# 비동기 순차회로 파형의 흐름도 변환에 의한 VHDL 코드 생성 알고리즘에 관한 연구

우경환* · 이 용희** · 임 태영*** · 이천희****
우송공업대학* · 하이닉스 반도체** · 한국전자통신연구원*** · 청주대학교****
전화 : 043-229-8448 / 핸드폰 : 019-415-8448/ FAX : 043-213-6392

# A Study on the VHDL Code Generation Algorithm by the Asynchronous Sequential Waveform Flow Chart Conversion

Woo, Kyong-Hwan* · Lee, Yong-Hui** · Lim, Tae-Yong*** · Yi, Cheon-Hee****
Woosong Technical College* · Hyniex Semiconductor** · ETRI*** · Chong Ju University****
E-Mail : yicheon@chongju.ac.kr

**요 약**

본 논문에서는 IP(Intellectual Property)와 IP 간의 핸드쉐이킹 신호를 비동기 논리회로로 대체 하도록 할 수 있는 인터페이스 논리의 생성 방법에 대하여 기술한다. 특히 핸드쉐이킹을 위하여 레벨형 입력과 펄스형 입력이 혼합된 비동기 타이밍 파형만 제시되었을 경우 이 파형을 흐름도로 변환시키고 변환된 흐름도에 의하여 VHDL 코드로 대체하는 새로운 "파형 변환 알고리즘:Wave2VHDL"을 제안한다. 또한 제안된 알고리즘으로부터 추출한 VHDL 원시 코드를 기존의 국내외 CAD 툴(Tool)에 적용함으로서 IP 인터페이스를 위한 비동기식 전자회로가 생성됨을 확인하고 시뮬레이션 결과와 제시된 타이밍도가 일치함을 증명한다.

**Abstract**

In this paper we described the generation method of interface logic which can be replace between IP and IP handshaking signal with asynchronous logic circuit. Especially, we suggest the new "Waveform Conversion Algorithm : Wave2VHDL", if only mixed asynchronous timing waveform suggested which level type input and pulse type input for handshaking, we can convert waveform to flowchart and then replaced with VHDL code according to converted flowchart. Also, we assure that asynchronous electronic circuits for IP interface are generated by applying extracted VHDL source code from suggested algorithm to conventional domestic/abroad CAD Tool, and then we proved that coincidence simulation result and suggested timing diagram.

## I. 서 론

프로세서가 포함된 시스템을 설계 할 때 CPU 또는 MPU와 주변장치간의 제어신호를 핸드쉐이크 하는 방법은 시스템 운영 소프트웨어(Operating System)내부에서 해당신호들을 정의하고 제어하여 처리하는 소프트웨어 방법과, MPU와 주변장치 사이에 인터페이스용 집적회로를 설계하여 하드웨어로 처리하는 방법이 있다.

소프트웨어로 처리하는 방법은, MPU가 포트들의 동작상태를 수시로 감시해야 하기 때문에 소프트웨어의 추가부담이 발생하게되며, 감시 및 제어에 소요되는 시간만큼 지연시간이 발생된다.

하드웨어로 처리하는 방법은, 주변장치의 입력신호 또는 상태 신호들을 이용하여 실시간으로 MPU를 직접 인터럽트 시키는 비동기식 신호처리 방법을 생각할 수 있다. 이 방법은 주 클럭(Main

Clock)을 사용하지 않고, 명령이나 데이터의 도착에 근거하여 명령을 실행하는 데이터-흐름 회로 구조가 있으며, 감시루틴을 사용하는 소프트웨어적 신호처리보다 고속으로 동작 할 수 있는 장점이 있다. 또한 클럭을 사용하지 않으므로 명령이나 데이터 신호가 없을 때에는 IC가 전력을 거의 사용하지 않게 되어 저 전력 소자를 구현할 수 있는 장점이 있다[1],[2].

