비동기식 회로 설계 기술

Design Method for Asynchronous Circuit

오명훈 (M.H. Oh) 서버플랫폼연구팀 선임연구원
김영우 (Y.W. Kim) 서버플랫폼연구팀 선임연구원
신치훈 (C.H. Shin) 서버플랫폼연구팀 UST 박사과정
김성남 (S.N. Kim) 서버플랫폼연구팀 선임연구원

목 차

비동기식 회로는 전역 클록이 없이 모듈끼리의 핸드셰이크 프로토콜에 의해 데이터를 동기화하고, 전송하는 회로로 전역 클록에 기반한 동기식 회로에 비해 전역 클록으로 인한 문제점들, 예를 들면, 타이밍 종결 문제, 전력 소모 문제, 다중 클록 도메인 설계 문제 등에서 이점을 갖는다. 최근에는 이 두 가지 회로의 장점을 모아 서로 다른 클록에 기반한 비교적 작은 규모의 동기식 모듈을 기반으로 모듈끼리의 데이터 전송을 비동기식으로 수행하는 GALS 구조도 많이 연구되고 있다. 본 고에서는 이러한 비동기식 회로를 위한 설계 방식을 설명하기 위해 먼저, 비동기식 회로의 특성과 설계 동향, 설계 방식에 영향을 미치는 핸드셰이크 프로토콜 및 지연 모델을 소개한다. 그리고, 크게 세 가지의 설계 방식을 간단한 예제를 통해 설명한다.

Ⅰ. 비동기식 회로

1. 비동기식 회로의 특성

비동기식 회로는 전역 클록을 사용하지 않고 주변 모듈끼리의 핸드셰이크 프로토콜에 의해서 데이터를 전송하는 방식으로 전송 시의 동기화는 해당 모듈끼리 localize됨을 기본으로 한다. 따라서, 가장 느린 컴포넌트의 시간을 기준으로 전역 클록의 주기를 결정하는 worst-case 형태의 동기식 설계 기법에 비해 모듈간의 다양한 종료시점으로 인한 average-case 형태의 동작을 수행한다. 이는 이론적으로 높은 성능의 특성을 가지며, 실제로 고속 파이프라인 회로[1]-[3]에 적용되었다.

한편, 비동기식 회로 설계 시에 전역 클록이 없으므로 timing closure를 위한 클록 트리 밸런스를 고려하지 않아도 되기 때문에 이로 인해 발생하는 문제점 특히 전력 소모가 문제가 되는 동기식 설계 방법[4]에 비해 장점을 지닌다. 더구나 동작이 필요할 때만 구동되므로, clock gating과 같이 강제적으로 클록 전파를 방지하는 기법을 사용하지 않더라도 특성상 전력 효율이 우수한 on-demand 형태의 동작을 수행할 수 있다. 비동기식 설계 기법이 저전력 설계를 의미하는 것은 아니지만, 이러한 이유 때문에 특정 애플리케이션에서 저전력 소모를 목적으로 성공적으로 적용되었다[5]-[8].

또한, 전역 클록을 사용하지 않음으로 인하여 클록 스파이크(clock spike)가 없기 때문에 평탄한 방사 스펙트럼(smooth radiation spectra) 특성을 갖는다. 이는 향상된 전자기적인 호환성(better electromagnetic compatibility)을 제공하여 매우 민감한 리시버 등과 함께 사용 시에 상호간섭이 감소될 수 있다[8].

아울러, 시스템 내의 모든 모듈들은 핸드셰이크 프로토콜을 통하여 통신하므로 가변 동작속도 환경에서도 잘 적응하여 우수한 이식성 및 재사용 특성을 갖는다.

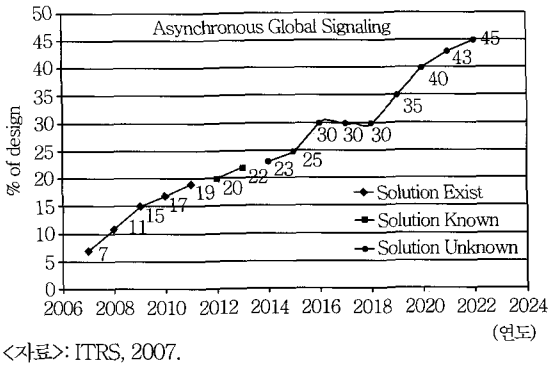
2. 비동기식 회로 설계 동향

이러한 비동기식 회로의 장점에도 불구하고 초기의 비동기식 설계방식이 갖고 있었던 여러 가지 어려움들—비동기 회로에서 발생하는 해저드 문제, 비동기식 회로 설계에 testability를 적용할 수 있는 방법, 상용화된 CAD 툴의 부재 및 성공적인 상용화 칩의 부재 등—은 비동기식 회로 설계의 장애 요소였다.

그러나, 1990년대부터 다시 주목 받기 시작한 비동기식 설계 방식은 유럽과 미국의 주요대학 및 몇몇 기업을 중심으로 앞서 언급한 문제점들의 해결책에 대한 지속적인 연구가 진행되어 왔다.

비동기식 회로 설계에서 hazard-free한 특성을 구현할 수 있는 몇몇 실현 가능한 기술들이 미국의 Stanford와 Columbia 대학에서 개발되었고, 네덜란드의 필립스사는 비동기식 회로 설계에 testability를 적용할 수 있는 솔루션을 개발하였다. 비동기를 위한 CAD 툴의 경우 필립스사와 미국의 Columbia 대학, 영국의 Manchester 대학 등에서 비동기식 회로 설계를 자동화 할 수 있는 소프트웨어를 개발하였다. 최근에는 이를 기반으로 Achronix Semiconductor[9], Elastix[10], Silistix[11], Tiempo[12], Fulcrum Microsystems[13], Camgian Microsystems[14], FTL Systems[15], Handshake Solutions[8]와 같은 비동기 회로 기반 설계 회사들이 활동하고 있다. 이들은 각각 고유한 설계 플로(flow)와 검증 방법을 가지고 있으며, 비동기식 회로의 특성이 강조되는 응용분야에 솔루션을 제공하고 있다.

