비동기 회로 및 시스템 설계

Asynchronous Circuit and System Design

박영수(Y. S. Park) 자동설계연구실 연구원
박인학(I. H. Park) 자동설계연구실 선임연구원, 실장

전역 클럭을 사용하는 동기 회로 설계 기술은 설계의 단순화 및 자동화가 용이하기 때문에 현재 많이 사용하는 설계 기술이다. 그러나 다양한 기능과 고성능을 필요로 하는 대규모 시스템이나 회로 설계에서는 전역 클럭 사용으로 인한 신호 지연, 전력 소모 등이 문제로 부각되면서 비동기 회로 설계 기술이 각광을 받고 있다. 비동기 회로 설계 기술은 1940년대에 개발된 기술이지만 설계 자체가 어렵고 면적 증가 등의 단점으로 제한된 분야에서 이용되었다. 현재 이러한 단점을 극복하기 위한 연구가 회로 설계, 검증, 동기/비동기 인터페이스, 그리고 저전력 회로 등의 분야에서 많이 진행되고 있다.

I. 서 론

동기 및 비동기 논리의 차이점은 동기 논리의 경우 클럭을 사용하여 연속적인 시스템의 상태를 분리하며 특정 시간 또는 기간 동안에 신호의 값을 유지하지만, 비동기 논리는 시스템의 상태를 입력된 값이나 내부의 동작으로 표시하고 어떤 사건(event)이 발생된 후 또는 다른 사건까지 신호의 값을 유지시키는 점이다.

비동기 논리의 사용은 이미 1940년대 비동기로 설계된 프로세서가 있었으며, 1950년대에는 비동기 논리에 대한 두 가지 중요한 이론적 모델 즉, Huffman 모델과 Muller 모델이 제안되었다. 그럼에도 현재 대부분의 컴퓨터가 동기 시스템으로 구성되었다. 이것은 중앙 집중적인 클럭의 사용이 설계를 단순하게 만들며, 자동화하는데 쉬운 방법을 제공하는 반면에 비동기 시스템은 설계 자체가 어려웠기 때문이다.

최근에 비동기 설계가 점점 각광을 받기 시작하고 있다. 여기에는 몇 가지 이유가 있는데 첫째, 디지털 시스템을 구현하기 위한 공정 기술이 하나의 칩에 많은 트랜지스터를 집적할 수 있도록 발달되었다. 이러한 공정 기술은 트랜지스터 크기를 줄이고 배선(wiring)과 셀의 신호 지연 시간의 비를 변경시켰다. 배선에 의한 지연 시간은 클럭이 넓게 분포되어 있는 동기 시스템 설계에서는 제약 요소가 되며 고속의 디지털 시스템 구현에도 약점으로 작용한다. 이와 대조적으로 비동기 설계에서는 이러한 제약 요소가 없으며 사건 중심(event-driven), 고속 및 시간 지연에 무관(delay-insensitive)한 동작을 한다. 둘째, 기기의 이동성 증가로 저전력 소모에 대한 요구가 증대되었다. 동기 클럭은 실제 기능 동

작에 필요한 전력보다 많은 전력을 소비한다. 전역 클럭이 없는 비동기 시스템은 기능 동작을 하기 위하여 발생한 신호 천이에 의해서만 전력이 소비된다. 세째, 비동기 설계 방법에서 발생하는 난제를 해결하기 위한 기술과 툴에 대한 학문적인 연구가 활발하다는 점이다. 따라서 초기 비동기 설계에서 나타난 고장(hazard) 등의 문제를 해결할 수 있다. 이러한 비동 설계 기술은 고속, 저전력 특성을 갖는 대규모 회로 및 시스템 설계에 많이 사용되고 있다.

본 고의 내용은 II 장에서 비동기 설계 특징, III 장에서 비동기 설계 방법, IV 장에서 주요 연구 방향, V 장에서 결론의 순서로 서술한다.

II. 비동기 설계 특징

비동기 회로의 개념은 1950 년대에 시작되었으나 현재 인기가 없는 이유는 초기 설계에서 보여지는 해저드의 제거 문제가 어려웠기 때문이다. 이후 여러 시간 가정하에서 비동기 회로를 생성할 수 있는 방법이 개발되어 문제를 해결할 수 있게 되었다. 또한 비동기 회로는 기존의 동기화 시스템보다도 많은 장점을 가지고 있다. 클럭 스큐의 제거, 동기화 실패 문제 해결 이외에 물리적 변수 변화에 대한 허용 오차의 확대, 쉽게 시스템의 모듈 방식을 사용한 합성, 저 에너지 소모, 최악 케이스 대신 평균 케이스 성능 등이다.

