UNIVERSITY OF TEHRAN

Electrical and Computer Engineering Department

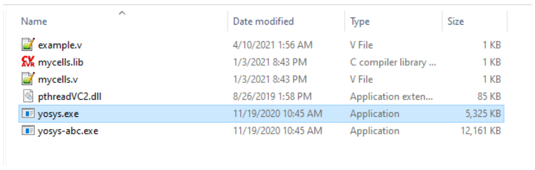
Digital Logic Design, ECE 367, ECE 894, Spring 1399-1400

Yosys , Synthesis tool

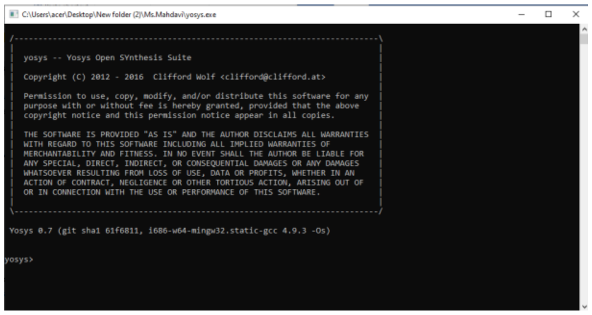
Yosys is an open-source tool that can be used to synthesize RTL design. you can use it in each level and it can convert behavior or structural designs to gate-level. See http://www.clifford.at/yosys to know more about yosys.

It needs components library for synthesis, which can be designed in accordance with the desired technologies.” yosys_presentation.pdf” and “yosys_manual.pdf” files are attached for more information. A number of files are attached to the document. To perform the synthesis, these files and the design code are placed in same directory. The synthesis steps are as follows.

Open *yosys.exe* as shown in Fig. 1, then you will see the yosys window on the right.



Name	Date modified	Type	Size
example.v	4/10/2021 1:56 AM	V File	1 KB
mycells.lib	1/3/2021 8:43 PM	C compiler library ...	1 KB
mycells.v	1/3/2021 8:43 PM	V File	1 KB
pthreadVC2.dll	8/26/2019 1:58 PM	Application exten...	85 KB
yosys.exe	11/19/2020 10:45 AM	Application	5,325 KB
yosys-abc.exe	11/19/2020 10:45 AM	Application	12,161 KB



C:\Users\acer\Desktop\New folder (2)\Ms.Mahdavi\yosys.exe

```
/----------------------------------------------------------------------------\
|                                                                            |
|  yosys -- Yosys Open SYnthesis Suite                                       |
|                                                                            |
|  Copyright (C) 2012 - 2016  Clifford Wolf <clifford@clifford.at>           |
|                                                                            |
|  Permission to use, copy, modify, and/or distribute this software for any  |
|  purpose with or without fee is hereby granted, provided that the above    |
|  copyright notice and this permission notice appear in all copies.         |
|                                                                            |
|  THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES  |
|  WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF          |
|  MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR   |
|  ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES    |
|  WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN     |
|  ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF   |
|  OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.            |
|                                                                            |
\----------------------------------------------------------------------------/

 Yosys 0.7 (git sha1 61f6811, i686-w64-mingw32.static-gcc 4.9.3 -Os)


yosys>
```

Fig. 1

1- Step1

In Fig. 2 according to red box 1, write command “*read_verilog YYYY.v*” to read your Verilog source file with name of *YYYY.v*. After execution, a successful message is given.

2- Step2

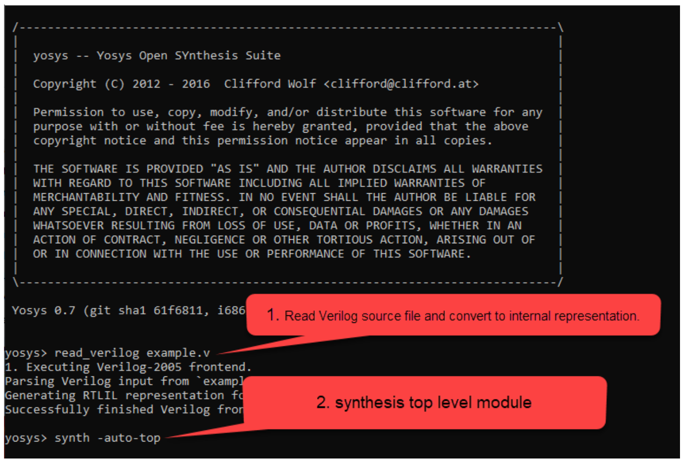
In Fig. 2 according to red box 2, write command “*synth -auto-top*” to synthesis top level module. The last few lines of the output of step 2 can be seen in Fig. 3.

