논문 01-26-7B-19 한국통신학회논문지 '01-7 Vol.26 No.7B

Safe Mode를 갖는 동기 클럭 발생 회로의 ASIC 구현

정회원 최 진 호*, 강 호 용**, 전 문 석***

ASIC Implementation of Synchronization Circuit with Safe Mode

Jin-ho Choi*, Ho-Yong Kang**, Moon-Seog Jun*** *Regular Members*

요 약

본 논문에서는 다른 클럭원들을 갖는 서로 다른 오실레이터에 의해 발생된 비동기 클럭을 입력으로 받아 동기 신호로 변환시키는 기능과 그 중 어느 한 클럭이 동작하지 않더라도 동작하는 클럭을 계속 유지하여 클럭 중단의 위험을 제거한 안전모드를 추가한 기능의 구현을 기술한다. 특히, 통신 분야에서 ASIC으로 Chip을 개발할 때 다중 클럭의 사용은 필연적이며 비동기 신호를 동기신호로 변환하는 기능의 구현은 기본적이면도 중요한 부분이다. 이 회로는 VHDL로 구현이 되었으며 다중 클럭 관련 ASIC구현에 기본적으로 응용이 가능하다.

ABSTRACT

In this paper, we describe the ASIC implementation of synchronization circuit, which has asynchronized inputs of different clock sources and has a synchronized output by multiplexing according to a based clock of input clocks. Additionally, this circuit supports the safe mode in which the clock output can operate without a stop even if the clock to be changed is not run. Especially this circuit is very important for the communication area using multiple clocks. This circuit is implemented by VHDL and available to the ASIC implementations related to multiple clocks.

Ⅰ. 서 론

통신시스템, 마이크로 프로세서등의 디지털회로를 설계하는데 다중클럭원을 갖는 설계가 불가피한 경우가 많다. 이런 경우에 신호나 클럭의 비동기로 인해 글리치(glitch)가 발생하던지 setup 또는 hold 위반(violation)에 의해 Unknown값이 발생이 되어 예상치 못한 결과가 생성된다. ASIC등 디지털회로 설계에 있어서 비동기 신호의 동기화는 무척이나 중요한 부분이며 비동기로 인한 에러를 방지하는 가장 기본적인 방법이 된다. 일반적인 경우는 클럭의 발생상황을 미리 알고 글리치나 타이밍위반을 피하도록 설계를 하고 있으나 본 논문에서 제안한 방법들은 클럭의 발생상황을 고려하지 않아도 된다. 그림 1에 글리치를 고려하지 않은 기본적인 클럭 스위칭 회로를 보여주며, 그림 2에는 이에 대한 타이밍도를 보여주며 비동기시에 선택된 출력 클럭에 글리치가 발생함을 보여준다. 이러한 글리치들은 회로내부에서 원하지 않는 출력을 발생시키는 원인이 될 수 있다.

본 논문에서는 비동기 클럭이나 신호를 타이밍 위반이나 글리치를 제거하고 안전모드를 갖는 클럭 다중화 회로의 ASIC구현에 따르는 구조를 제안 설명한다. 여기에서 안전 모드란 변환하고자 하는 클럭이 동작을 하지 않을 때에는 select신호가 동작하지 않는 클럭으로 변환하도록 활성화된다 해도 기존의 동작하는 클럭을 계속 출력하여 클럭의 연속성을 보장하도록 하는 것이다.

II장에서는 종전의 특허 기술인 동기신호 출력장치[1]와 비동기 클럭 신호 다중화 장치에 대하여 설

* LG전자 디지털미디어연구소 선임연구원(jinhochoi@lge.com),
** 한국전자통신연구원 네트워크기술연구소 선임연구원(hoyong.kang@etri.re.kr),
*** 숭실대학교 컴퓨터학과 통신연구실(mjun@computing.ssu.ac.kr)
논문번호 : 010004-0216, 접수일자 : 2001년 2월 16일

