論文 86-23-2-16

VLSI의 논리 설계 자동화를 위한 ASM도표와 SDL

(ASM Chart and SDL for VLSI Logic Design Automation)

趙 仲 彙*, 鄭 正 和*

(Joung Hwee Cho and Jung Wha Chong)

要 約

본 논문에서는 VLSI 의 논리 설계 자동화 위한 새로운 ASM(Algorithmic State Machine) 도표와 새로운 하드웨어 기술 언어를 제안한다.

설계 요구에 대한 동작 특성의 표현을 위해 종래의 ASM도표를 수정 제안하며, 이 도표의 각 기호에 1대1 대응하며 계층적 설계가 가능한 새로운 하드웨어 기술 언어인 SDL(Symbolic Description Language)을 제안한다. 다음에 실제의 설계 예에 대하여 ASM 도표로 표현하고 SDL로 기술한 다음 SDL 하드웨어 컴파일러에 이용하여 게이트 레벨의 논리 회로도를 얻을 수 있음을 보인다.

Abstract

THis paper proposes a new algorithmic state machine (ASM) chart and a new hardware description for automatic logic design of VLSI. To describe the behavioral characteristics of the design specification, the conventional ASM chart is mofified, and a new hardware description language, SDL, is proposed. The SDL is one-to-one correspondent to the proposed ASM chart symbol, and can be used in a hierachical design of VLSI. As a design example, we obtain a logic circuit diagram of gate lebel utilizing a SDL hardware compiler after drawing an ASM chart and describing in SDL.

I. 序 論

VLSI의 집적도가 증가함에 따라 디지탈 시스템 설계에 대한 정확성과 설계시간을 단축하기 위해 논리 설계자동화 대한 요구가 증가되고 있다.[1]

종래에는 설계 요구 조건을 설계자가 분석하여 상태도표 또는 진리치 표등으로 표시한 후 레이아웃(layout)의 입력 데이타인 게이트 레벨의 논리 회로도를 얻는 설계 방식이 사용되었으나 이 방식에 의하면 설계 면적의 최적화 목표는 어느 정도 달성할 수 있으나 설계의 잘못을 포함할 가능성이 높을뿐만 아니라 설계 시간이 매우 길게 된다는 단점이 지적되고있다.[1,2]

최근 이러한 문제점을 해결하기 위해 설계하고자 하는 디지탈 시스템의 동작 또는 구조 특성에 따라 1) Algorithmic 레벨 2) Processor, memory, switch (PMS) 레벨 3) Instruction 레벨 4) Register-transfer 레벨(RTL) 5) Gate 레벨 6) Circuit 레벨에서[2] 하드웨어 기술 언어(HDL:Hardware Description Language)로 기술한 후 그보다 아래 레벨의 기술 표현식을 얻는 논리 설계자동화 대한 연구가 발표되고 있다.[1,10] 특히, 설계하고자 하는 디지탈 시스템의 동작 특성을 register-transfer 레벨에서 기술하여 게이트 레벨의 논리 회로도를 얻는 연구가 가장 활발히 진행되고 있는데 이와같은 하드웨어 기술 언어로는 AHPL[3,5],

*正會員, 漢陽大學校 電子工學科
(Dept. of Elec. Eng., Hanyang Univ.)

接受日字 : 1985年 12月 12日

DDL[4,12] 및 CDL[2,10] 등이 발표되어 있다. 이와같은 하드웨어 기술 언어들은 설계요구 조건으로부터 직접 기술하기가 어려우며[2] 이미 기술한 것에 대한 이해와 수정이 어렵다는 단점이 지적되고 있다.[2,7] 최근에는, 하드웨어의 동작 특성을 register-transfer 레벨에서 소프트웨어의 흐름도(flowchart)인 PAD(Problem Analysis Diagram)[6]로 표시한후, 이를 그래픽(graphic) 터미날에 입력하여 PASCAL언어로 변환한 다음 게이트 레벨의 논리 회로도를 얻는 연구가 발표되었으나[7] 소프트웨어의 흐름도와 언어로는 하드웨어의 병렬 동작을 표현하기가 매우 어렵다는 단점이 지적되고 있다.[2,7]

한편, CLARE[8]에 의하여 FSM(Finite State Machine)의 설계 요구 조건을 표시할 수 있는 ASM(Algorithmic State Machine) 도표가 제안되어 하드웨어의 설계에 이용되고는 있으나[8] 기호의 제약성으로 인하여 하드웨어의 병렬 동작을 표현할 때 중복 표현되는 부분이 많으며, 더욱 하드웨어 기술 언어와 직접 연관되어 사용되지는 못하고 있다.

