Keypad

논리회로실습

부경대 컴퓨터·인공지능공학부 최필주

목차

- 키 패드 개요
- 키 패드 동작 원리
- 키 패드의 구현

개요

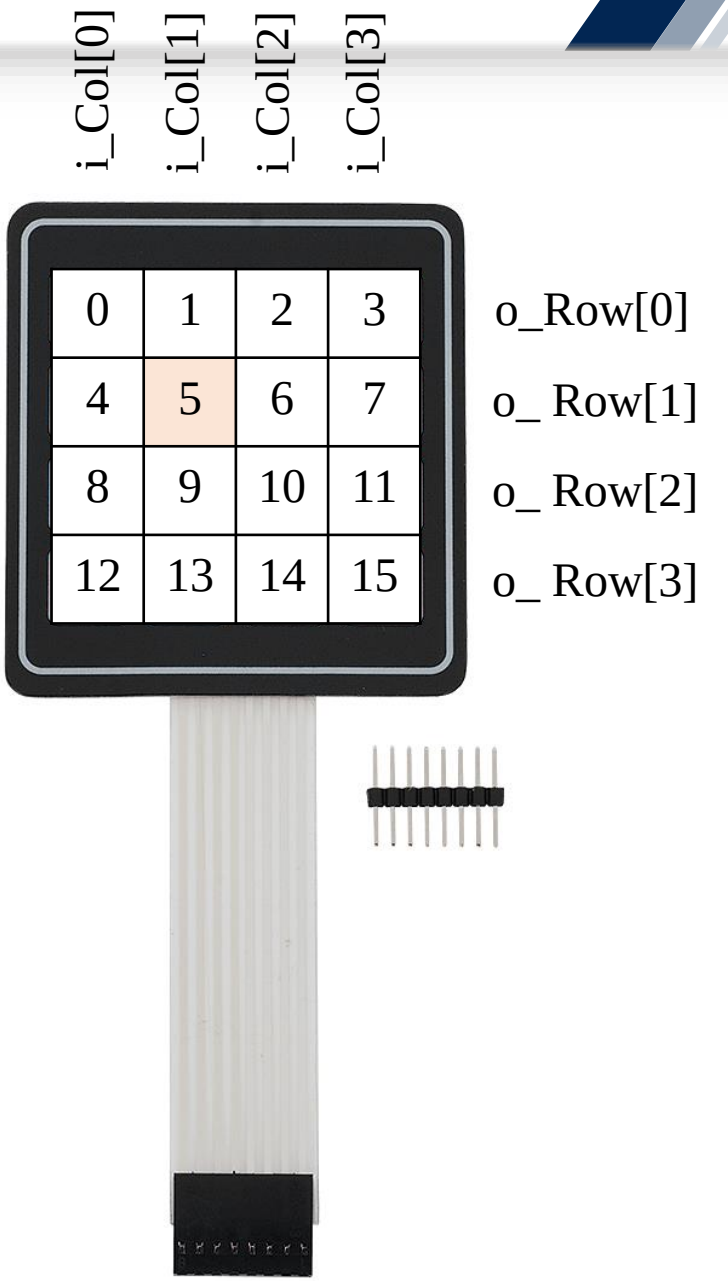
- 키 패드의 구조
  - 16개의 핀 사용 대신 열/행 각 4핀씩 사용

- 동작
  - Dot matrix
    - 각 열에 1출력
    - 각 열 모양을 4개의 행 핀에 출력
  - Keypad
    - 각 열/행에 0출력
    - 버튼 눌린 행/열에 0 입력

개요

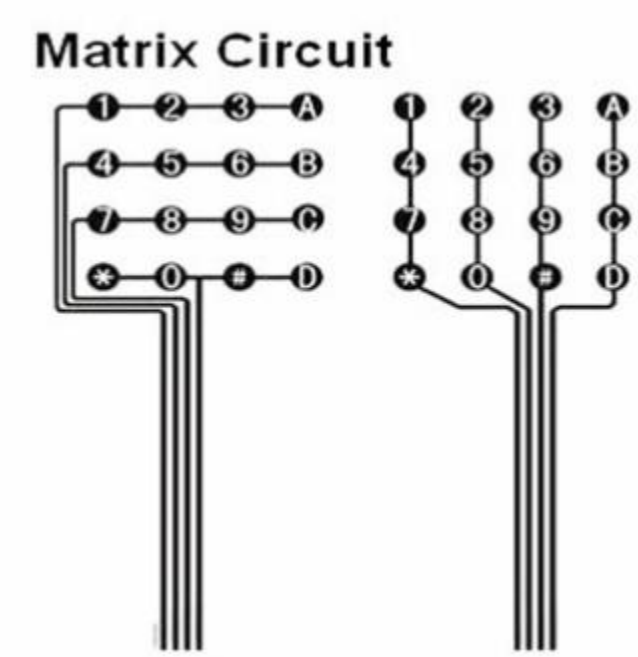
- 키 패드의 구조
  - 16개의 핀 사용 대신 열/행 각 4핀씩 사용

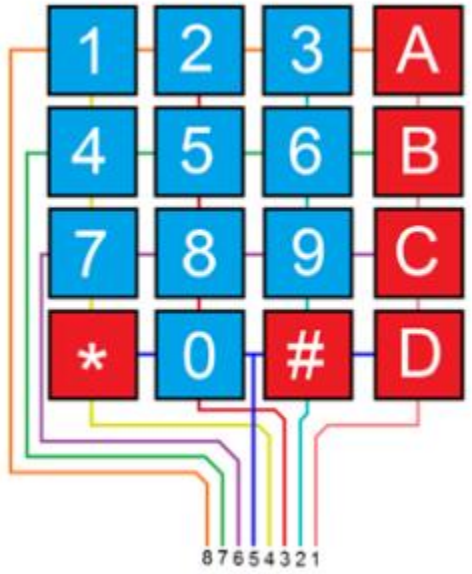
- 동작
  - 각 행/열에 0 출력
  - 버튼이 눌리면 눌린 열/행에 0 입력
  - 예) 5가 눌렸을 때
    - o_Row[1] = 0로 출력될 때 i_Col[1] = 0이 입력됨



