# SYNTHESIS
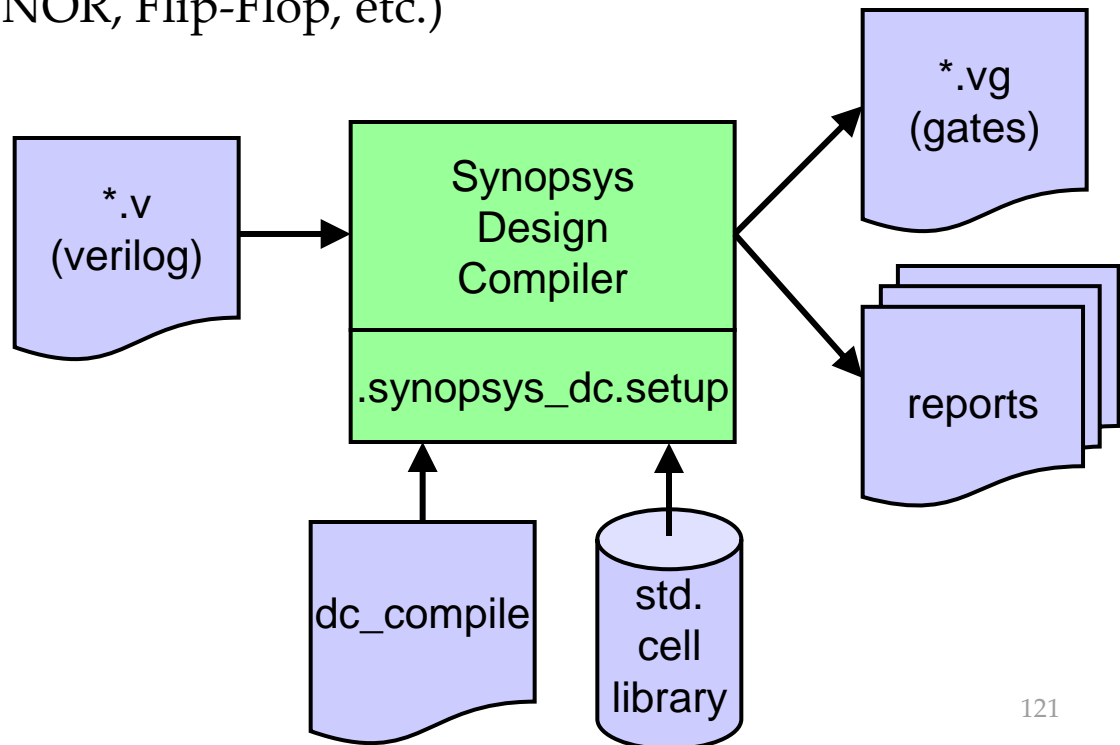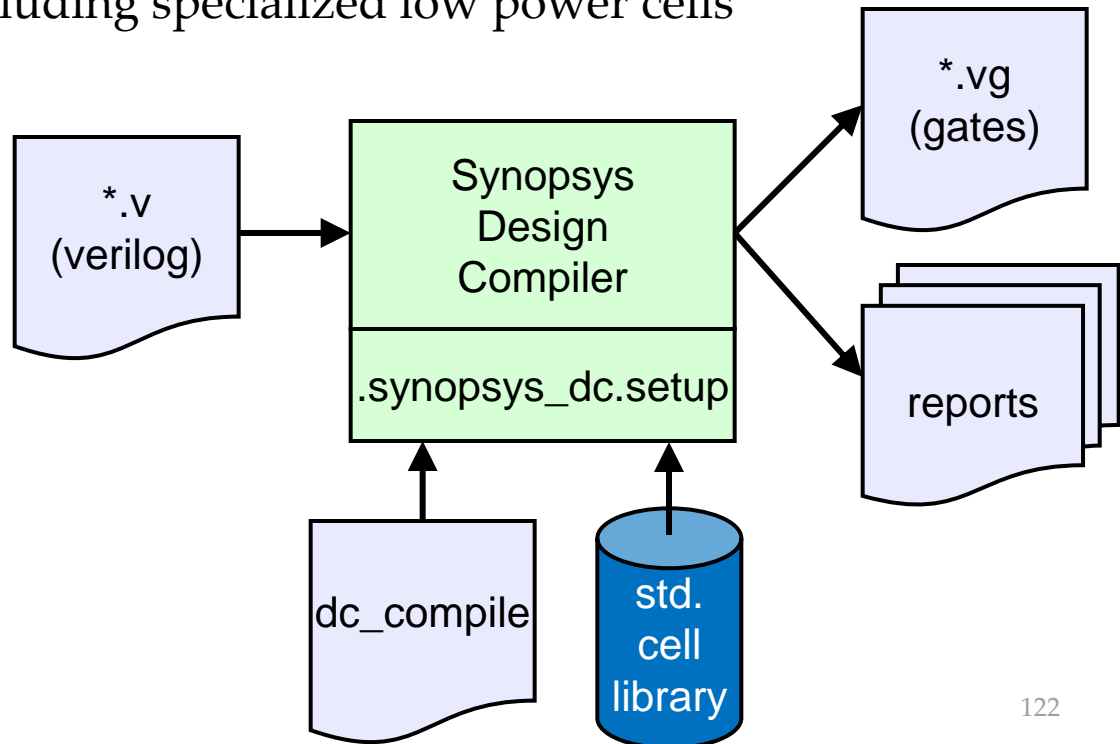
# Synthesis

- Involves synthesizing a gate netlist from verilog source code
- We use *Design Compiler* (DC) by Synopsys which is the most popular synthesis tool used in industry