따라서, 본 논문에서는 register-transfer 레벨에서 하드웨어의 병렬 동작 특성의 표현이 용이하며 하드웨어 기술 언어와 직접 연관되어 사용될 수 있도록 새로운 ASM 도표 기호를 제안한다. 다음에, 디지탈 시스템 제어부의 기술이 용이하도록 앞에서 제안한 ASM 도표 기호에 1대1 대응되는 비 절차적 언어 구조를 지니며, 대규모 디지탈 시스템을 여러 모듈로 나누어 계층적 기술이 가능하게 함으로써 이해와 수정이 용이한 새로운 하드웨어 기술 언어인 SDL(Symbolic Description Language)을 제안한다. 또한, 본 논문에서 제안하는 새로운 ASM도표와 SDL의 효용성을 보이기 위해 실제의 예에 적용 설명하고 SDL하드웨어 컴파일러[11]에 입력하여 게이트 레벨의 논리 회로도와 등가인 NDL(Network Description Language) 표현식을 얻을 수 있음을 보인다.

Ⅱ. ASM 도표

ASM 도표는 CLARE[8]에 의해 FSM의 동작 특성을 표현하기 위한 흐름도로 발표되었으나 기호의 제약성으로 인하여 하드웨어의 모든 동작을 표현하기가 용이하지 못하여 소규모 회로 설계에 이용되고 있다.[9]

따라서, 본 논문에서는 register-transfer 레벨에서 하드웨어의 제어부 동작에 대한 표현과 이해 및 수정이 용이하고, 데이타 전달부의 표현도 함께 행할수 있으며 또한 하드웨어 기술 언어와 직접 연관되어 디지탈 시스템 설계에 이용될 수 있는 새로운 ASM 도표의 기호를 정의하면 다음과 같다.

1. 상태 기호(state symbol)

상태 기호는 FSM의 동작중 시스템 클럭의 소비를 필요로 하는 동작을 표현하기 위한 것으로 그림 1과 같이 표시한다. 기호 내부에는 그 상태에서 발생하는 출력들을 기술하는데 그 상태에서 조건에 대한 판단을 먼저 행하여 그에 따라 동작하면서 출력이 없는 경우는 공백으로 둔다. 기호 이름은 Qi(i ≥ 1인 정수)로 기호 왼쪽에 표시한다.

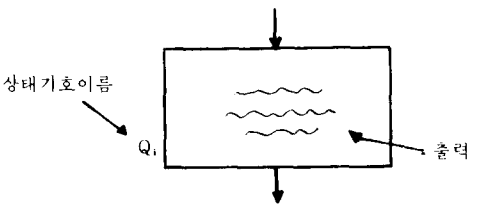


그림 1. 상태 기호
Fig. 1. State symbol.

2. 조건 판단 기호(condition decision symbol)

조건 판단 기호는 여러 조건들이 각각 참(true)인 경우 다음 기호로 천이함을 표현하기 위한 것으로 그림 2와 같은데 하드웨어의 병렬 동작 표현이 용이하도록 조건은 동시에 2개이상 참일 수도 있으며 항상 참임을 표시하는 상수1일 수도 있다는 특징을 갖는다. 기호 이름은 Ci(i≥1인 정수)로 기호 왼쪽에 표시한다.

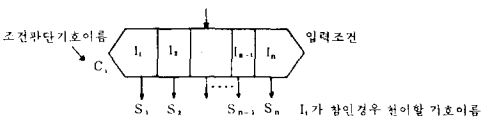


그림 2. 조건 판단 기호
Fig. 2. Condition decision symbol.

3. 조건 출력 기호(conditional output symbol)

조건 출력 기호는 2.의 입력 조건이 참인 경우 시스템 클럭을 소비하지 아니면서 하드웨어의 출력이 존재하는 경우를 표시하기 위한 것으로 그림 3과 같이

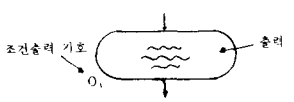


그림 3. 조건 출력 기호
Fig. 3. Conditional output symbol.

표시한다. 기호 내부에는 상태 기호와는 달리 적어도 1개 이상의 출력이 존재하여야 하며, 이 기호에의 입력은 종래의 ASM 도표 기호와는 달리 조건 판단 기호뿐만아니라 조건 출력 기호로부터도 입력될수 있다는 특징을 갖는다. 기호 이름은 Oi(i≥1인 정수) 로 기호 왼쪽에 표시한다.

1. 에서 정의한 상태 기호는 상태 기호 내부의 출력에 관계없이 시스템 클럭 1개를 소비하면서 동작하나, 3. 의 조건 출력 기호는 이와 연결된 상부의 상태 기호와 같은 시스템 클럭에서 동작하는 특징을 갖도록 정의한다. 따라서, 하나의 상태 기호와 그의 출력 경로에 연결된 조건 판단 기호 및 조건 출력 기호로 구성되는 ASM 도표 일부분을 ASM 블럭(block)이라 정의하는데 각 ASM블럭이 1개의 시스템클럭 내에서 동작하므로 ASM 블럭 내의 하드웨어 제어부 회로는 1개의 플립 플롭과 조합 논리 회로로 형성된다.