1. 동기 및 비동기 설계 비교 [1]

동기 설계에서 레지스터 사이의 논리는 매 클럭 주기마다 새로운 입력으로 계산된다. 이러한 동기 설계에서는 전력을 줄이기 위하여, 유효한 동작을 하지 않을 때 실행 부분에 대한 전원을 감소시키거나 천이 수를 최소화시키는 것이 중요하다. 따라서 동기 회로에서는 사용되지 않는 모듈에 대한 검출 회로나 전원 감소 등의 주의를 필요로 한다. 비동기 설계 기술인 self-timed 논리는 본래 사용되지 않는 모듈에 대하여는 전원 감소의 특성이 있다. 그러나 self-timed 논리로 구현된 논리 모듈의 출력은 완전한 신호 발생을 필요로 하기 때문에 부가된 회로로 인하여 비용이 증가하는 단점이 있다. 이러한 단점의 영향을 줄이고 필요한 신호를 생성시키는 방법으로 이중 회선 코딩(dual-rail coding)을 사용한다. 이 방법에서는 DCVSL(differential cascade voltage switch logic)과 같은 논리 회로를 사용하며, 종속된 DCVSL 게이트로 구성된 조합 매크로 셀의 신호들은 맨 끝 게이트들의 출력을 오아링하여 구성하기 때문에 추가 비용이 적게 증가한다. 그러나 이중 회선 코딩은 출력들 중에 최소한 한 개가 '0'으로 되어야 하기 때문에 천이가 반드시 발생한다. 일반적으로 기존의 정적 제품(static family)보다도 입력이 천이하는 경우 최소한 두 배 정도의 전력을 소비한다. 그러므로 self-timed 논리로의 구현은 데이터 패스와 같은 연속적인 계산을 필요로 하는 곳에서는 에너지를 많이 소비하기 때문에 비효율적이다.

2. 고속성(Clock Limitation)

동기 설계의 발전은 컴퓨터의 설계 방식이나 성능에 종속되기 때문에 시스템이 복잡해짐에 따라

서 근본적인 제약 요소이다. 또한 동기 회로는 클럭에 따라서 동시에 동작한다. 이러한 동작은 클럭이 전기적 신호이며, 다른 신호들과 같이 동일한 전달 지연을 갖고 전달되는 경우에만 발생한다.

칩과 칩 사이에서는 최대의 클럭 주기로 동기적 구성이나 유지가 어렵다. 이러한 경우 온칩 위상 록트 루프(on-chip phase-locked loops)가 칩 사이의 오차를 보상해 주지만 최대 50MHz 정도이므로 불충분한 상태이다. 하나의 단일 칩에 완전한 CPU를 만들어도 클럭 스큐는 존재한다. 또한 고성능 프로세서의 구현은 실리콘 면적을 증가시키며 클럭 구동기가 허용하는 스큐가 증가하기 때문에 면적 증가에 대한 제약이 있다. 통로(tracks)가 좁아질수록 칩은 커지고 칩 내의 전기적 신호들의 전달을 위하여 클럭 속도는 증가한다. 따라서 스큐 문제가 심각하게 된다. 클럭은 선 지연(wire delay)을 보상해주기 위하여 넣지만 클럭 또한 선에 따른 전달 지연 시간을 갖기 때문에 스큐 문제는 해결되지 않는다.

클럭 배치에서 또 다른 물리적 제한은 열의 발생이다. CMOS는 게이트가 천이하는 경우에만 에너지를 소모하기 때문에 저전력에 적합한 기술이다. 그러나 모든 경우에 이것이 만족되지는 않는다. 이유는 새로운 입력에 의한 동작이 아니라 클럭에 연결된 많은 게이트들이 동작하기 때문이다. 가장 큰 게이트가 클럭 구동기이고 이 게이트 셀은 단지 칩의 일부분만이 유효한 동작을 하더라도 시간 기준(timing reference)을 위하여 모든 시간 동안에 동작해야 한다. 또한 칩이 실질적인 동작을 하지 않더라도 고속의 클럭을 멈추거나 시작시키는 것이 쉽지 않다.

3. 저전력(커패시턴스 최소화 및 회로 최적화)

초기 CMOS 소자는 매우 저전력이지만 공정 규칙이 감소함에 따라서 CMOS의 동작 속도는 증가되고 조밀해졌다. 또한 공정 규칙이 1정도 줄어들면 최대 성능을 내기 위한 전력의 소비는 2배 정도 증가한다. 소비 전력의 증가는 열 발생의 원인이며, 공급 전원의 전압을 낮추는 방법을 사용하더라도 열 발생 문제는 여전히 존재한다. 따라서 고성능 프로세서와 같이 많은 면적과 전력을 소비하는 칩은 배터리를 전원으로 하는 장치에 사용할 수 없다. 또한 데스크 톱에서도 냉각 시스템이나 열 제거 기술을 필요로 하기 때문에 사용하기 어렵다. Sub-micron 이하로 소자의 크기가 감소하고 칩에 많은 기능을 첨가할수록 칩의 평균적인 균형의 감소로 전역 클럭은 점점 비효과적이다.

