마이크로프로세서 설계 무작정 따라하기 part-III (2)

부제: 합성, 배치 및 배선

KAIST 전자전산학과 박사과정 배영돈(http://www.donny.co.kr)

이번 강좌에서는 Synopsys 사의 Design Compiler 를 이용하여 회로를 합성하고 최적화 하는 방법에 대하여 설명한다.

1. 서론: 합성이란?

합성은 Verilog 를 비롯한 상위단계의 언어로부터 회로를 생성하는 과정을 의미한다.
같은 동작을 하는 회로이더라도 합성방법에 따라 다른 결과를 얻게 된다. 먼저 합성 툴에 따라 결과가 다르게 된다. 우리가 사용할 툴은 Synopsys 사의 Design Compiler 로 강력한 기능을 제공하는 고가의 툴이다. 두번째는 설계방식에 의한 것으로, 지난 회에 설명한 것과 같이 설계 방법(Verilog 기술방법, 구조설계, 등)에 따라 합성이 불가능할 수도 있으며, 아무리 좋은 툴이라도 좋은 성능을 낼 수 없을 수 있다. 세번째, 합성에 사용된 제약조건(constraint)이다. 합성과정은 크게 translation(번역)과 optimization(최적화)의 과정으로 나눌 수 있다(그림 1). 번역과정은 Verilog 코드를 회로로 나타내는 과정이며, 최적화과정은 같은 동작(functionality)을 하는 회로를 여러 가지 조건에 맞추어 최적화하는 과정이다. 대표적인 경우가 면적을 최소화하는 경우와 동작속도를 최대로 하는 경우이다. 그림 2 와 같이 결과 회로가 큰 차이를 보일 수 있게 된다. 무엇을 기준으로 최적화를 할 것인가를 결정하는 것이 바로 제약조건(constraint)이다.

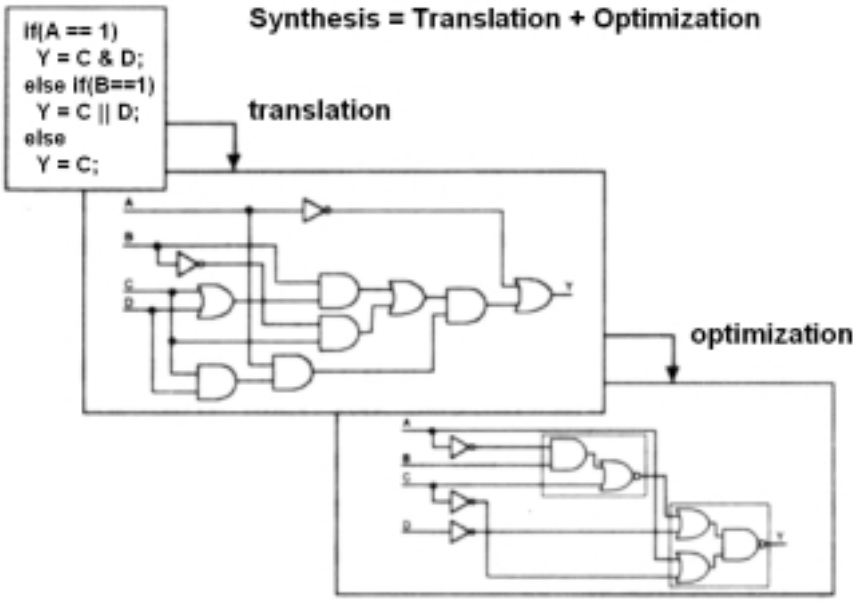
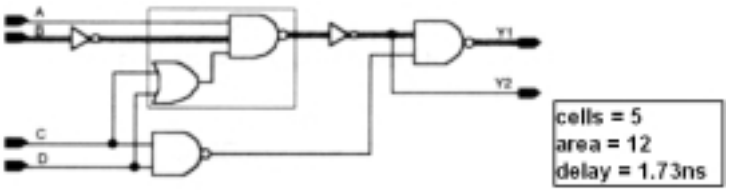
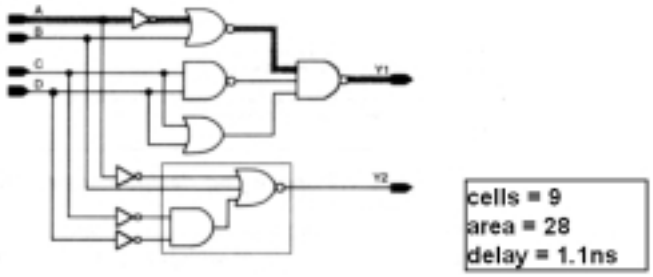