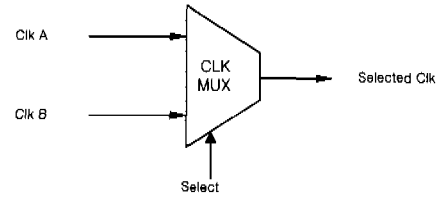


그림 1. 일반적인 클럭 다중화 회로

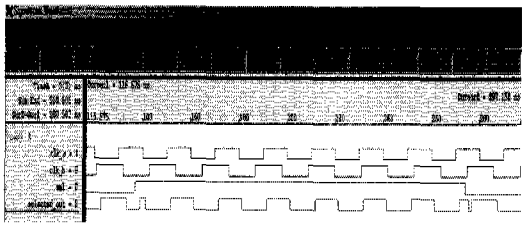


그림 2. 글리치를 갖는 클럭 다중화 회로의 Gate Simulation 타이밍도

명하고[2], III장에서는 안전 모드를 가지고 비동기 신호를 기준 클럭에 동기 시켜 출력하는 회로의 구현을 설명하고 결론을 맺는다. 참고 문헌 또한 각 ASIC Library회사에서 코어(core)회로에 해당하는 부분이라 이론을 설명한 논문은 있으나[5] 구현 구조를 가지고 실제적으로 특허나 논문으로 공개된 것이 거의 없는 실정이며 기존에 사용되는 클럭 다중화회로는 PLL을 이용하여 글리치를 제거하여 다중화를 시키므로[3],[4] 회로의 복잡성이 많은 반면 본 논문에서는 PLL을 사용하지 않은, 로직으로만 구성한 구현 구조를 제시하였다. 또한 종래 기술로서 [1], [2], [6]을 제시하였는데 [1]과 [2]는 다음 II장에서 종래 기술로 설명을 할 것이다.

참고문헌 [6]의 Computer Systems에 대한 클럭 스위칭 구조에 대한 특허기술은 3개의 Mux들과 2개의 Edge Detector들 그리고 하나의 래치(Latch), 하나의 F/F을 사용하여 구현을 하였는데 본 논문에서는 이 구조와 다른 구조를 보여주므로 차별화를 두었다. 본 논문의 구조와 다른 점들은 우선, Edge detector를 사용하여, 입력되는 클럭의 Edge를 교차적으로 래치를 하여 보호구간을 만들어주는데 이때 글리치와 같은 Edge 신호들이 만들어 진다. 이러한 Edge 신호들은 ASIC Design시에 Timing 위반 발생의 원인이 된다. 즉, ASIC Design을 고려하지 않은 Schematic구현의 구조가 될 것이다. 두 번째는, Select 신호가 자유스럽지 못하다는 점이다. 즉, 현재 출력되는 클럭의 high구간과 동기가 되어야만 글리치가 발생하지 않는 구조가 된다. 이 기술은 Computer System에서 CPU, Memory등 주변 장치와의 인터페이스하는 버스와 연결되어 사용되는데 클럭과 Select신호의 발생 관계를 예측가능 한 상황에서 설계를 한 구조가 될 수 있다. 마지막으로, 본 논문은 안전모드의 기능을 갖는 구조에서도 차별을 두고 있다. 그림 3에 종래 구조의 블록도를 보여준다.

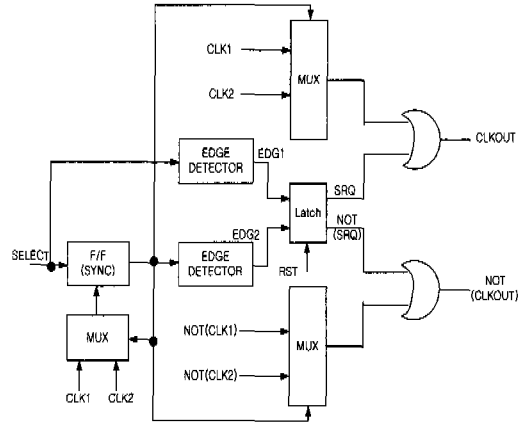


그림 3. Computer Systems을 위한 클럭 스위칭의 종래 구조[6]

본 논문에서 보여주는 Timing Simulation결과는 2가지 Tool을 사용하는데 VHDL Code를 기능검증(RTL Level Simulation)하기 위해 Mento사의 시뮬레이터인 Modelsim 을 사용하였고 합성 후에 Netlist를 검증(Gate Level)하기 위해 Cadence사의 시뮬레이터인 Verilog-XL과 Signalscan을 사용하였다.

Ⅱ. 비동기 신호 출력회로와 클럭 다중화 회로의 종래 기술들

2.1 비동기 신호 출력 회로

이 장에서는 비동기 신호를 기준이 되는 특정 클럭에 동기시켜 출력을 하는 회로에 대하여 설명한다. 일반적으로 송신단에서 보내준 송신신호는 수신단에서 수신단의 클럭과 동기되어야만 원하는 신호를 왜곡없이 출력할 수 있으므로 클럭 제어분야에서 가장 기본적인 부분이 비동기신호를 동기신호로 변환하여 안정적인 동기신호를 제공하는 것이다. 이러한 동기신호 변환 회로의 구성은 그림 4와 같고 하나의 래치(latch)와 하나의 멀티플렉서 그리고 AND 게이트로 구성된다.