CMOS 회로에서 소비되는 전력은 천이하는 동안에만 소비하기 때문에 천이 수를 최소화하는 것이 저전력 설계 방법중의 하나이다. 천이 수를 줄이기 위한 방법으로는 게이트 클럭(gated clock)의 사용이나 최적화된 회로 구조 사용 등과 같은 방법이 있다. 또한 천이 수와 관계 있는 스위치 커패시턴스(switched capacitance) 값을 줄이는 것도 저전력 설계를 위해 필요하다. 이러한 커패시턴스의 최적화는 알고리즘, 아키텍처, 논리 설계, 회로 설계, 그리고 도면 설계의 모든 수준에서 시도되고 있다. 특히 회로 수준에서의 최적화에서 보면, 여러 가지 논리 및 연산 기능을 구현하는데 기본적인 회로 및 위상을 선택하는데 몇 가지의 선택 사양이 있다. 정적 또는 동적 구현, 패스 게이트 또는 CMOS 논리 게이트, 그리고 동기 또는 비동기 타이밍 등이

<표 1> 비동기 논리의 장점[3]

특 징	설 명
평균 케이스 성능	- 데이터 종속적 처리 - 평균과 최악 상태의 차이가 큰 경우에 효과적 - 제품 완성에 많은 시간 불필요
저전력 소모	- CMOS는 천이 동안에만 전력 소모 - 클럭의 경우 유효 동작이 아닌 경우에도 전력 소모 - 요구에 의해서만 전력 소모
모듈 구성 용이	- 레고 방식 - 증진적인 향상 가능 - 동작 파라미터의 확실성
인터페이스에서 클럭 동기 불필요	- 클럭에 대한 입력 신호 동기에 많은 주의와 시간 소모 - 준-안정 상태가 발견하기 어려운 오류 발생 가능 - 본질적으로 다양한 데이터 속도에 적합
준-안정 시간	- 예측할 수 없는 시간 동안 준-안정 영역에서 쌍안정 가능 - 고정된 분석 시간에는 오류 발생 가정 - 정확함을 증명하기 위하여 중재자 사용 가능
클럭 배치 문제 없음	- 설계 시간 감소 - 전력 비용 감소 - 칩 면적 감소
동시성 향상	- 동시성이 많은 시스템 표현에 적절 - 인터리빙 계획보다는 동시성 사용 가능
지적인 도전	- 더 좋은 퍼즐 - 비형식의 추리는 위험 - 기술 혁신을 위한 여지
본질적인 정밀	- 순서 영역으로의 직접 매핑 제공 - 이론적 작업에 대한 확실한 목표 - 조립 설계에 의한 교정 - 측정에 대한 신뢰
전역 동기의 불필요	- 고속 클럭, 큰 칩 및 시스템에 사용 - 전역 동기 사용은 간결하나 실현성 부족 - 비동기 기술로 문제의 해결과 능력 인정

시스템 및 회로 설계자에게 주어지는 선택 사양이다. 그러므로 비동기 설계는 회로 수준에서의 저전력 설계 기술이다.

4. 비동기 논리의 장단점[2]

비동기 논리의 장점은 동기 회로의 경우 소자의 크기가 감소할수록 클럭 스큐가 중요하지만 비동기 회로는 전역 클럭이 없기 때문에 클럭 스큐가 없다. 또한 동기 회로의 경우 클럭 라인은 항상 토글(toggle)되기 때문에 실제 기능을 하지 않더라도 충전 및 방전에 의한 전력 소비가 있지만, 비동기 회로는 실제 동작에 참여하는 부분만이 천이하기 때문에 전력 소비가 적다. 성능면에서 동기 회로의 경우 모든 가능한 동작이 완료된 후 까지, 비동기 회로는 동작 종료까지의 성능을 나타내기 때문에

최악 케이스 대신 평균 케이스의 성능을 갖는다. 동기 회로의 경우 임계 경로 상에서 드물게 사용되는 부분까지 포함하여 최적화하지만, 비동기 회로는 임계 경로 상에서 드물게 사용하는 부분은 비최적화 된 채로 사용하기 때문에 전역 타이밍 문제의 해결이 용이하다. 그리고 쉽게 다른 공정 기술로의 설계 변경이 가능하다. 그 밖에 비동기 논리에 대한 장점은 <표 1>에 정리하였다.