동작 원리

- Keypad의 구조





{Row[0], …, Row[3], Col[0], …, Col[3]}

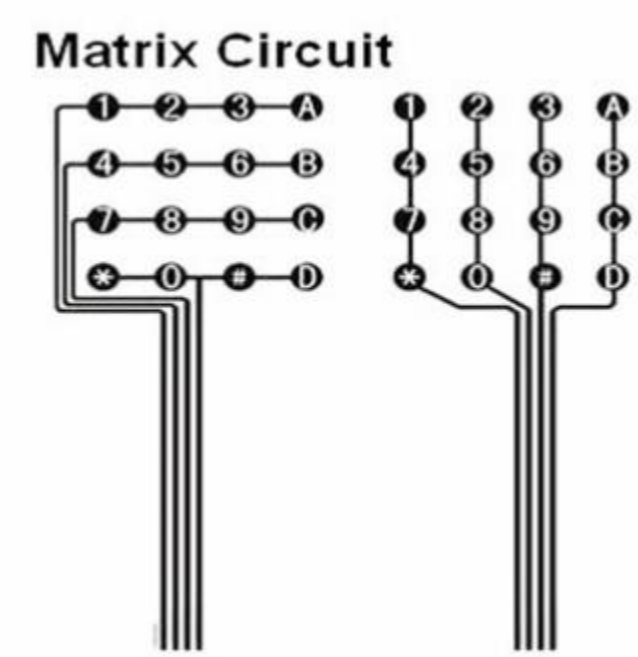
동작 원리

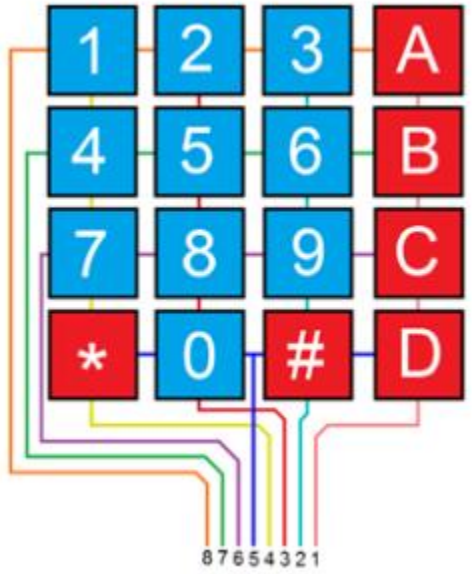
- 숫자의 결정
  - 효과적인 결정방법은?

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

=

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111





{Row[0], …, Row[3], Col[0], …, Col[3]}

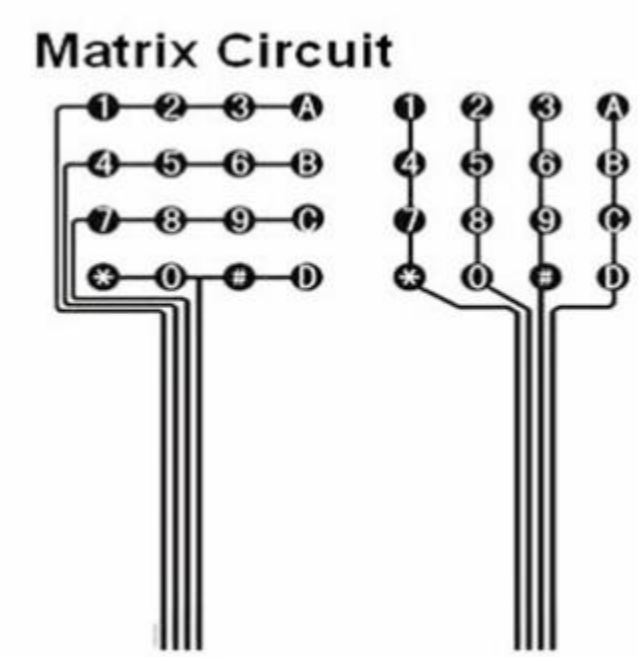
동작 원리

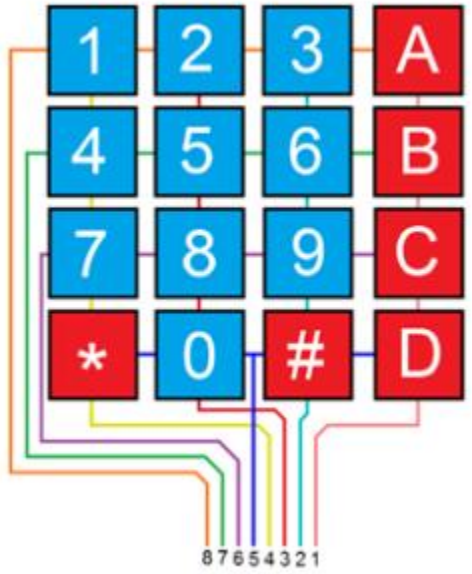
- 숫자의 결정
  - 최상위 2비트: Row에 의해 결정
  - 최하위 2비트: Col에 의해 결정

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

=

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111





{Row[0], …, Row[3], Col[0], …, Col[3]}

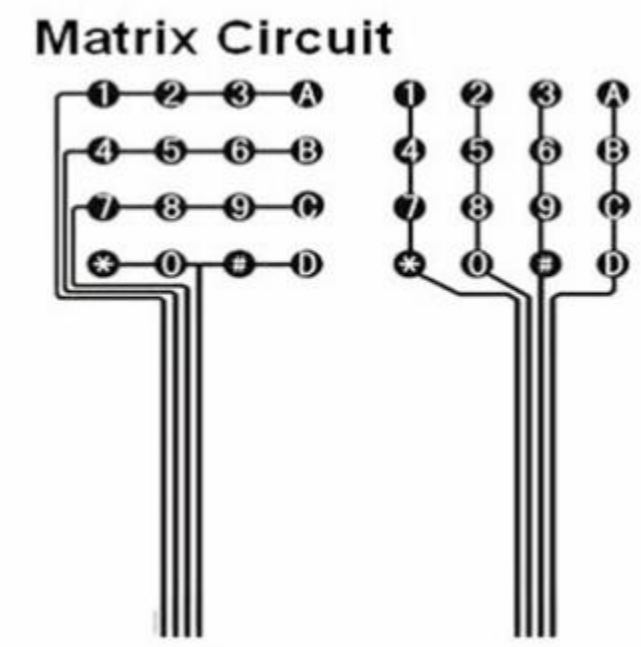
동작 원리

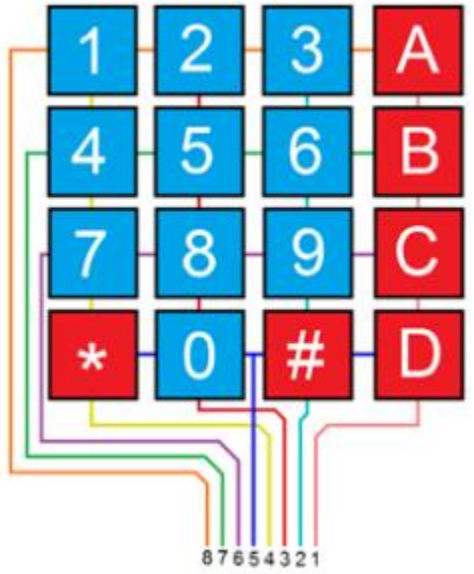
- 숫자의 결정
  - Num[3]: o_Row == 4’b0011일 때 눌림
  - Num[2]: o_Row == 4’b0101일 때 눌림
  - Num[1]: i_Col[2] 또는 i_Col[3]가 0
  - Num[0]: i_Col[1] 또는 i_Col[3]가 0

