PLD 설계용 툴 개발에 관한 연구

김 희 석† 원 충 상††

요 약

본 논문에서는 디지틀 회로를 PLD 소자로 설계하는 PLD 설계용 툴인 PLD Designer을 개발하였다. PLD Designer는 FSM(finite state machine)의 상태수가 제한적(20개 미만)일 경우, 상태표로부터 부울식을 추출할 수 있는 상태 그래픽 편집기(state graphic editor)와 상태 그래픽 편집기에서 생성된 부울식에 적합한 PLD 소자(PAL16R4, PAL22V10, GAL16V8 등)를 선정하여 핀 할당을 실현하는 핀 맵 편집기(pin map editor)로 구성되어 있다. 또한 핀 맵 편집기는 fuse map, checksum, JEDEC 화일을 생성하며 PLD 디바이스 구현에 사용된다. 생성된 부울식을 검증하기 위해 테스트 벡터(test vector) 생성 알고리듬을 개발하였으며 PLD Designer에 의해 생성된 JEDEC 화일과 PALASM의 JEDEC 화일과 비교한 결과 동일함을 입증하였다.

A Study on the Development of a Tool for PLD Design

Hee Suk Kim† and Chung Sang Won††

ABSTRACT

In this paper, we have developed a PLD Designer which is a design tool for digital circuits design using PLD device. PLD designer consists of a state graphic editor to extract boolean equations from state table within 20 states of FSM and a pin map editor to assign pin map for PLD device(PAL16R4, PAL22V10, GAL16V8, etc), which is suitable for extracted boolean equations. Also pin map editor generates a necessary JEDEC file to implement PLD device by using fuse map and checksum algorithm. To verify extracted boolean equation, we have developed simulation test vector generation algorithm. The results of JEDEC files generated by PLD designer is same with the results of JEDEC files generated by PALASM.

1. 서 론

디지틀 시스템 설계에 사용되었던 소자는 TTL이나 CMOS logic의 MSI, SSI와 같은 소자들이었으나, 최근에는 PLA, PAL, GAL등과 같은 PLD(Programmable Logic Device)[1, 2, 3] 소자의 출현으로 설계시간 단축과 설계의 정확성 및 최적회로를 설계할 수 있는 PLD 설계용 툴(tool)에 관한 연구가 활발히 진행되고 있다[8, 9].

PLD 설계 자동화 툴은 현재 ABEL, CUPL, PALASM, AMAZE등과 같은 설계용 툴들이 사용되고 있다[4, 5].

그러나 기존의 PLD 설계 자동화툴은 FSM (finite state machine) 기술언어인 HDL(hardware description language)의 기술형태가 다양하여 일반 사용자가 사용하기 불편하며, 시스템의 동작 특성을 나타내는 상태도(state graph)를 자동적으로 HDL로 변환하는 기능이 없어 시스템의 동작특성을 일일이 기술해야 하는 단점이 있으며, JEDEC 화일 생성과정 또한 공개되지 않고 있다[6, 9].

따라서 본 논문에서는 이러한 단점을 보완한 PLD 설계용 툴인 PLD Designer을 개발하였다. 개발한 PLD Designer의 구성은 상태수가 제한

· 이 논문은 교육부의 1993년부 학술연구조성비에 의해 연구되었음.

† 정 회 원 : 청주대학교 전자공학과 교수
†† 정 회 원 : 충주산업대학교 컴퓨터공학과 부교수

논문접수 : 1994년 5월 7일, 심사완료 : 1994년 9월 16일

적(20개 미만)일 경우, 입력을 상태도로 받아들이기 위해 상태 그래픽 편집기(state graphic editor)를 개발하여 상태도를 상태표로 변환한 뒤, 상태표에서 부울식(boolean equation)을 추출하였다.

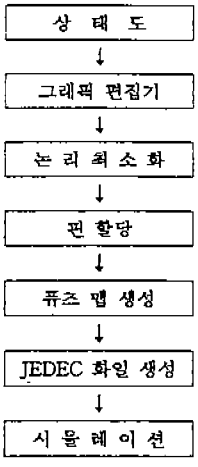
또한, 상태 그래픽 편집기에 의해 추출된 부울식을 논리최소화 툴을 이용하여 최소화하고 부울식의 입 출력갯수를 고려하여 여기에 적합한 PLD 소자(PAL16R4, PAL16L8, PAL16R8, PAL22V10, GAL16V8)를 선택하는데 필요한 data book기능을 추가 보완하고, 선택된 PLD 소자에 핀(pin)을 할당할 수 있는 핀 맵 편집기(pin map editor)를 개발하였다.