비동기 논리의 단점은 비동기 회로의 설계가 어려운 점이다. 동기 회로의 경우 조합 논리와 래치를 사용하고 해저드를 고려하여 클럭 주기를 충분히 길게 해줄 수 있기 때문에 회로의 동적 상태를 제거할 수 있지만 전역 클럭을 사용하지 않는 비동기 회로는 동적 상태에 세심한 주의와 함께 해저드 제거나 해저드가 초기에 입력되지 않도록 주의해야 한다. 또한 동작 조건의 순서에 주의해야 하며 결합, 교환, 그리고 DeMorgan's 법칙과 같은 대수적 관계가 성립되지 않는 경우가 발생하기도 한다. 동기 회로에 비하여 높은 성능을 갖지만 신호를 입력(signaling policies)시키기 위한 추가의 시간 필요, 평균적인 지연 증가 등에 대한 비용이 비동기 회로의 장점보다 큰지 또는 작은지 불명확하다. 그리고 회로의 검증 및 테스트가 복잡한 점 등이다.

III. 비동기 설계 방법

비동기 회로의 많은 장점에도 불구하고 동기 회로가 지배적인 이유는 비동기 회로가 동기 회로에 비하여 설계하기 어렵기 때문이다. 또한 해저드도 회로에서 오동작을 하지 않도록 제거되어야 하고 동기 시스템에서 래치의 배치에 따른 동작 순서도 비동기 제어 논리에 의하여 검증되어야 한다.

비동기 설계에 영향을 주는 요소는 지연 모델, 신호 인가 모델, 그리고 환경 모델이다. 지연 모델에는 게이트의 지연만 고려하는 게이트 지연 모델, 게이트 및 선에 의한 지연 값을 모두 고려하는 선 지연 모델이 있다. 또한 이 지연 값에 따라서 유한(bounded) 및 무한(unbounded) 타이밍 특성을 갖는다. 신호 인가 모델은 출력이 변하기 전에 입력 값이 인가되는 경우 이들 입력들을 모두 제거하는 관성(inertial)과 제거하지 않는 순(pure) 메모리 특성이 있다. 환경 모델로는 비동기 회로가 동작하는 상태에 따라서 구분된다. 동작 상태는 입력의 변화에 따라서 비동기 시스템이 안정된 후에 입력이 변하는 기본 모드(fundamental mode)와 이전 입력 값의 변화에 의하여 첫번째 출력 신호가 변하자마자 입력이 변하는 입출력 모드(input/output mode)가 있다. 또한 두 가지 모드는 입력 변화의 수에 따라서 단일 입력 변경(single input change: SIC), 다중 입력 변경(multiple input change: MIC), 그리고 무제한 입력 변경(unrestricted input change: UIC)의 세 가지로 구분된다.

비동기 설계에 대한 주요 기술로 타이밍 모델, 신호 발신 프로토콜, 그리고 시스템 사양 및 구조에 대하여 서술하고, 주요 기술에 대한 분류와 특징은 <표 2>에 요약하였다.

1. 타이밍 모델

타이밍 모델은 동기 회로와 유사한 유한 지연(bounded delays), 게이트의 임의 지연을 사용하는 SI(speed independent), 게이트 및 선의 임의 지연을 사용하는 DI(delay insensitive), 그리고 DI 의 특성을 갖