3- Step3

In Fig. 3 according to red box3, write command “*dfflibmap -liberty mycells.lib*” to map registers to hardware flip flops. The results of step 3 can be seen in Fig. 4.

4- Step4

In Fig. 4 according to red box4, write command “*abc -liberty mycells.lib*” to map logic to available hardware gates and performs optimization. This step done based on “mycells” library. The result of step 4 can be seen in Fig. 5.


1. Read Verilog source file and convert to internal representation.
2. synthesis top level module

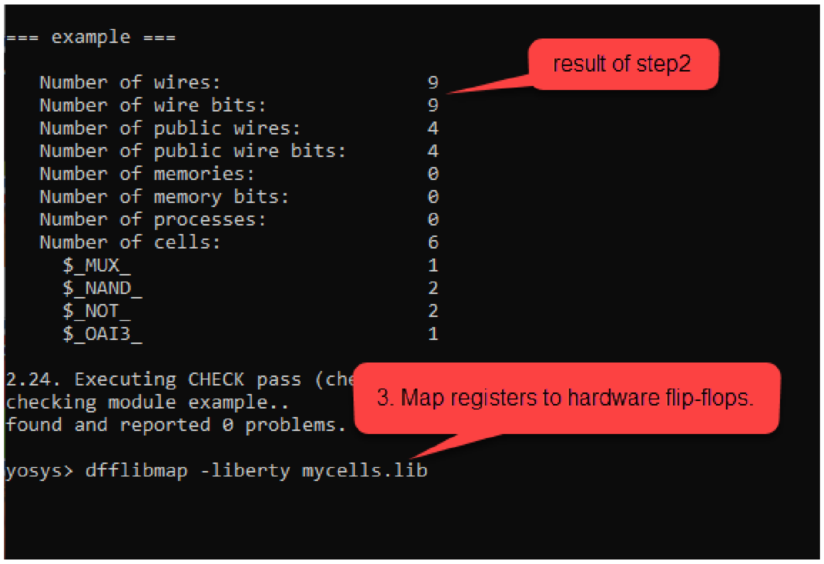
```
/----------------------------------------------------------------------------\
|                                                                            |
|  yosys -- Yosys Open SYnthesis Suite                                       |
|                                                                            |
|  Copyright (C) 2012 - 2016  Clifford Wolf <clifford@clifford.at>           |
|                                                                            |
|  Permission to use, copy, modify, and/or distribute this software for any  |
|  purpose with or without fee is hereby granted, provided that the above    |
|  copyright notice and this permission notice appear in all copies.         |
|                                                                            |
|  THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES  |
|  WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF          |
|  MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR   |
|  ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES    |
|  WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN     |
|  ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF   |
|  OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.            |
|                                                                            |
\----------------------------------------------------------------------------/

 Yosys 0.7 (git sha1 61f6811, i686

yosys> read_verilog example.v
1. Executing Verilog-2005 frontend.
Parsing Verilog input from `exampl
Generating RTLIL representation fo
Successfully finished Verilog fron

yosys> synth -auto-top
```