핀 할당 후, 논리최소화를 거쳐 생성된 부울식을 검증하기 위해 테스트 벡터(test vector) 생성 알고리듬을 개발하여 파형 형태(wave form)로 출력하였다[7].

검증을 마친 부울식을 PLD 소자(PAL 또는 GAL)로 구현하기 위해 퓨즈 맵(fuse map)과 JEDEC 화일, checksum 생성 알고리듬을 개발하여 PLD 소자로 구현하였다.

2. PLD을 이용한 디지틀 회로 설계

PLD로 설계된 회로는 설계자가 즉시 검증할 수 있고, 수정 및 설계 시간과 설계 비용면에서 유리하다. 특히, PLD중에서 PAL(Programmable Array Logic), GAL(Generic Array Logic)등은 규칙적인 구조를 가지고 있기 때문에 설계하기가 편리하며 현재 널리 사용되고 있다.



(그림 1) 전체 설계도
(Fig. 1) Total flow chart

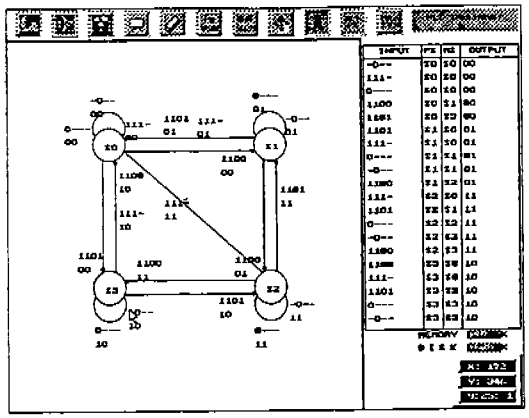
따라서, 본 논문에서는 PAL과 GAL을 이용한 PLD 설계용 툴인 PLD Designer를 개발하였으며 설계 과정은 (그림 1)과 같다.

2.1 상태 그래픽 편집기

PLD 소자를 이용하여 디지틀 시스템을 설계하는 방법은 동작특성을 PLD 설계 언어(text 형태)인 상태 CHDL로 기술하는 방법과 상태 그래픽 편집기를 이용하여 작성하는 방법으로 나눌 수 있는데, 본 논문에서 개발한 상태 그래픽 편집기로 작성하는 경우, 상태도를 그래픽 모드(graphic mode)에서 그린 후, 마우스(mouse)의 click 상태에 따라 상태표를 추출하게 되고 이 상태표에서 부울식을 추출하게 된다.

상태 그래픽 편집기의 환경은 아이콘 메뉴(icon menu) 방식을 사용하였으며 (그림 2)와 같다.

(그림 2)의 상태 그래픽 편집기에서 상태도를 그리는 방법은 주(main) 화면 중 아이콘 검출기에서 상태 생성기를 선택하여 상태를 그리고, 상



(그림 2) 상태 그래픽 편집기를 이용하여 설계한 스텝모터 제어도
(Fig. 2) State diagram of stepper motor controller using state graphic editor

태검색기에서 마우스의 click 상태에 따라 x 좌표, y 좌표를 받아들여 상태를 그릴 때에 구조체에 저장된 데이타와 비교한 후 상태도를 완성하고, 상태표 화일 생성기에서 최종적인 상태표를 생성한다.