본 논문에서 제안하는 새로운 ASM도표의 효용성을 보이기 위해[3]의 그림5. 25회로에 대하여 종래의 ASM 도표와 본 논문에서 제안하는 ASM도표로 표시하면 그림4와 같으며 (b)의 점선 내부가 ASM 블럭의 한 예이다. 한편, 기호의 이름은 하나의 ASM블럭 내에는 동일한 것이 없도록 정하며 상태 기호 바로 뒤에 나오는 조건 판단 기호의 이름은 C_i으로 한다.

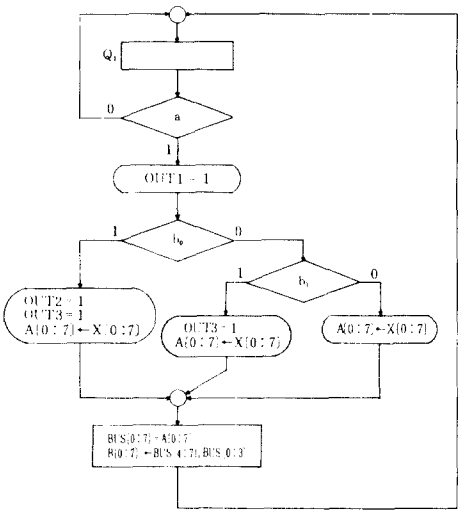


그림 4. (a) Clare's의 ASM chart
Fig. 4. (a) ASM chart of clare's.

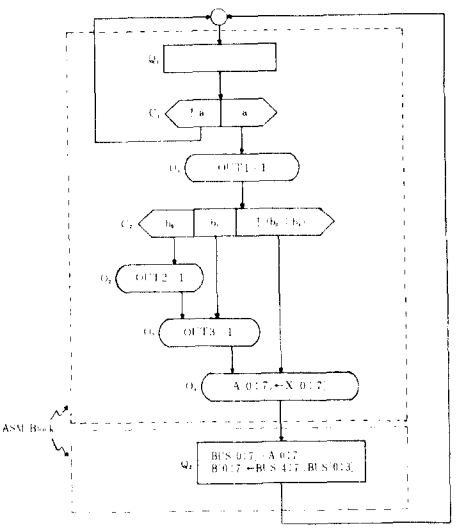


그림 4. (b) 본 논문의 ASM chart
Fig. 4. (b) ASM chart of this apper.

Ⅲ. 하드웨어 기술 언어 : SDL

하드웨어 기술 언어는 하드웨어의 모든 동작 특성을 쉽게 기술할 수 있어야 하며 구문이 간단하여 기술, 이해 및 수정이 용이하여야 한다. 또한, 대규모 디지탈 시스템 설계가 가능하도록 모듈단위로 나누어 기술하는 계층적 설계가 가능하여야 하며, 컴퓨터 처리(processing)가 용이하도록 구문이 설정되어야 한다.

따라서, 본 논문에서는 위의 모든 조건을 만족하며 Ⅱ에서 정의한 ASM 도표 기호와 1대1 대응하며 비절차적 언어 구조를 지니며 디지탈 시스템의 제어부 및 데이타 전달부를 함께 기술하는 새로운 하드웨어 기술 언어인 SDL 구문을 설정 제안하면 다음과 같다.

SDL은 시스템 선언부와 시스템 실행부로 크게 구성되는데 시스템 선언부에서는 시스템의 입·출력및 메모리에 해당하는 레지스터, 선 및 버스(bus)를 선언 기술하며 시스템 실행부는 SYSBEGIN : 으로 시작하여 계층적 설계가 가능하도록 모듈단위로 나누어 기술한다. 각 모듈에 대한 기술은 모듈 선 언부와 모듈 기술부로 다시 구성되는데 선언부에는 기술하고 있는 모듈의 입·출력 및 메모리에 해당하는 레지스터, 선 및 버스등을 선언한다. 모듈 기술부는 BEGIN : 으로 시작하여 ASM블럭 단위로 기술한다. 각 ASM블럭의 기술은 도표 기호와 1대1 대응하도록 기호 이름을 표기하고 내용을 기술하는데 먼저 ASM블럭의 시작점인 상태 기호 이름을 기술한다. 다음에 상태 기호 뒤에 따른 출력들을 나열하고 다음 기호로 천이함을 표기하는데 상태 기호 뒤에 바로 조건 판단 기호가 있는 경우는 출력들만을 나열한다. 기호 이름이 조건 판단

기호인 경우는 입력 조건들을 나열하고 다음 기호로 천이함을 표시한다. 또한, 기호 이름이 조건출력인 경우는 그에 따른 출력들을 나열하고 다음 기호로 천이함을 표시한다.