한편, 하드웨어로 처리하는 방법을 사용하여 내장형 시스템(Embedded System) 또는 SOC(System On a Chip)를 구현하려면, 그림 1과 같이 타이밍 파형만 제시된 상태에서 요구(Request)와 응답 및 핸드쉐이크에 해당되는 회로 부분을 직접 설계해야한다. 특히, CPU와 IP(Intellectual Property) 또는 IP와 IP 사이의 인터페이스를 위한 타이밍 파형만 알고, 여기에 필요한 논리회로를 구현해야할 때는 시스템 설계자가 직접 설계해야할 필요성이 더욱 높아진다.

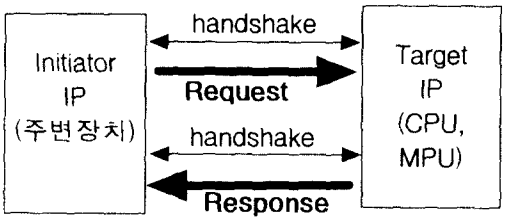



그림 1. IP 인터페이스 블럭도

이와같은 IP 인터페이스 논리회로는 주 클럭이 없는 상태의 비동기 순차회로로 구성해야 구현 할 수 있으나, 이를 용이하게 설계하기가 어려운 단점이 있으므로 이 문제를 해결하기 위하여 본 논문에서는 새로운 "파형변환 알고리즘 :Wave2VHDL"을 제안하고 이 알고리즘을 IP 인터페이스 논리회로의 설계에 적용하여, 파형변환 알고리즘의 구현 결과인 VHDL 코드를 LODECAP, Renoir99 등의 CAD용 설계 툴(tool)를 이용하여 유효성을 입증하고자 하였다[3].

## II. 파형에 의한 비동기 펄스형 순차회로 합성 알고리즘의 고찰

IP 인터페이스에 있어서 일반적인 요구에 관련된 타이밍 파형을 그림 2에 나타내었다.

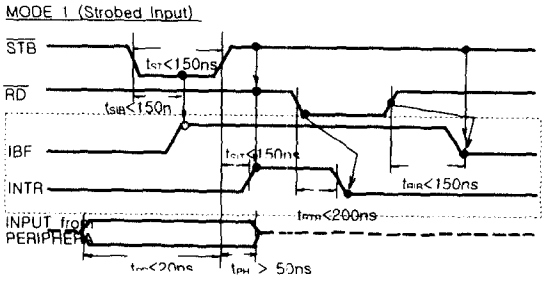



그림 2. IP 인터페이스의 요구 타이밍도

그림 2처럼 제시된 타이밍 파형만 가지고, IP 인터페이스 논리회로와 유사한 비동기 펄스형 순차회로를 합성할 수 있는 알고리즘 중에 "현재상태와 다음상태"를 상태도로 표현하여, 천이도를 작성한 후 논리방정식을 추출하고, 마스터 플립플롭용 회로를 합성하여 슬레이브 플립플롭용 회로를 추가로 합성하는 기법이 있다. 이 방법은 주 클럭을 반드시 인가해야하기 때문에 IP 인터페이스 논리회로의 설계에는 적용할 수 없다는 문제점이 있다[4].

제시된 타이밍 파형만으로 펄스형 순차회로를 합성할 수 있는 알고리즘으로 Janus 알고리즘이 있다. 이 알고리즘은 두 IP에 대한 이벤트(Event) 중심으로 기술된 타이밍도를 입력으로 받아들여 이를 이벤트 그래프로 변환시킨 후 SR-래치와 D-FF으로 구성된 세 종류의 템플리트를 동기와 비동기 특성에 맞추어 적용시키는 합성 절차를 거쳐서 논리회로를 생성시키는 방법이다[5].