그림 1.합성(Synthesis)



a) Minimum Area



b) Maximal Speed

그림 2. 최적화(optimization)

본 강좌에선 Design Compiler 의 기본 적인 사용법과 회로 합성에 유용한 constraint 의 사용방법을 예제를 통해 설명하고자 한다.

2. Design Compiler 의 실행

Design Compiler 를 실행하는 방법은 두 가지가 있다. 첫번째는 dc_shell 로 텍스트 화면에서 직접 명령을 입력하는 것이며, 두번째는 design_analyzer 로 GUI 환경에서 메뉴방식으로 기능을 실행한다. design_analyzer 가 시각적으로 보여준다는 점을 제외하면 기능상의 차이점은 없다. 초보자의 경우 GUI 방식의 design_analyzer 를 사용하는 것이 더 편리하지만, 어느 정도 익숙해진 상황에서는 dc_shell 환경이 빠르고 편리하다. design_analyzer 에서도 Setup 메뉴의 Command Window 를 사용하면 dc_shell 과 마찬가지로 command 를 사용할 수 있다. 본 강좌에서는 command 방식을 기준으로 설명하도록 한다.

Design Compiler 를 실행하기 위해선 지난강좌에서 설명한 것과 같이 셀 라이브러리가 준비되어있어야 한다. 본 강좌에서는 IDEC MPW (Hynix 0.35um)용 라이브러리를 사용하였다.

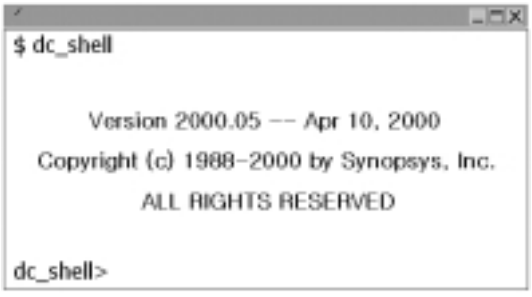


그림 3. Design Compiler 의 실행

먼저 가장 기본적인 기능인 Verilog code 를 읽어서 합성하고 저장하는 과정을 알아보자.

1) Verilog code 를 읽어오기 (SimpleCore 의 alu.v)

dc_shell> read –format verilog alu.v <ENTER>

2) 합성

dc_shell> compile –map_effort high <ENTER>

3) 합성 결과의 저장 (저장형식는 Synopsys 의 DB 형식)

dc_shell> write –format db –hierarchy –output "alu.db"<ENTER>

3. 합성 결과 확인

1) 면적

```
dc_shell> report_area
****************************************
Report : area
Design : alu
Version: 2000.05
Date   : Tue Jun 18 16:02:59 2002
****************************************

Library(s) Used:

    cb35os142d_typ (File: /home/donny/library/hyundai035/HSC350/1.0.0/logic/syno

Number of ports:            55
Number of nets:            185
Number of cells:           133
Number of references:       16

Combinational area:      269.000000
Noncombinational area:     0.000000
Net Interconnect area:     0.502800

Total cell area:         269.000000
Total area:              269.502808
```