한편, SDL에서 디지탈 시스템의 데이타 전달부를 기술하기 위한 연산자와 그에 대한 의미는 표1과 같다.

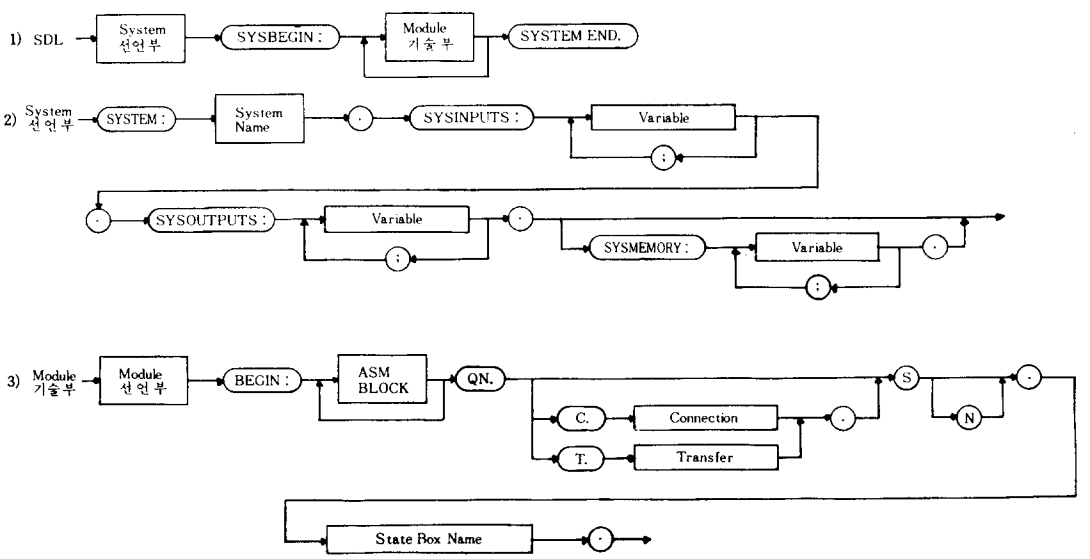
표1의 기호 ←은 오른쪽 신호(입력)가 기본 지연시간(unit delay time)후에 왼쪽 신호(출력)에 전달됨을 표시하기 위한 것으로 실제 회로도의 입·출력 신호 사이에는 D 또는 J-K 플립 플롭등의 지연 소자가 형성된다. 기호 =는 입력이 지연 시간 없이 출력에 연결됨을 표시하기 위한 것으로 입·출력 신호 사이에는 조합 논리 회로가 형성된다. 한편, 본 논문에서 사용하는 macro함수는 전체 칩(chip) 면적의 최소화를 위해 설계가 완료되어 있는 것을 시스템 설계가 완료된 후에 대체한다. 또한, 시스템 클럭에 관계없이 동작하는 연결(=)과 모듈 내의 임의의 ASM 블럭이 동작할 때 항상 동작하는 전달(←)은 각 모듈 기술부의 종료 표시인 QN. 다음에 기술하며 마지막으로 모듈이 시스템 클럭에 의해 동기가 시작되는 상태 기호 이름을 기술한다. 위의 기술 규칙에 따라 본 논문에서 제안하는 SDL의 구문 그래프를 도시하면 그림5와 같다.

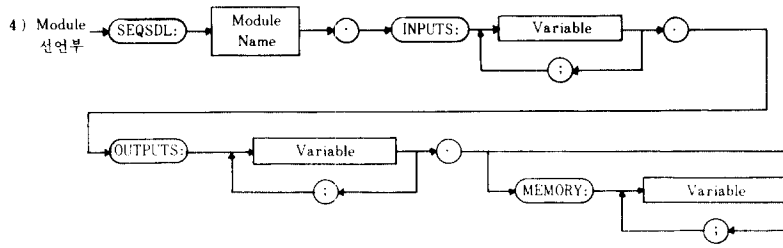
그림6.(a)는 그림4.(b)의 ASM 도표에 대한 SDL 기술이며 그림 6.(b)는[3]의 p.242에 기술된 AHPL에 대한 SDL 기술이다.

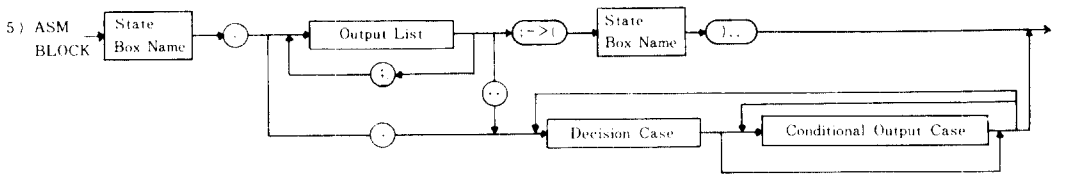
표 1. SDL 연산자
Table 1. SDL Operator.