상용화된 비동기 회로 제품의 경우를 살펴보면, 네덜란드의 필립스사에서 최초의 상용화된 비동기 마이크로컨트롤러인 80C51을 1990년대 후반에 발표하고 이를 1998년에 pager에 적용하였고, 2001년에는 스마트카드에 적용하였다. 영국의 Manchester 대학에서는 2000년 비동기 ARM 프로세서를 발표하였고, 모토롤라사에서는 2000년에 PowerPC 칩에 비동기 제산기를 적용하였다. 또한, HAL사에서는 1990년대 초반에 자사의 HAL-I, HAL-II 프로



<자료>: ITRS, 2007.

(그림 1) 비동기식 회로의 사용 전망

세서에 비동기 부동소수점 연산 제산기를 적용하였다. 그리고 최초의 상용 32비트 비동기 ARM 프로세서인 ARM966HS가 ARM사와 Handshake Solution사에 의하여 개발되는 등 비동기식 회로 설계 방식을 적용한 다양한 제품들이 출시되고 있는 상황이다.

비동기식 회로 설계 플로가 안정화되었다 하더라도 전체 칩을 전적으로 비동기식으로 설계하는 것은 동기식 설계 기법에 익숙한 설계자들에게 일반적으로 쉽게 접근할 수 없다. 그러나, 동기식 설계 기법을 보완하는 형태, 혹은 GALS 구조 형태로 비동기식 설계 기법은 점점 더 널리 사용될 것이다[16]. (그림 1)은 ITRS의 자료로 2022년에는 비동기식 기반의 디자인이 전체 디자인의 45%를 차지할 것으로 예측하고 있다.

본 고에서는 이러한 비동기식 회로를 실제 설계할 수 있는 방법들을 조사, 분석한다.

Ⅱ. 비동기식 회로 개념

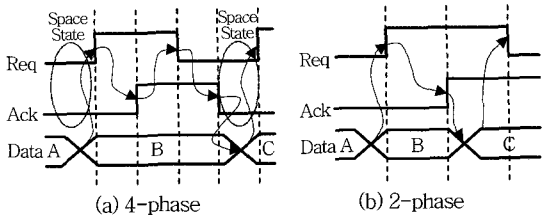
비동기식 회로 설계에 있어서 중요한 두 가지 개념인 핸드셰이크 프로토콜과 지연 모델을 소개한다. 실제로 어떤 프로토콜과 지연 모델을 사용하고자 하는 것에 따라서 다른 설계 방법이 적용될 수 있다.

1. 핸드셰이크 프로토콜

비동기 회로에서는 모든 데이터 전달이 이 핸드셰이크 프로토콜에 의해 수행된다. 즉 전송 데이터의 심볼과 그 시작 시점이 가변적인 비동기 회로 환경에서는 이벤트의 유효성을 알려주는 정확한 transition이 필요하다. 이 정확한 transition을 생성(sender 입장에서) 혹은 감지(receiver 입장에서)하기 위해 시그널링 방식, 데이터 인코딩 방식을 사용한다. 각 방식에 따라서 여러 가지 구현 방식이 가능하고, 아울러 비동기식 핸드셰이크 방식은 채널 종류에 따라서도 다양한 방식으로 구현될 수 있다.

시그널링 방식은 데이터의 유효 시점과 데이터 처리의 완료를 의미하는 request, acknowledge 신호 쌍의 rising, falling 중 한쪽만 사용하느냐, 아니면 모두 사용하는가에 따라서 4-phase(한쪽 사용), 2-phase(양쪽 사용)로 구분된다. 4-phase 시그널링이 항상 초기화 상태(space state)를 거쳐야 하고, 2-phase 시그널링에는 space state가 존재하지 않는다.

Bundled data 방식은 별도의 데이터 인코딩을 하지 않고 동기식 회로에서 데이터의 유효 시점을 클록 신호가 알려주는 것처럼 요구신호가 데이터의 유효 시점을 알려 준다. 즉 요구신호와 전송 데이터가 동기화되어 발생한다. 유효한 데이터는 요구신호보다 먼저 생성되어야 한다는 제약 조건이 따른다. (그림 2)는 4-phase bundled data 방식과 2-phase bundled data 방식을 나타내고 있다. 4-phase 방식은 동기식의 표준셀로도 쉽게 설계 가능하므로 널리 사용되고 있다[17]. 2-phase 방식은 이론상 4-phase에 비해 2배의 성능 효과가 기대되나, 표준셀이 아닌 특화된 셀을 따로 설계해야 하므로 구현 수준에서는 성능 향상을 기대하기 어렵다. 대표적인 구현 사례로는 micropipeline[18]이 있다.



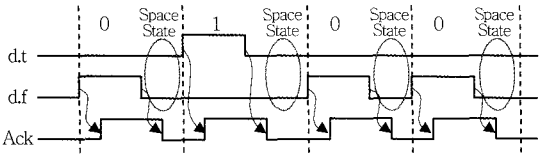
(그림 2) Bundled Data 방식

데이터 인코딩 방식은 주로 지연 무관(DI) 데이터 전송 방식에 사용되는 방식으로 데이터의 유효 시점을 따로 신호(request)로 표현하지 않고, 전송 데이터에 인코딩하여 나타낸다. Dual-rail 인코딩[19], 1-of-4 인코딩[20]의 방식이 사용된다.