이러한 상태표 생성 알고리듬을 표현하면 (그림 3)과 같이 기술된다.

```
struct DATA {
      /*  상태도를 생성할 때 x 좌표, y 좌표, */
      /*  반경을 그리고 상태에 대한 정보 저장 */
}data[MAX_STATE];
struct LNE {
      /* 마우스로 click한 후 line에 대한 x 좌표, y 좌표, */
      /*      반경 그리고  상태에 대한 정보 저장        */
}line[MAX_CONDITION];

for (i=0; i < MAX_STATE; i++) {
    if (data[i] = ●)   Display_circle( );  /*  편집기내에 배치 */
    do {
        Mouse_click1_position(x1, y1, pre_state[n]);
            /* x1좌표, y1좌표, 현재 상태값들을 pre_state[n]에 저장 */
        Mouse_click2_position(x2, y2, next_state[n]);
            /* x2좌표, y2좌표, 다음 상태값들을 next_state[n]에 저장 */
        Line_display(x1, y1, x2, y2);  /* 현재 상태 -> 다음 상태 선 표현 */
        Getstring(input_x, y, 입력값);     /* 상태표에 입력값 입력 */
        Getstring(output_x, y, 출력값);    /* 상태표에 출력값 입력 */
        MAX_LINE <= MAX_LINE+1;
    }
}
Output_stream(State_table_file);

Getstring(int x, int y, char *s) {     /* 상태표의 조건 입력 */
    while( 입출력문자 입력 )
          from_graphic_editor_to_state_table_set;
}
```

(그림 3) 상태표 생성 알고리듬
(Fig. 3) State table generation algorithm

본 논문에서는 상태 그래픽 편집기의 사용예로서 스텝모터 제어회로(Stepper motor controller)를 선정하였으며 스텝모터 제어회로의 상태도를 (그림 2)의 상태 그래픽 편집기에서 그린 다음 상태테이블 알고리듬으로 상태표를 추출하면 (그림 4)와 같다.

/e1	/e2	s	d	q0	q2	q0	q2	PS	NS	q1	q3
0	-	-	-	1	1	1	1	S0	S0	0	0
-	0	-	-	1	1	1	1	S0	S0	0	0
1	1	1	-	1	1	1	1	S0	S0	0	0
1	1	0	0	1	1	1	0	S0	S1	0	0
1	1	0	1	1	1	0	1	S0	S3	0	0
1	1	0	1	1	0	1	1	S1	S0	0	1
1	1	1	-	1	0	1	1	S1	S0	0	1
0	-	-	-	1	0	1	0	S1	S1	0	1
-	0	-	-	1	0	1	0	S1	S1	0	1
1	1	0	0	1	0	0	0	S1	S3	0	1
1	1	1	-	0	0	1	1	S2	S0	1	1
1	1	0	1	0	0	1	0	S2	S1	1	1
0	-	-	-	0	0	0	0	S2	S2	1	1
-	0	-	-	0	0	0	0	S2	S2	1	1
1	1	0	0	0	0	0	1	S2	S3	1	1
1	1	0	0	0	1	1	1	S3	S0	1	0
1	1	1	-	0	1	1	1	S3	S3	1	0
1	1	0	1	0	1	0	0	S3	S2	1	0
0	-	-	-	0	1	1	0	S3	S3	1	0
-	0	-	-	0	1	0	1	S3	S3	1	0

(그림 4) 스텝모터의 상태표
(Fig. 4) State table of stepper motor

(그림 4)에 생성한 상태표에서 입력변수의 출력이 '1'인 경우와 출력이 '0'인 경우의 부울식을 추출할 수 있는데 부울식 추출 알고리듬은 (그림 5)와 같다.

```
for (n=0; n <= MAX_CLICK-1; n++)   {
   if(out_array == '1') {   /* 출력이 '1'인 경우만 부울식 생성 */
     for(i=0;i <= in_size-1; i++)
       if(in_array[n][i] == '1')  /* in이 '1'인 경우 정논리 입력변수 file write */
          File_print();
       else  File_print(inv); /* in이 '0'인 경우 부논리
                                    입력변수 file write */
     for(i=0;i <= state_size-1; i++)
       if(pre_state[n][i] == '1') /* 상태가 '1'인 경우 정논리 상태변수
                                                    file write */
          File_print();
       else  File_print(inv); /* 상태가 '0'인 경우 정논리
                                 상태변수 file write */
   }
Output_stream(Boolean_equ_file);
```