- Target library examples:
  - Standard cell (NAND, NOR, Flip-Flop, etc.)
  - FPGA CLB
- Other key files
  - source verilog (or VHDL)
  - compile script
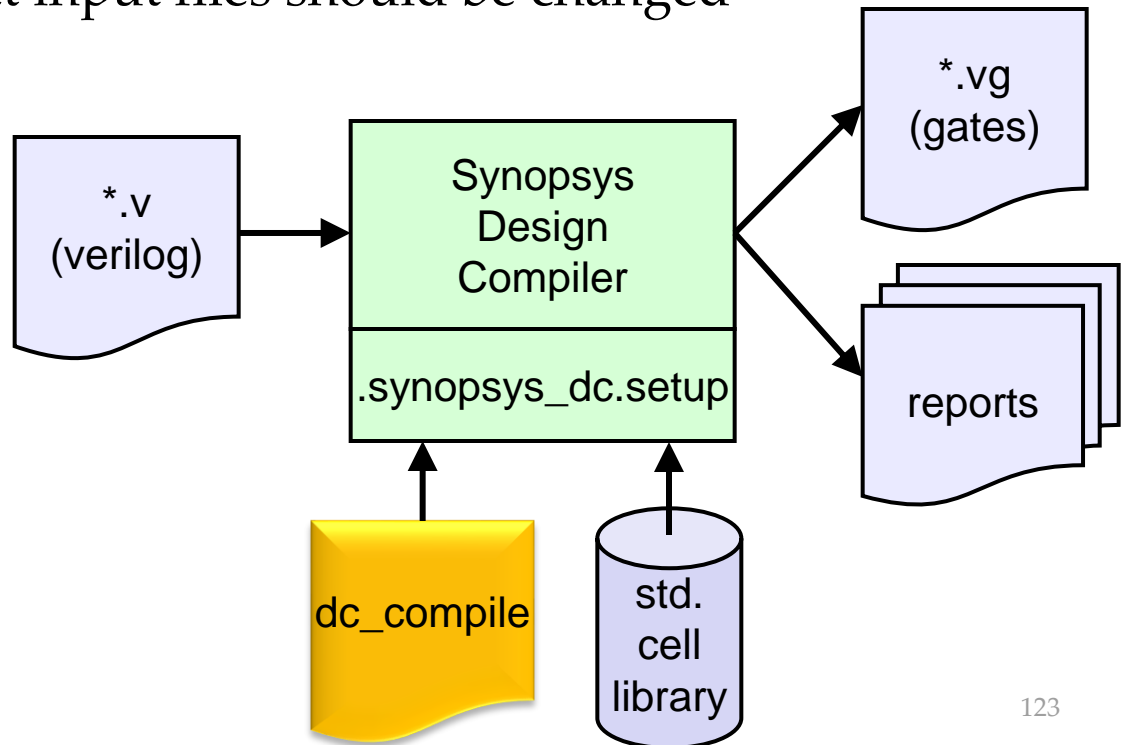  - output gate netlist
  - many reports