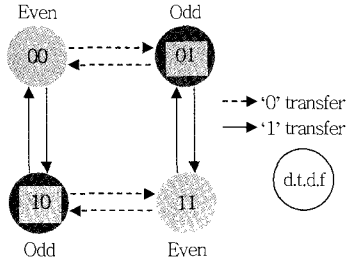
4-phase dual-rail 인코딩 기법은 (그림 3)처럼 data '0'을 d.t=0, d.f=1로, data '1'을 d.t=1, d.f=0으로 인코딩하고 space state를 d.t=0, d.f=0으로 표현한다. Pair 신호 중 한 개만 active 하므로 유효 시점을 쉽게 detect 할 수 있고 구현도 쉽지만, 인코딩에 의해 두 배의 신호 선이 필요하다.

2-phase dual-rail 인코딩 기법은 (그림 4)와 같이 d.t의 변화는 데이터 '1' 전송을, d.f의 변화는 데이터 '0' 전송을 의미한다. 이보다 구현 측면에서 유리한 LEDR[21]도 2-phase dual-rail 인코딩 기법에 속한다.

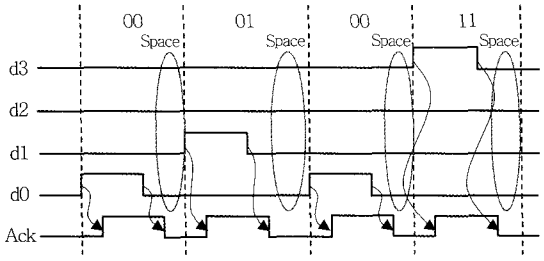
4-phase 1-of-4 인코딩 방식은 (그림 5)와 같이



(그림 3) 4-phase Dual-rail 인코딩 방식



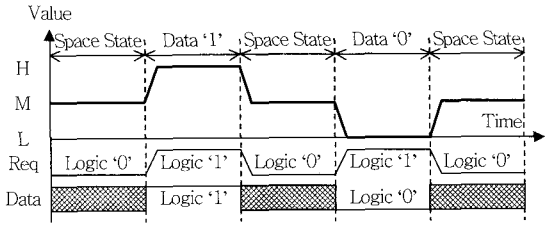
(그림 4) 2-phase Dual-rail 인코딩 방식



(그림 5) 4-phase 1-of-4 인코딩 방식

4개의 wire를 one-hot 방식으로 인코딩한다. 즉, 4개의 wire(d3, d2, d1, d0)에 대해서 0001을 '0'(2진 데이터로 00), 0010을 '1'(2진 데이터로 01), 0100을 '2'(2진 데이터로 10), 1000을 '3'(2진 데이터로 11)으로 인코딩하고, space state는 0000로 할당한다. 4개 wire 중 1개만 변화하므로 4개의 wire 중 2개가 변화하는 dual-rail에 비해서 1/2 수준의 신호 변화 특성을 갖는다. 이는 구현 시에 전력 소모 측면에 유리한 특성이다.

앞서 소개한 데이터 인코딩 방식에 기반한 것들은 N비트 데이터를 2N개의 wire로 인코딩하는 방식이며, 이외에 ternary 신호를 이용하여 N개의 wire로 인코딩 할 수도 있다. (그림 6)은 4-phase ternary 인코딩 방식[22]을 나타내고 있다.



(그림 6) 4-phase Ternary 인코딩 방식

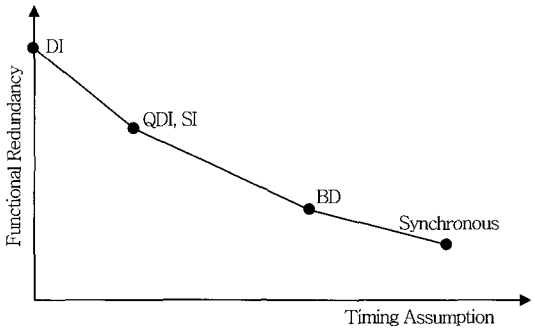
2. 지연 모델

비동기식 회로는 지연 모델에 따라서도 구분될 수 있는데 회로의 소자, 도선의 지연유무에 따라 bounded delay, speed independent, delay insensitive, quasi-delay insensitive 형태가 존재한다. <표 1>은 지연 모델을 요약[19]하고 있다.

BD 모델은 소자, wire 모두 유한한 지연을 가정하고 있다. 입력이 인가된 후 회로가 안정화되고 출력이 발생할 때까지 다른 입력이 발생할 수 없다는 fundamental mode 수행을 기본으로 한다. 이는 동기식

〈표 1〉 소자 도선에 따른 지연 모델

지연 모델	소자	도선
Bounded Delay(BD)	bounded	bounded
Delay Insensitive(DI)	unbounded	unbounded
Quasi Delay Insensitive(QDI)	unbounded	isochronic fork
Speed Independent(SI)	unbounded	zero delay



(그림 7) 각 지연 모델의 특성

회로 특성과 유사하고 실제로 이 기반 모델의 비동기식 회로 설계 시에는 최대 지연시간을 고려해야 한다.

DI 모델은 소자, wire 모두 알려지지 않는 지연을 가정하므로 어떠한 시간 가정도 필요 없다. 따라서 가장 robust한 가정이지만, 실제적으로 적용되는 회로는 극히 제한적이다.

SI 모델은 소자의 지연 시간만 알려지지 않는 것으로 가정한 것이고, QDI는 SI와 유사하나 한 wire에서 분기되는 wire에 대해서는 같은 지연 시간을 갖는 것으로 가정하는 isochronic fork 개념이 내포되어 있다. (그림 7)은 타이밍 가정 정도와 회로의 redundancy 정도를 기준으로 각 지연 모델의 특성을 나타내고 있다.