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

=

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111





{Row[0], …, Row[3], Col[0], …, Col[3]}

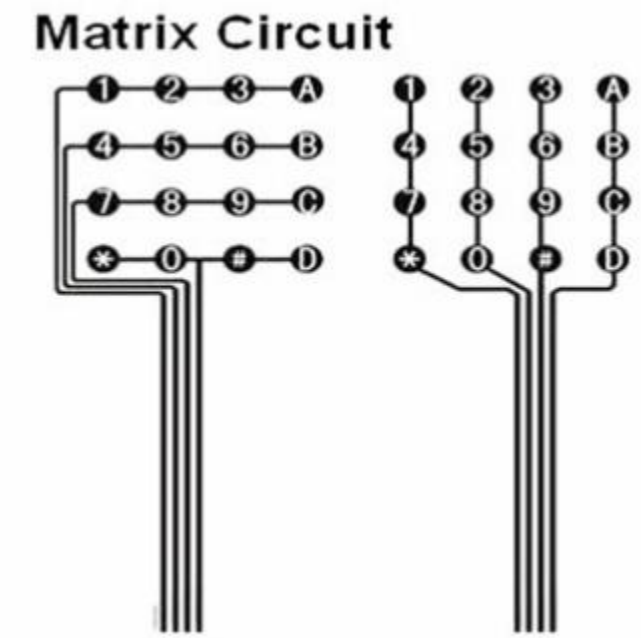
동작 원리

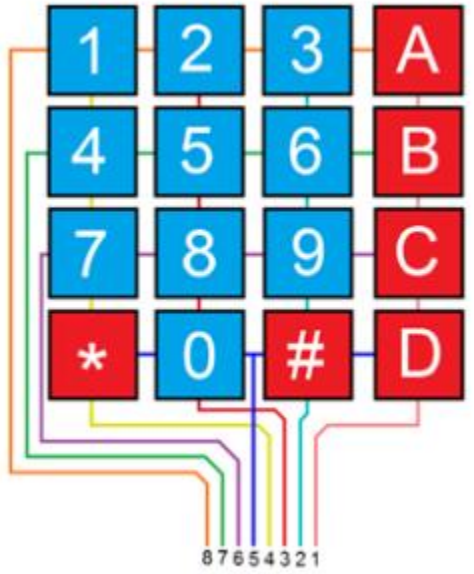
- 숫자의 결정
  - Num[3]: o_Row == 4’b0011일 때 눌림
  - Num[2]: o_Row == 4’b0101일 때 눌림
  - Num[1]: i_Col[2] 또는 i_Col[3]가 0
  - Num[0]: i_Col[1] 또는 i_Col[3]가 0

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

=

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111





{Row[0], …, Row[3], Col[0], …, Col[3]}

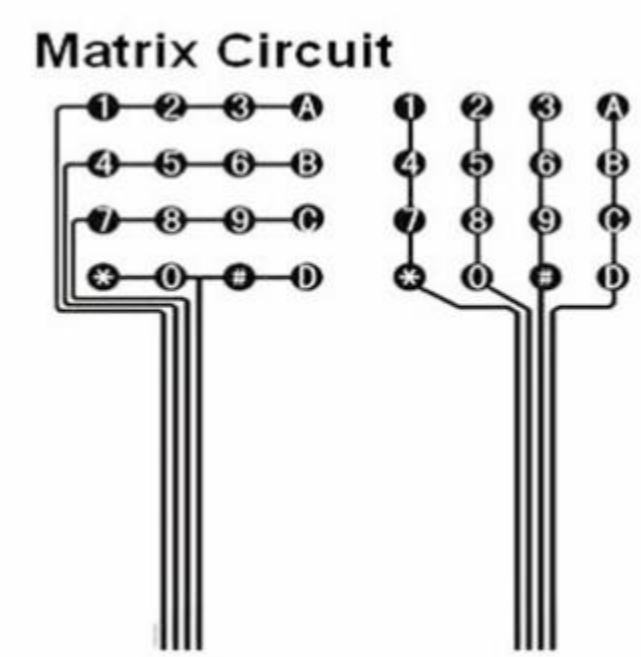
동작 원리

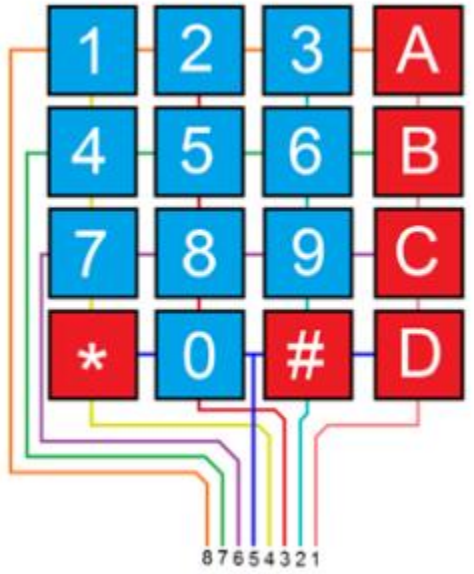
- 숫자의 결정
  - Num[3]: o_Row == 4’b0011일 때 눌림
  - Num[2]: o_Row == 4’b0101일 때 눌림
  - Num[1]: i_Col[2] 또는 i_Col[3]가 0
  - Num[0]: i_Col[1] 또는 i_Col[3]가 0