<표 2> 비동기 설계의 주요 기술

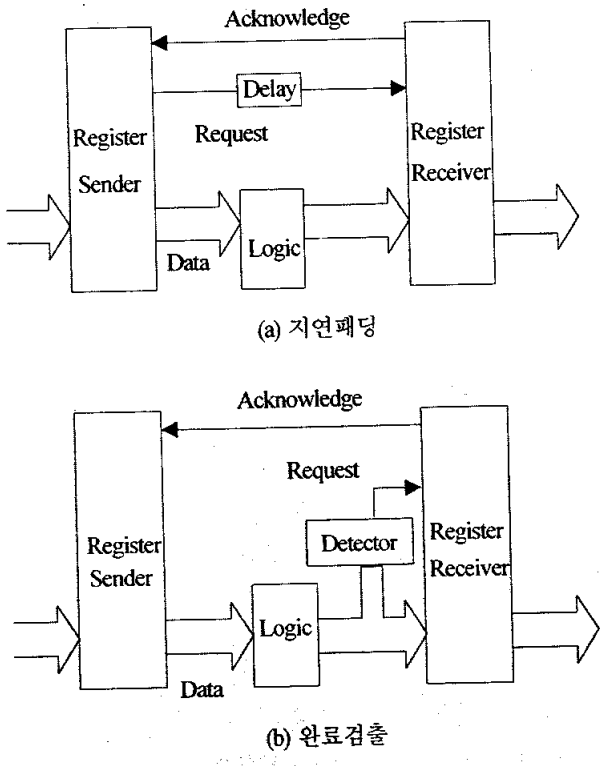
분 류	기 술	모델 및 특징
타이밍 모델	유한 지연	- 동기 회로와 비슷함 - 회로 또는 지연 범위 가정하에 최대 지연 예측 - 필요한 경우 추가 모델 사용
	속도 독립적 회로	- 게이트에 임의 지연 사용 - 선 지연 없음
	지연 무감각	- 게이트 및 선에 임의 지연 사용 - 장점을 갖는 모델이나 회로 종류에 따라 상이
	준 지연 무감각	- Isochronous forks - 지연이 비슷하다고 가정 - 지연 무감각 모델과 매우 유사
신호 발신 프로토콜	제어 신호 발생	- 4상 신호 및 2상 천이 신호 발생 - 데이터 전달 시점을 결정하는 프로토콜
	데이터 신호 발생	- 묶음 데이터: 비트 당 1개의 선 / 데이터의 유효 신호 제어 - 이중 회선 데이터: 비트 당 2개의 선 / 단일 인정 제어 선
시스템 사양 및 구조	유한상태머신	- 고전적인 비동기 기술 - 비동기 유한상태머신 모델 - 버스트 모드 비동기 유한상태머신 - 확장된 버스트 모델
	페트리 네트 베이스	- 인터페이스 상태 그래프 모델 - 암호화된 인터페이스 상태 그래프 - 인터페이스 네트 - 신호 천이 그래프
	매크로 모듈	- Sutherland's 마이크로파이프라인 - 기본 마이크로파이프라인
	구문 변환 프로그램	- 비동기 회로로 변환 - CSP, OCCAM 및 Tangram 언어

지만 등시적 분기를 갖는 QDI(quasi delay insensitive) 모델이 있다. 지연 가능 범위에서는 SI 또는 DI가 유용하며 데이터패스에서는 유한 지연 모델이 적합하다

2. 신호 발신 프로토콜

상위에서의 신호 전달에 필요한 프로토콜로는 self-timed 논리 종류에 속하는 request/acknowledge 방식이 가장 대중적인 구조이다. 이 방식은 (그림 1) 과 같은 구조를 가지며 전역 클럭 대신 신호 핸드셰이크를 사용한다. 신호 핸드셰이크는 지연 패딩(delay padding)과 완료 검출(completion detection) 방법을 사용하여 구현한다. 지연 패딩은 동기 논리와 유사하며 최대 논리 지연 값을 측정하거나 예측하여 *Logic* 블록이 안정되기 전에 request 신호가 들어오면 추가적인 지연 소자(*Delay*)를 넣는 것이다. 또한 완료 검출은 *Logic* 블록의 완료 상태를 검출하여 request 신호를 발생시키는 것이다. 따라서 추가적

인 지연 소자나 완료 신호 검출 블록의 사용은 면적에 대한 오버헤드를 증가시킨다.



(그림 1) 신호 핸드셰이크

3. 시스템 사양 및 구조

시스템의 구조는 유한 상태 머신, 페트리 네트 베이스 (Petri-net based), 매크로 모듈(macro-modules), 그리고 구문 변환 프로그램(syntax-direct program translation) 등의 종류가 있다.

가. 유한 상태 머신

유한 상태 머신에서는 전통적인 비동기 기술을 사용하는 Huffman 종류[(그림 2)]의 상태 머신, 비동기 유한 상태 머신 모델 (asynchronous FSM model: AFSM), 버스트(burst) 모드 비동기 유한 상태 머신 모델, 그리고 확장된 버스트 모드 비동기 유한 상태 머신 모델 등의 종류가 있다.

비동기 유한 상태 머신 모델은 입력이 변경된 후, 다음 입력이 변경되기 전에 안정된 상태로 되어야 하는 기본적인 방식의 동작을 한다. 이러한 동작은 동기 머신의 셋업 및 홀드 사양과 유사하다. 입력 변경은 단일 입력 변경, 다중 입력 변경, 그리고 무제한 입력 변경의 형식으로 한다. 이러한 비동기 유한 상태 머신 모델의 설계 과정은 초기의 흐름 표 생성, 흐름 표를 줄이기 위한 상태 최소화, 상태 지정, 그리고 모든 입력 변화에 대하여 해저드가 없는 논리생성의 순서로 된다. 그리고 버스트 모드로의 확장이 가능한데, 단일 입력 변경 형식의 경우 속도가 느리고 다중 입력 변경 및 무제한 입력 변경 형식은 어려운 단점이 있다. 이러한 단점은 기본적인 방식의 동작을 하지만 단일 입력 대신 버스트 입력, 버스트 내에서도 임의 순서로 입력 변경 허용, 버스트 출력, 그리고 다른 버스트 입력들이 입력되기 전에 출력을 안정화 시키는 방법으로 해결할 수 있다.