또한 "진리치 비교 알고리즘(TRUth values COMparison algorithm : TRUCOM)"이라고 명명한 "제시된 펄스형 순차회로 파형을 합성 할 수 있는 알고리즘"도 제안되어있다[6]. 이 알고리즘은 펄스형 순차회로의 합성 알고리즘으로서, 레벨형 입력과 펄스형 입력이 혼합된 비동기 회로에서, 사용하려는 플립플롭을 설정한 후 파형 동작 순서에 따라서 신호와 매칭 시킬 단자를 가정하여 진리치를 대입한 후, 가정한 단자의 논리 방정식을 구현하여 회로를 생성하는 설계 알고리즘이다. 위에서

제시한 알고리즘들은 본 논문에서 구현하려는 비동기식 회로의 합성이 가능하기는 하지만, 단계별로 논리 회로도를 그려나가야 하기 때문에 디자인 룰(Rule)이 바뀔 경우에 타이밍 문제점들을 해결하기가 어렵다. 따라서 본 논문에서는 새로운 "파형 변환 알고리즘"을 제안하여 이 문제점들을 해결하고자 하였다.

본 논문에서는 그림 2처럼 제시된 인터럽트 발생기의 타이밍도를 예로 적용 방법을 기술하였다. 이 타이밍도는 주변장치와 IP, CPU와 IP 사이를 인터페이스 시키는 일반적인 파형이며, 주 클럭이 없는 비동기식으로 구동되어야한다.

그림 2를 간략히 설명하면, 인터페이스 회로는 주변장치 IP로부터 스트로브(STB) 입력을 받아서 그림의 점선 부분과 같이 IBF(Interrupt Buffer Full)를 출력하고, 읽기(RD) 신호의 상승 엣지에서 IBF를 종료 해야한다. 그림에서 STB와 RD가 핸드쉐이킹되며, IBF가 요구(Request) 신호가 된다. 또한 STB 신호는 레벨 입력(Level Input)이고, RD는 엣지입력(Edge Input)이다.

## III. 파형 변환 알고리즘의 제안

### III-1. 파형변환 알고리즘의 개념

파형변환 알고리즘(Asynchronous sequential **wave**form **to VHDL** code Generation algorithm by the flow chart conversion : 이하 Wave2VHDL로 함)이란 핸드쉐이킹을 위한 비동기식 순차회로 파형이 제시되었을 때, 이 파형을 흐름도로 변환시키고 변환된 흐름도에 의하여 VHDL 코드로 대체하는 알고리즘으로서, "레벨형 입력과 펄스형 입력이 혼합된 비동기 파형에서, 파형의 진리표를 작성하고 성분을 분류한 후 파형의 동작 순서에 따라서 개별신호들을 흐름도로 바꾸고, 흐름도의 단계별로 VHDL 코드를 대입시킨 후 VHDL 소스를 완성함으로써 CAD 툴을 활용하여 비동기식 전자회로를 생성" 시킬 수 있는 전자회로 설계에 알맞은 알고리즘이다. 이 알고리즘의 체계도를 그림 3에 나타냈다.