Ⅲ. 비동기식 회로 설계 기법

비동기식 회로 설계 기법은 크게 그래프 기반 방식과 상위 수준 언어 기반 방식의 두 가지로 구분된다. 전자는 주로 입출력 신호들의 스펙을 그래프에 기반하여 기술하고 이를 합성하여 SI 혹은 BD 모델의 비동기식 제어 회로 생성에 사용된다. 제어 패스는 위와 같은 비동기식 전용 툴을 사용하지만, 데이터 패스는 동기식 설계 방법과 같이 설계한다. 실제 설계 시에는 bottom-up 방식으로 integration 한다.

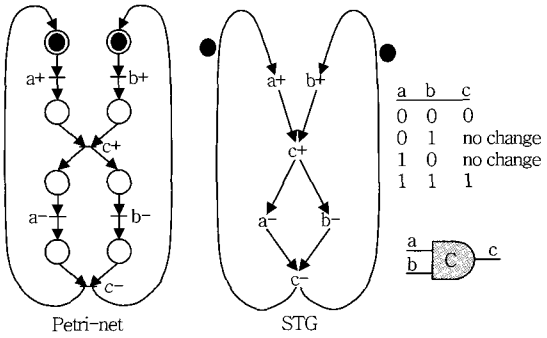
후자는 동기식의 HDL 기반 합성 플로와 유사하게 비동기식 전용 상위 수준 언어를 사용하여 시스템의 스펙을 기술하고 이를 컴포넌트 기반으로 합성하여 설계한다. 이때에는 제어 패스뿐만 아니라 데이터 패스도 비동기식으로 합성된다. 전자와 비교하여 top-down 방식의 특성을 갖는다.

이 모두는 표준셀로 설계가 가능하며, 별도의 비동기식 셀라이브러리를 후공정에서 사용할 수 있는 조건에서는 좀더 최적화된 설계가 가능하다. 다음에서 이 두 가지 방법에 대해 상세히 기술한다.

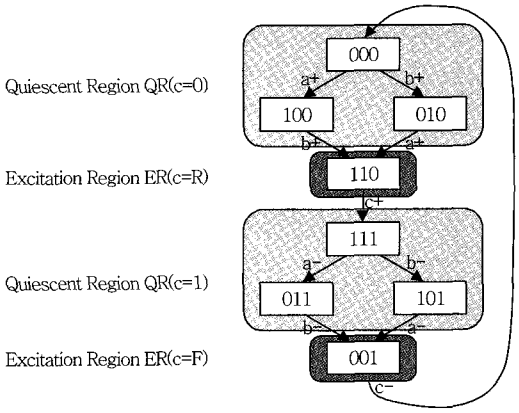
1. SI 모델 그래프 기반 설계 기술

Petri-net 혹은 제한된 형식의 Petri-net, 특히 interpreted free-choice Petri-net 형식인 STG를 기반으로 하는 합성 툴들이 많이 연구되었다.

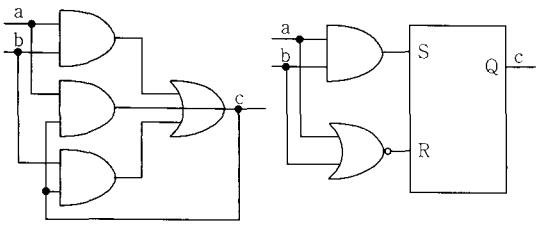
(그림 8)은 입력 신호 a, b, 출력 신호 c에 대한 C-element의 Petri-net과 STG를 나타내고 있다. 토큰 firing 룰에 따라 모든 arc를 마킹한 후의 abc 신호의 상태를 나타내는 SG는 (그림 9)에 표시되어 있다. 이 SG를 가지고 구현 방식(SOP의 게이트 혹은 SR 래치) 출력 c에 대한 조건식을 생성한다.



(그림 8) Petri-net과 STG 예제



(그림 9) (그림 8)의 예제에 따른 SG



(a) 2-level AND-OR 게이트 (b) SR latch

(그림 10) 합성 결과 회로

(그림 9)에서 c에 대해 모두 4개의 region을 나눌 수 있다. QR은 c를 유지, ER은 c set, 혹은 reset 시키는데, 여기에서 c를 1로 set 시키는 ER과 1로 유지시키는 QR에 대해 abc로 K-map을 구성하면 c=ab+ac+bc의 SOP 형태의 조건식이 생성된다. 반면 SR latch 구현을 위해서는 c를 set하는 ER, reset하는 ER을 중심으로 set=ab, reset=!a!b의 식을 얻을 수 있다. (그림 10)은 두 합성 방식에 따른 결과 회로를 나타낸다.

이러한 합성 과정을 자동화 할 수 있는 대표적인 합성 툴로는 Petrify[23]가 많이 사용되고 있다. 일반적으로 STG의 장점으로는 회로의 병행성을 쉽게 표현할 수 있는 능력을 들 수 있으나, input choice를 다루기가 어렵다는 것이 단점으로 지적되고 있다. 즉, free-choice net을 기반으로 하는 STG는 단지 입력 신호 상태 천이에 의해서만 회로가 향후 패스를 선택하게 되는데, 이때에는 입력 신호들 사이의 상호 배타적인 발생을 가정해야 한다.

아울러, STG 기반의 설계 기법에서는 데드록이 없는 liveness, 신호의 +, - 순서를 지켜야 하는 consistent state assignment, SG에서 같은 상태가 없어야 하는 complete state coding 특성을 만족시킬 수 있게 스펙을 기술하여야 하며[19], 4-phase signaling을 지원한다.