*.v
(verilog)

Synopsys
Design
Compiler

.synopsys_dc.setup

*.vg
(gates)

reports

dc_compile

std.
cell
library

# Standard Cell Library

- We currently use the 45 nm NanGate FreePDF45 Open Cell Library
  http://www.nangate.com/?page_id=2325
  which is an open-source library developed by NanGate Inc,
  http://www.nangate.com

- It contains "62 different functions ranging from buffers to scan flip-flops with set and reset, including specialized low power cells with multiple drive strength variants, the library includes over 170 different standard cells."

*.v
(verilog)

Synopsys
Design
Compiler

.synopsys_dc.setup

*.vg
(gates)

reports

dc_compile

std.
cell
library

# Compile Optimization Parameters

- There are many many configuration parameters which can be tuned to optimize the result of synthesis
- For homework/projects in this class, do not change any of these optimization parameters since this is not the focus of the class
- Only parameters to select input files should be changed
- Please talk to me if you would like to modify the script more extensively

*.v
(verilog)

Synopsys
Design
Compiler

.synopsys_dc.setup

*.vg
(gates)

reports

dc_compile

std.
cell
library

# Synthesis Key Files

- `Makefile`
  - Contains all commands needed for simulation and synthesis
  - Requires you to enter the top-level design name at the top of the file
  - Type "make <return>" to see make targets and instructions
- `dc-template.tcl`
  - Template used to generate a customized command file for Design Compiler
  - Do not edit this file unless you are told you need to
- `.synopsys_dc.setup`
  - Do not edit this file
  - Watch out for it since it appears in linux only with 'ls –a' and not just 'ls'
- `abc.v`
  - Very simple example design with 2-bit and 32-bit adders, and registers
- `abc.vfv`
  - File that contains all source verilog files for simulation (NCVerilog or Verilog-XL)
- `abc.vfs`
  - File that contains all source verilog files for synthesis (Design Compiler)

# Makefile

```
rattle_179> make

Make targets. Either change module name in Makefile line 35 or add the
text 'NAME=xyz' after 'make' for simulation and synthesis targets below.
  make                       print this help summary
  make clean                 deletes some generated files
  make cleanall              deletes all generated files
Make targets for simulation
  make compile               compile only with .vfv and NCVerilog
  make run                   run with NCVerilog
  make viewer                start simvision
Make targets for synthesis
  make check                 compile only with .vfs and NCVerilog
  make synth                 synthesize default module

Recommended procedure to synthesize top-level module 'xyz'
  1) change 'NAME := CHANGE_ME' to 'NAME := xyz' at top of Makefile
  2) add filenames of all modules to be synthesized to file xyz.vfs
  3) 'make check' and fix any errors
  4) 'make synth'

Alternate procedure
  1) add filenames of all modules to be synthesized to file xyz.vfs
  2) 'make NAME=xyz check' and fix any errors
  3) 'make NAME=xyz synth'
```

# Synthesis Timing

*clock*

*two timing paths* ②

① ± 50 ps

- cycle time = 10 nsec
- clock skew = 50 psec
- 2-input OR gate delay = 129.78 psec
- Example circuit where inputs and outputs are not registered so timing is extra complex
- `abc.tim`