(그림 5) 부울식 생성 알고리듬
(Fig. 5) Boolean generation algorithm

(그림 5)의 부울식 생성 알고리듬으로 (그림 4)의 상태표에서 다음 상태와 출력이 '1'인 경우의 부울식으로 추출하면 (그림 6)과 같다.

```
q0  =  e1 * q0 * q2
     + e2 * q0 * q2
     + /e1 * /e2 * s * q0 * q2
     + /e1 * /e2 * /s * /d * q0 * q2
     + /e1 * /e2 * /s * d * q0 * /q2
     + /e1 * /e2 * s * q0 * /q2
     + e1 * q0 * /q2
     + e2 * q0 * /q2
     + /e1 * /e2 * s * /q0 * /q2
     + /e1 * /e2 * /s * d * /q0 * /q2
     + /e1 * /e2 * /s * /d * /q0 * q2
     + /e1 * /e2 * s * /q0 * q2

q2  =  e1 * q0 * q2
     + e2 * q0 * q2
     + /e1 * /e2 * s * q0 * q2
     + /e1 * /e2 * /s * d * q0 * q2
     + /e1 * /e2 * /s * d * q0 * /q2
     + /e1 * /e2 * s * q0 * /q2
     + /e1 * /e2 * s * /q0 * /q2
     + /e1 * /e2 * /s * /d * /q0 * /q2
     + /e1 * /e2 * /s * /d * /q0 * q2
     + /e1 * /e2 * s * /q0 * q2
     + e1 * /q0 * q2
     + e2 * /q0 * q2

q1  =  /e1 * /e2 * s * /q0 * /q2
     + /e1 * /e2 * /s * d * /q0 * /q2
     + e1 * /q0 * /q2
     + e2 * /q0 * /q2
     + /e1 * /e2 * /s * /d * /q0 * /q2
     + /e1 * /e2 * /s * /d * /q0 * q2
     + /e1 * /e2 * s * /q0 * q2
     + /e1 * /e2 * /s * d * /q0 * q2
     + e1 * /q0 * q2
     + e2 * /q0 * q2

q3  =  /e1 * /e2 * /s * d * q0 * /q2
     + /e1 * /e2 * s * q0 * /q2
     + e1 * q0 * /q2
     + e2 * q0 * /q2
     + /e1 * /e2 * /s * /d * q0 * /q2
     + /e1 * /e2 * s * /q0 * /q2
     + /e1 * /e2 * /s * d * /q0 * /q2
     + e1 * /q0 * /q2
     + e2 * /q0 * /q2
     + /e1 * /e2 * /s * /d * /q0 * /q2
```

(그림 6) 상태표에 의해 추출된 부울식
(Fig. 6) Extracted boolean equation from state table

(그림 6)의 부울식을 two-level 논리 최소화 툴인 CPLAMIN[8]으로 최소화하면 (그림 7)과 같다.

```
q1=/q0
q3=/q2
q0 = /e1 * q0 + /e2 * q2 + e1 * e2 * s
     + e1 * e2 * /d * q2 + e1 * e2 * d * /q2
q2 = /e1 * q2 + /e2 * q2 + e1 * e2 * s
     + e1 * e2 * /d * /q0 + e1 * e2 * d * q0
```

(그림 7) 최소화된 부울식
(Fig. 7) Minimized boolean equation

2.2 핀 맵 편집기(PIN map editor)

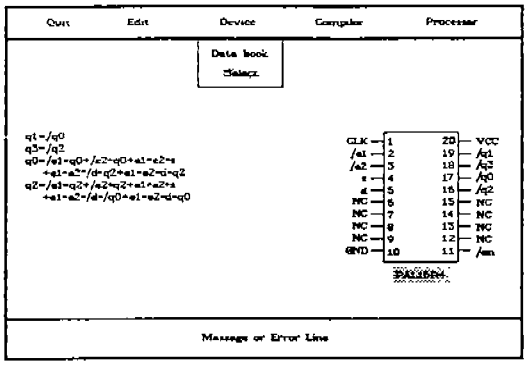
기존의 PLD 설계 툴은 핀 할당과 소자 선정을 입력 화일에서 기술하기 때문에 최소화된 부울식을 참고하는 것이 아니라 입 출력의 갯수에 맞추어서 임의로 소자를 선정한다.

그러나 본 논문에서는 핀 할당과 소자 선정을 논리 최소화를 수행한 뒤, 여기에 적합한 소자를 선정하고 핀 할당을 수행할 수 있는 핀 맵 편집기를 개발하였다.

핀 맵 편집기는 논리최소화된 부울식에 적합한 소자를 선정하기 위해 PLD 소자들의 입 출력 및 소비전력등과 정보를 가지고 있는 data book에서 소자를 선정하고 선정된 PLD 소자에 핀 할당을 하게 되는데 핀 맵 편집기는 (그림 8)과 같다.