래치는 멀티플렉서의 출력된 값을 피드백(feed-back)신호로 받아서 유지시켜 주는 역할을 하게 된

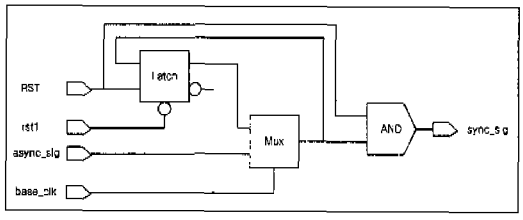


그림 4. 동기신호 출력회로 다이어그램[1]

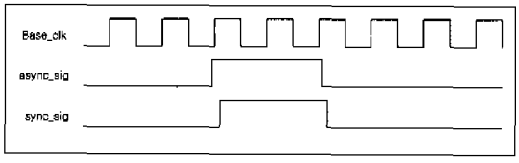


그림 5. 동기신호 출력회로의 타이밍 다이어그램[1]

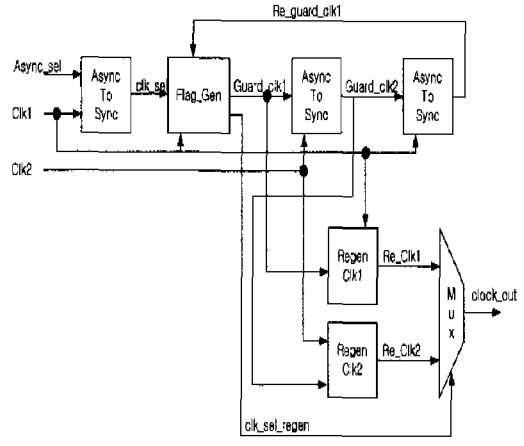


그림 6. 비동기 클럭 다중화 블록도[2]

다. 즉, 멀티플렉서의 선택신호인 base_clk이 low일 때에는 멀티플렉서의 이전 출력값을 계속 유지를 시켜주도록 하는 역할을 하며 또한 ASIC설계의 합성시에 타이밍 루프를 없애기 위한 소자로도 사용이 된다. 멀티플렉서는 전술한 바와 같이 base_clk을 선택신호로 받아 active high일때에는 async_sig 신호를 출력하고 low일때는 래치로부터 인가되는 피드백신호를 출력하게 된다. 즉, active high일때, 래치된 async_sig신호는 low구간동안에는 변화되지 않고 멀티플렉서의 출력으로 보장이 되므로 글리치가 발생이 안되고 unknown값도 발생이 되지않는 것이다. AND 게이트는 RST 신호가 인가되었을 때, sync_sig값을 초기화 시키는 역할을 한다.[1]

여기에서 시스템 리셋신호인 RST는 active low이며 rst1은 RST의 하나의 버퍼 지연을 갖는 신호이다. 이는 최종출력인 sync_sig값이 처음 초기화시에 글리치가 발생이 안되도록 타이밍을 조정하기 위한 신호이다. RST신호가 high로 되면 그때부터 정상적인 회로동작이 이루어 진다.

2.2 비동기 클럭 다중화 회로

이번 절에서는 1절에서 설명한 비동기 신호를 동기신호로 변환하는 ATS(AsyncToSync) 블록을 이용한 클럭 다중화 회로를 설명한다.

그림 6의 비동기 클럭 다중화회로에 대한 타이밍 다이어그램은 III장의 안전모드 구현시 타이밍도에 포함되므로 생략하였다. 이 회로는 서로 다른 클럭들, 즉 clk1과 clk2를 가지며 두 클럭의 변환 선택을 위하여 Async_sel입력을 갖는다. 여기에서 Async_sel이라 부르는 이유는 이 선택신호 역시 clk1과 clk2의 어느 쪽과도 동기가 안된 다른 클럭원으로부터의 입력이 될 수도 있거나 버퍼나 조합논리회로의 지연으로 인한 비동기적인 입력을 의미한다. 여기에서는 기준이 되는 클럭은 clk1으로 정하였고 clk1을 중심으로 다중화가 이루어짐을 알 수가 있다. 우선 비동기 선택신호인 Async_sel신호는 첫번째 ATS블록을 통해 clk1에 동기된 clk_sel로 변환되어 clk1에 동기된 블록인 FLAG_GEN 블록으로 입력된다.