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

=

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111





{Row[0], …, Row[3], Col[0], …, Col[3]}

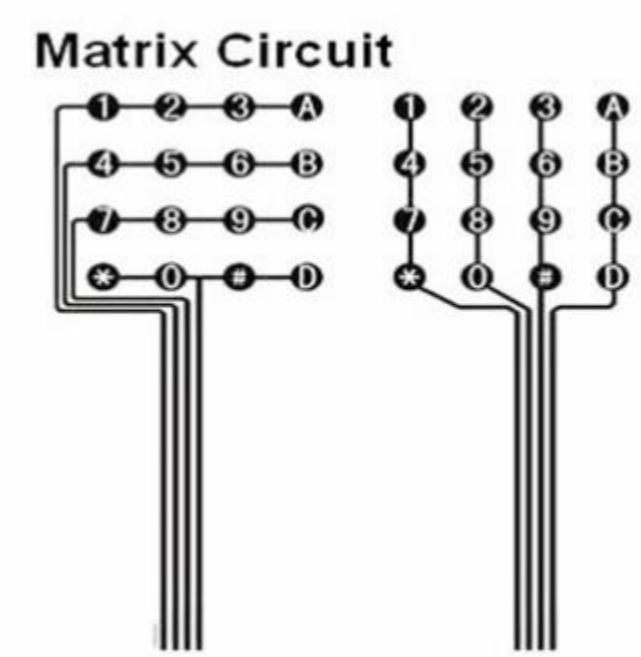
동작 원리

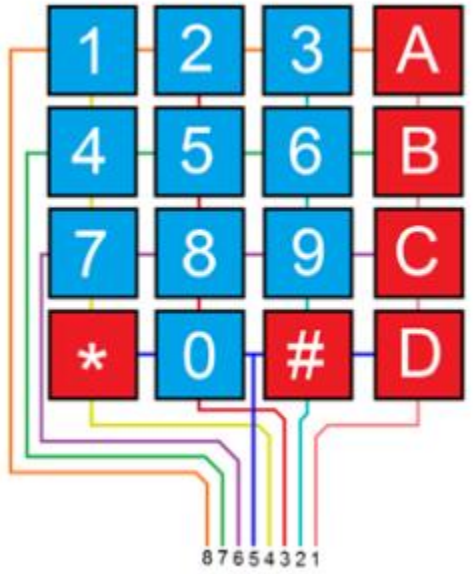
- 숫자의 결정
  - Num[3]: o_Row == 4’b0011일 때 눌림
  - Num[2]: o_Row == 4’b0101일 때 눌림
  - Num[1]: i_Col[2] 또는 i_Col[3]가 0
  - Num[0]: i_Col[1] 또는 i_Col[3]가 0

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

=

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

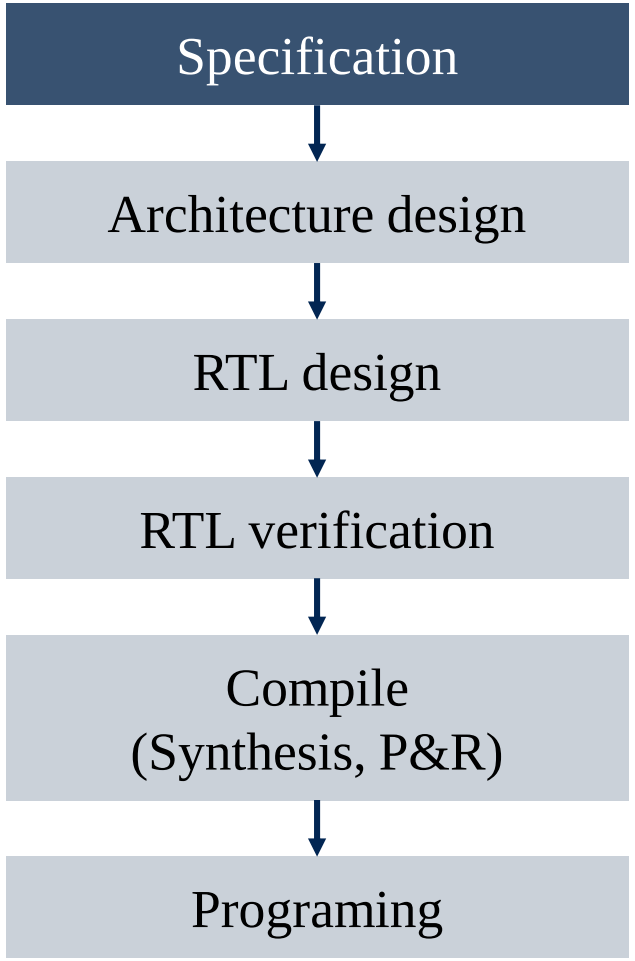




{Row[0], …, Row[3], Col[0], …, Col[3]}

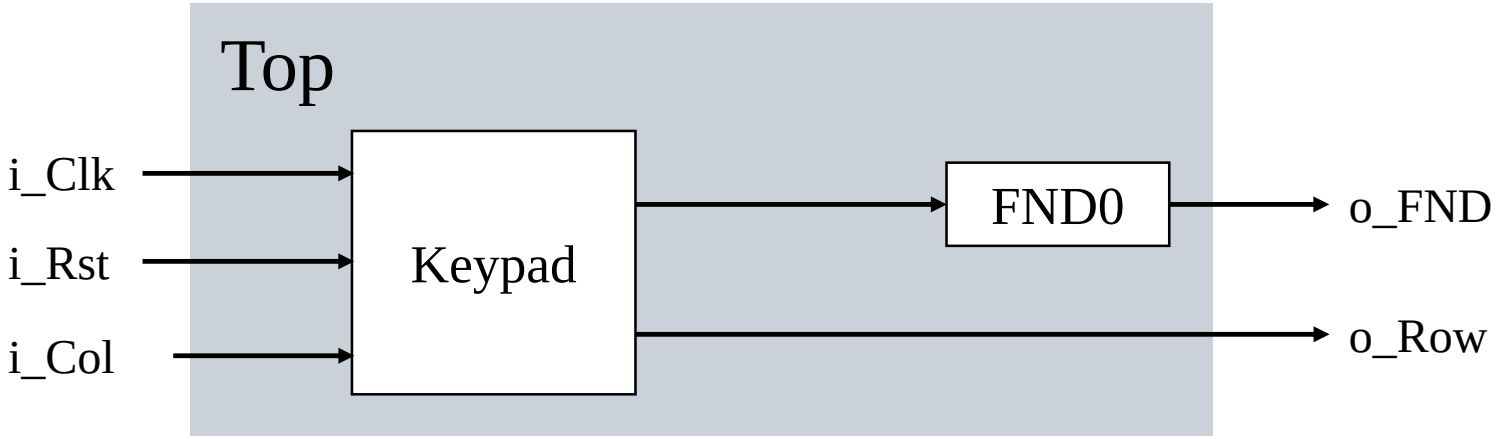
Keypad의 구현

- 명세
  - 입력
    - i_Clk
    - i_Rst ➔ push[3]
    - i_Col ➔ Extension
  - 출력
    - o_Row ➔ Extension
    - o_FND