*in0[0]*   *out[0]*

4000 ps   OR   750 ps

synthesized logic

```
      Startpoint: in0[0] (input port clocked by clk)
      Endpoint: out[0] (output port clocked by clk)
      Path Group: clk
      Path Type: max

      Point                                   Incr        Path
      -------------------------------------------------------------
```

| | Point | Incr | Path |
|---|---|---|---|
| **clock edge *n*** ① | clock clk (rise edge) | 0.00 | 0.00 |
| | clock network delay (ideal) | 0.00 | 0.00 |
| | input external delay | 4000.00 | 4000.00 f |
| | in0[0] (in) | 0.00 | 4000.00 f |
| | U10/op (or2_2) | 129.78 | 4129.78 f |
| | out[0] (out) | 0.00 | 4129.78 f |
| | data arrival time | | 4129.78 |
| | | | |
| **clock edge *n*+1** ② | clock clk (rise edge) | 10000.00 | 10000.00 |
| **(@ 100 MHz)** | clock network delay (ideal) | 0.00 | 10000.00 |
| | clock uncertainty | -50.00 | 9950.00 |
| | output external delay | -750.00 | 9200.00 |
| | data required time | | 9200.00 |

```
      -------------------------------------------------------------
      data required time                                  9200.00
      data arrival time                                  -4129.78
      -------------------------------------------------------------
      slack (MET)                                         5070.22
```
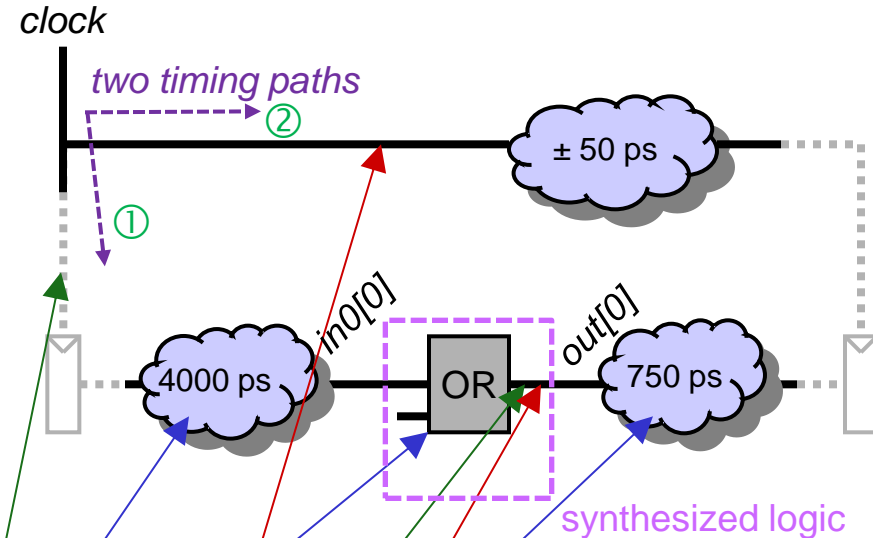
- Even thought the circuit does not contain registers (grayed out), the input and output signals are still timed with respect to them as if they were there.
- Timing is more straightforward when there are registers in the circuit.

126

# Synthesizing Hardware without Registers

- For quick estimates of combina-tional blocks, it is often helpful to synthesize a hardware design that does not contain registers

- But Design Compiler will give an "Unconstrained Paths" error if there is no clock signal and the *.tim timing report will not have necessary information

- To solve this, add a "clk" clock input to the hardware module that does not connect to anything. There will still be the following error which can be ignored:

  **port 'clk' is not connected to any nets. (LINT-28)**

```verilog
// or2.v
//
// practice with a pure-combinational circuit

`timescale 10ps/1ps
`celldefine
module or2 (
   in0,
   in1,
   out,
   clk
   );

   //----- Inputs/outputs
   input  [1:0]        in0;
   input  [1:0]        in1;
   output [1:0]        out;

   input               clk;

   //----- 2-bit adder
   wire [1:0]   out;

   assign out = in0 | in1;

endmodule   /* or2 */
`endcelldefine
```