(그림 7)의 논리최소화된 부울식을 (그림 8)의 핀 맵 편집기에서 소자 선정 및 핀 할당을 하면 부울식의 입력이 5개이고, 출력이 4개이기 때문에 여기에 적합한 PLD 소자중 PAL16R4를 선정한 후, 각 핀에 핀 할당을 수행한다.

핀 할당과정에서 NC는 핀 할당이 되지 않는 핀을 나타낸다.



(그림 8) PLD Designer에서의 핀 맵 편집기
(Fig. 8) PIN map editor for PLD Designer

2.3 퓨즈 맵(fuse map)과 JEDEC 화일 생성 알고리듬

핀 할당이 끝나고 나면 PLD 소자로 구현하기 위해 JEDEC 화일과 checksum 값을 생성하여야 한다. JEDEC 화일과 checksum 생성과정은 핀 할당된 부울식의 입력 변수와 출력 변수의 수를 고려하여 JEDEC 화일 생성 알고리듬에 의해

```
pinname[MAX_PIN]  /* PAL 소자의 핀 할당 정보 */
equation[MAX_TERM]  /* 최적화된 부울식 */
logic[MAX_ROW][MAX_COL]  /* 소자의 fuse pattern */
Input_stream(pin_assignment_file);
Input_stream(boolean_equation_file);
fuse_pattern_생성:
각_멤에_대한_index_값_설정          /* 각 핀의 연결을 알려준다. */
for ( i=0 ; i<MAX_TERM ; i++ ) {
  j = Search(equation[i], pinname),
       /* 부울식으로 된 할당 변수와 비교
  k = Search(*equation[i], pinname);
       /* 부울식의 각 literal에 해당되는
                          핀 할당 변수와 비교 */
  logic[j][k] = 'X';
     /* fuse pattern 의 지정된 교점에 'X'를 표시 */
  for ( i=0 ; i<MAX_COL ; i++ )
    if (logic[j][i] != 'X')  logic[j][i] = '-';
    for ( k=0 ; k<MAX_COL ; k++ )
         logic([k][(MAX_PIN-j-1)*8+k] = 'X';
     /* fuse blown의 수를 계산 */
    fuse_blown = ( logic[i][i] != 'X' )? fuse_blown+1 : fuse_blown;
}
Output_stream(fuse_map_file) ;   /* fuse map 화일 생성 */
for ( i=0 ; i<MAX_ROW ; i++ )
  for ( j=0 ; j<MAX_COL ; j++ ) {
    logic[i][j] =( logic[i][j] = 'X' )? '0' : '1';
  }
for ( i=0 ; i<MAX_ROW ; i++ )
  for ( j=0 ; j<MAX_COL ; j+=8 )
    /* checksum : JEDEC 의 값을 summation */
    checksum = checksum + ( (logic[i][j]*2^0) + (logic[i][j+1]*2^1)
                          + (logic[i][j+2]*2^2) + (logic[i][j+3]*2^3)
                          + (logic[i][j+4]*2^4) + (logic[i][j+5]*2^5)
                          + (logic[i][j+6]*2^6) + (logic[i][j+7]*2^7));
Output_stream(JEDEC_file);  /* JEDEC 화일과 checksum이 생성 */
```

(그림 9) JEDEC 화일 생성 알고리듬
(Fig. 9) JEDEC file generation algorithm

fuse map과 JEDEC 화일을 생성하게 되는데, JEDEC 화일중 checksum은 fuse된 상태의 합으로 나타난다.

JEDEC과 checksum 생성 알고리듬은 (그림 9)와 같다.

(그림 8)에서 PLD 소자인 PAL16R4로 핀 할당된 스텝모터 제어회로의 부울식을 JEDEC 화일 생성 알고리듬으로 구현하면 (그림 10), (그림 11)과 같은 fuse map, JEDEC 화일, 그리고 checksum값을 생성한다.

(그림 10)에서 TOTAL FUSES BLOWN은 퓨즈의 끊어진 수를 계산한 값이고, (그림 11)에서는 핀이 할당되지 않은 퓨즈는 생략하였다.