FLAG_GEN 블록은 2bit의 FSM(Finite State Machine)으로 이루어져 있으며 초기단계 "00"에서는 현재의 Re_guard_clk1신호와 새로운 clk_sel 입력과 비교를 하게 된다. 즉, 기존의 신호가 clk1을 선택하고있다고 가정을 할 때, 새로운 clk_sel신호가 clk2를 선택하도록 활성화된다면 "00" state에서는 이를 탐지하여 state "01"로 전이를 한다. 동시에 clk1을 보호하는 guard구간을 만들기 위한 신호인 guard_clk1신호를 active low로 발생시킨다. 여기에서 보호신호구간이란 변환하고자 하는 두 클럭이 변환하는 시점에서 안정된 변환을 위해 강제로 high로 유지시키는 구간을 말한다.

Clk1의 보호 구간에는 clk1을 high상태를 만들고 clk2의 보호 구간에는 clk2를 high상태로 만들어 놓은 후 이때 변환 신호인 clk_sel_regen을 발생시켜 안정한 clock_out을 출력한다는 것이다. FLAG_GEN에서는 state는 "00"에서 "01"로 전이된 후에 자동적으로 다음 클럭에는 "10"으로 전이시켜 최소한 두 clk1 주기의 보호구간을 보장하여준다. 즉, Guard_clk1은 두 구간이상 active low를 유지할 것이다.

Clk1의 보호 신호인 Guard_clk1이 발생되면 이

신호를 가지고 clk2에 동기된 Guard_clk2를 발생시켜야 한다. 그러므로 두 번째 ATS를 사용하여 clk2에 동기된 Guard_clk2신호를 발생하게 된다. 두 번째 ATS에서 발생된 Guard_clk2신호가 active low로 되었다는 것은 clk1과 clk2 모두에 대한 보호 구간이 만들어 졌음을 의미하는 것이다. 그러므로 이러한 사실을 clk1으로 동작하는 FLAG_GEN 블록에 알려주어야 하므로 Guard_clk2신호를 clk1에 동기된 Re_guard_clk1신호로 변환을 시켜주기 위하여 세 번째 ATS블록을 사용한다. 이 블록을 통해 재생성된 Re_guard_clk1신호를 FLAG_GEN블록에서 받아 홀딩하고 있는 새로운 sync_sel신호를 clk_sel_regen로 바꾸어 전달을 하게 된다. 이때 state는 "10"에서 "00"으로 되돌아가서 초기상태가 되며 다음 clk_sel이 변화되기를 기다리는 "준비상태(ready state)"가 될 것이다.

최종 블록인 Regen_Clk1과 Regen_Clk2 블록들은 평상시에는 Clk1과 Clk2를 Pass하는 기능만을 수행하다가 각각의 Guard_clk신호가 입력되면 그 구간에서는 자신들의 clk들을 high상태로 만들어 주는 기능을 수행한다. 두 Guard_clk신호 모두 각기 자신의 clk에 동기되어 입력되므로 글리치의 발생이 생기지 않는다. 이렇게 재 발생된 클럭들은 각각 re_clk1과 re_clk2로 변환되어 FLAG_GEN블록으로부터 안전구간 내에 발생된 clock_sel_regen신호에 맞추어 clock_out으로 스위칭되어 출력된다.[2]

Ⅲ. 안전 모드를 갖는 비동기 클럭 다중화 회로

2장에서 기존의 비동기 신호를 동기신호로 변환하는 기술과 비동기 클럭을 동기 클럭으로 변환하는 종래의 기술을 살펴보았다. 그러나 전술한 비동기 클럭 다중화 회로에는 두 가지 단점을 가지고 있다.