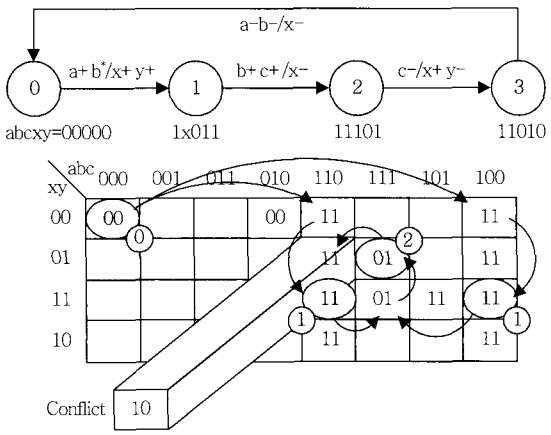
2. BD 모델 그래프 기반 설계 기술

AFSM을 기본 입력 스펙으로 하는 그래프 기반 방식의 합성 기술은 여러 입력값들이 변화된 이후에 회로에 인가하여 상태 천이를 발생시키는 다중 입력 변화를 가정한다. 즉, 모든 입력이 유효할 때까지 기다렸다가 유효 출력값을 동시에 발생시키는 burst 모드 회로를 기반으로 한다. 동기식의 mealy 머신의 FSM과 기술 방법이 유사하여 설계가 용이한 장점이 있으나, STG와는 달리 burst 입력의 제한때문에 병렬 수행 기술이 어렵다는 단점이 있다. 이러한 문제점을 해결하기 위해서 EBM 기반의 합성방식[24]이 연구되었다.

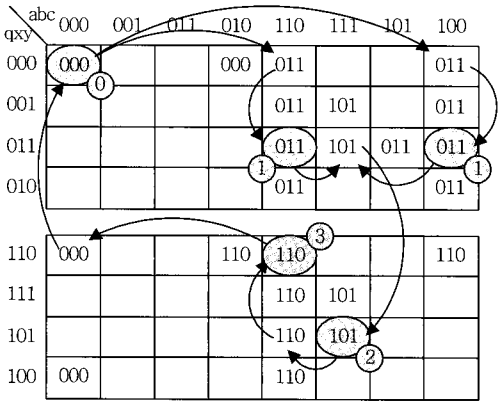
EBM에서는 burst 모드에 입력 신호 발생이 가변적인 경우 그 신호 발생 여부와 상관없이 상태 변환이 가능하도록 "direct-don't care" 기능을 추가시켰다. 이로 인해 입력 burst의 대기 시간을 감소시킬 수 있고, 출력 신호와 입력 신호의 변화에 약간의 병렬성을 기술할 수 있다. 또한 EBM에 추가된 "conditional"의 기능은 입력 신호의 level 값으로 상태 변환의 흐름을 조절할 수도 있다.

(그림 11)은 EBM 기반으로 기술된 AFSM 예제를 나타내고 있다. 입력 burst(a, b, c)가 모두 발생하고 출력 burst(x, y)를 발생시킨 다음 상태가 변하는데, b 신호에 대해서는 상태 0에서 상태 1까지 "direct-don't care"로 선언되어 있다. (그림 11)의 아래는 모든 입출력값으로 상태를 인코딩한 값에 따라 K-map 상에서의 출력값에 따른 상태변환을 나타낸다.

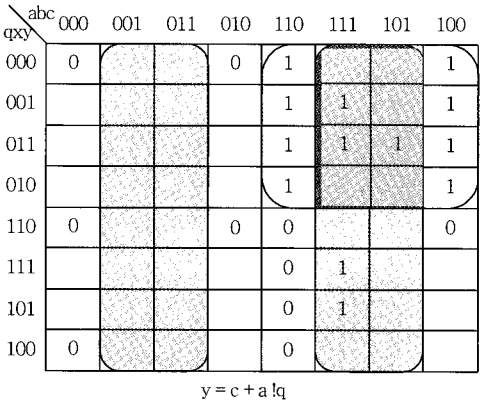
상태 2에서 3으로 변환 시에 이미 할당된 상태와 충돌이 발생한다. 이를 해결하기 위해 (그림 12)와 같이 내부 신호(q)를 발생시켜 충돌될 상태를 분리시킨



(그림 11) AFSM 예제와 K-map 상의 상태 변화



(그림 12) (그림 11)의 예제에 따른 K-map



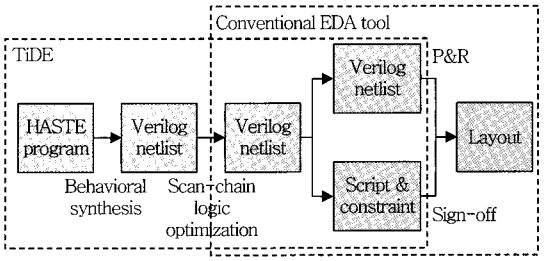
(그림 13) (그림 11)의 예제에 출력 y 추출 K-map

다. (그림 13)은 최종적으로 출력 y에 관한 equation을 위한 K-map과 식을 나타낸다.

AFSM 혹은 EBM 기반의 대표적인 툴로는 3D [25], MINIMALIST[26],[27] 등이 있다. STG 기반의 합성 툴과 마찬가지로, 회로 스펙을 기술할 때는 고유한 제약조건이 따른다. 즉, 입력 인가 후 출력과 내부 상태가 안정화되기 전에는 다른 입력을 발생시킬 수 없는 "multiple-input change fundamental-mode environment" 제약 조건과 단일 상태에서 여러 상태로 분기할 때는 조건의 모호성이 없어야 한다는 distinguishability 제약 조건을 만족시켜야 한다[24].