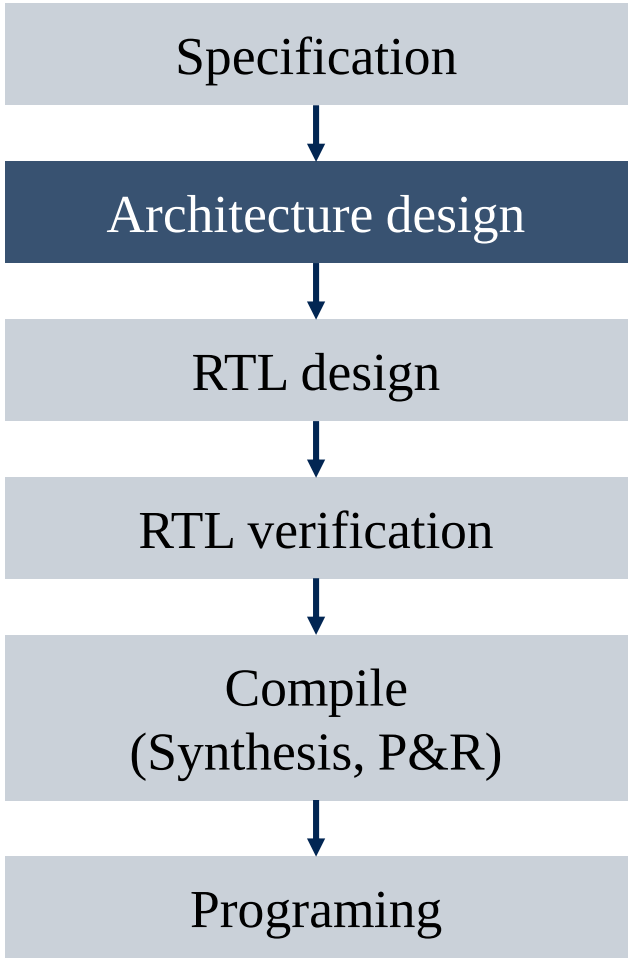


Keypad의 구현

- 구조 설계

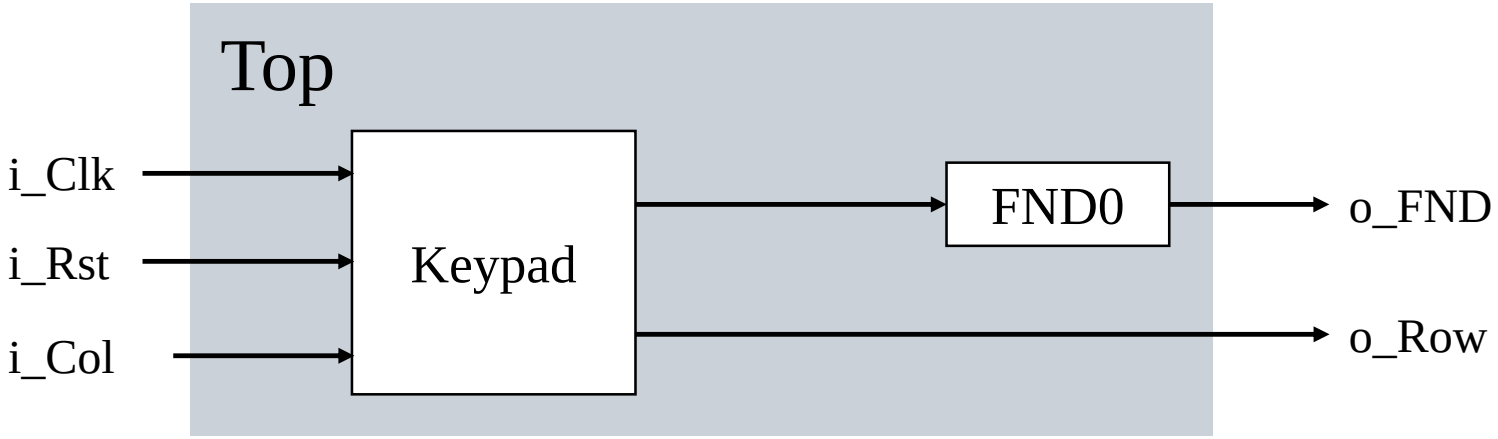


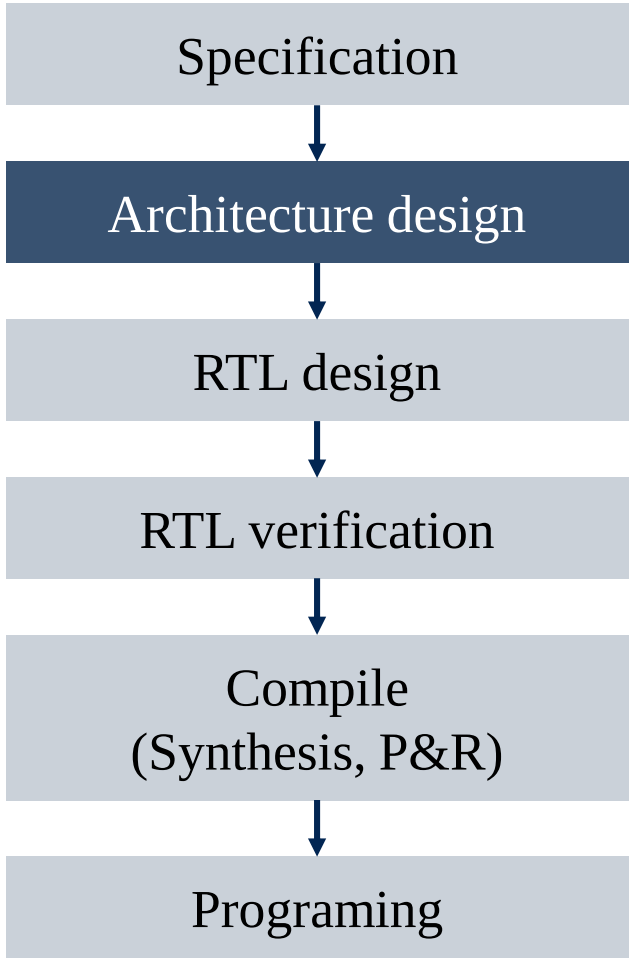
  - Keypad module에 필요한 레지스터는?