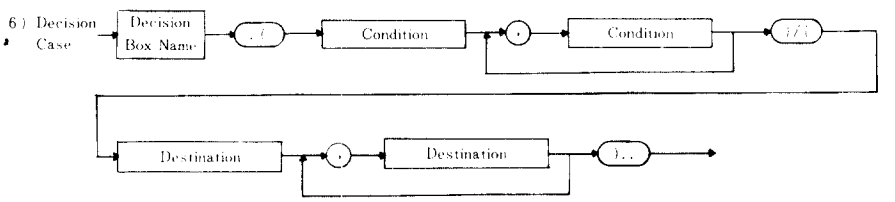
연 산 자	의 미
;	상태 기호 또는 조건 출력 기호의 구분 및 출력과 천이 내용이 구분 표시
%	macro 함수의 연산부 구분표시
,	1) 레지스터의 비트 연결표시 2) 조건 판단기호의 조건과 천이기호 각각의 구분표시
/	조건 판단 기호에 대한 조건과 천이내용 사이의 구분표시
$n_1 : n_2$	레지스터의 비트 시작점(n_1)과 끝점 표시($n_1 > n_2$)
$n_1 \# n_2$	n_2(0 또는 1)가 n_1개 연속된 데이타의 표시
*1레지스터	레지스터의 모든 비트를 논리적으로 AND한 결과로서 0 또는 1을 표시
+1레지스터	레지스터의 모든 비트를 논리적으로 OR한 결과로서 0 또는 1을 표시
!	NOT
&	AND
|	OR
←	오른쪽 신호(입력)를 왼쪽 신호(출력)에 전달
=	오른쪽 신호(입력)를 왼쪽 신호(출력)에 연결
→	무조건 오른쪽 기호로 천이

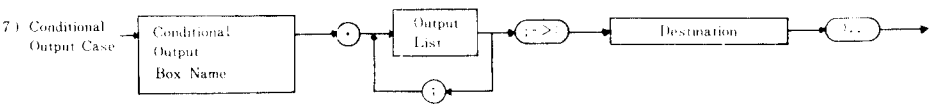
그림6.(c)는 그림4에 대한 DDL 기술인데 종래의 하드웨어 기술 언어는 분기의 종료점이 상태 기호이므로 상태 사이에 조건이 많은 경우 중복 기술이 많아 DDL보다 SDL에 의한 기술이 매우 간단하다.

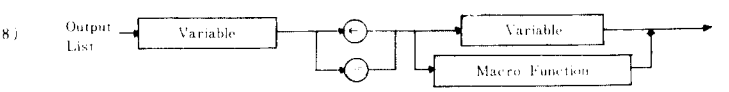


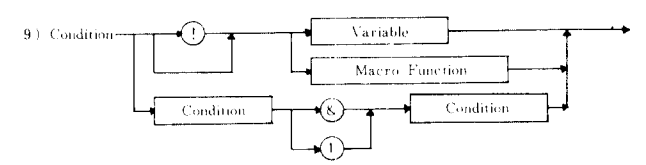

4) Module 선언부
SEQSDL:
Module Name
INPUTS:
Variable
OUTPUTS:
Variable
MEMORY:
Variable

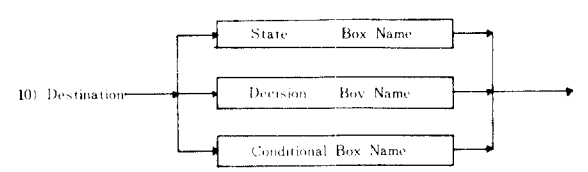

5) ASM BLOCK
State Box Name
Output List
State Box Name
Decision Case
Conditional Output Case