3. 상위 수준 언어 기반 설계 기술

상위 수준 언어를 기반으로 회로 스펙을 기술하는



(그림 14) TiDE를 이용한 설계 흐름

방법으로는 초기에 CSP[28] 언어가 개발되었다. 병행 수행을 하는 여러 모듈들을 정의된 문법으로 기술하고 각 모듈들의 통신은 역시 정의된 채널을 통하여 수행된다. 기술된 회로 스펙은 production rule이라고 불리는 연산자들로 직접 변환되며, 이들 연산자들은 모두 비동기 회로 셀 라이브러리 형태로 미리 구현되어 있으므로, 합성 시에는 연산자들과 회로들을 직접 매핑하는 작업을 수행한다.

지금까지 연구된 비동기 회로 관련 상위 수준 언어로는 trace theory[29], Occam[30], Tangram [31], Balsa[19] 등이 있다. 최근에는 필립스에서 개발된 Tangram을 기반으로 한 HASTE를 사용하여 게이트 수준 netlist를 생성하는 상위 수준 및 다른 EDA 툴에서 ASIC화 할 수 있는 하위 수준까지 지원하는 TiDE가 상용화 되었다[8]. (그림 14)는 TiDE를 이용한 설계 플로를 나타내고 있다. 행위수준 합성을 통해 HASTE를 합성한 후 최적화 및 scan-chain 삽입부터 상용 EDA 툴을 사용한다. 그리고 최종 레이아웃을 위한 스크립트 및 constraint를 자동 생성하고 이것으로 레이아웃 작업이 수행된다. 로직 최적화 단계 이후는 상용 EDA 툴의 것을 사용하므로 다음에서 행위 수준 합성에 대해서 살펴본다. 즉, HASTE로 작성된 코드가 TiDE를 통해 어떻게 표준 netlist로 합성되는지를 설명한다[32],[33].

행위 수준의 합성 과정은 컴파일러가 HASTE 코드를 분석해 핸드셰이크 회로를 생성하는 단계와 이 핸드셰이크 회로를 분석하여 표준셀로 매핑하는 단계로 구성되어 있다. 이 두 단계에서 이미 설계된 컴포넌트들을 기반으로 한 syntax-directed 컴파일 방식을 지향한다. 즉 핸드셰이크 회로를 생성할 때

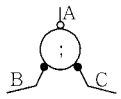
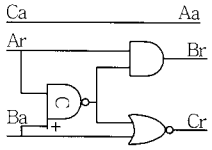
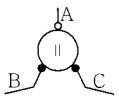
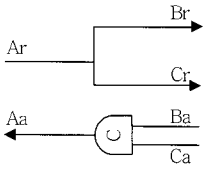
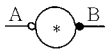
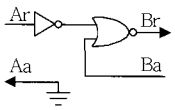
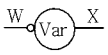
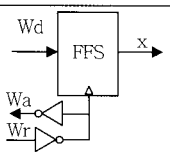
HASTE 구문 상의 여러 오퍼레이터(operator)들은 라이브러리에 정의된 컴포넌트들로 대체되고, 이 컴포넌트들은 이미 표준셀로 구현되어 매핑된다. <표 2>는 대표적인 컴포넌트들로써 핸드셰이크 회로 심볼과 구현 회로, 그리고 컴포넌트 입출력 신호의 발생 순서를 보여준다.

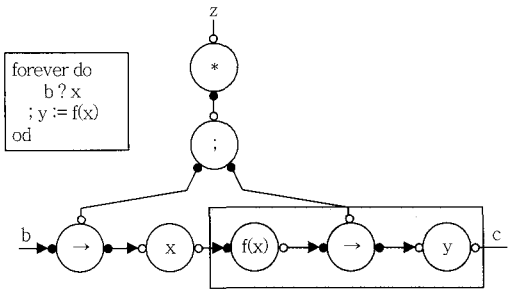
각 컴포넌트의 심볼에는 신호 발생경로에 따라 회로에 요청을 발생시키는 능동형과 요청을 받는 수동형 포트가 있고, 각각 선 끝에 달려 있는 흰색과 검은색으로 마킹된 작은 원으로 표현된다. 즉 능동형 포트의 신호의 요청에 따라 수동형 포트의 신호가 active 되어 컴포넌트의 고유한 동작을 수행한다. 이때에는 모든 포트는 4-위상 시그널링에 기반하여 요구신호(Ar, Br, Cr)와 응답 신호(Aa, Ba, Ca)로 핸드셰이크 프로토콜을 발생시킨다.

Sequencer 컴포넌트는 A 포트의 요청으로 B 포트를 active 시키고 동작이 끝난 후 C 포트를 active 시킨다. 반면 parallel 컴포넌트는 B, C 포트를 동시에 active 시킨다. Repeater는 A 포트의 요청 이후 B 포트가 무한 반복 동작을 수행하고, filp-flop variable component는 실제 표준셀의 플립플롭의 클록 신호를 이용하여 핸드셰이크 프로토콜을 수행하고 플립플롭의 저장시간 동안의 지연소자가 필요하다. 이 밖에서도 fork, join, merge, mux, demux와 같은 컴포넌트들이 있다.

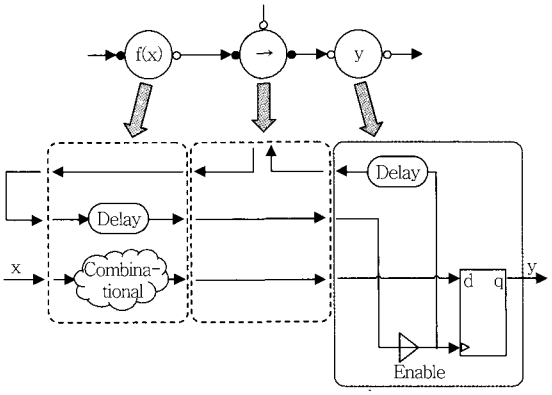
(그림 15)는 간단한 HASTE 코드와 컴파일 후의 핸드셰이크 회로를 나타내고 있다. 이 회로는 forever do에서 od까지의 내부 내용을 무한 반복 수행하게 되는데, 오퍼레이터 '?'는 좌측으로부터 우측으로의 채널 입력을 의미한다. 즉 채널 b에서 데이터를 읽어 변수 x에 저장하게 된다. 오퍼레이터 ':='는 우측에서 좌측으로 변수 할당을 의미한다. 즉 변수 y에 기능회로 f(x)의 결과 데이터를 저장하게 된다. 이 회로는 repeater, sequencer, variable 컴포넌트 및 기능회로 컴포넌트로 구성된다.