Keypad의 구현

- 구조 설계

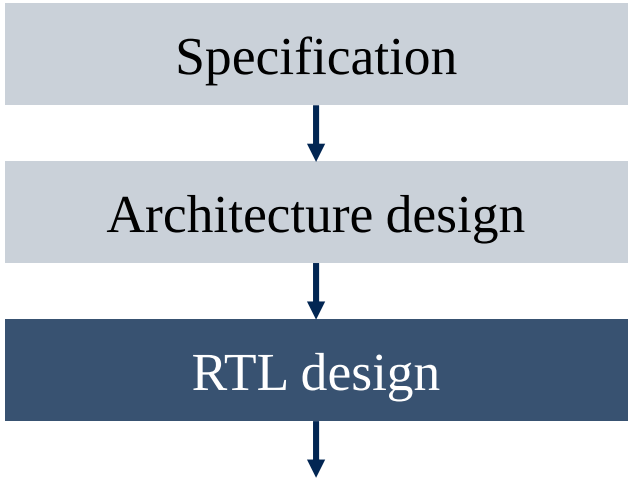




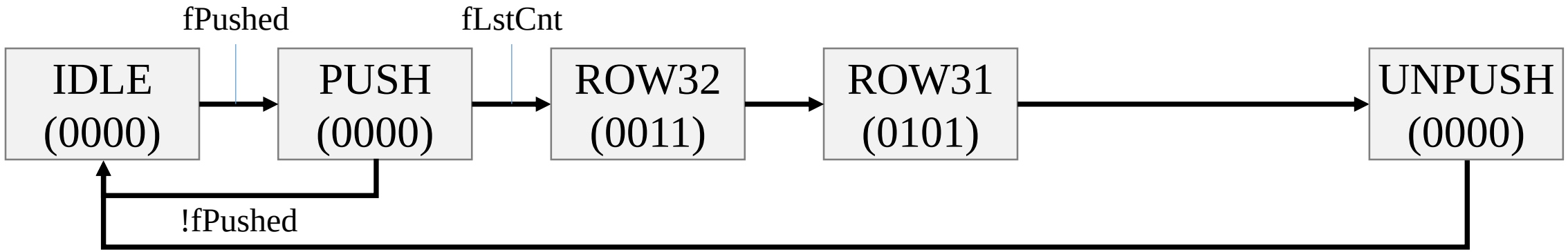
  - Keypad module에 필요한 레지스터
    - State
    - Num (4)
    - Count

Keypad의 구현

- 숫자의 결정 원리
  - Num[3]: o_Row == 4’b0011일 때 눌림
  - Num[2]: o_Row == 4’b0101일 때 눌림
  - Num[1]: i_Col[2] 또는 i_Col[3]가 0
  - Num[0]: i_Col[1] 또는 i_Col[3]가 0



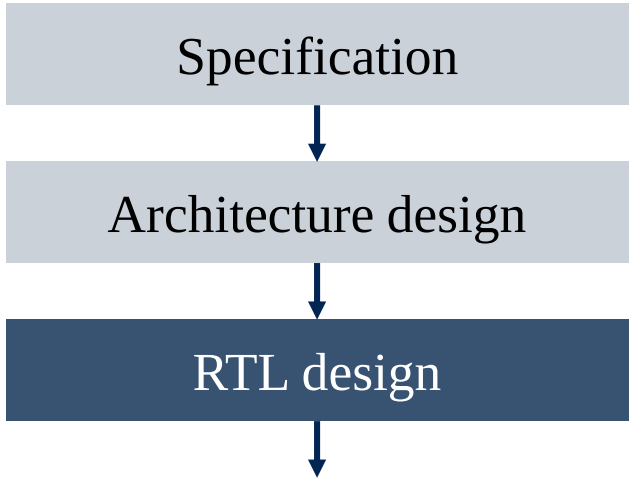
- FSM (o_Row)



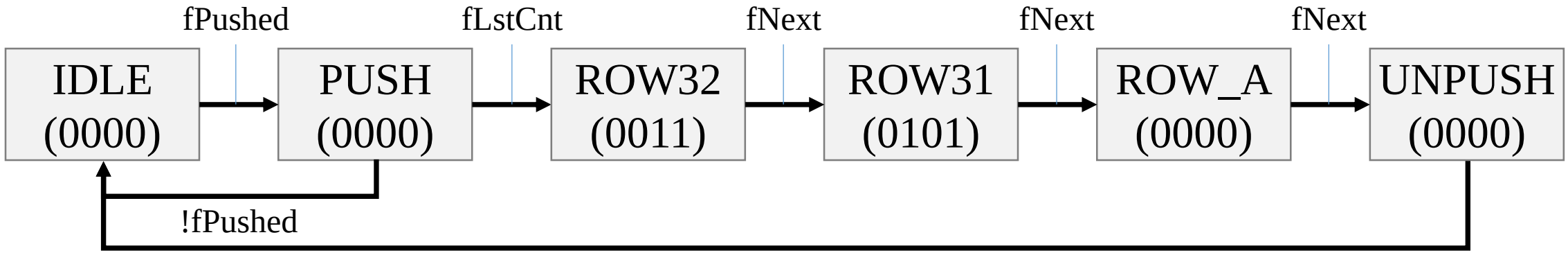
- fPushed = i_Col != 0, fLstCnt = &c_Cnt

Keypad의 구현

- 숫자의 결정 원리
  - Num[3]: o_Row == 4’b0011일 때 눌림
  - Num[2]: o_Row == 4’b0101일 때 눌림
  - Num[1]: i_Col[2] 또는 i_Col[3]가 0
  - Num[0]: i_Col[1] 또는 i_Col[3]가 0



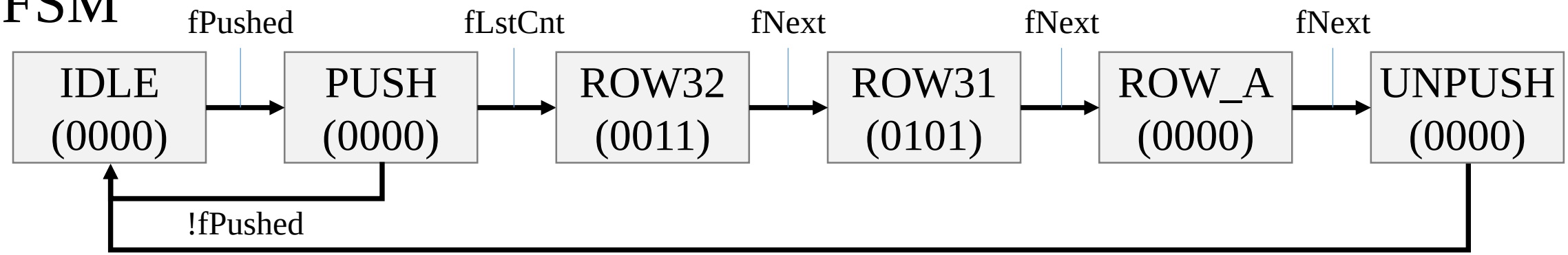
- FSM (o_Row)