Fig. 2



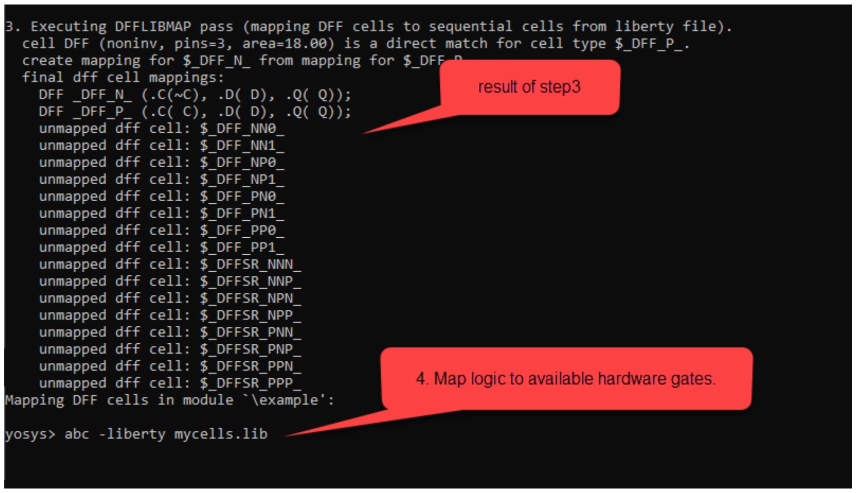
```
=== example ===

   Number of wires:                  9
   Number of wire bits:              9
   Number of public wires:           4
   Number of public wire bits:       4
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  6
     $_MUX_                          1
     $_NAND_                         2
     $_NOT_                          2
     $_OAI3_                         1

2.24. Executing CHECK pass (che
checking module example..
found and reported 0 problems.

yosys> dfflibmap -liberty mycells.lib
```

Fig. 3



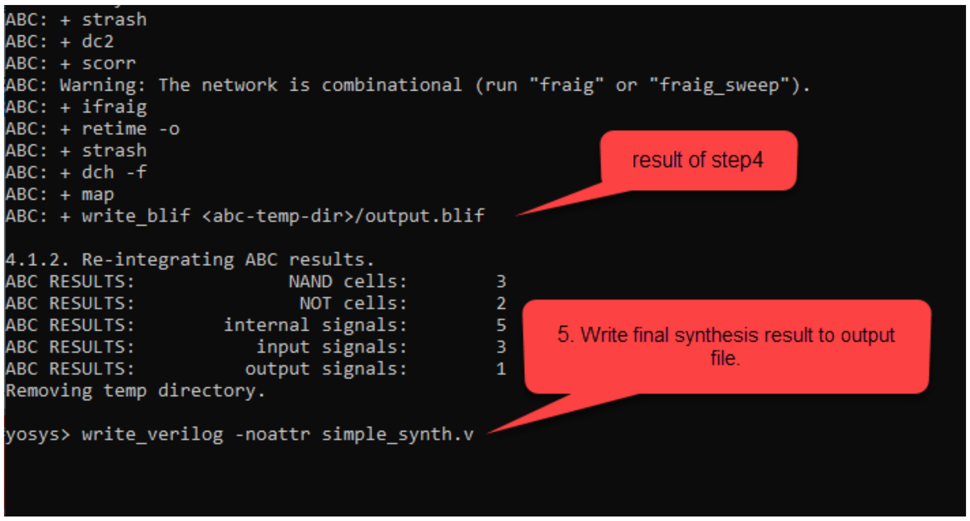
```
3. Executing DFFLIBMAP pass (mapping DFF cells to sequential cells from liberty file).
  cell DFF (noninv, pins=3, area=18.00) is a direct match for cell type $_DFF_P_.
  create mapping for $_DFF_N_ from mapping for $_DFF_P
  final dff cell mappings:
    DFF _DFF_N_ (.C(~C), .D( D), .Q( Q));
    DFF _DFF_P_ (.C( C), .D( D), .Q( Q));
    unmapped dff cell: $_DFF_NN0_
    unmapped dff cell: $_DFF_NN1_
    unmapped dff cell: $_DFF_NP0_
    unmapped dff cell: $_DFF_NP1_
    unmapped dff cell: $_DFF_PN0_
    unmapped dff cell: $_DFF_PN1_
    unmapped dff cell: $_DFF_PP0_
    unmapped dff cell: $_DFF_PP1_
    unmapped dff cell: $_DFFSR_NNN_
    unmapped dff cell: $_DFFSR_NNP_
    unmapped dff cell: $_DFFSR_NPN_
    unmapped dff cell: $_DFFSR_NPP_
    unmapped dff cell: $_DFFSR_PNN_
    unmapped dff cell: $_DFFSR_PNP_
    unmapped dff cell: $_DFFSR_PPN_
    unmapped dff cell: $_DFFSR_PPP_
Mapping DFF cells in module `\example':

yosys> abc -liberty mycells.lib
```