〈표 2〉 TiDE의 대표 컴포넌트

컴포넌트	심볼	구현 회로
Sequencer		
	*[Ar↑;Br↑;Ba↑;Br↓;Ba↓;Cr↑;Ca↑; Aa↑;Ar↓;Cr↓;Ca↓;Aa↓]	
Parallel		
	*[Ar↑;(Br↑;Ba↑||Cr↑;Ca↑);Aa↑;Ar↓; (Br↓;Ba↓||Cr↓;Ca↓);Aa↓]	
Repeater		
	Ar↑;*[Br↑;Ba↑;Br↓;Ba↓]	
Filp-Flop variable		
	*[Wr↑(…);Wa↑;Wr↓(v);x:=v;Wa↓]	



(그림 15) HASTE 예제 코드와 핸드셰이크 회로



(그림 16) HASTE 예제 코드의 구현 일부

(그림 15)의 박스 안의 핸드셰이크 회로는 (그림 16)과 같이 매핑되어 구현된다. 기능 회로 컴포넌트는 응답 신호가 가는 시점과 출력단의 데이터가 유효해지는 시점을 동기화시키기 위한 지연소자를 포함한다. 또한 variable 컴포넌트에도 홀드 타임(hold time)을 보장하기 위한 지연소자가 포함된다.

앞서 설명한 설계 방법은 모두 4-phase bundled-data 방식의 핸드셰이크 프로토콜에 기반한다. 그러므로, 제어 회로, 특히 STG에 기반한 설계 방식의 제어 회로가 SI 특성을 갖더라도 데이터 패스는 BD 특성을 고려하여 설계해야 한다. 그러므로, 데이터 패스의 종료 시점을 위해서는 데이터 패스의 critical path 만큼의 match 지연 소자를 사용해야 한다. 데이터 패스에 DI 혹은 QDI 특성을 적용할 수 있는 DIMS[34]나 NCL[35]를 이용하여 데이터 패스를 구현하는 방법도 있다. 이 방법을 이용하면 앞서 설명한 제어 패스 설계가 훨씬 단순해지는 반면, dual-rail 인코딩에 기반하므로 면적, 전력 소모 측면에서 단점이 있다.

이외에도 micropipeline 구조를 사용하여 동기식의 회로에서 저장 소자 사이의 클록을 제거하고 핸드셰이크 프로토콜을 삽입시키는 형태로 비동기 회로를 설계하는 de-synchronization 방식[36]도 연구되었다.

Ⅳ. 결론

본 고에서는 전역 클록이 없이 설계 가능한 비동기식 회로의 설계 방법에 관하여 설명하였다. 초기 관련 연구단계에서 지적되었던 문제점들은 상당 수준 해결되어 최근에는 비동기식 회로 설계에 기반한 각종 솔루션들을 제공하는 EDA 회사들이 속속 생겨나고 있으며, GALS 형태로 최근의 설계 패러다임에 영향을 미치고 있다.

이러한 비동기식 회로는 근간이 되는 핸드셰이크 프로토콜과 지연 모델에 따라서 다양한 가정의 구현 방식이 존재한다.

비동기식 회로의 제어 회로의 스펙을 그래프로 기술하고 합성한 후 데이터 패스는 기존의 EDA 툴에서 설계하여 integration 하는 방법이 있고, 회로의 스펙 자체를 상위 수준의 언어로 기술하여 컴포넌트 기반의 라이브러리로 합성하는 방법이 있다. 또한, 데이터 패스를 DI 특성을 갖게 설계하여 제어 회로의 복잡성을 제거할 수 있는 방법이 있고, 동기식으로 설계한 후 비동기식 회로로 변환하는 식의 설계 방법도 존재함을 알 수 있었다.

앞으로의 동향은 설계 안정성을 높이기 위해 기존 동기식의 설계 흐름을 가능한 많이 중첩하여 이용할 것으로 예상된다. 그러나, 회로의 최적화 이슈로 인하여 여전히 시스템의 특정 부분은 bottom-up 방식이 사용될 것이고, 다양한 핸드셰이크 프로토콜을 지원할 수 있는 설계 방식도 개발될 것으로 전망된다.

● 용어해설 ●

C-element: 비동기식 회로에 자주 사용되는 일종의 제어 회로로서 모든 입력들이 '1' 혹은 '0'가 되었을 때 '1', '0'를 출력한다. 모든 입력이 같은 값을 갖지 않으면 그전 출력값을 유지한다. Sum-of-product 형태의 게이트, SR 래치, 트랜지스터 수준의 다양한 구현 방식이 존재한다.

약어 정리

AFSM	Asynchronous Finite State Machine
BD	Bounded Delay
CAD	Computer-Aided Design
CSP	Communicating Sequential Processes
DI	Delay-Insensitive
DIMS	Delay Insensitive Minterm Synthesis
EBM	Extended Burst Mode
FSM	Finite State Machine
GALS	Globally Asynchronous Locally Synchronous
NCL	Null Conventional Logic
QDI	Quasi Delay Insensitive
SI	Speed Independent
SG	State Graph

SOP	Sum-of-Product
STG	Signal Transition Graph
TiDE	Timeless Design Environment

참 고 문 헌

[1] I. Sutherland and S. Fairbanks, "Gasp; a Minimal FIFO Control," *Proc. Int'l Symp. Advanced Research in Asynchronous Circuits and Systems*, 2001, pp.46-53.

