하계학술회의
논문85-8-4

Prolog를 이용한 논리회로 합성
(Logic Circuit Synthesis Using Prolog)

공 기석 서울대학교
조 동섭 전자계산기공학과
황 희웅

< 요 약 >

논리회로의 합성이란 minimize된 Boolean Expression을 실제로 존재하는 TTL IC로 implement시키는 과정을 말한다. 즉, IC pin assignment의 과정인 것이다. 본 논문에서는 논리회로를 합성하는 expert system의 초보적인 형태를 제안하고 있다.

1. 서론

일반적으로 논리회로 설계는 register transfer level과 gate level의 두단계로 나누어지는데, register transfer level design은 DDL(Digital system Design Language)로 기술되어 진다 [3]. DDL로 표현된 형태는 자동적으로 각 부분의 동작을 나타내는 table로 바꾸어지는데, 현재 논리회로의 합성은 이 table을 사용하여 수동적으로 이루어 지고 있다. 그런데, 자료에 의하면 첫째, 논리회로를 손으로 합성하는 경우 소요되는 manpower는 design과 testing의 전체과정에서 요구되는 manpower의 약 60% 라는 것, 둘째, 논리회로 합성과정에서 error가 없으면 hardware상에 남아있는 design error는 1/10 이하로 감소된다는 것이다 [2].

이러한 사실들이 논리회로 설계에서 CAD이용의 한 형태로서, 논리회로의 합성을 도와주는 expert system의 필요성을 설명해 줄 수 있으리라 생각된다.

본론에서는, 인공지능 분야에서 널리 사용되고있는 언어인 Prolog를 이용하여 논리회로를 합성하는 간단한 예를 보이고 있다 [8].

2. 본 론

minimize된 Boolean Expression (sum of products form 형태)에서는 AND, OR gate의 input갯수가 제한되어 있지 않다. 그러므로, input갯수가 제한되어 있는 standard TTL IC로 이 Boolean Expression을 구현하기 위해서는, 각 gate를 취급하는 특별한 지식이 필요하다 다음은 이러한 지식들의 예이다.

[Rule 1]

INVERTER gate가 AND gate의 출력단에 연결되어 있으면, 그 INVERTER gate와 AND gate를 NAND gate로 대체한다.

[Rule 2]

AND gate의 input갯수 (N)이 AND IC의 최대 input갯수 (M)보다 작거나 같으면, N보다 크거나 같은 input갯수를 가진 gate중에서 최소의 input갯수를 가진 TTL IC를 사용한다.

[Rule 3]

N이 M보다 크다면, AND gate를 M-1 개의 original input과 두번째 AND gate의 출력을 input으로 받는 첫번째 AND gate와 N-M+1 개의 original input을 갖는 두번째의 AND gate로 대체한다.

[Rule 4]

sum of products form인 형태로 gate가 연결되어 있을 때에는 AND, OR gate를 NAND, NAND gate의 연결형태로 바꾼다.

논리회로 합성과정에 들어가는 knowledge의 크기를 줄이기 위해, 본 논문에서는 다음과 같은 6가지 형태의 IC만을 사용할 수 있다고 가정하였다.

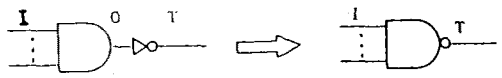
gate type	number of inputs	IC name
AND	2	SN74LS09
	3	SN74LS11
NAND	2	SN74LS00
	3	SN74LS10
INVERTER	1	SN74LS04
OR	2	SN74LS32

표 1. 6가지 형태의 IC

위에서 든 rule들을 Prolog로 표현한 형태를 그림1에 보였다. 예시되지 않은 rule들도 이와 마찬가지 방법으로 Prolog로 표현가능하다. 논리회로 합성 expert system내에서 이와 같은 rule들은 일종의 knowledge-base를 구축하게 된다 [9], [10]. rule들의 적용순서를 결정하는 지식(지식의 지식 즉 meta knowledge)도 또한 Prolog로 쓰여져 있다.

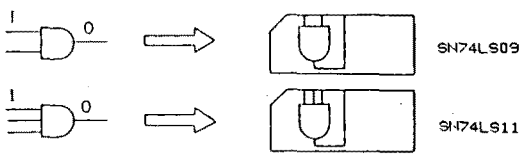
rule들을 사용하여 AND, OR gate로 이루어진 logic diagram (sum of products form)에 대해서 pin을 assign한 예가 그림2이다. 그림2의 (a)에서는 logic diagram을 보였고, (b)에서는 이것을 Prolog input으로 바꾼 형태를, (c)에서는 이 Prolog program의 output을 나타내었다. (d)에서는 (c)의 결과를 사용한 배선도를 보였다.

