POLITÉCNICA

"Ingeniamos el futuro"

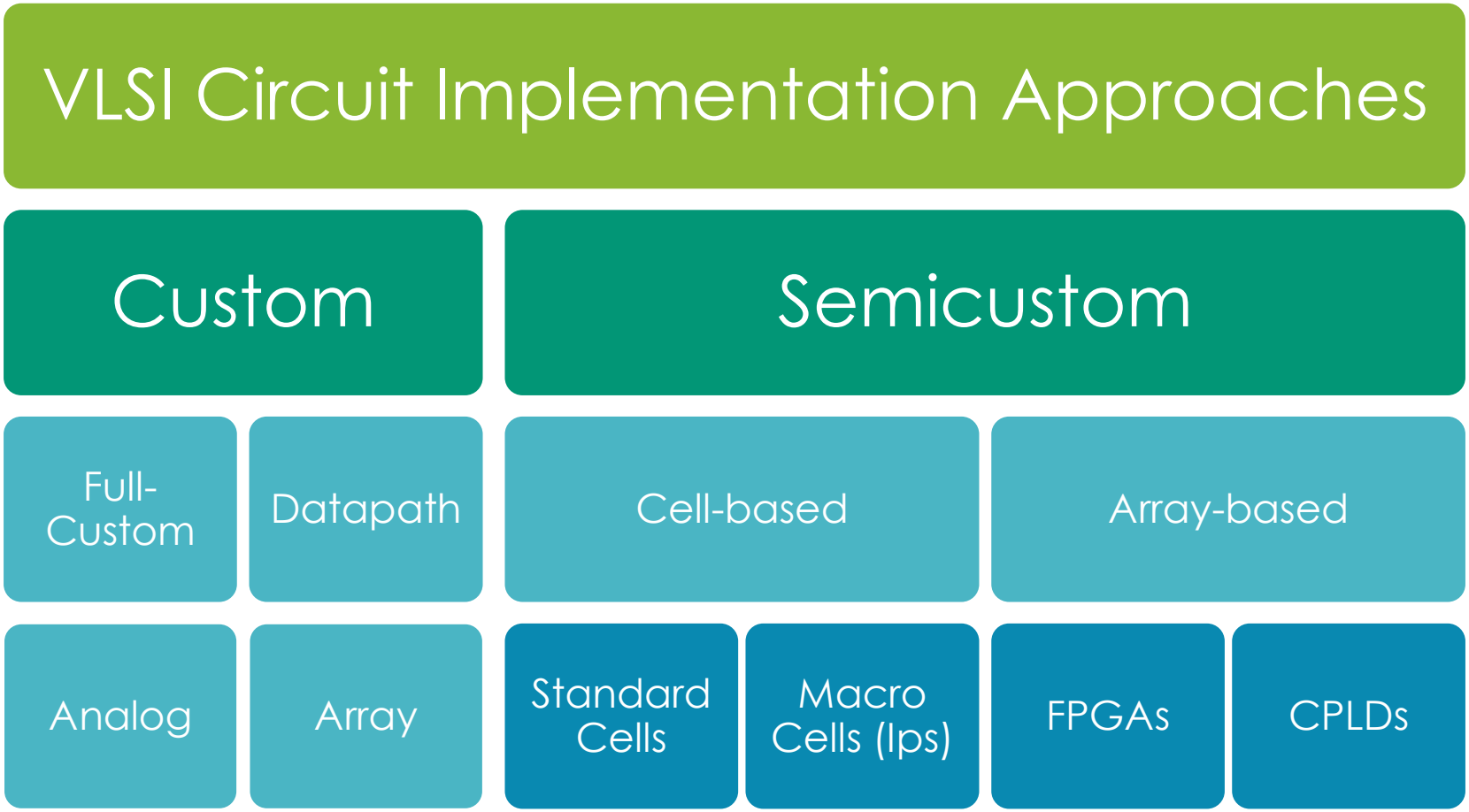
Departamento de Ingeniería Electrónica

Síntesis Lógica con Synopsys y Simulación con QuestaSim

LDIM, Curso 2013-2014

Pablo Ituero

VLSI Implementation Choices



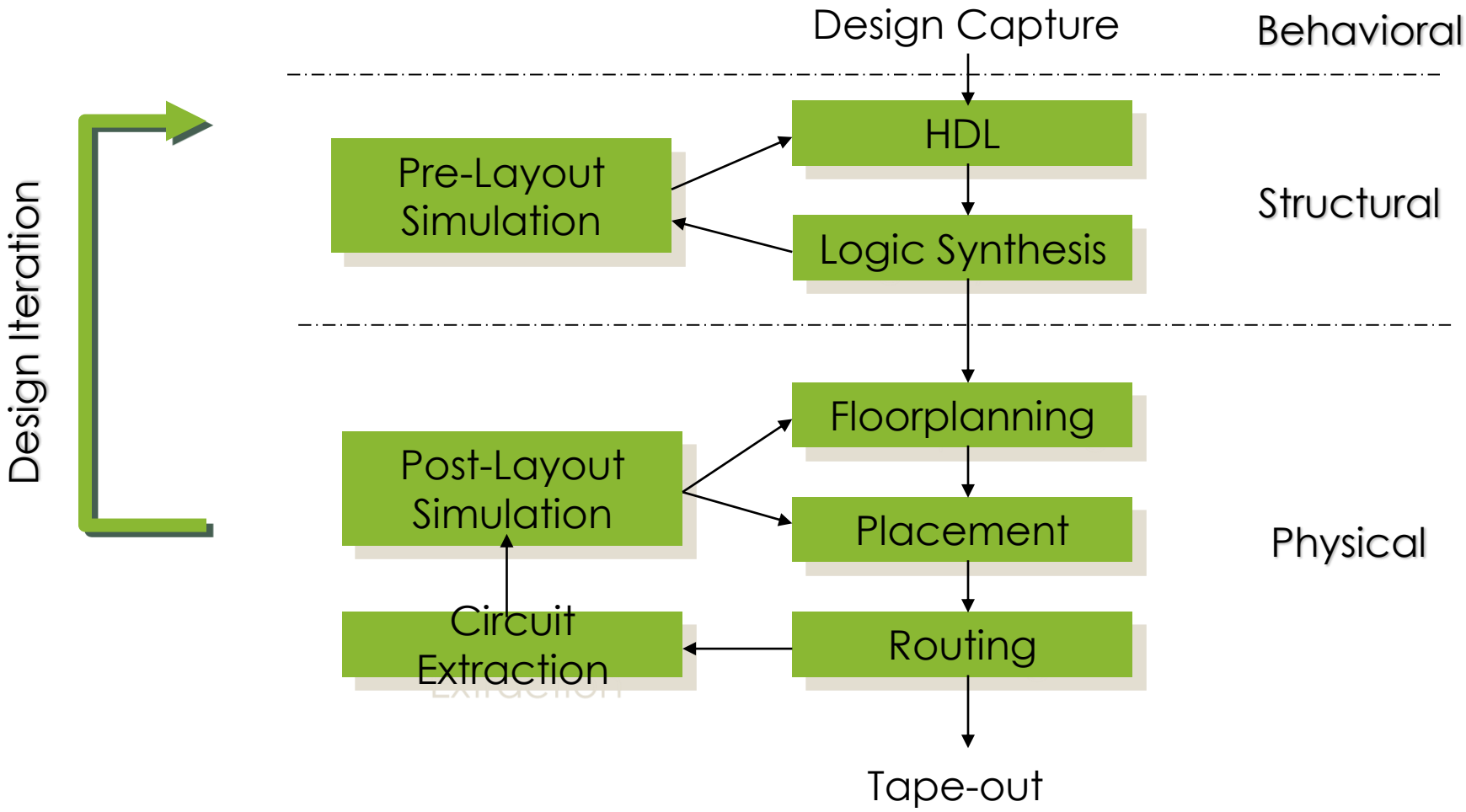
Circuitos Semi-custom

- El esfuerzo de diseño se desplaza a la arquitectura y a la algorítmica (síntesis, colocación, rutado, verificación, …)
- Circuitos *Basados en Células*:
  - Utilizan una biblioteca de células y generadores de memorias.
  - Síntesis desde un código HDL, placement y rutado automático.
  - Actualmente es el tipo de ASIC más popular. Muy buenos resultados de prestaciones en tiempo menor. Prestaciones medias-altas. Tendencia actual para grandes tiradas.
- *FPGAS y CPLD*:
  - Arrays de puertas ya fabricados, toda la parte de interacción con el fabricante, encapsulado, prototipado, testeo, etc., desaparece.
  - Ciclo de trabajo muy rápido. Ideal para prototipado o diseños de tiradas muy bajas.
  - Prestaciones limitadas debido a las estructuras prediseñadas.

Parte 2: Semi-Custom

- Diseño físico digital a partir de un código HDL
- Foco en metodología y automatización
- Desde síntesis hasta place&route.
- Entorno de trabajo profesional. Herramientas CAD de la industria. Librería de células estándar real fabricable AMS 0.35um.
  - Síntesis: Synopsys
  - Diseño físico: Cadence
  - Simulación: Modelsim