[2] M. Singh and S.M. Nowick, "MOUSETRAP: High-speed Transition-signaling Asynchronous Pipelines," *IEEE Trans. on VLSI Systems*, Vol.15, No.6, June 2007, pp.684-698.

[3] S. Schuster and P. Cook, "Low-power Synchronous-to-asynchronous-to-synchronous Interlocked Pipelined CMOS Circuits Operating at 3.3-4.5 GHz," *IEEE J. of Solid-State Circuits*, Vol.38, No.4, Apr. 2003, pp.622-630.

[4] J. Pangjun and S.S. Sapatnekar, "Low-power Clock Distribution Using Multiple Voltages and Reduced Swings," *IEEE Trans. on VLSI Systems*, Vol.10, No.2, June 2002, pp.309-318.

[5] J. Kessels and R. Marston, "Designing Asynchronous Standby Circuits for a Low-Power Pager," *Proc. of the IEEE*, Vol.87, No.2, Feb. 1999, pp.257-267.

[6] L.S. Nielsen and J. Sparso, "Designing Asynchronous Circuits for Low Power: an IFIR Filter Bank for a Digital Hearing Aid," *Proc. of the IEEE*, Vol.87, No.2, Feb. 1999, pp.268-281.

[7] H. van Gageldonk et al., "An Asynchronous Low-power 80c51 Microcontroller," *Proc. Int'l Symp. Advanced Research in Asynchronous Circuits and Systems*, 1998, pp.96-107.

[8] http://www.handshakesolutions.com

[9] http://www.achronix.com

[10] http://www.elastix-corp.com

[11] http://www.silistix.com

[12] http://www.tiempo-ic.com

[13] http://www.fulcrummicro.com

[14] http://www.camgian.com

[15] http://ftlsystems.com

[16] "International Technology Roadmap for Semiconductors," *Semiconductor Industry Association*, 2007.

[17] S.B. Fuber and P. Day, "Four-phase Micropipeline Latch Control Circuits," *IEEE Trans. on Very Large Scale Integration Systems*, Vol.4, No.2, June 1996, pp.247-253.

[18] I.E. Sutherland, "Micropipelines," *Comm. of the ACM*, 1989, Vol.32, No.6, pp.720-738.

[19] J. Sparsø and S. Furber, "Principles of Asynchronous Circuit Design: a System Perspective," Kluwer Academic Publishers, 2001.

[20] W.J. Bainbridge and S.B. Furber, "Delay Insensitive System-on-chip Interconnect Using 1-of-4 Data Encoding," *in Proc. Int'l Symp. on Advanced Research in Asynchronous Circuits and Systems*, Mar. 2001, pp.118-126.

[21] M. Dean, T. Williams, and D. Dill, "Efficient Selftiming with Level Encoded 2-phase Dual-rail(LEDR)," in Carlo H. Sequin, editor, *Advanced Research in VLSI: Proc. 1991 UC Santa Cruz Conf.*, MIT Press, 1991, pp.55-70.

[22] M.H. Oh and D.S. Har, "Low Delay-power Product Current-mode Multiple Valued Logic for Delay-Insensitive Data Transfer Mechanism," *IEICE Trans. Fundamentals*, Vol.E88-A, No.5, May 2005, pp.1379-1383.

[23] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavgno, and A. Yakovelv, "PETRIFY: A Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers," *IEICE Trans. on Information and Systems*, Vol.E80-D, No.3, Mar. 1997, pp.315-325.

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

[25] K.Y. Yun and D.L. Dill, "Automatic Synthesis of Extended Burst-mode Circuits. II. (Automatic Synthesis)," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol.18, No.2, Feb. 1999, pp.118-132.

[26] M.Y. Agyekum and S.M. Nowick, "A Cycle-based Decomposition Method for Burst-mode Asynchronous Controllers," *Proc. Int'l*

Symp. Advanced Research in Asynchronous Circuits and Systems, Mar. 2007, pp.129-142.

[27] S.M. Nowick, "Automatic Synthesis of Burst-mode Asynchronous Controllers," Ph.D. dissertation, Stanford University, Mar. 1994.

[28] A.J. Martin, "Compiling Communicating Processes into Delay-insensitive VLSI Circuits," *Distributed Computing*, Vol.1, 1986, pp.226-234.

[29] J.C. Ebergen, "A Formal Approach to Designing Delay-insensitive Circuits," *Distributed Computing*, Vol.5, No.3, 1986, pp.226-234.

[30] E. Brunvand and R.F. Sproull, "Translating Concurrent Programs into Delay-insensitive Circuits," *Proc. of the 1987 IEEE Int'l Conf. on Computer Aided Design*, 1989.

[31] K. van Berkel et al., "A Fully Asynchronous Low-power Error Corrector for the Dcc Player," *Proc. of the 1994 IEEE Int'l Conf. on Solid-state Circuits*, 1994, p.TA5.4.

[32] TiDE Manual, Handshake Solutions, 2009.

[33] HASTE Manual, Handshake Solutions, 2009.

[34] J. Sparsø and J. Staunstrup, "Delay Insensitive Multi-ring Structures," *Integration, the VLSI Journal*, Vol.15, No.3, Oct. 1993, pp.313-340.

[35] K.M. Fant and S.A. Brandt, "NULL Convention Logic: a Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," *Int'l Conf. on Application Specific Systems, Architectures, and Processors*, 1996, pp.261-273.

[36] J. Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou, "Desynchronization: Synthesis of Asynchronous Circuits from Synchronous Specifications," *IEEE Trans. on Computer-Aided Design*, Vol.25, No.10, Oct. 2006, pp. 1904-1921.