첫번째는 기준 클럭인 clk1보다 clk2의 주기가 많이 차이가 날 때 출력 clock_out이 high로 오랫동안 유지한다는 점이다. 예를 들어, clk2가 clk1보다 긴 주기를 갖는다면 clk1에 의해 guard_clk1을 active low로 만들어 보호구간을 만들고 나서 clk2의 주기가 바뀔 때까지, 그리고 Guard_clk2신호가 발생이 될 때까지 clk1의 보호구간을 유지할텐데 이 경우 clock_out이 장기간 high로 유지되는 결과를 초래하게 된다. 이 또한 장기간동안의 클럭의 부재를 뜻하는 것이므로 좋은 스위칭 방법이라고는 할 수 없다. 그림 7은 종전의 비동기회로인 그림 6의 단점을 극복한 타이밍도를 보여준다. 그림 6의 Guard_clk1은 그림 8의 guard1과 같고 Guard_clk2는 guard2와 같다. 그림 7에서 볼 수 있듯이 이전의 구조는 clk_sel이 입력되고 delay_cnt의 상태가 00에서 01로 전환되면서 guard1이 active low로 떨어지면 이 때부터 re_clk_a와 clock_out이 high를 유지하며 guard2가 active low 구간까지 지속된다[2]. 그러나 본 논문의 제안한 구조는 그림 7과 같이 clk_sel이 인가된 후 clk1의 두 주기 동안 clk2의 전이가 발생하지 않을 경우에는 clk2가 2주기 이상 큰 클럭이라 가정하고 state를 11로 옮겨 이미 발생시킨 clk1의 보호구간인 guard1을 다시 high로 복귀한다. 그리고 clk2의 event가 발생할 때까지 기다렸다가 유효한 guard1을 다시 발생시켜 두 번째 guard1의 구간동안만 high로 보호구간을 허용한다.

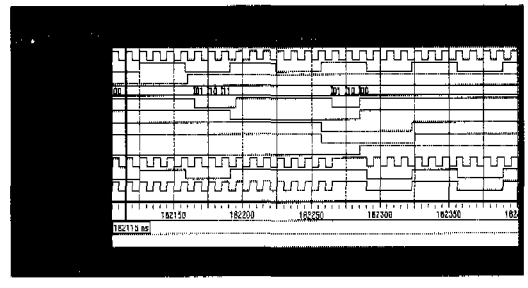


그림 7. 제안한 비동기 다중화 회로의 타이밍 다이어그램 (clk2 >> clk1)

두 번째는 현 상태가 clk_sel이 low가 되어 clk1이 clock_out을 통해 출력될 때 clk_sel이 high가 되어 clk2로의 변환 신호가 들어 왔다고 가정하자. 이때 clk2가 동작을 하지 않아 Low나 high로 고정이 되어있다면 이전 회로[2]에서는 clock_out이 동작하지 않는 clk2로의 변환을 가지고 high또는 low의 값을 가지게 될 것이다. 이것은 클럭 발생의 중단을 의미하며 적어도 클럭이 중단되지 말아야 하는 중요한 회로에서는 치명적인 시스템 결함이 발생할 것이다. 이러한 단점을 해결하고자 본 논문에서는 안전 모드를 추가한 비동기 클럭 다중화 회로를 제안한다.

clk2가 동작하지 않는다 해도 clk1으로 계속 clock_out을 유지해주면서 clk2가 복원되기만을 기다린다. 결국 clk2가 복원되어 정상적으로 동작을 한다면 자동적으로 clk2로 변환이 발생될 것이다. 그림 8과 9에 제안한 회로의 블록 다이어그램과 타이밍도를 보여준다.

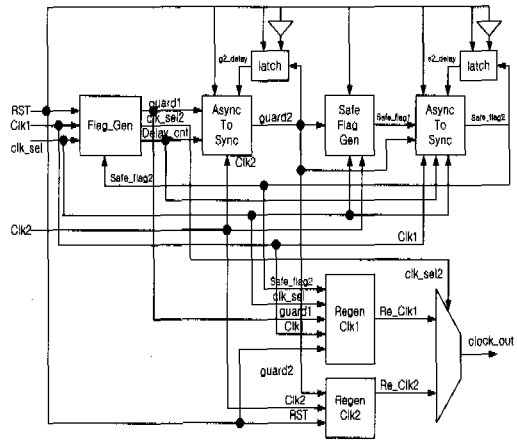


그림 8. 안전모드를 갖는 비동기 클럭 다중화 블록도

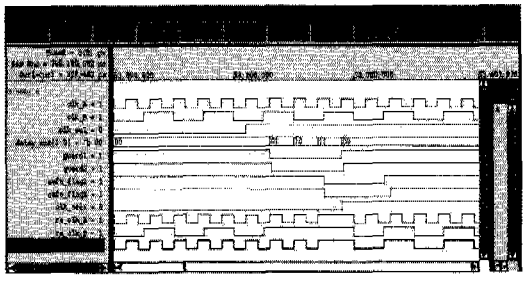


그림 9. 정상모드에서 클럭 다중화 Gate Level 타이밍도

블록도와 타이밍도를 통해 구현 기능을 설명을 한다면 그림 8의 안전모드를 갖는 비동기 클럭 다중화회로는 Flag_Gen블록, ATS블록, Safe_Flag_Gen 블록, Regen_Clk1, Regen_Clk2, 멀티플렉서 블록으로 구성된다. 전체에서 사용된 Cell들의 수는 Flip-Flop 5개, Latch 2개 그리고 논리 조합회로로 구성된다. 여기에서 사용되는 FSM의 state는 delay_cnt[1:0]과 같다.