# Synthesis Timing

*Estimated achievable cycle time = Target cycle time − Timing slack*

- A **positive** slack value implies that the circuit is estimated to perform at a higher clock rate than the target clock rate (1/*TargetCycleTime*)

- A **negative** slack value implies that the circuit is estimated to be unable to achieve the target clock frequency

# Synthesis Circuit Area

- `abc.area`

```
****************************************
Report : area
Design : abc
Version: V-2004.06-SP2
Date   : Thu Feb  3 15:56:49 2005
****************************************

Library(s) Used:

    vtvtlib25 (File: /afs/ece/classes/Html/Winter04/eec281/lib/vtvtlib25.db)

Number of ports:             31
Number of nets:              30
Number of cells:             10
Number of references:         1

Combinational area:       699.840027
Noncombinational area:      0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:          699.840027
Total area:                 undefined
```

# Synthesis Gate Netlist

- `abc.vg`

```
module prac ( in0, in1, out, clk );
  input [9:0] in0;
  input [9:0] in1;
  output [9:0] out;
  input clk;

  or2_2 U1 ( .ip1(in0[9]), .ip2(in1[9]), .op(out[9]) );
  or2_2 U2 ( .ip1(in0[8]), .ip2(in1[8]), .op(out[8]) );
  or2_2 U3 ( .ip1(in0[7]), .ip2(in1[7]), .op(out[7]) );
  or2_2 U4 ( .ip1(in0[6]), .ip2(in1[6]), .op(out[6]) );
  or2_2 U5 ( .ip1(in0[5]), .ip2(in1[5]), .op(out[5]) );
  or2_2 U6 ( .ip1(in0[4]), .ip2(in1[4]), .op(out[4]) );
  or2_2 U7 ( .ip1(in0[3]), .ip2(in1[3]), .op(out[3]) );
  or2_2 U8 ( .ip1(in0[2]), .ip2(in1[2]), .op(out[2]) );
  or2_2 U9 ( .ip1(in0[1]), .ip2(in1[1]), .op(out[1]) );
  or2_2 U10 ( .ip1(in0[0]), .ip2(in1[0]), .op(out[0]) );
endmodule
```

# Synthesis, Other Output Files

- `abc.cell`
  - All cells used
  - Area per cell
- `abc.logv`
  - General log file for simulations
  - Skip reading it for errors and warnings at your peril!
- `abc.logs`
  - General log file for synthesis
  - Skip reading it for errors and warnings at your peril!

# Reported Circuit Power

- Don't trust power numbers
- Don't trust any uncalibrated CAD tool anyway
- Power *comparisons* are likely useful to estimate the value or cost of design modifications, however
- The greatest sources of inaccuracies come from:
  - Interconnect (wires) dissipates a majority of the power in almost any design in a modern CMOS technology
  - The synthesis tool may or may not even have an equation to estimate wire capacitance (and by $E = CV^2$, a way to estimate energy per operation)
  - Even if it tries to estimate wire capacitance, it will surely be wrong!  The only way to know it accurately is to do the actual cell placement and wire routing ("place & route") of the design