Semicustom Design Flow

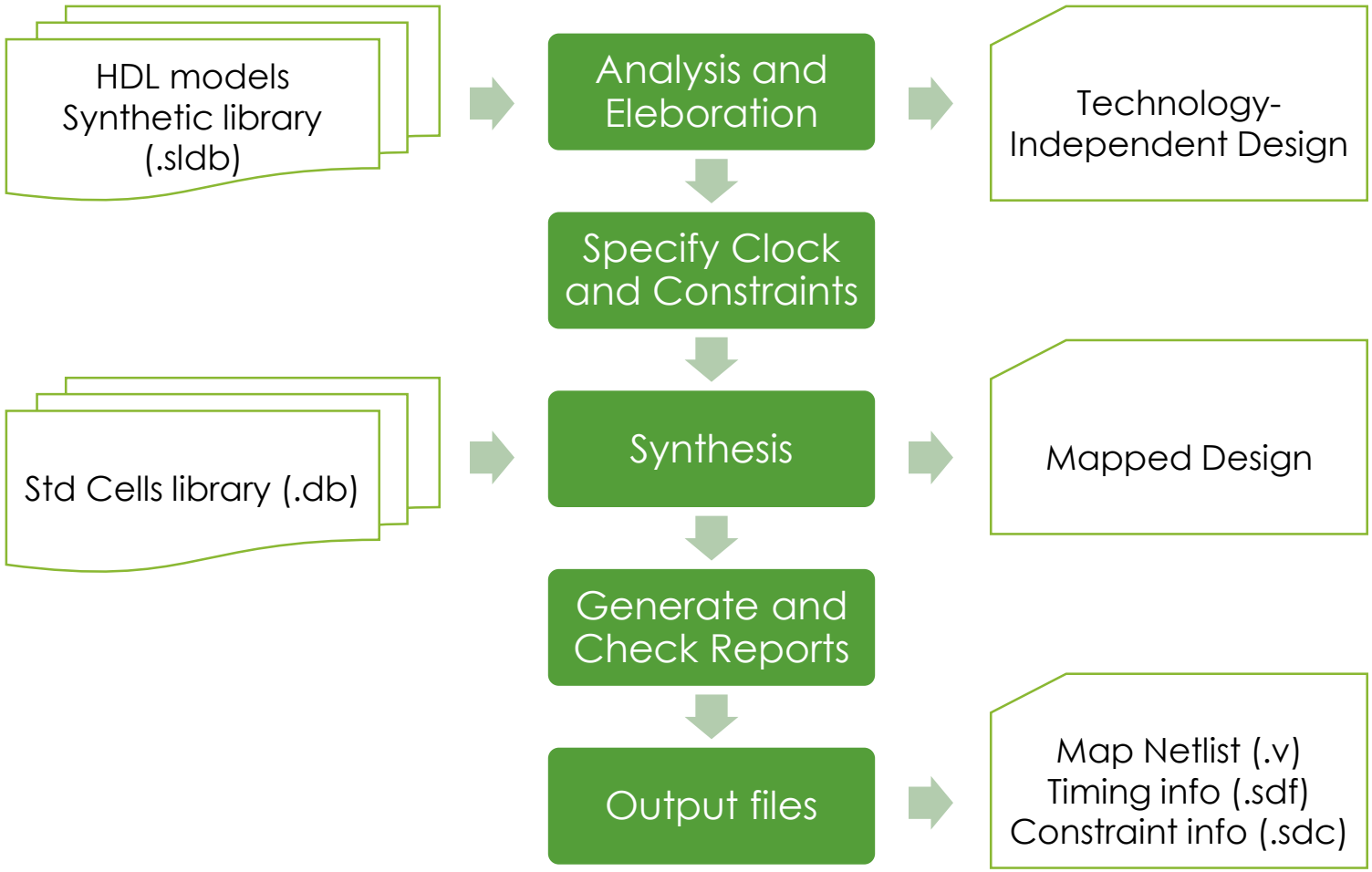


Organización y Evaluación

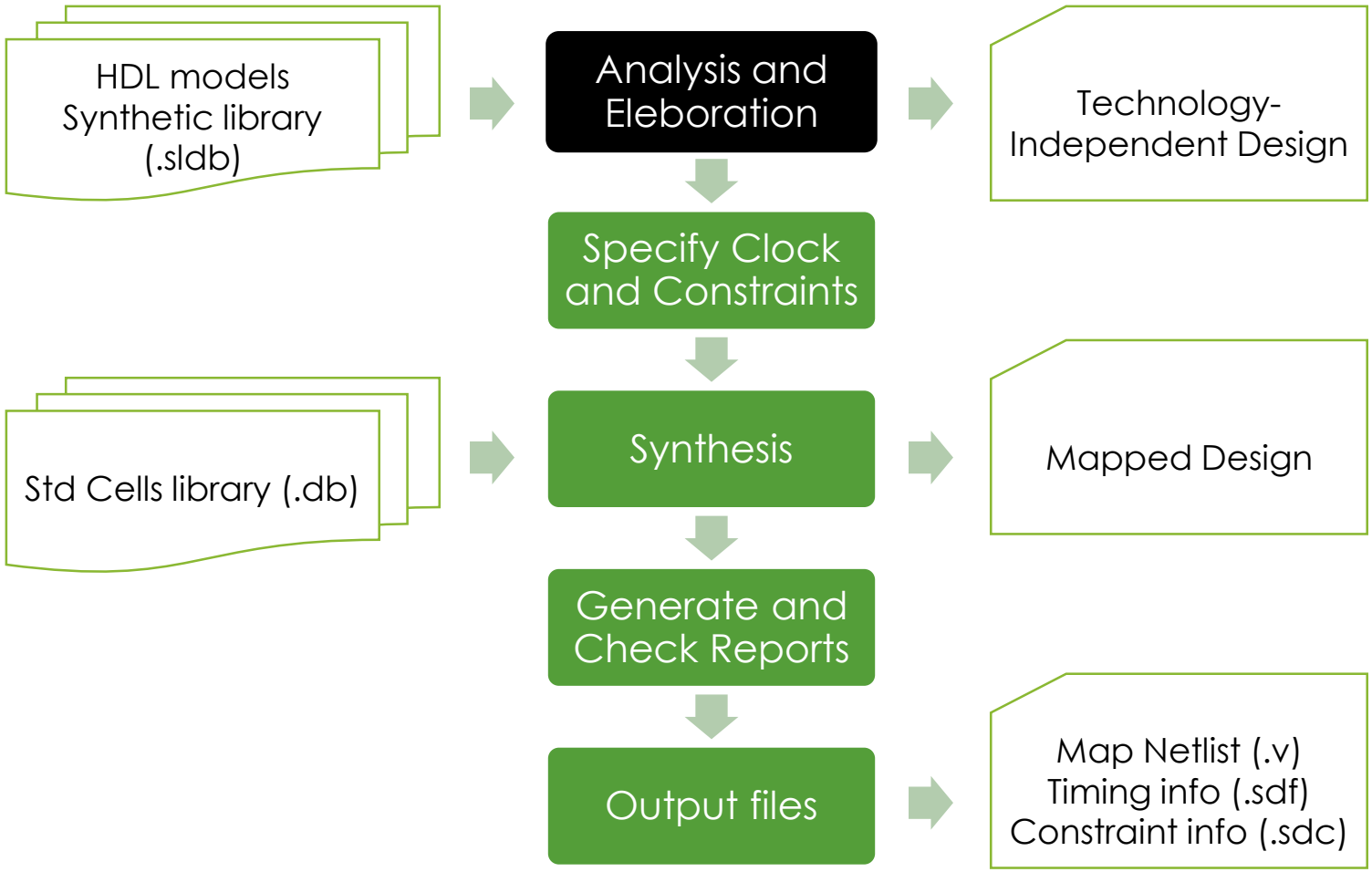
- El laboratorio permanecerá abierto la mañana del jueves de 10 a 13h y todas las tardes de lunes a viernes de 15 a 18h.