버스트 모드 비동기 유한 상태 머신 모델은 버스트 입력들은 임의의 순서나 시간에 도착하고, 각 상태는 단일 등록 포인트를 가지며, 주어진 상태에서 버스트 입력은 다른 버스트의 부분 집합이 아닌 사양을 갖는다. 또한 회로를 구성하기 위하여 지역적 클럭 사용, 클럭 미사용, 3-D 머신 등과 같은 기술이 사용된다.

확장된 버스트 모드 비동기 유한 상태 머신 모델은 기본 버스트 모드 사양에 두 가지 특성이 부가되는데 "directed don't-cares" 및 수준 상태 신호

(level condition signals)이다. "Directed don't-cares"는 입력 가장자리와 최종적인 동작에 의하여 종료되도록 하며 수준 상태 신호는 상태 변경 방향을 결정하도록 하는 견본 신호이다.

나. 페트리 네트 베이스

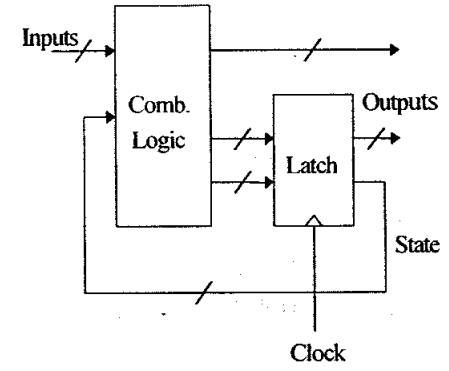
페트리 네트 베이스는 시스템의 상태보다는 신호 천이에 근거한 기술이며 신호 천이는 회로의 인터페이스 동작을 표현한다. 이 기술은 I-net 신호 천이 그래프 (interface-net signal transition graphs), 변경 도표(change diagrams), 그리고 명령어 등을 포함한다.

I-nets 는 인터페이스 신호 이름으로 표시된 천이를 이용하여 페트리 네트를 표현한다. 또한 허용된 인터페이스 동작으로 기술되고 상태 머신으로 변환되어 구현된다.

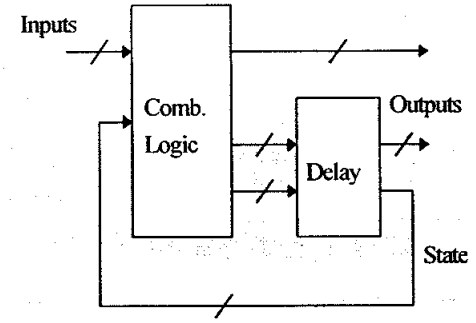
신호 천이 그래프는 I-nets 와 같이 인터페이스 신호 천이에 중점을 두지만, 특성을 표현하는데 그래프 이론을 사용한다. 신호 천이는 방향에 따라서 표시되며 구현하는데 허용되는 형식은 제한이 있다. 이러한 신호 천이 그래프의 특성은 문법적 확인만으로 "deadlock free" 및 해저드에 대한 확인이 가능하다. 또한 여러 형태로의 확장이 가능하고 동작과 회로에 대한 많은 특성을 확인할 수 있는 특징이 있다.

다. 매크로 모듈

매크로 모듈은 비동기 시스템으로 만들 수 있는 블록으로 구성되며 모듈간 DI 인터페이스를 사용한다. 매크로 모듈의 종류에는 Sutherland's 마이크로 파이프라인과 기초적인 마이크로 파이프라인의 두 종류가 있는데, Sutherland's 마이크로 파이프라인은 대중적인 매크로 모듈 라이브러리로 2상 (two-phase) 동작 특성을, 그리고 기초적인 마이크로 파이프라인은 FIFO 와 같은 동작 특성을 갖는다.



(a) 동기 상태 머신



(b) Huffman 비동기 상태 머신

(그림 2) 유한 상태 머신

라. 구문 변환 프로그램

구문 변환 프로그램은 프로그램 언어를 사용하여 기술된 구문을 비동기 회로로 변환시킨다. 이 프로그램은 동시 발생에 대한 표기가 가능하고 기본적인 전달 구조는 채널이다. 프로그램 기술 언어로는 CSP(communicating sequential processes), OCCAM 및 Tangram 등이 있다.

Ⅳ. 주요 연구 방향

비동기 논리를 이용한 연구는 형식 방법(formal methods), 검증, 회로 기술, 조정, 동기 및 비동기 인터페이스, 데이터 패스 설계, 그리고 저전력 회로 등의 여러 분야에서 연구가 진행되고 있다. 이 중에 비동기 설계 분야, 비동기 회로의 테스트[4], 전력 예측[5], 그리고 비동기 셀 라이브러리[6]에 대하여 살펴본다.