또한 스텝모터 제어회로의 핀 할당된 소자가 PAL16R4가 아니라 GAL16V8 소자로 구현한 경우, JEDEC 화일은 그림 12와 같다.

```
PAL16R4
_STEP.
 0   ----  ----  ----  ----  ----  ----  ----  ----
 1   ----  ----  --X-  ----  ----  ----  ----  ----
 8   ----  ----  ----  ----  ----  ----  ----  ----
 9   ----  ----  ----  --X-  ----  ----  ----  ----
16   X---  ----  ---X  ----  ----  ----  ----  ----
17   ----  X---  ---X  ----  ----  ----  ----  ----
18   -X--  -X--  X---  ----  ----  ----  ----  ----
19   -X--  -X--  ----  -X-X  ----  ----  ----  ----
20   -X--  -X--  ----  X-X-  ----  ----  ----  ----
24   X---  ----  ----  ---X  ----  ----  ----  ----
25   ----  X---  ----  ---X  ----  ----  ----  ----
26   -X--  -X--  X---  ----  ----  ----  ----  ----
27   -X--  -X--  --X-  -X--  ----  ----  ----  ----
28   -X--  -X--  ---X  X---  ----  ----  ----  ----
  SUMMARY
  -------
     TOTAL FUSES BLOWN    = 416
```

(그림 10) PAL16R4를 사용한 스텝모터 제어회로의 퓨즈 맵

(Fig. 10) Fuse map of stepper motor controller using PAL16R4

```
PAL16R4
_STEP.*
QP20*      /* pin 수 : 20 pin */
QF2048*    /* 전체 fuse 수 */
G0*F0*     /* G0:용단방법 F0:미지정데이타 0으로 set
L0000 11111111111111111111111111111111*
L0032 11111111101111111111111111111111*
L0256 11111111111111111111111111111111*
L0288 11111111111101111111111111111111*
L0512 01111111111011111111111111111111*
L0544 11110111111101111111111111111111*
L0576 10111011011111111111111111111111*
L0608 10111011111101011111111111111111*
L0640 10111011111010111111111111111111*
L0768 01111111111111011111111111111111*
L0800 11110111111111011111111111111111*
L0832 10111011011111111111111111111111*
L0864 10111011110110111111111111111111*
L0896 10111011111001111111111111111111*
C3458*          /* checksum */
E9D8            /* transmission checksum */
```

(그림 11) 스텝모터 제어회로의 JEDEC 화일

(Fig. 11) JEDEC file of a stepper motor controller using PAL16R4

```
GAL16V8
_STEP.*
QP20*      /* pin 수 : 20 pin */
QP2194*    /* 전체 fuse 수 */
G0*F0*     /* G0:용단방법 F0:미지정데이타 0으로 set */
L0000 11111111111111011111111111111111*
L0256 11111111101111111111111111111111*
L0512 01111111111011111111111111111111*
L0544 11110111111101111111111111111111*
L0576 10111011011111111111111111111111*
L0608 10111011111101011111111111111111*
L0640 10111011111010111111111111111111*
L0768 01111111111111011111111111111111*
L0800 11110111111111011111111111111111*
L0832 10111011011111111111111111111111*
L0864 10111011110110111111111111111111*
L0896 10111011111001111111111111111111*
L2112 00000000000001111111111111111111*
L2144 11111111111111111111111111111111*
L2176 11111111111111101*
C354A*          /* checksum */
01FC            /* transmission checksum */
```

(그림 12) GAL 16V8을 사용한 스텝모터 제어회로의 JEDEC 화일

(Fig. 12) JEDEC file of a stepper motor controller using GAL16V8

2.4 시뮬레이션(simulation)

생성된 JEDEC 화일을 PLD 소자에 프로그램하기 전에 부울식의 논리적 검증을 위해 시뮬레이션을 수행하여야 한다.

시뮬레이션을 수행하기 위해서는 클럭입력, 입출력변수, 출력파형 반복횟수등의 조건을 기술하여야 하는데, 본 논문에서는 앞의 조건들을 SET 문, CLOCK문, FOR문으로 기술하여 시뮬레이션을 수행할 수 있는 시뮬레이터[9]를 개발하였다.