그림 9와 같이 Flag_Gen Block의 기능은 clk_sel 신호가 바뀌면 클럭 변환을 알리는 flag신호들을 발생시킨다. 즉, guard신호를 active low로 떨어뜨리며 delay_cnt[1:0]의 상태를 01 로 전이하면서 클럭변환이 시작되도록 한다. 이 블록은 4가지의 FSM으로 동작을 하는데 각 state의 기능은 다음과 같다.

State "00" (dealy_cnt[1:0] = "00") : clk_sel 입력을 기다린다. 이때 이전의 clk_sel 신호인 clk_sel2신호와 값이 다르면, 즉 클럭변환 신호가 입력되면 guard1신호를 active low로 발생시키며 state를 "01"로 전이한다. 그렇지 않고 클럭변환 신호가 없으면, 즉 clk_sel과 clk_sel2신호가 같으면 계속 "00" state를 유지한다.

State "01" : 안정된 동작을 위하여 guard구간을 최소 두 클럭을 유지해야 하는데 이런 클럭의 시간 여유를 두기위해 state "01"에서 state "10"으로 자동 전이한다. 즉, Guard구간을 최소 2 클럭을 유지하는 것이다.

State "10" : 여기에서는 세가지 조건으로 동작을 하는데, 우선 clk_sel이 high인 상태에서 safe_flag2가 Low이면 clk_sel2에다 clk_sel값을 전달하고 guard1신호를 active high로 환원하고 다시 state "00"으로 전이하여 초기상태로 간다. 이 조건에서 clk_sel이 high라 함은 clk2로의 변환을 알리는 것이고 safe_flag2가 low라 함은 현재 clk2가 동작중임을 알리는 것이다. 그렇지 않고 clk_sel이 Low이면 clk1으로 변환을 의미하는데, 이 블록자체가 clk1으로 동작하므로 clk1으로의 전환은 동기에 문제가 발생하지 않음을 뜻한다. 그래서 clk_sel2를 Low로 변환(clk1선택)시키고 초기상태 "00"으로 전이한다. 마지막으로 두 조건이외의 조건일 경우에는 clk2로 변환되어야 하나 clk2가 동작을 하지 않는 상태(safe_flag2가 active가 안된 상태)이므로 clk_sel2는 Low(clk1선택)를 그대로 유지하며 state "11"로 전이한다.

State "11" : 여기에서는 clk_sel이 high 상태일 것이고 safe_flag2가 active low가 되기를 기다린다. 즉 clk2가 동작하기를 기다리는 것이다. 이 state에서 Clk2가 동작을 안 한다면 이 state는 clk_sel이 다시 low로 바뀌거나 clk2가 동작될 때까지 계속 유지되며 clk1을 최종출력으로 보낼 것이다. 이는 clk_sel이 high로 바뀌어 변환 요청을 한다해도 clk2가 동작을 하지 않으면 최종 Mux의 선택신호 clk_sel2는 계속 Low를 유지하고 clk1을 출력하는 것이다. 중간에 clk2가 동작한다면 safe_flag2가 active low가 될 것이며 이때 clk_sel이 high이면 clk2로의 변환을 해야 하는데, guard1상태가 low상태를 유지하고 있다면 clk_sel2를 high로 바꾸고 guard1을 high로 바꾸며 state를 초기상태 "00"으로 전이한다. 그러나 guard1이 high이면 변환과정을 다시 시작하기 위해 guard1을 active low로 전환하고 state를 "01"로 전이한다. 이러한 안전모드에서의 타이밍도는 그림 10에 설명된다.

