의 크기가 변화함에 따라 wrapper 설계 시에 사용된 게이트의 수를 나타내고 있다. 데이터의 크기가 증가하면, 사용되는 FIFO의 면적에 따라 normal 모델의 면적도 커지지만, Decoupled 모델의 면적은 일정하므로 같은 성능이라도 면적 측면에서 Decoupled 모델이 유리함을 알 수 있다.

<그림 17>의 결과는 다양한 클록 주파수를 적용한 후에, 250개의 전송 데이터에 대해서 리시버 모듈의 종료 시간을 측정한 것이다. 직관적으로 알 수 있듯이, 4 ns 주기의 클록을 갖는 센더와 리시버 모듈이 가장 빨리 종료되며, 두 모듈이 모두 30 ns 주기의 클록으로

동작할 때 가장 긴 종료 시간을 갖는다. Decoupled 모델이 Normal 모델에 비해 얻는 성능 향상 정도는 센더와 리시버 모듈의 클록 속도 차이가 적을수록 크게 나타남을 알 수 있는데, 이는 클록 속도 차이가 크면, 두 모듈이 통신을 요구하는 시점이 한쪽만 burst하게 발생하게 되므로, Normal 모델에 비해서 Decoupled 모델이 크게 이득을 얻지 못함을 의미한다. 다시 말하면, Decoupled 모델은 센더, 리시버 모듈의 동작 속도가 비슷한 구조에서 Normal 모델에 비해 더 큰 성능 이득을 얻을 수 있다.

Ⅷ. Wrapper를 사용한 예제 회로

논문에서 제안된 wrapper를 실제 GALS 시스템에 적용하여 올바른 동작을 수행하는지를 확인하기 위해서 먼저 동기식의 회로를 설계하였다. 이 동기식 회로로부터 GALS 시스템의 센더와 리시버를 구성하기 위해, 데이터 패스와 제어 패스 부분을 분리시킨 후, wrapper를 장착하였다.

예제로 선택한 회로는 순차적으로 들어오는 입력 비트를 검사하여 "00*1"이나, "11*0"의 입력 시퀀스발견 시에, 각 비트에 상응하는 32비트 데이터를 누적 덧셈 후, 일정 값을 곱하는 연산을 수행한다. <그림 18>은 예제 회로의 블록 다이어그램을 나타낸다. 크게 덧셈기와 곱셈기 두개의 연산 블록과, 내부 데이터 흐름을 제어하기 위한 FSM 블록으로 나뉜다.

<그림 19>는 예제 회로에 사용된 Mealy 형식의 FSM을 나타낸다. 입력 데이터에 따른 상태 천이과정과 각 상태에서의 동작을 설명하고 있으며, 각 상태에서 데이터 패스를 제어하는 신호들을 다르게 발생시킨다. S3과 S4에서 S5로 상태 천이가 이루어질 때 가장 긴 연산 시간을 소모하므로, 클록의 주기는 이때의 시간으로 결정된다. 즉, 입력 데이터를 받고 덧셈기와 곱셈기를 거친 후 최종 출력 값을 발생시키는 시간이 클록의 한 싸이클로 정의된다.

GALS 시스템을 구성하기 위해, 센더는 덧셈기로, 리시버는 곱셈기로 LS 모듈들을 선택하였다. 또한 각각의 모듈에서 덧셈기와 곱셈기를 제어하기 위한 지역화된 FSM도 원래 동기 회로의 FSM으로부터 분리하여 재 설계하였다. 각 LS 모듈의 클록 주기는 센더의 경우는 덧셈기의 연산 시간으로 결정되고, 리시버의 경우는 곱셈기의 최대 지연시간으로 결정 할 수 있었다.

지역화된 데이터 패스와 제어 패스로 구성된 LS 모듈과 wrapper와의 인터페이스 방법을 <그림 20>에 나타내었다. 센더의 경우, 입력 데이터 비트를 검사한 결과 "00*1"이나 "11*0"와 매치되었을 때 그때까지 누적 덧셈 값을 출력하기 위한 제어신호를 사용하여 wrapper에서 필요한 en 신호를 재생성시켰다. 또한, 최종 데이터를 출력하기 위해 data_clk 신호로 래치를 제어했다. 마찬가지로 리시버의 경우도 센더와 같이, 연산이 종료된 상태에서 발생하는 FSM의 제어 신호로 en 신호를 구동하였다.

시뮬레이션 결과, 동기식 방법에 의해 설계된 회로의 동작과 wrapper를 장착한 GALS 형태의 회로 동작이

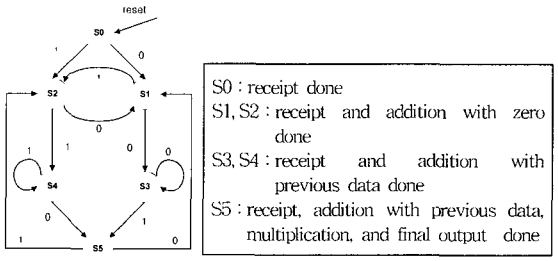


그림 19. 동기식 예제 회로의 FSM
Fig. 19. FSM for synchronous example circuit.