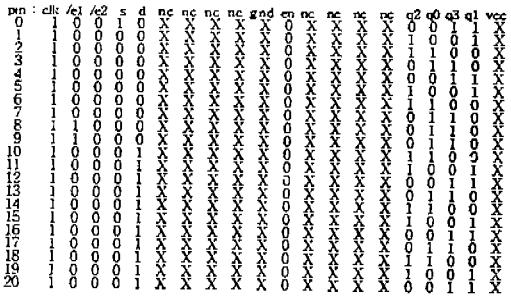
예를 들어 스텝모터 제어회로의 부울식에 대한 시뮬레이션을 수행하기 위해 작성한 소스 화일(source file)을 (그림 13)에 나타내었고, 시뮬레이션 소스 화일로부터 추출된 테스트 벡터는 (그림 14)에 나타내었다.

```
SIMULATION

SET : /CLK,EN,E1,E2,S,/D;
CLOCK : CLK;
SET : /S, /D, E1;
FOR J:=1 TO 9 {
        CLOCK : CLK;
        }
SET : E1,D;
FOR J:=1 TO 11 {
        CLOCK : CLK;
        }
TRACE_OFF
```

(그림 13) 스텝모터 제어회로를 시뮬레이션을 하기 위한 소스 화일

(Fig. 13) Stepper motor controller source file for simulation



(그림 14) 소스 화일로 부터 추출한 테스트 벡터
(Fig. 14) Test vector extracted from simulation source file

2.5 결과 비교

본 논문에서 개발한 PLD Designer에서 설계한 몇 가지 예제의 실행시간과 소자, 그리고 checksum 및 transmission checksum을 상용 툴인 PALASM과 비교한 결과를 〈표 1〉에 나타내었다. 〈표 1〉에서 PLD Designer의 실행시간이 PALASM에 비해 약간 우세함을 보였고, PALASM의 checksum과 본 논문에서 개발한 PLD Designer의 checksum 값이 동일함을 입증하였으며, transmission checksum의 값이 다른 것은 checksum이 fuse의 연결 상태를 나타내는 값인데 반해, transmission checksum값은 사용된 툴마다 생성되는 JEDEC화일의 구조가 약간씩 틀리기 때문이다.

3. 결 론

본 논문에서는 PLD 설계용 툴인 PLD Designer를 개발하였다.

PLD Designer의 구성은 상태도를 그래픽 모드에서 마우스를 이용하여 그린 다음, 이를 상태표로 변환하고 부울식을 추출해 주는 상태 그래픽 편집기를 개발하였다.

또한, 상태 그래픽 편집기에서 생성된 부울식을 논리최소화하여 여기에 적합한 범용성 소자(PAL16R4, PAL16R8, PAL10H8, PAL10L8, PAL12L6, PAL16L8, PAL12H6, PAL22V10, GAL16V8)를 선택하여 핀 할당을 수행하기 위해 소자들의 특성을 수록한 data book기능을 포함한 핀 맵 편집기를 개발하였고, PAL과 GAL의 fuse map 및 JEDEC 화일 생성 알고리듬도 개발하였다.

특히, 논리최소화 과정을 거쳐 생성된 부울식을 검증하기 위해 테스트 벡터를 생성하여야 한

〈표 1〉 PLD Designer와 PALASM과의 결과비교
〈Table 1〉 Result of PLD Designer and PALASM

예 제 명	사용 소자 (I/O)	PALASM			PLD Designer		
		실행시간 (sec)	checksum	transmission checksum	실행시간 (sec)	checksum	transmission checksum
4진 카운터	PAL16R4 (1/4)	3.5	47F0	E9F3	3	47F0	EB4C
BCD 가산기	PAL16R4 (7/5) PAL16L8 (9/8)	3.5 3.5	3E3F 90D0	E8F6 EB6B	3.1	3E3F 90D0	EA4F ECC4
Stepper motor controller	PAL16R4 (4/4) GAL16V8 (4/4)	4 4.5	3458 354A	E87F 00A3	4	3458 354A	E9D8 01FC
Digital clock	PAL22V10 ×3 (4/8)	5	1A4D	18F4	4.5	1A4D	1D4D
Asynchronous state machine	PAL16L8 (2/3)	4	25A2	E96B	3	25A2	EAC4

※ (I/O) : 입력/출력(Input/Output)

다. 본 논문에서는 시뮬레이션 조건들을 SET문과 CLOCK문으로 정의하여 테스트 벡터를 생성하여 검증하였고 파형형태로 출력하였다.