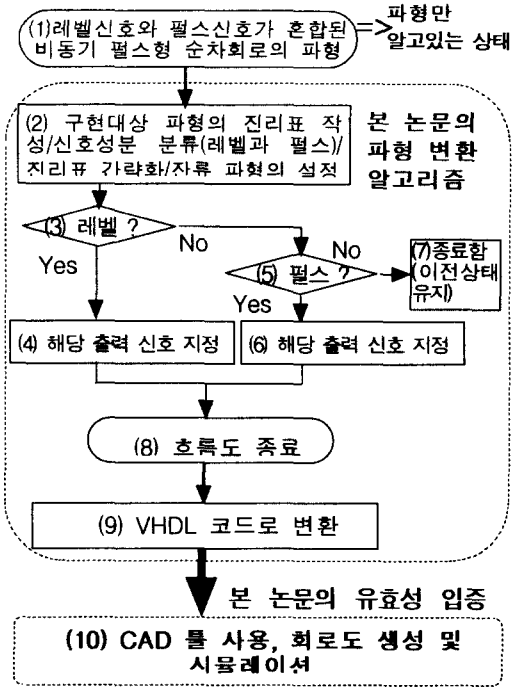


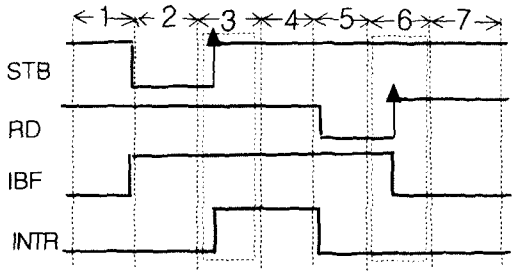

그림 3. wave2VHDL 알고리즘의 체계도

### III-2. 파형변환 알고리즘의 절차 및 구현

두개의 IP 사이에서 데이터를 주고받는 인터페이스 논리회로 설계 방법에 대하여 그림 2와 같이 필요한 파형만 제시된 상태를 예로하여 5단계의 절차를 거쳐서 설명한다.

**1) 제 1단계 : 구현대상 파형의 라이징 엣지 구간을 확대한 진리표 작성**

그림 4와 같이 그림 2에서 제시된 파형에서 출력의 변화가 발생하는 구간 및 라이징 엣지 구간을 확대하여 구획한 후 파형 동작순서를 정하고 이에 따른 진리표를 작성한다.




| | | 파형 동작순서 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 입력 | STB | 1 | 0 | ↑ | 1 | 1 | 1 | 1 |
| | RD | 1 | 1 | 1 | 1 | 0 | ↑ | 1 |
| 출력 | IBF | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| | INTR | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

그림 4. 라이징 엣지 구간을 확대한 진리표

**2) 제 2단계 : 신호성분 분류(레벨과 펄스)**

구현대상 파형의 진리표에서, 라이징 엣지를 기준으로 출력신호의 변화가 발생하였을 경우에는 펄스로, 이전상태를 유지하고 있을 때 레벨로, 그림 5처럼 신호성분을 분류한다.

그림에서 파형 동작순서 2와 3을 살펴보면 출력 IBF는 파형 동작순서 3에서 파형 동작순서 2의 이전상태를 유지하고 있으므로 "레벨"이며, 파형 동작순서 5와 6은 "1"에서 "0"으로 신호의 변화가 발생하였으므로 "펄스"로 분류한다.

**3) 제 3단계 : 진리표 간략화 및 잔류 파형의 설정**

그림 6과 같이 2 단계에서 분류한 "레벨과 펄스가 공존" 하는 파형 동작순서의 우측방향들을 연속적으로 비교하여, 동일한 출력값의 파형 동작순서는 찾아서 생략한다.

| | | 파형 동작순서 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 입력 | STB | 1 | 0 | ↑ | 1 | 1 | 1 | 1 |
| | RD | 1 | 1 | 1 | 1 | 0 | ↑ | 1 |
| 출력 | IBF | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| | INTR | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

그림 5. 출력 신호성분 분류

예를 들어 동작순서 3과 4를 동일 출력행 끼리 XOR하여, 결과값이 "0"이면 동일하고 "1"이면 상이하다. 결과적으로 출력이 동일하므로 파형 동작순서 4를 생략할 수 있다.

| | | 파형 동작순서 | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 입력 | STB | 1 | 0 | ↑ | 1 | 1 | 1 | 1 |
| | RD | 1 | 1 | 1 | 1 | 0 | ↑ | 1 |
| 출력 | IBF | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| | INTR | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

그림 6. 동일 출력치 비교

**4) 제 4단계 : 잔류 파형의 흐름도 작성**

3 단계 잔류 파형에 대하여 출력신호당 1개씩의 흐름도를 작성한다. 이 때 레벨 신호를 흐름도의 시작 지점(Start point)으로 하며, 우측방향으로 진행하도록 작성한다. 그림 8은 그림 7의 잔류파형에 대한 진리표를 흐름도로 작성한 예를 나타내었다.

| | | 파형 동작순서 | | | | |
|---|---|---|---|---|---|---|
| | | 2 | 3 | 5 | 6 | |
| 입력 | STB | 0 | ↑ | 1 | 1 | (펄스신호) |
| | RD | 1 | 1 | 0 | ↑ | (레벨신호) |
| 출력 | INTR | 0 | 1 | 0 | 0 | |