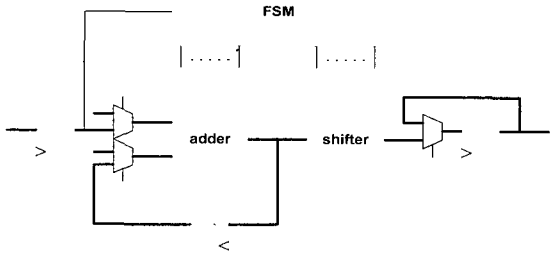


그림 18. 동기식 예제 회로의 블록도
Fig. 18. Block diagram of synchronous example circuit

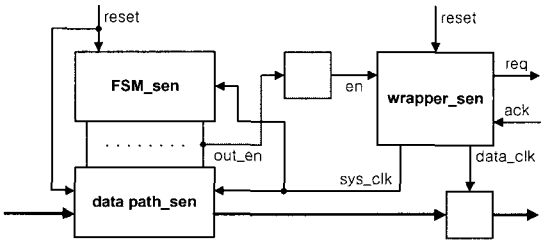


그림 20. wrapper를 장착한 센더 모듈
Fig. 20. Sender module with wrapper.

서로 일치함을 확인하였다. 현재, 좀더 실제적인 애플리케이션에 적용하기 위하여 스마트카드용 보안 칩을 GALS 구조로 설계하는 연구를 수행중이다.

Ⅸ. 결 론

본 논문에서는, 대규모의 칩 설계 시 발생할 수 있는 문제점들을 구조적으로 해결할 수 있는 방안으로 제시되고 있는 GALS 시스템에서, 전체 시스템 동작의 신뢰성와 안정성을 보장하고, 성능 향상을 도모할 수 있는 형태의 비동기식 접속장치 회로를 제안하였다. 전형적인 데이터 전송 프로토콜 방식에 따라 데이터 전송시에 내부 클록을 정지시키는 구조와, 데이터 전송과 내부 클록을 부분적으로 병렬 수행할 수 있는 구조의 두가지 형태로 설계 하였다. 모두 stoppable 클록에 기반하고 있으며 기본적으로, 표준 셀 라이브러리로 설계 가능하다.

0.25㎛공정의 게이트 레벨 시뮬레이션을 통한 성능 평가 결과, one-to-one 구조에서 병렬 수행 가능한 형태의 접속장치가 내부 클록을 항상 정지시키는 구조에 비해 약 5%에서 12%까지 향상됨을 알 수 있었고, FIFO를 추가한 시뮬레이션에서도 FIFO를 장착한 같은 성능의 전형적인 방식에 비해서 면적 측면에서 유리함을 알 수 있었다. 또한, 사용될 접속장치의 선택은 적용할 애플리케이션에 따라서 달라 질 수 있음을 시뮬레이션을 통하여 추론할 수 있었다. 마지막으로 간단한 애플레케이션 회로에 적용하여 설계된 접속장치의 실제 사용 예를 제시하였다.

참 고 문 헌

[1] International Technology Roadmap for Semiconductors, Semiconductor Industry Association, 1999.

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

[3] J. N. Seizvic, "Pipeline synchronization," in Proc. International Symposium on Advanced Research in Asynchronous circuits and Systems, pp. 87~96, Jun. 1996.

[4] M. Pechoucek, "Anomalous Response Times of Input Synchronizers," IEEE Trans. Comput., vol. C-25, no. 2 , Feb. 1976.

[5] D. M. Chapiro, "Globally-Asynchronous Locally-Synchronous Systems," Ph.D. thesis, Stanford University, Oct. 1984.

[6] C. Mead and L. Conway, Introduction to VLSI Systems, Addison-Wesley, 1980.

[7] VanScheik, W.S. and Tinder, R.F., "High speed externally asynchronous/internally clocked systems," IEEE Trans. Comput., vol. 46, no. 7, pp. 824~829, Jul. 1997.

[8] K. Y. Yun and A. E. Dooply, "Pausible clocking based heterogeneous systems," IEEE Trans. VLSI Systems, vol. 7, no. 4, pp. 482~487, Dec. 1999.

[9] Jens Muttersbach, et al., "Practical Design of Globally-Asynchronous Locally-Synchronous Systems," in Proc. International Symposium on Asynchronous circuits and Systems, pp. 52~59, Apr. 2000.

[10] S. W. Moore et. al., "Self Calibrating Clocks for Globally Asynchronous Locally Synchronous Systems," in Proc. International Conf. Computer Design, pp. 73~78, Sep. 2000.

[11] Furber, S.B. and Day, P., "Four-phase micro pipeline latch control circuits," IEEE Trans. VLSI Systems, vol. 4, no. 2, pp. 247~253, Jun. 1996.

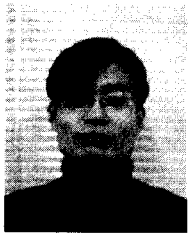
[12] K. Y. Yun, "Synthesis of Asynchronous Controllers for Heterogeneous Systems," Ph.D. thesis, Stanford University, Aug. 1994.

[13] IDEC Cell Library Data Book : IC Design Education Center, Jun. 2000.

저 자 소 개

吳 明 勳(學生會員) 第38卷 SD編 第6號 參照

崔 淏 鎔(正會員) 第40卷 SD編 第3號 參照
현재 : 충북대학교 전기전자컴퓨터공학부 교수



朴 錫 在(學生會員)
2002년 2월 : 충북대학교 전기전자공학부 졸업(공학사). 2003년 3월~현재 : 충북대학교 반도체공학 석사과정. <주관심분야 : VLSI설계, GALS system설계>

李 東 翊(正會員) 第38卷 SD編 第6號 參照
현재 : 광주과학기술원 정보통신공학과 교수