1. 비동기 설계 분야

비동기 설계는 프로세서나 메모리 등의 설계에 많이 사용된다. 비동기 프로세서의 설계에서는 주로 페트리 네트를 이용한다[7]. 페트리 네트는 하드웨어 시스템의 설계 형식으로 주목 받고 있으며 회로 설계자에게는 페트리 네트 표기의 그래픽적 특징이 대수적 표기보다 효과적이다. 페트리 네트 사용의 주요 특징을 보면, 수학적인 기반이 우수하며 회로의 해저드의 확인에도 사용할 수 있다. 또한 형식 합성(formal synthesis) 실행에 필요한 모델링 언어로 복잡한 프로세서 설계나 신호 처리 칩에 대하여 상위 수준에서의 분석이 가능하다. 그리고 "PNs to VHDL" 변환, 또는 역으로 "VHDL subset to PNs" 변환이 가능하며, 페트리 네트 툴들을 기존 설계 환경에 접목시킬 수 있다. 이 기술을 비동기 회로 설계 분야에서 많이 사용하는 이유는 페트리 네트의 사건 중심 특성이 비동기 회로의 사건 중심 특성과 잘 일치하기 때문이다. 현재 타이밍 정보와 같은 회로 특성을 정확하게 모델링할 수 있도록 페트리 네트 표기의 확장에 대한 연구가 진행중이다.

비동기 프로세서의 설계로는 맨체스터 대학의 AMULET1, CALTECH의 비동기 프로세서, "HP Lab's" 마이크로 프로세서인 Mayfly, 그리고 Tokyo 기술 연구소의 TITAC 등이 있다. 이들 설계에서는 AMULET 마이크로 프로세서를 이용한 제어 회로 모델링 및 분석, 회로 페트리 네트를 이용한 비동기 회로 모델링 및 분석, 그리고 형식화(formalism) 및 신호 천이 그래프를 이용한 비동기 인터페이스 회로의 합성 기술이 사용되었다.

비동기 논리를 사용하여 설계된 메모리는 회로가 동작할 때 에너지를 소비하기 때문에 에너지 소비면에서 월등한 성능을 나타낸다[8]. 이러한 비동기 메모리의 성능은 동작 실행에 필요한 에너지 측정으로 평가하며, 에너지는 명령 수행, 메모리로부터 데이터 인출, 그리고 인터럽트 실행 등에 소비되는 에너지이다. 이들 동작에 필요한 에너지를 줄이는 것은 주어진 에너지 내에서 명령 실행의 수를 최대로 하는 것과 같다.

2. 비동기 회로의 테스트

비동기 회로가 마이크로 프로세서와 같은 큰 회로를 대상으로 하고, 상업적인 툴(Mosaic)에도 사용되기 때문에 테스트의 필요성이 증대되었다. 일반적으로 테스트는 동기회로보다 어렵다. 전역 동기 신호가 없어서 제어 부분이 감소되었지만 상태 유지를 위한 구성 요소 부분이 증가하기 때문에 테스트 벡터의 생성이 어렵다. 또한 쉽게 테스트하기 위한 회로의 추가는 면적 오버헤드를 증가 시키고 해저드나 레이스 문제는 "delay fault"로는 검출이 어렵다. 이와 같은 단점과는 다르게 테스트를 쉽게

<표 3> 비동기 설계 툴[9]

툴 이름	특 징
SIS	비동기 및 동기 합성 / 다단 조합 논리 합성
FORCAGE	속도에 독립적인 회로 분석 및 합성
MEAT	가장 우수한 비동기 툴
SMV, CSML, and MCB	- SMV: CTL 심벌 모델 확인 - CSML: 유한상태머신 표현을 위한 상위 수준 언어 - MCB: EMC 모델 확인
VERDECT	속도에 독립적이거나 지연 무감각한 회로 검증
SYN	속도에 독립적인 회로 합성
ASSASSIN	비동기 제어 회로의 분석 및 합성
Synchronized Transition	C 프로그램 또는 Larch Prover 인식 표현으로 변환
3D	비동기 및 동기 혼합 합성
Analyze	회로 합성 및 검증을 위한 형식 방법
Petrify	신호 천이 그래프 읽기 및 생성
Verify	속도에 독립적인 회로 검증
HORN Project Software	저전력 VLSI 분석 및 설계
LARD	비동기 시스템 표현을 위한 하드웨어 표현 언어

하는 비동기 회로의 특성도 있다. 비동기 회로는 동기 동작의 전역 클럭 대신 부분적 신호 변경(local handshakes)을 사용한다. 여기서 사용된 신호들에 대한 stuck-at-fault는 무한정 모듈간 연결이 가능하기 때문에 쉽게 관찰이 가능한 점이다. 이러한 차이점들을 이용하여 테스트함으로써 적용된 기술에 대한 트레이드 오프로 재평가가 가능하다.