2) 동작 속도

```
dc_shell> report_timing
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : alu
Version: 2000.05
Date   : Tue Jun 18 16:02:59 2002
****************************************

Operating Conditions: NCCOM   Library: cb35io122d_typ
Wire Load Model Mode: segmented

  Startpoint: aluCtl[1] (input port)
  Endpoint: condFlag[0]
            (output port)
  Path Group: (none)
  Path Type: max
  Des/Clust/Port     Wire Load Model       Library
    ------------------------------------------------
    alu              1000                cb35io122d_typ

    Point                               Incr       Path
    ------------------------------------------------------------
    input external delay                  0.00       0.00 f
    aluCtl[1] (in)                        0.00       0.00 f
    r24/U22/zn (inv0d0)                   1.28       1.28 r
    r24/U21/zn (aoi22d1)                  0.10       1.38 f
    r24/U1_0/co (ad01d1)                  0.44       1.82 f
    r24/U1_1/co (ad01d1)                  0.33       2.15 f
```

```
r24/U1_2/co (ad01d1)                0.33      2.48 f
r24/U1_3/co (ad01d1)                0.33      2.82 f
r24/U1_4/co (ad01d1)                0.33      3.15 f
r24/U1_5/co (ad01d1)                0.33      3.48 f
r24/U1_6/co (ad01d1)                0.33      3.81 f
r24/U1_7/co (ad01d1)                0.33      4.14 f
r24/U1_8/co (ad01d1)                0.33      4.48 f
r24/U1_9/co (ad01d1)                0.33      4.81 f
r24/U1_10/co (ad01d1)               0.33      5.14 f
r24/U1_11/co (ad01d1)               0.33      5.47 f
r24/U1_12/co (ad01d1)               0.33      5.81 f
r24/U1_13/co (ad01d1)               0.33      6.14 f
r24/U1_14/co (ad01d1)               0.33      6.47 f
r24/U1_15/s (ad01d1)                0.50      6.97 r
U77/zn (aoi221d1)                   0.21      7.18 f
U119/zn (inv0d0)                    0.55      7.73 r
U114/zn (aoi22d1)                   0.17      7.90 f
U75/zn (nr02d0)                     0.37      8.27 r
condFlag[0] (out)                   0.00      8.27 r
data arrival time                             8.27
-------------------------------------------------------------
(Path is unconstrained)
```