- fPushed = ~&i_Col, fLstCnt = &c_Cnt, fNext = &c_Cnt[1:0]

Keypad의 구현

- FSM

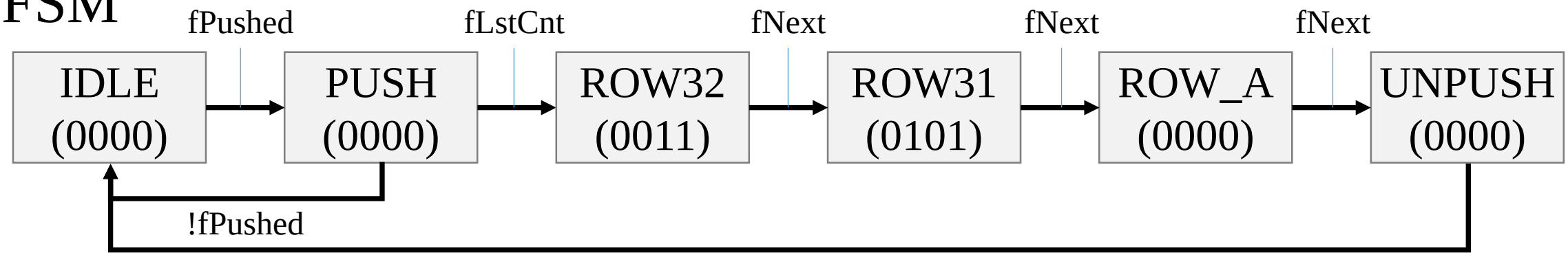


- 출력 정의
  - o_Num = c_Num
  - o_Row[3] = 0
  - o_Row[2] = c_State == ROW31
  - o_Row[1] = c_State == ROW32
  - o_Row[0] = c_State == ROW32 || c_State == ROW31

- fPushed = ~&i_Col
- fLstCnt = &c_Cnt
- fNext = &c_Cnt[1:0]

Keypad의 구현

- FSM

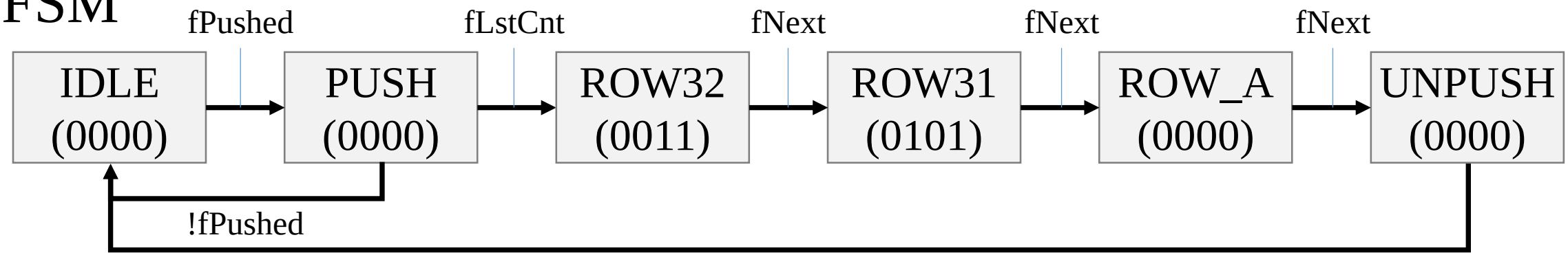


- 레지스터 정의
  - n_Cnt = fIncCnt ? c_Cnt + 1 : 0;
    - fIncCnt = c_State == PUSH || c_State == ROW32 || c_State == ROW31 || c_State == ROW_A
  - n_Num ??

- fPushed = ~&i_Col
- fLstCnt = &c_Cnt
- fNext = &c_Cnt[1:0]

Keypad의 구현

- FSM

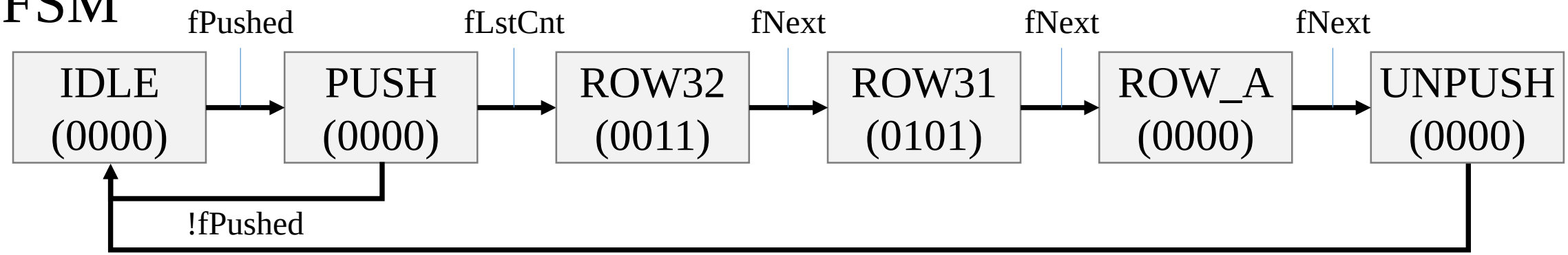


- 레지스터 정의
  - n_Cnt = fIncCnt ? c_Cnt + 1 : 0;
  - n_Num[3] = fCheck && c_State == ROW32 ? fPushed : c_Num[3];
  - n_Num[2] = fCheck && c_State == ROW31 ? fPushed : c_Num[2];
    - fCheck = c_Cnt[1:0] = 2'b01

- fPushed = ~&i_Col
- fLstCnt = &c_Cnt
- fNext = &c_Cnt[1:0]