3. 전력 예측

비동기 회로 또는 전역 클럭을 사용하지 않는 회로는 저전력 분야에 적합하지만 이에 대한 증명으로 비동기 회로에서의 전력 예측 연구는 아직 미흡한 상태이다. 대부분의 설계자는 저전력 에러 정정기, 프로세서, 인터페이스 회로 등과 같이 저전력을 필요로 하는 VLSI 회로 설계에 비동기 회로의 장점을 사용하는 것에 주된 관심을 갖는다.

저전력 설계에서 다른 구조로 변형할 경우 실험적인 예측 기술이 필요하다. 이러한 기술은 소비되는 전력을 빠르고 어느 정도 정확하게 예측할 수 있어야 한다. 또한 저전력 비동기 회로를 설계하기 위하여 여러가지 종류의 회로와 구조적 가능성에 대하여 빠르게 선택할 수 있는 수학적 모델 및 예측 기술이 필요하다. 따라서 비동기 회로 영역에서 소비 전력의 예측에 필요한 규칙적인 방법이 필요하게 되었다. 최근에 Berkel와 Beerel 등에 의하여 예측 기술이 제안되었으며 특히 self-timed 회로에서의 전력 예측을 위하여 Devadas, Ghosh-Devadas, Moteri-Devadas, Alidina-Monteiro-Devadas 등이 알고리즘을 제안하였다.

4. 비동기 셀 라이브러리

동기 VLSI 시스템의 성능은 전역 클럭의 속도에 의하여 제한된다. Muller 모델에 기초한 self-timed 설계 기술은 전역 클럭을 사용하지 않기 때문에 성능을 향상시킬 수 있다. 그러나 해저드를 막기 위하여 self-timed 시스템은 몇 가지의 가정과 타이밍 제약을 만족해야 하기 때문에 특별히 설계된 셀을 필요로 한다.

비동기 설계를 위한 셀 라이브러리는 Muller C-elements, DCVSL 회로, 래치, 지연 소자 등과 같은 셀로 구성된다. 모든 셀이 가정과 타이밍 제약에 대한 위반이 없어야 한다. 또한 self-timed 시스템의 검증을 위하여 SPICE 나 Verilog 등의 시뮬레이터에 사용할 수 있는 빠르고 신뢰성 있는 모델과, 기존의 셀 라이브러리와 호환되는 적응성 등이 필요하다.

V. 결 론

비동기 논리의 특징, 설계 및 응용 분야 등에 대하여 살펴 보았듯이 비동기 논리 기술의 연구 영역은 광범위하다. 또한 대규모 회로나 시스템의 설계에서 비동기 논리 기술이 사용되고 있다. 아직 면적이 증가하는 문제점을 가지고 있지만 앞으로 동작 속도나 소비 전력보다는 중요도가 낮아질 것이다. 그리고 이러한 장점을 최대로 하는 상용화된 비동기 논리 설계 툴도 곧 등장하리라 예상된다.

참 고 문 헌

[1] A. P. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, 1995, pp. 253-254.

[2] Scott Hauck, "Asynchronous Design Methodologies: An Overview," *Proceeding of the IEEE*, Vol. 83, No. 1, January 1995, pp. 69-93.

[3] Erik Brunvand, *Tutorial Introduction to Asynchronous Circuits and Systems*, http://maveric0.uwaterloo.ca:80/amulet/async/info.htm(current Sep. 2, 1997).

[4] H. Hulgaard, Steven M. Burns, and G. Borriello, "Testing Asynchronous Circuits: A Survey," *Technical Report 94-03-06*, Department of Computing Science and Engineering, Univ. of Washington, March 1994.

[5] P. Kudva and V. Akella, "A Technique for Estimating Power in Self-Timed Asynchronous Circuits," *International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Nov. 1994, pp. 165-175.

[6] Yuk-Wah Pang, Wing-yun Sit, Chiu-sing Choy, Cheong-fat Chan, and Wai-kuen Cham, "An Asynchronous Cell Library for Self-Timed Designs," *IEICE Trans. Information & Systems*, Vol. E80, No. 3, March 1997, pp. 296-307.

[7] A. Semenov, A. M. Koemans, L. Lloys, and A. Yakovlev, "Designing an Asynchronous Processor Using Petri-nets," *Technical Report 539*, Department of Computing Science, Univ. of Newcastle, 1995.

[8] Jose A. Tierno and Alain J. Martin, "Low-Energy Synchronous Memory Design," *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Nov. 1994, pp. 176-186.

[9] Budi Rahardjo, *Asynchronous Tools*, http://www.cs.man.ac.uk/amulet/async/async_tools.html(current Sep. 2, 1997).