3) design_analyzer 를 이용하여 schematic 을 출력한 결과

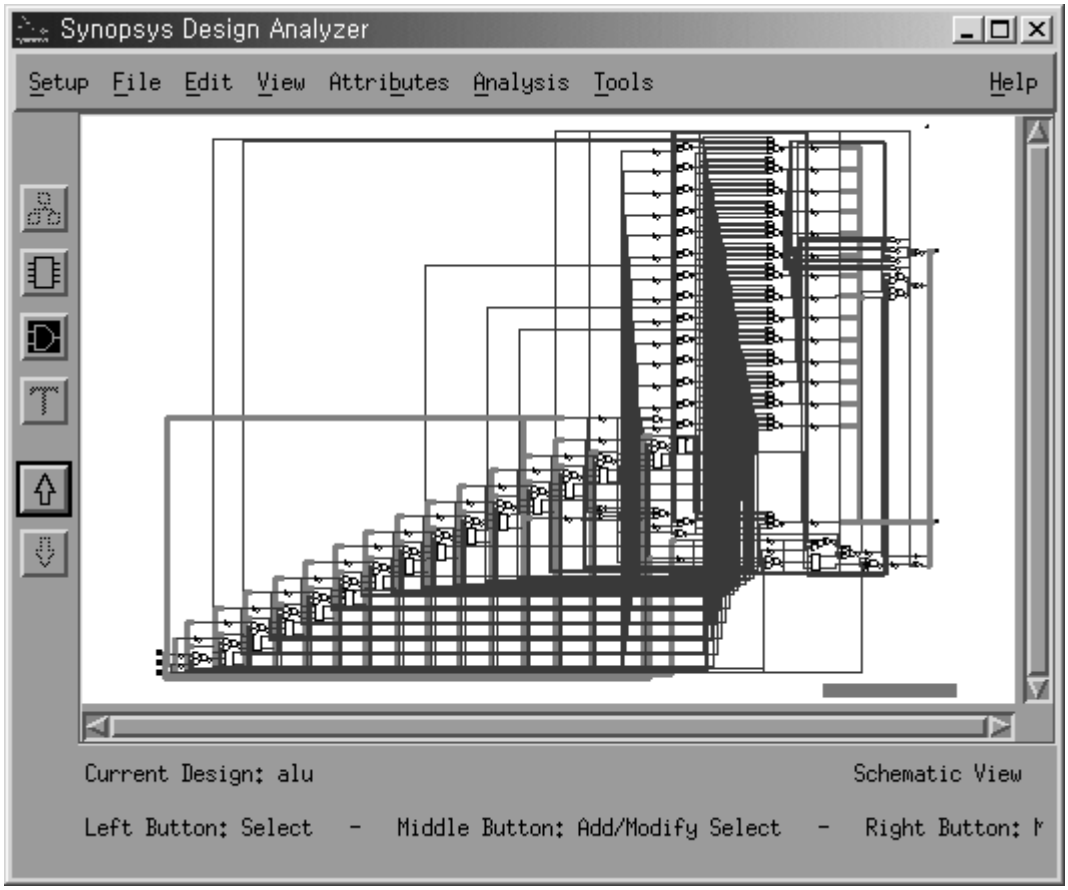


그림 4. alu 의 합성결과

4. 최적화 (Optimization)

단순히 Verilog 코드를 읽어서 합성하는 것 만으로도 원하는 동작(functionality)을 하는 회로를 얻을 수 있다. 그러나 제약조건(constraint)를 설정하면 더욱 효율적인 회로로 최적화 할 수 있다. 합성 결과는 사용방법에 따라 매우 비효율적일 수도 효율적일 수도 있게 된다.

먼저 alu 블록의 합성 결과를 보면 그림 5 와 같이 alu_DW01 이라는 새로운 블록이 생성된 것을 볼 수 있다. 이것은 덧셈/뺄셈기능을 합성하기 위해서 미리 제공하는 DesignWare 라고 부르는 회로를 사용한 것이다.

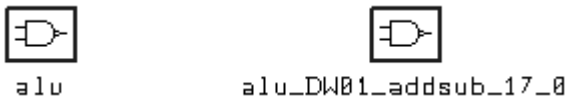


그림 5. alu 의 합성결과 (DesignWare 의 사용)

DesignWare 를 이용하여 합성된 회로들은 DW01 등의 이름을 갖는 별도의 module 로 합성 된다. 이 모듈을 flatten 하여 한 개의 모듈로 만드는 방법은 다음과 같다.

```
dc_shell> current_design alu
dc_shell> ungroup –all –flatten
```

그림 4 는 DesignWare 모듈이 flatten 된 결과이다.

먼저 면적을 최소화하는 방법을 알아보자

```
dc_shell> set_max_area 0
dc_shell> compile –map_effort high
```

그림 6 은 면적을 최소화하는 조건으로 합성한 결과이다 굵은 선은 critical path 를 나타내고 있다. 면적을 최소화 하여 266 게이트의 결과를 얻은 반면, 지연시간은 7.83ns 로 매우 느린 회로임을 알 수 있다. 합성결과를 보면 덧셈기 부분이 ripple carry adder 방식으로 합성되었음을 짐작할 수 있다.