계속해서 그림 8의 구조에 따르는 나머지 블록들을 설명하면 다음과 같고 각 블록을 설명할 때의 타이밍도는 그림 7, 그림 9, 그림 10을 참조할 수 있다. 첫번째 ATS블록은 II장에서 설명한 바와 같이 종전기술과 같은 동작을 한다. 즉, clk1에 동기

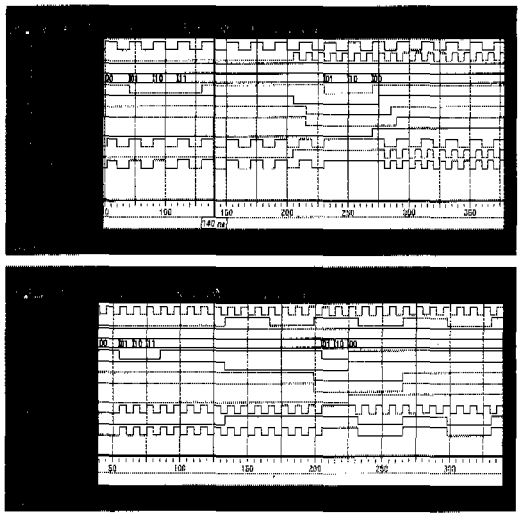


그림 10. 안전모드에서의 클럭 변환 타이밍도

된 guard1신호를 받아 clk2에 동기된 Guard2신호를 발생시키기 위한 블록이다.

Safe_Flag_Gen 블록은 clk2가 동작하는지의 여부를 알려주는 블록이므로 clk2로 구동된다. 즉, RST이 인가되면 safe_flag값은 high로 초기화되고, 이후 clk2의 riging edge에서 guard2신호가 active low이거나 guard2신호가 active low이면서 clk_sel이 high이면 safe_flag를 active low를 발생시킨다. 그렇지 않으면 high이다. 즉, safe_flag는 clk2를 클럭원으로 사용하는 F/F이므로 클럭 변환 신호가 인가 되고 guard2신호가 active low일때 발생되어 clk2의 동작을 알려주는 것이다. clk2가 동작하지 않는다면 safe_flag신호의 발생은 불가능할 것이다. 두 번째 ATS블록 역시 종전기술과 같은 동작을 한다. 즉, clk2에 동기된 safe_flag신호를 받아 clk1에 동기된 safe_flag2신호를 발생하여 safe_mode의 상황을 Flag_Gen 블록(clk1으로 구동)으로 전달해주어 clk_sel2를 안전하게 출력하도록 한다.

Regen_clk2블록은 clk2를 Re_clk2로 출력을 하는데, 평상시에는 clk2신호를 받아 전달하는 버퍼기능(Re_clk2 = clk2)을 하지만 클럭 변환시에 guard2가 active low되는 보호구간동안 high를 유지시켜 출력(Re_clk2 = high)하는 기능을 갖는다. 타이밍 동기를 위해 clk2의 high구간에 high 유지 구간을 시작하며 유지한다. 동작 설명은 clk2가 high일때 guard2가 클럭변환을 위해 active low이면 clk2를 high로 만들지만 guard2가 high면 그냥 high로 pass시킨다. 즉, guard2가 active low일때 clk2가 high구간에서는 항상 Re_clk2를 high로 고정시키는 것을 시작하는 것이다. Clk2가 low일때는 guard2가 high이면 clk2를 low로 pass시키지만 active low이면 clk2를 high로 만들어준다. 결론적으로 guard2가 active low인 구간에서 Re_clk2를 항상 high로 고정시켜놓아 다음 블록인 Mux블록에서 안전하게 클럭변환이 일어나게 한다. Regen_Clk1블록의 설명은 Regen_clk2 발생과 유사하며 다음과 같다. Clk1이 high일때, guard1이 active low라면 Re_clk1을 high로 Pass시키고, low가 아니라도 high로 pass시킨다. 그러나 clk1이 low일때는, guard1이 active low이면서 동시에 Safe_flag2가 low이거나 clk_sel이 low일때의 조건이 만족하면 Re_clk1을 high로 고정시킨다. 그 외의 조건이면 low를 pass한다.

마지막으로 본 논문에서 제안한 안전모드를 갖는 비동기 클럭 다중화 회로의 Synopsys Tool에 의한 합성그림을 그림 11에 보여준다.