[Rule 1]



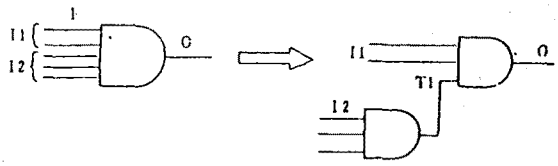

```
process(and,I,O):-inverter(O,T),
                  retract(and(I,O)),
                  retract(inverter(O,T)),
                  asserta(nand(I,T)).
```

[Rule 2]



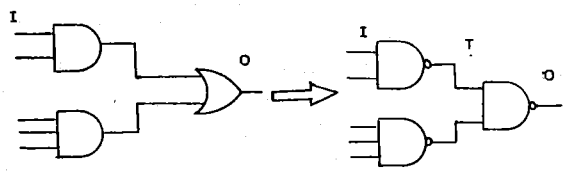

```
process(and,I,O):-length(I,2),
                  retract(and(I,O)),
                  asserta(sn74ls09(I,O)).
process(and,I,O):-length(I,3),
                  retract(and(I,O)),
                  asserta(sn74ls11(I,O)).
```

[Rule 3]




```
process(and,I,O):-length(I,N),
                  N>3,
                  split(I,N,I1,I2),
                  genlist(T1),
                  append(I1,T1,T),
                  process(and,I,O),
                  process(and,I2,T1).
```

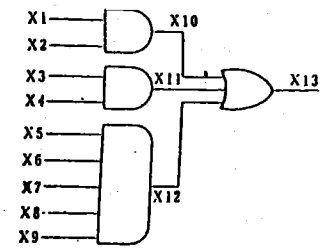
[Rule 4]




```
process(or,I,O):-
    find_and_chang_input_ics(I,O,New_inputs),
    asserta(nand(New_inputs,O)),
    collect_all_inputs([],X),
    rule_apply(X).
```

그림 1. Prolog로 표현된 rule들의 예

(a)



(b)

```
and([x1,x2],[x10]).
and([x3,x4],[x11]).
and([x5,x6,x7,x8,x9],[x12]).
or([x10,x11,x12],[x13]).
```

(c)

```
sn74ls00([[x1,1],[x2,2]],[t1,3]).
sn74ls00([[x3,4],[x4,5]],[t2,6]).
sn74ls11([[x7,13],[x8,1],[x9,2]],[t3,12]).
sn74ls10([[x5,1],[x6,2],[t3,13]],[t4,12]).
sn74ls10([[t1,10],[t2,9],[t4,11]],[x13,8]).
```

(d)

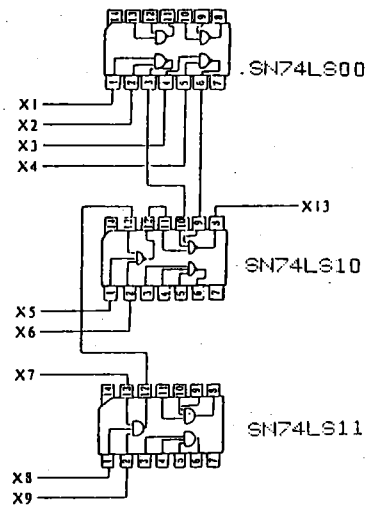


그림 2. 논리회로 합성의 예.

(a) logic diagram, (b) input data,
(c) output data, (d) IC assignment.

3. 결 론

Prolog를 이용한 논리회로 합성용 expert system을 개발하기 위한 기본적인 연구가 행해졌다. 그 결과로서, Prolog가 algorithm을 충분히 표현할 수 있으며, 효과적으로 지식을 나타낼 수 있음을 알았다. 이는 Prolog가 가진 unification mechanism과 automatic backtracking기법에 부분적으로 기인한다고 생각된다. 앞으로, 본 논문에서의 논의를 확장하면, intelligent CAD system의 구성도 가능하리라 전망된다 [1].

[참 고 문 헌]

1. F. Maruyama, et al., "Prolog-Based Expert System for Logic Design", Proc. of Int. Conf. on Fifth Generation Computers Systems, 1984, pp 563-567.

2. T.Uehara, N.Kawato, "Logic Circuit Synthesis Using Prolog", New Generation Computing, 1983.

3. N.Kawato, T.Uehara, S.Hirose, T.Saito, "An Interactive Logic Synthesis System Based upon AI Techniques", 19th Design Automation Conf., 1982.

4. 樹下行三編, 論理装置のCAD, 情報処理学会, 1980.

5. S.G.Shiva, "Automatic Hardware Synthesis", Proc. IEEE, vol. 71, no. 1, Jan., 1983.

6. H.M.Lipp, "Methodical Aspects of Logic Synthesis", Proc. IEEE, vol. 71, no. 1, Jan., 1983.

7. F.Hill, G.Peterson, Introduction to Switching Theory and Logical Design, John Willey & Sons, New York, 1974.

8. W.Clocksin, C.Mellish, Programming in Prolog, Springer - Verlag, Berlin, 1981.

9. P.Winston, Artificial Intelligence, Addison Wesley, 2nd ed., 1984.

10. R.Forsyth, ed., Expert Systems, Chapman and Hall, London, 1984.