# Synthesis Timing Example II

```
Startpoint: r_in31a_reg[0]
          (rising edge-triggered flip-flop clocked by clk)
Endpoint: out32_reg[31]
          (rising edge-triggered flip-flop clocked by clk)
Path Group: clk
Path Type: max

Des/Clust/Port     Wire Load Model     Library
-----------------------------------------------
abc                5K_hvratio_1_1      NangateOpenCellLibrary

Point                               Incr      Path
-----------------------------------------------------
clock clk (rise edge)               0.00      0.00
clock network delay (ideal)         0.00      0.00
r_in31a_reg[0]/CK (DFF_X1)          0.00      0.00 r
r_in31a_reg[0]/Q (DFF_X1)           0.08      0.08 f
U1/ZN (AND2_X1)                     0.04      0.13 f
U1_1/CO (FA_X1)                     0.09      0.22 f
U1_2/CO (FA_X1)                     0.09      0.31 f
U1_3/CO (FA_X1)                     0.09      0.40 f
U1_4/CO (FA_X1)                     0.09      0.50 f
U1_5/CO (FA_X1)                     0.09      0.59 f
U1_6/CO (FA_X1)                     0.09      0.68 f
U1_7/CO (FA_X1)                     0.09      0.78 f
U1_8/CO (FA_X1)                     0.09      0.87 f
U1_9/CO (FA_X1)                     0.09      0.96 f
U1_10/CO (FA_X1)                    0.09      1.06 f
U1_11/CO (FA_X1)                    0.09      1.15 f
U1_12/CO (FA_X1)                    0.09      1.24 f
U1_13/CO (FA_X1)                    0.09      1.33 f
U1_14/CO (FA_X1)                    0.09      1.43 f
U1_15/CO (FA_X1)                    0.09      1.52 f
U1_16/CO (FA_X1)                    0.09      1.61 f
U1_17/CO (FA_X1)                    0.09      1.71 f
U1_18/CO (FA_X1)                    0.09      1.80 f
U1_19/CO (FA_X1)                    0.09      1.89 f
U1_20/CO (FA_X1)                    0.09      1.99 f
U1_21/CO (FA_X1)                    0.09      2.08 f
U1_22/CO (FA_X1)                    0.09      2.17 f
U1_23/CO (FA_X1)                    0.09      2.27 f
U1_24/CO (FA_X1)                    0.09      2.36 f
U1_25/CO (FA_X1)                    0.09      2.45 f
U1_26/CO (FA_X1)                    0.09      2.54 f
U1_27/CO (FA_X1)                    0.09      2.64 f
U1_28/CO (FA_X1)                    0.09      2.73 f
U1_29/CO (FA_X1)                    0.09      2.82 f
U1_30/CO (FA_X1)                    0.09      2.92 f
U1_31/S (FA_X1)                     0.13      3.05 r
out32_reg[31]/D (DFF_X1)            0.01      3.06 r
data arrival time                             3.06

clock clk (rise edge)               4.00      4.00
clock network delay (ideal)         0.00      4.00
clock uncertainty                  -0.20      3.80
out32_reg[31]/CK (DFF_X1)           0.00      3.80 r
library setup time                 -0.03      3.77
data required time                            3.77
-----------------------------------------------------
data required time                            3.77
data arrival time                            -3.06
-----------------------------------------------------
slack (MET)                                   0.71
```

**clock edge *n***

①

**clock edge *n+1***
**(@ 250 MHz)**

②

*clock*

*two timing paths*

②

± 200 ps

①

logic
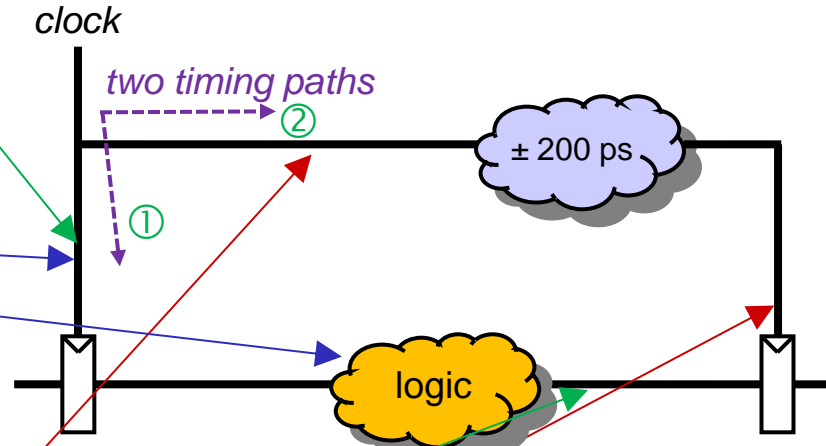
- cycle time = 4 nsec
- clock skew = 200 psec

133

© B. Baas