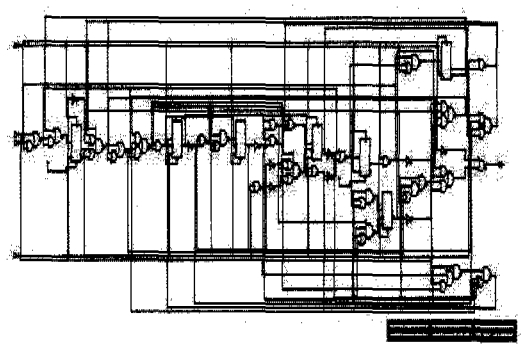


그림 11. 안전모드를 갖는 비동기 클럭 다중화 회로의 합성 결과

Ⅳ. 결 론

본 논문에서는 실제적으로 ASIC 설계에 응용될 수 있도록 비동기 신호의 동기 변환 블록을 이용하여 안전모드를 추가한 비동기 클럭 다중화 회로를 제안하였다. 제안된 회로는 종래 기술에서 단점이 될 수 있는 clk1과 장 주기를 갖는 clk2를 스위칭할 때, 다중화된 클럭이 high 상태를 길게 유지함을 제거하였으며 변환될 클럭이 동작을 하지 않더라도 최종 클럭 출력이 중단되는 위험을 막기 위하여 안전모드 기능을 추가하여 설계를 하였다. 나아가 중단된 클럭이 복구되어 다시 동작을 했을 때 자동으로 탐지하여 그 클럭으로 변환해 줄 수 있는 기능을 제공하여 클럭의 연속성을 유지 시켜주는데 아주 유용한 구조가 될 수 있다. 이 회로의 구현은

VHDL로 구현되었으며 Synopsys tool을 이용하여 합성되었고 0.35um공정으로 현재 16bit 마이크로 콘트롤러 제품에 적용되어지고 있다.

참 고 문 헌

[1] 최진호, "동기신호 출력장치", 대한민국 특허청, 공개특허공보, 공개번호 특2000-0044169, 공개일자 2000년 7월 15일.

[2] 최진호, "비동기 클럭신호 다중화장치", 대한민국 특허청, 공개특허공보, 공개번호 특2000-0044170, 공개일자 2000년 7월 15일.

[3] Sung-Sik Hwang, Dual-loop DLL-based Clock Synchronizer, *Electronics Letters*, Vol.36 No.14 July 2000.

[4] Klaus Schossmaier and Dietmar Loy, "An ASIC Supporting External Clock Synchronization for Distributed Real-Time System," Proceedings of *the 8th Euromicro Workshop on Real-Time System*, L'Aquila, Italy, June 12-14, 1996

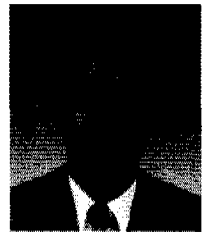
[5] Boaz Patt, "A Theory of Clock Synchronization," Massachusetts Institute of Technology, October 1994.

[6] Philip A, Ferolito and Sunnyvale, "Clock Switching Apparatus and Method For Computer Systems", United States Patent, Patent Number 5274678, Date of Patent Dec. 28, 1993.

최 진 호(Jin-Ho Choi) 정회원

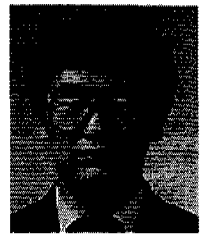
1992년 2월 : 숭실대학교 전자계산학과 졸업(학사)
1994년 2월 : 숭실대학교 컴퓨터학과 졸업(공학석사)
1994년~1999년 : 대우전자 반도체연구소 주임연구원
1999년~현재 : LG전자 디지털미디어연구소 선임연구원
<주관심 분야> 정보통신, 디지털 미디어 ASIC Design

강 호 용(Ho-Yong Kang) 정회원



1989년 2월 : 부산대학교 전자공학과 졸업
2001년 3월~현재 : 충남대학교 정보통신공학과 석사과정
1988년 12월~1993년 12월 : 대우통신 반도체연구소 주임연구원
1994년 1월~2000년 5월 : 대우전자 ASIC Center 선임연구원
2000년~현재 : 한국전자통신연구원 전송핵심모듈팀 선임연구원
<주관심 분야> 광가입자망, 기가비트 이더넷, VLSI 설계

전 문 석(Moon-Seog Jun) 정회원



1980년 : 숭실대학교 전자계산학과 졸업(학사)
1986년 : University of Maryland 전산과 졸업(석사)
1989년 : University of Maryland 전산과 졸업(박사)
1989년 : Morgan State University 전산수학과 조교수
1989년~1991년 : New Mexico State University 부설 Physical Science Lab. 책임연구원
1991년~현재 : 숭실대학교 정보과학대학 부교수
<주관심 분야> 컴퓨터 알고리즘, 병렬처리, VLSI 설계, 암호학