Keypad의 구현

- FSM

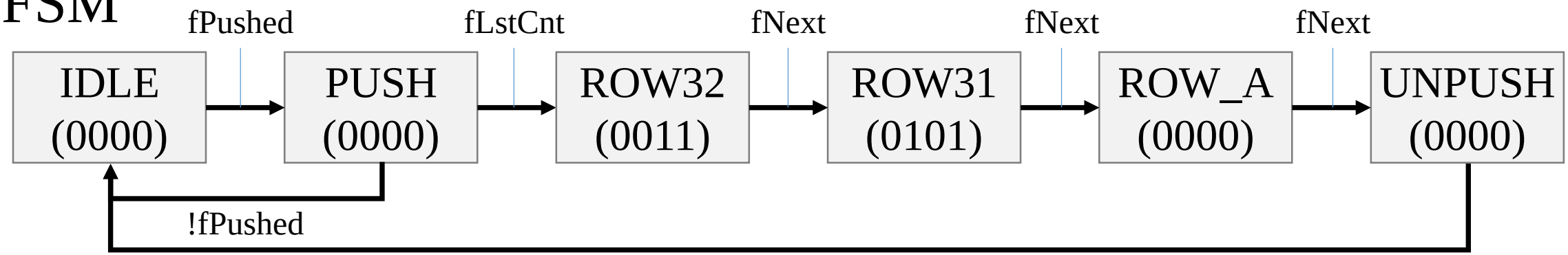


- 레지스터 정의
  - n_Cnt = fIncCnt ? c_Cnt + 1 : 0;
  - n_Num[3] = fCheck && c_State == ROW32 ? fPushed : c_Num[3];
  - n_Num[2] = fCheck && c_State == ROW31 ? fPushed : c_Num[2];
  - n_Num[1] = fCheck && c_State == ROW_A ? fPushed32 : c_Num[1];
    - fPushed32 = i_Col[3] == 0 || i_Col[2] == 0

- fPushed = i_Col != 0
- fLstCnt = &c_Cnt
- fNext = &c_Cnt[1:0]

Keypad의 구현

- FSM



- 레지스터 정의
  - n_Cnt = fIncCnt ? c_Cnt + 1 : 0;
  - n_Num[3] = fCheck && c_State == ROW32 ? fPushed : c_Num[3];
  - n_Num[2] = fCheck && c_State == ROW31 ? fPushed : c_Num[2];
  - n_Num[1] = fCheck && c_State == ROW_A ? fPushed32 : c_Num[1];
  - n_Num[0] = fCheck && c_State == ROW_A ? fPushed31 : c_Num[0];
    - fPushed31 = i_Col[3] == 0 || i_Col[1] == 0

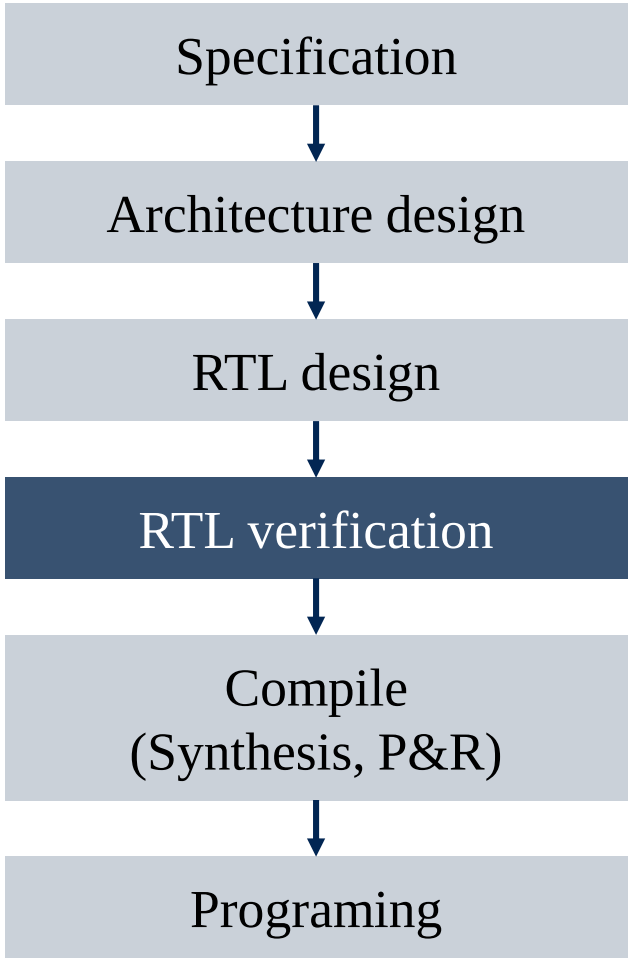
- fPushed = i_Col != 0
- fLstCnt = &c_Cnt
- fNext = &c_Cnt[1:0]

Keypad의 구현

- RTL 검증 - testbench module
  - tb_Keypad 작성

```
module tb_KeyPad;
reg             Clk;
reg             Rst;

reg             [N-1:0]       KP_i_Col, Num;
wire            [N-1:0]       KP_o_Row, KP_o_Num;

KeyPad KP0(Clk, Rst, KP_o_Row, KP_i_Col, KP_o_Num);

/* 50 MHz clock */
always
   #10 Clk = ~Clk;

always@*
begin
   KP_i_Col = 4'b1111;
   KP_i_Col[Num[1:0]] = KP_o_Row[Num[3:2]];
end
```

```
initial
begin
   Clk = 1;
   Rst = 0;
   KP_i_Col = 4'b1111;

   @(negedge Clk) Rst = 1;
   PushNum(0);
   PushNum(1);
   PushNum(2);
   …
   PushNum(15);

   $stop;
end

task PushNum;
input           [3:0]           i_Num;
begin
   Num = i_Num;     #1000000;
   KP_i_Col = 4'b1111;    #100;
end
endtask
endmodule
```



Keypad의 구현

- RTL 검증 - testbench module
  - tb_Keypad 작성

```
module tb_KeyPad;
reg             Clk;
reg             Rst;

reg             [N-1:0]       KP_i_Col, Num;
wire            [N-1:0]       KP_o_Row, KP_o_Num;

KeyPad KP0(Clk, Rst, KP_o_Row, KP_i_Col, KP_o_Num);

/* 50 MHz clock */
always
   #10 Clk = ~Clk;

always@*
begin
   KP_i_Col = 4'b1111;
   KP_i_Col[Num[1:0]] = KP_o_Row[Num[3:2]];
end
```

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

=

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

Keypad의 구현

- FPGA 구현 – pin 설정
  - 입력
    - i_Clk
    - i_Rst ➔ push[3]
    - i_Col ➔ GPIO0(24, 22, 20, 18)
  - 출력
    - o_Row ➔ GPIO0(16, 14, 12, 10)
    - o_FND

	GPIO0		
0			1
2			3
4			5
6			7
8			9
VCC		GND	
10	o_Row[0]		11
12	o_Row[1]		13
14	o_Row[2]		15
16	o_Row[3]		17
18	i_Col[0]		19
20	i_Col[1]		21
22	i_Col[2]		23
24	i_Col[3]		25
VCC		GND	
26			27
28			29
30			31
32			33
34			35