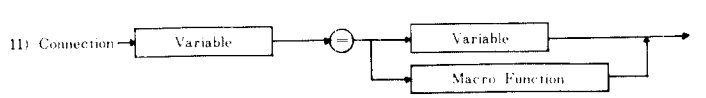

6) Decision Case
Decision Box Name
Condition
Condition
Destination
Destination

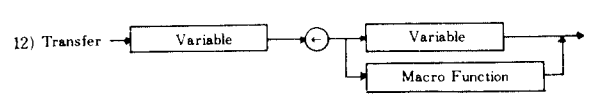

7) Conditional Output Case
Conditional Output Box Name
Output List
Destination


8) Output List
Variable
Variable
Macro Function


9) Condition
Variable
Macro Function
Condition
Condition


10) Destination
State Box Name
Decision Box Name
Conditional Box Name


11) Connection
Variable
Variable
Macro Function


12) Transfer
Variable
Variable
Macro Function

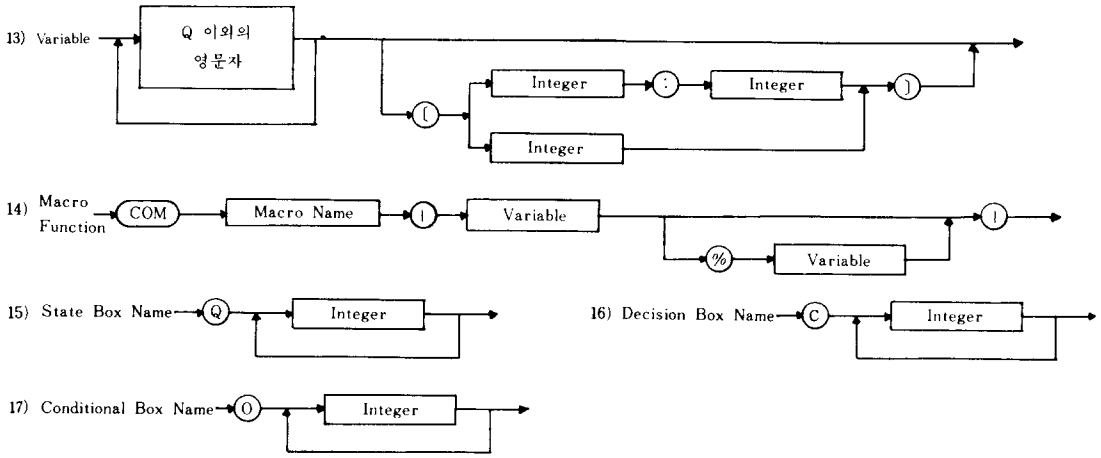


그림 5. SDL 구문 그래프
Fig. 5. SDL syntax graph.

한편, 본 논문에서 제안하는 register-transfer 레벨의 SDL기술은 게이트 레벨의 논리 회로도를 얻는 SDL 하드웨어 컴파일러의 입력으로 사용된다. 일반적으로 하드웨어 컴파일러의 출력은 부울 함수 또는 진리치 표로 표시되는데 SDL 하드웨어 컴파일러[11]의 출력은 게이트 레벨 회로도와 등가인 NDL 표현식인데 NDL표현식은 다음과 같다.

출력신호이름=gate이름 입력신호1 입력신호2

본 논문에서 제안하는 SDL의 효용성을 보이기 위하여 자연 및 메모리 소자로 D 플립 플롭을, 조합 논리 회로 소자로는 2-입력 AND 및 OR 게이트와 NOT 게이트를 사용하는 SDL하드웨어 컴파일러에 그림6(a)의 SDL을 입력하여 NDL로 표시하면 표 2

```
SEQSDL:NO DELAY.
INPUTS:a;b0;b1;X[8].
OUTPUTS:B[8];OUT1;OUT2;OUT3.
MEMORY:A[8];BUS[8].
BEGIN:
Q1..
        C1. (a, ! a)/(O1, Q1)..
        O1. OUT1=1; -> (C2)..
        C2. (b0, b1, ! (b0 | b1))/ (O2, O3, O4)..
        O2. OUT2=1; -> (O3)..
        O3. OUT3=1; -> (O4)..
        O4. A[0:7] <- X[0:7]; -> (Q2)..
Q2. BUS[0:7]=A[0:7];B[0:7] <- BUS[4:7], BUS[0:3]; -> (Q1)..
QN.
SN. Q1.
```

그림 6. (a) 그림 4(b)의 SDL 기술
Fig. 6. (a) SDL Description of Fig. 4 (b).

```
SEQSDL:MULTISHIFT.
INPUTS:a;b;X[6].
OUTPUTS:Z;LOOK;A[0:5].
MEMORY:A[18];CNT[3].
BEGIN:
Q1. A[0:17] <- 0, A[0:16]..
    C1. (a, ! a)/(Q3, Q2)..
Q2. A[0:17] <- 0, A[0:16]; -> (Q1)..
Q3. CNT[0:2] <- a, b, 0; A[0:11] <- X[0:5], A[0:5]..
    C1. (b, ! b)/(Q1, Q4)..
Q4. A[0:17] <- 1, A[0:16]; CNT[0:2] <- COMINC(CNT[0:2])..
    C1. (* / CNT[0:2], ! (* / CNT[0:2]))/(Q5, Q4)..
Q5. LOOK=1..
    C1. (! b, b)/(O1, O2)..
    O1. A[0:5] <- A[6:11] & A[12:17]; -> (Q1)..
    O2. A[0:5] <- A[6:11]; A[6:17] <- X[0:5], A[0:5]; -> (Q1)..
QN.
C. Z=A[17].
SN. Q1.
```

그림 6. (b) SDL 기술의 다른 예
Fig. 6. (b) Another example of SDL description.

```
⟨STATE⟩
Q1:
    | *a&b0 * | OUT1=1, OUT2=1, OUT3=1, A[0:7] <- X[0:7], ->Q2.,
    | *a&¬b0&b1 * | OUT1=1, OUT3=1, A[0:7] <- X[0:7], ->Q2.,
    | *a&¬b0&¬b1 * | OUT1=1, A[0:7] <- X[0:7], ->Q2.,
    | *¬a * | ->Q1.,
Q2:BUS[0:7]=A[0:7], B[0:7] <- BUS[4:7] | | BUS[0:3], ->Q1..
⟨END⟩
```

그림 6. (c) Fig. 4의 DDL 기술 실행부
Fig. 6. (c) DDL description execution unit of Fig. 4.