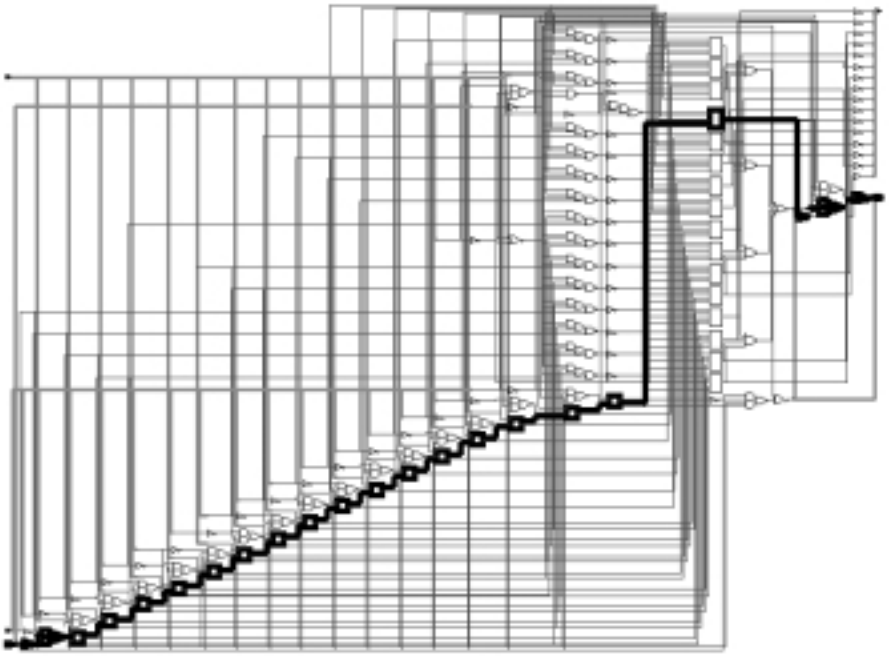


그림 6. 면적 최소화 (area: 266, delay: 7.83ns)

다음으로 동작 속도를 최대로 하는 방법을 알아보자.

```
dc_shell> set_max_delay 0 –from {aluAIn, aluBIn} –to {aluOut, condFlag}
dc_shell> compile –map_effort high
```

그림 7 은 동작속도를 최대로 합성한 결과로 carry look-ahead adder 방식을 사용하여 합성되었다. 면적을 최소로 한 경우에 비하여 면적이 6 배가량 증가하였으나 지연시간은 2.4ns 로 줄어들었다.

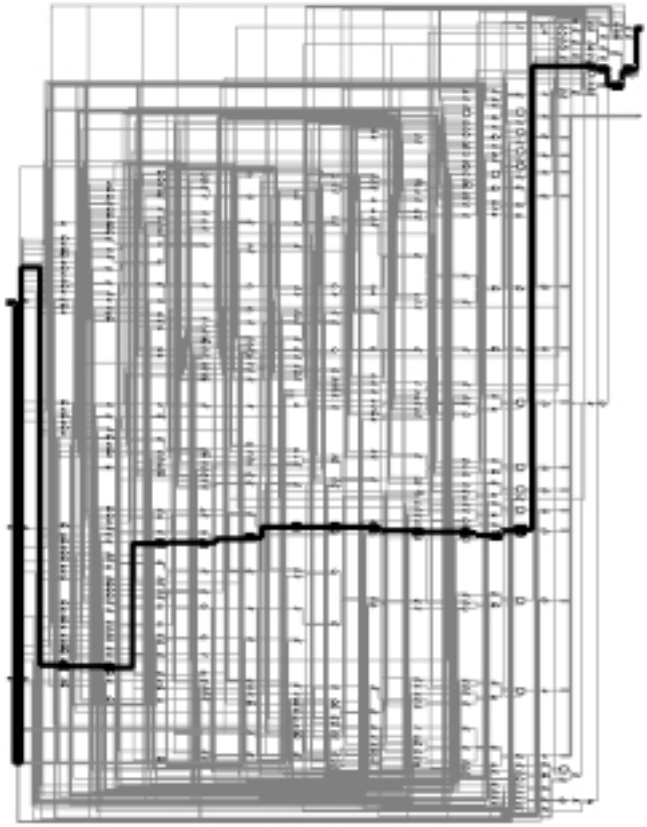


그림 7. 동작 속도 최적화 (area: 1576, delay: 2.4ns)

지금까지의 결과는 alu 출력에 걸린 부하를 고려하지 않은 결과이다. 지연시간은 출력 부하에 따라 달라지게 된다. set_load 명령을 사용하여 0.5pF 의 출력부하를 추가해보자.

```
dc_shell> set_load 0.5 {aluOut, condFlag}
dc_shell> report_timing
data arrival time 4.49
```

이와 같이 같은 회로에 동작조건을 바꿈에 따라 결과가 달라진다.
달라진 조건을 반영하기 위하여 다시 합성을 수행한 결과 면적 2158 게이트, 지연시간 2.8ns 의 결과를 얻었다.

다음 강좌에서는 clock 이 있는 순차 회로를 최적화 하는 방법과 복잡한 회로를 효율적으로 합성하는 방법에 대하여 알아보도록 한다.