Fig. 4

5- Step5

In Fig. 5 according to red box5, write command “*write_verilog -noattr XXX.v*” to write final synthesis result to output files with name of XXX.v. sub command “*-noattr*” removes attributes in synthesis result.



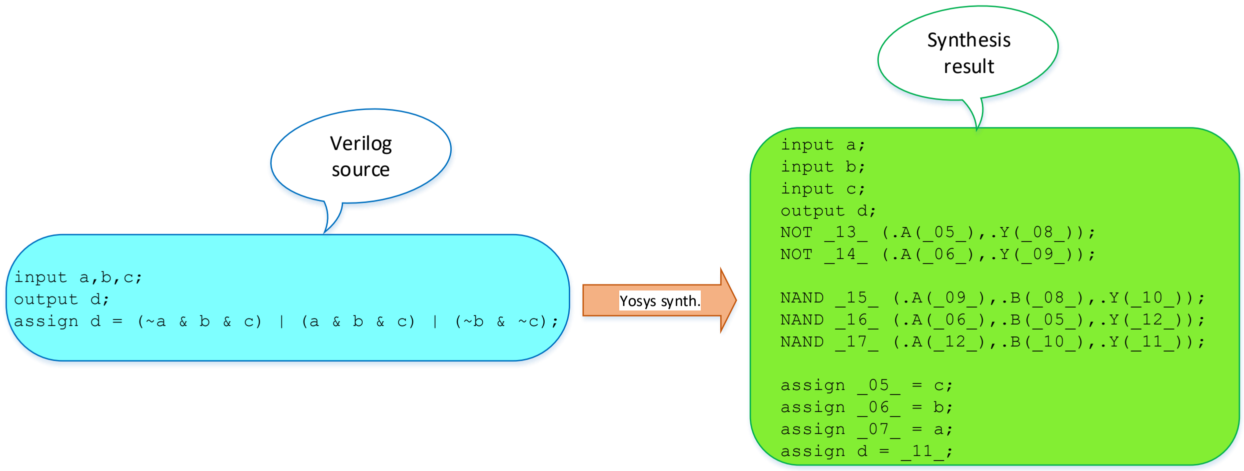
```
ABC: + strash
ABC: + dc2
ABC: + scorr
ABC: Warning: The network is combinational (run "fraig" or "fraig_sweep").
ABC: + ifraig
ABC: + retime -o
ABC: + strash
ABC: + dch -f
ABC: + map
ABC: + write_blif <abc-temp-dir>/output.blif

4.1.2. Re-integrating ABC results.
ABC RESULTS:              NAND cells:       3
ABC RESULTS:               NOT cells:       2
ABC RESULTS:        internal signals:       5
ABC RESULTS:           input signals:       3
ABC RESULTS:          output signals:       1
Removing temp directory.

yosys> write_verilog -noattr simple_synth.v
```

Fig. 5

Finally, your Verilog source converts to gate-level structure and stores in output Verilog file. Left side of Fig. 6 shows Verilog source by assign statement and right side shows synthesis output by gate structure.



```
input a,b,c;
output d;
assign d = (~a & b & c) | (a & b & c) | (~b & ~c);
```

```
input a;
input b;
input c;
output d;
NOT _13_ (.A(_05_),.Y(_08_));
NOT _14_ (.A(_06_),.Y(_09_));

NAND _15_ (.A(_09_),.B(_08_),.Y(_10_));
NAND _16_ (.A(_06_),.B(_05_),.Y(_12_));
NAND _17_ (.A(_12_),.B(_10_),.Y(_11_));

assign _05_ = c;
assign _06_ = b;
assign _07_ = a;
assign d = _11_;
```

Fig. 6