와 같다. 표 2(a)는 하드웨어의 제어부에 대한 NDL이며 (b)는 데이타 전달부에 대한 NDL인데 이를 논리 회로도로 도시하면 그림 7과 같다. 앞의 SDL에서 정의한바와 같이 ASM블럭이 Q_1과 Q_2에 대한 2개이므로 제어부에 대한 지연 소자는 2개이며, 데이타 전달부에서 기본 지연 시간후에 입력이 출력에 전달(←)되는 것은 Q_2에서 비트 연결 연산자(,)에 의한 2회와 O_4에서 1회로 지연 소자가 3개 요구된다. 또한, Q_1에 대응하는 ASM 블럭내의 조건 출력 기호뒤에 나열된 출력들은 조합 논리 회로를 형성함을 알 수 있어 본 논문에서 제안하는 ASM 도표와 SDL의 효용성을 확인하였다.

Ⅳ. 結　　論

본 논문에서는 VLSI의 논리 설계 자동화를 위해 설계 요구 조건을 표현하는 흐름도와 이에 대응하는 하드웨어 기술 언어를 제안하였다.

설계 요구에 대한 register-transfer 레벨에서의 동작 특성 표현이 용이하도록 종래의 ASM도표 기호를 수정하여 새로운 ASM 도표 기호를 제안하였다. 또한, 이 도표의 각 기호에 1대1 대응하며 register-transfer 레벨에서 설계하고자 하는 하드웨어의 모든 동작에 대한 기술과 이해 및 수정이 용이함은 물론, 설계 시스템을 여러개의 모듈로 나누어 설계할 수 있어 대규모 디지탈 시스템 설계에 이용될 수 있는 새로운 하드웨어 기술 언어인 SDL을 제안하였다. 다음에, 실제의 디지탈 시스템에 대하여 SDL로 기술한 후 SDL 하드웨어 컴파일러에 입력하여 게이트 레벨의 논리 회로도와 등가인 NDL 표현식을 얻을 수 있어 본 논문에서 제안하는 ASM 도표와 SDL의 효용성을 보였다. 따라서, 본 논문에서 제안하는 ASM 도표와 SDL을 사용하여 VLSI를 설계하는 경우 설계 시간이 매우 단축되며 설계의 정확성을 보장할 수 있어 설계 비용이 크게 줄어들 것으로 기대된다.