실제로 본 논문에서 개발한 PLD Designer을 이용하여 스텝모터 제어회로(PAL16R4)와 4진 카운터(PAL16R4 또는 GAL16V8), BCD 가산기(PAL16L8와 PAL16R4), Digital clock (PAL22V10)등을 PLD writer에 연결하여 구현한 결과 현재 상용 툴인 PALASM의 결과와 동일함을 입증하였다.

본 논문의 PLD Designer는 회로의 동작 특성을 text 형태인 HDL로 기술하는 기존의 툴과는 다른 그래픽 형태로 작성하기 때문에 사용자가 기술언어를 몰라도 사용할 수 있다는 장점을 가지고 있으며, 최소화된 부울식을 보면서 적합한 소자를 선택하여 핀 할당을 수행하기 때문에 효율적이다.

앞으로의 과제는 PLD 분할 알고리듬(partition algorithm), 테스트 벡터를 자동 생성할 수 있는 ATPG 알고리듬과 대규모 PLD 소자인 CPLD (Complex PLD)에 관한 연구가 절실히 요구된다.

참 고 문 헌

[1] Parag K. Lala, *Digital system design using PLD*, 1990.

[2] Martin Bolton, *Digital system design with programmable logic*, 1990.

[3] Cole, Bernards, "Programmable Logic Devices:Faster, Denser, and a Lot More of Them," Electronics:pp. 61-72, Sept. 1987.

[4] 김 회석, 박 광현, 변 상준, 류 정호, "PAL synthesis를 위한 상태 CHDL (C-based Hardware Description Language) 기술에 관한 연구," 대한전자공학회 CAD, 전자 계산, 반도체·재료 및 부품 연구회 합동학술발표 논문집, pp. 34-37, 1991.

[5] 김 회석, 변 상준, 하 재희, 정 홍구, "PAL 합성을 위한 JEDEC 화일 생성알고리듬에 관한 연구," 대한전자공학회 CAD, 전자 계산, 반도체·재료및 부품 연구회 합동학술발표 논문집, pp. 91-94, 1992.

[6] 김 회석, "상태 합성기 설계를 위한 상태 CHDL 기술및 기호 최소화 algorithm 개발," 전자공학회 논문집 제26권 5호, pp. 127-136, 1989.

[7] 김 회석; 변 상준, "논리 최소화를 위한 simplify algorithm 개선에 관한 연구," 대한전자공학회 반도체·재료및 부품 CAD 연구회 합동 학술 발표 논문집, pp. 185-188, 1989.

[8] 김 회석, 이 근만, 임 인칠, "상태합성기 (State Machine Synthesizer) 설계를 위한 상태 CHDL개발 및 Two-evel minimizer 개발에 관한 연구," 대한전자공학회 논문집-A, 제 29권 4호, pp. 83-89, 1992.

[9] 정 홍구, 김 재진, 김 성무, 변 상준, 김 회석, "PLD 설계를 위한 상태 graphic editor 개발에 관한 연구," 대한전자공학회 전자계산, 반도체-재료 및 부품, CAD 및 VLSI설계 합동 학술발표논문집, pp. 28-31, 1993.

김 회 석

1977년 한양대학교 전자공학과 졸업(학사)
1980년 한양대학교 대학원 전자공학과 졸업(공학석사)
1985년 한양대학교 대학원 전자공학과 졸업(공학박사)
1981~83년 청주대학교 전자공학과 전임강사
1983~87년 청주대학교 전자공학과 조교수
1987~88년 미국 Colorado Univ. 전자공학과 VLSI 설계실 연구원
1987~92년 청주대학교 전자공학과 부교수
1993~현재 청주대학교 전자공학과 교수
관심분야 : IC 설계환경 구축, VISI 설계자동화

원 충 상

1973년 한양대학교 전자공학과 졸업(학사)
1983년 한양대학교 대학원 전자공학과(공학석사)
1993년 청주대학교 전자공학과 박사과정 수료
1981년 충주공업전문대학 전임강사
1993~현재 충주산업대학교 컴퓨터공학과 부교수
관심분야 : IC 설계환경 구축, VISI 설계자동화