그림 7. INTR에 대한 입력신호 성분

그림 7에서 레벨 신호는 파형 동작순서 5이며, 여기서부터 흐름도가 시작된다. 따라서 RD가 "0" 이면 INTR이 "0"이 되어 (1)번째 비교문으로 작성한다. 이 후 RD가 "1" 일 경우, 파형 동작순서 3에서 STB가 라이징 될 때만 IBF가 "1"이 되어, (2)번째 비교문으로 작성한다. 이 이외의 경우는 파형 동작순서 6과 2가 되며, 이 때는 이전 상태를 유지하며 종료하게된다.

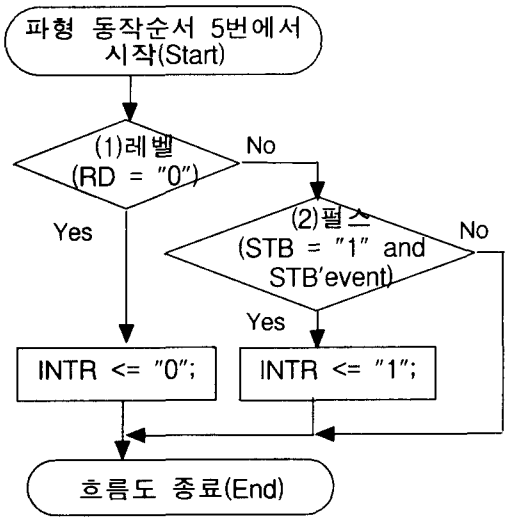



그림 8. INTR 진리표를 흐름도로 작성한 예

**5) 제 5단계 : 흐름도로부터 VHDL 작성**

제 4 단계의 흐름도로부터 VHDL 파일을 작성한다. 이 때 각 흐름도당 1개씩의 Process문을 작성하되, 파형 동작순서가 우선하는 레벨신호가 있는 흐름도부터 작성한다. 그림 9에 VHDL 파일로 작성한 예를 나타냈다.

## IV. 파형 변환(Wave2VHDL) 알고리즘의 시뮬레이션

본 논문의 파형변환 알고리즘은 멘토(Mentor) 사의 르노아르99(Renoir99)에서 그림 8 처럼 제안된 흐름도를 입력하여 VHDL로 합성 후 모델심(ModelSim)으로 시뮬레이션을 수행하였으며, 시놉시스(Synopsys)사의 브이에스에스(VSS)로도 시뮬레이션을 수행하였다.