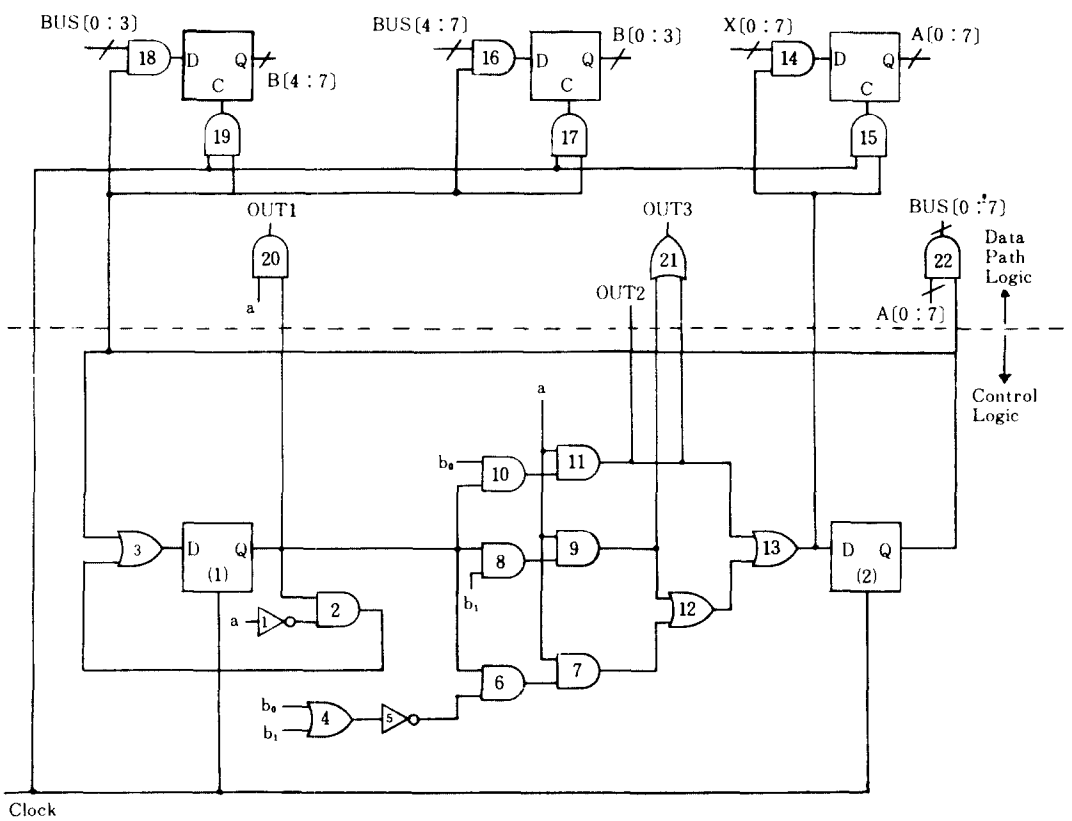


그림 7. 게이트 회로도
Fig. 7. Gate diagram.

표 2. (a) 제어부의 NDL 표현
Table 2. (a) NDL description of conts control logic.

NETWORK	DESCRIPTION	LANGUAGE FOR CONTROL	LOGIC
Q1 D F/F DATA INPUT			
1	= NOT	a	
2	= AND	Q1	1
3	= OR	2	Q2
Q2 D F/F DATA INPUT			
4	= OR	b0	b1
5	= NOT	4	
6	= AND	Q1	5
7	= AND	6	a
8	= AND	Q1	b1
9	= AND	8	a
10	= AND	Q1	b0
11	= AND	10	a
12	= OR	7	9
13	= OR	12	11

표 2. (b) 데이타 전달부의 NDL 표현
Table 2. (b) NDL description of data path logic.

NETWORK	DESCRIPTION	LANGUAGE FOR DATH PATH	LOGIC
A[0 : 7]		D F/F DATA INPUT	
14	= AND	X[0 : 7]	13
A[0 : 7]		D F/F CLOCK INPUT	
15	= AND	13	CLK
B[0 : 3]		D F/F DATA INPUT	
16	= AND	BUS[4 : 7]	Q2
B[0 : 3]		D F/F CLOCK INPUT	
17	= AND	Q2	CLK
B[4 : 7]		D F/F DATA INPUT	
18	= AND	BUS[0 : 3]	Q2
B[4 : 7]		D F/F CLOCK INPUT	
19	= AND	Q2	CLK
OUT 1		DATA INPUT	
20	= AND	Q1	a
OUT 2		DATA DATA INPUT	
11			
OUT 3		DATA INPUT	
21	= OR	9	11
BUS[0 : 7]		DATA INPUT	
22	= AND	A[0 : 7]	Q2

앞으로의 연구과제로는 SDL연산자에 대한 보완 연구와 이에따른 좀더 효율적인 하드웨어 컴파일러에 대한 연구가 계속되어야 할 것이다.

參 考 文 献

[1] Parker, A.C., "Automated synthesis of digital systems," *IEEE Trans. Design & Test*, vol. 1 no. 4, Nov. pp. 75-81, 1984.

[2] Shiva, S.G., "Computer hardware description language-a tutorial," *Proceedings of IEEE*, Dec. pp. 1605-1615, 1979.

[3] Hill, F.J. and Peterson, G.R. *Digital Systems: Hardware Organization and Design.*, 2nd ed., Wiley Press, 1978.

[4] Duley, J.R. and Dietmeyer, D.L., "A digital system design language," *IEEE Trans. Computers*, vol. C-17, no. 9, Sept. pp. 850-861, 1968.

[5] Swanson, R., Navabi, Z. and Hill, F.J., "An AHPL Compiler/Simulator System," *Proc. 6th Texas conference on computing systems*, Austin, Tex., 1977.

[6] Y. Futamura, T. Kawai, H. Horikoshi and M. Tsutsumi, "Design and implementation of programs by problam analysis diagram (PAD)," *Trans. IPSJ*, vol. 21, no. 4, July pp. 259-267, 1980.

[7] G. Odawara, J. Sato and M. Tomita, "A symbolic functional description language," *Proc. IEEE ACM 21st Design Automation Conf.*, pp. 73-80, 1984.

[8] Clare, C.R., *Designing Logic Systems Using State Machines, McGraw-Hill*, New York, pp. 8-9, 1973.

[9] Hill, F.J. and Peterson, G.R., *Digital Logic and Microprocessors*, Wiley Press, 1984.

[10] Shiva, S.G., *Computer Design and Architecture*, Little, Brown and Company, 1985.

[11] 조중휘, "Hardware Compiler for Symbolic Description Language," 한양대학교 CAD 연구실 보고서, 1985.

[12] 박성범, "논리설계 자동화를 위한 하드웨어 컴파일러에 관한 연구," 한양대학교 석사학위 논문, 1985.