또한 ETRI의 0.8um SOG 디자인 룰을 적용한 로드캡(LODECAP)의 VHDL 합성기에서 그림 9처럼 제안된 VHDL 코드를 입력함으로서, 그림 2에서 제시된 타이밍도에 대하여 그림 10과 같은 인터페이스 회로도를 합성 할 수 있었다. 또한 로드캡의 파형편집기를 이용하여 INTR에 대하여 그림 11와 같은 시뮬레이션 결과를 얻을 수 있었으며, 이는 그림 2에서 제시된 타이밍도와 동일함을 입증 할 수 있었다. 한편, 그림 10의 회로도는, 각 입력신호(STB, RD, ack, wr)들이 F/F의 클럭단자에 접속되고, 구동되므로 주 클럭과 관계없이 비동기로 동작함을 알 수 있으며, 각 출력들은 입력신호(STB, RD, ack, wr)들에 대하여 리얼타임으로 동작함을 알 수 있다.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity doubleEdge is
   port ( stb : IN  std_logic;
          rd : IN  std_logic;
          intr : OUT std_logic);
end doubleEdge;
architecture doubleEdge of doubleEdge is
begin
   process (rd, stb)
   begin
      if stb = '0' then
         intr <= '1';
      elsif (rd = '1' and rd'event) then
         intr <= '0';
      end if;
   end process;
end doubleEdge;
```

그림 9. VHDL 파일의 예

## V. 결론

본 논문에서는 IP와 IP 간에 핸드쉐이킹을 위하여 레벨형 입력과 펄스형 입력이 혼합된 비동기 파형만 제시했을 때, 파형을 흐름도로 변환하고 이것을 VHDL 코드로 대체시키는 알고리즘으로, 이 알고리즘을 파형변환 알고리즘(Wave2VHDL)이라고 제안하였다. 파형변환 알고리즘으로부터 완성된 VHDL 소스코드를 LODECAP의 VHDL 합성기(VHDL Synthesis)에 적용하여 IP 인터페이스를 위한 비동기 전자회로가 생성됨을 확인하였고, 파형 편집기에 적용한 시뮬레이션 결과와 제시된 타이밍도가

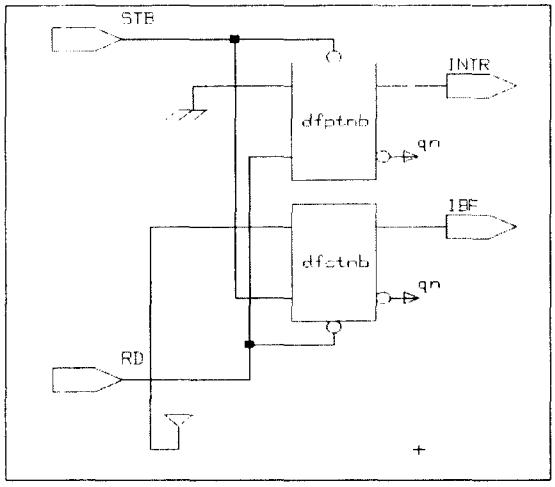



그림 10. 요구 타이밍의 인터페이스 회로도

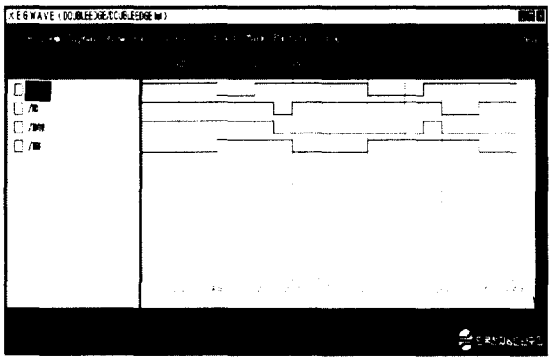


그림 11. 시뮬레이션 결과

일치함으로 알고리즘의 유효성을 입증하였다.

본 논문에서 제안한 파형변환 알고리즘은 비교적 간단하면서 전자회로 구현이 매우 용이하며, 결과물이 VHDL 코드로 생성되어 주 클럭을 생략할 수 있기 때문에, 칩 구현시 디자인 룰이 바뀌어도 손쉽게 설계할 수 있을 것으로 기대되므로 IP화하기도 적합하다고 판단된다.

앞으로의 연구 과제는 파형변환 알고리즘을 이용하여 제시된 타이밍도의 파형들만 입력하는 것으로도 VHDL 코드가 생성될 수 있도록 프로그램화하는 방향으로 연구가 진행되어야 하겠다.

## 참고 문헌

[1] C.L. Seitz, *System timing in Introduction to VLSI Systems*, Addison-Wesley, 1980.

[2] A. Bellaouar and M.I. Elmasry, "Low-power Digital VLSI Design." Kluwer Academic Publisher, 1995.

[3] Y. H. Bae, etc. "VHDL based ASIC design CAD system suppotting manual & automatic design," World Korean Scientist Workshop, July 1996.

[4] J. F. Wakerly, *Digital Design Principles and Practices*, Prentice-Hall, Englewood Cliffs, N.J., P265-295, 1990.

[5] Gaetano Borriello, and Randy H. Katz, "Synthesis and Optimization of Interface Transducer Logic," Proc. ICCAD '87, pp.274-277, (Event Graph)

[6] 이천희 외 1명, "인터럽트 발생기를 사용한 접속 비트 전환식 양방향 접속장치의 설계," 대한전자공학회 36권 C편 7호, P17-26, 1999년 7월