- **Parte 1: Diseño Full-Custom**
  - **Semana 0** (10-14 Febrero 2014): Elección de turno e inicio del curso el viernes 15 a las 13.00 en el laboratorio B-043.
  - **Semana 1** (17-21 Febrero 2014): Aprendizaje de la herramienta icfb. Diseño, simulación y caracterización de un inversor. Diseño, simulación y caracterización de dos células básicas: NAND, NOR de dos entradas o similar.
  - **Semana 2** (24-28 Febrero 2014): Parámetros, análisis de corners y análisis estadístico.
  - **Semana 3** (3-7 Marzo 2014): Trazados, DRC, LVS y backannotation.
  - **Semanas 4-6** (10-28 Marzo 2014) Realización de la práctica final de la parte 1, diseño, simulación, caracterización y trazado de un bloque de complejidad media-baja.
- **Parte 2: Diseño Semi-Custom**
  - **Semana 7** (31 Marzo - 4 Abril 2014): Síntesis lógica con Synopsys. Simulación. Optimización de consumo.
  - **Semana 8** (6-11 Abril 2014): Colocación y rutado con Encounter. Utilización de scripts.
  - **Semanas 9-10** (22 Abril - 6 Mayo 2014): Realización de la práctica final de la parte 2. Diseño físico de un circuito digital de complejidad alta descrito en VHDL.

Synthesis Flow



Synthesis Flow



Analysis and Elaboration

- Analysis:
  - Reads the HDL source and checks it for syntactical errors
  - Creates HDL library of objects in an HDL-indepentdent format
- Elaboration:
  - Translates the design into a technology-independent design (GTECH)
  - Generics are defined
  - Arithmetic operators replaced by DesignWare components
  - Links the whole design to resolve references

DesignWare Synthetic Library

- Technology-independent microarchitectural-level libraries from Synopsys.
- Implementation for various IP blocks.
- Advanced implementation for built-in operators that improve performance.
- Also complex operators:
  - MAC, SOP and vector adders
  - DSP blocks (FIR, IIR filters)
  - Memories
  - Microcontrolers

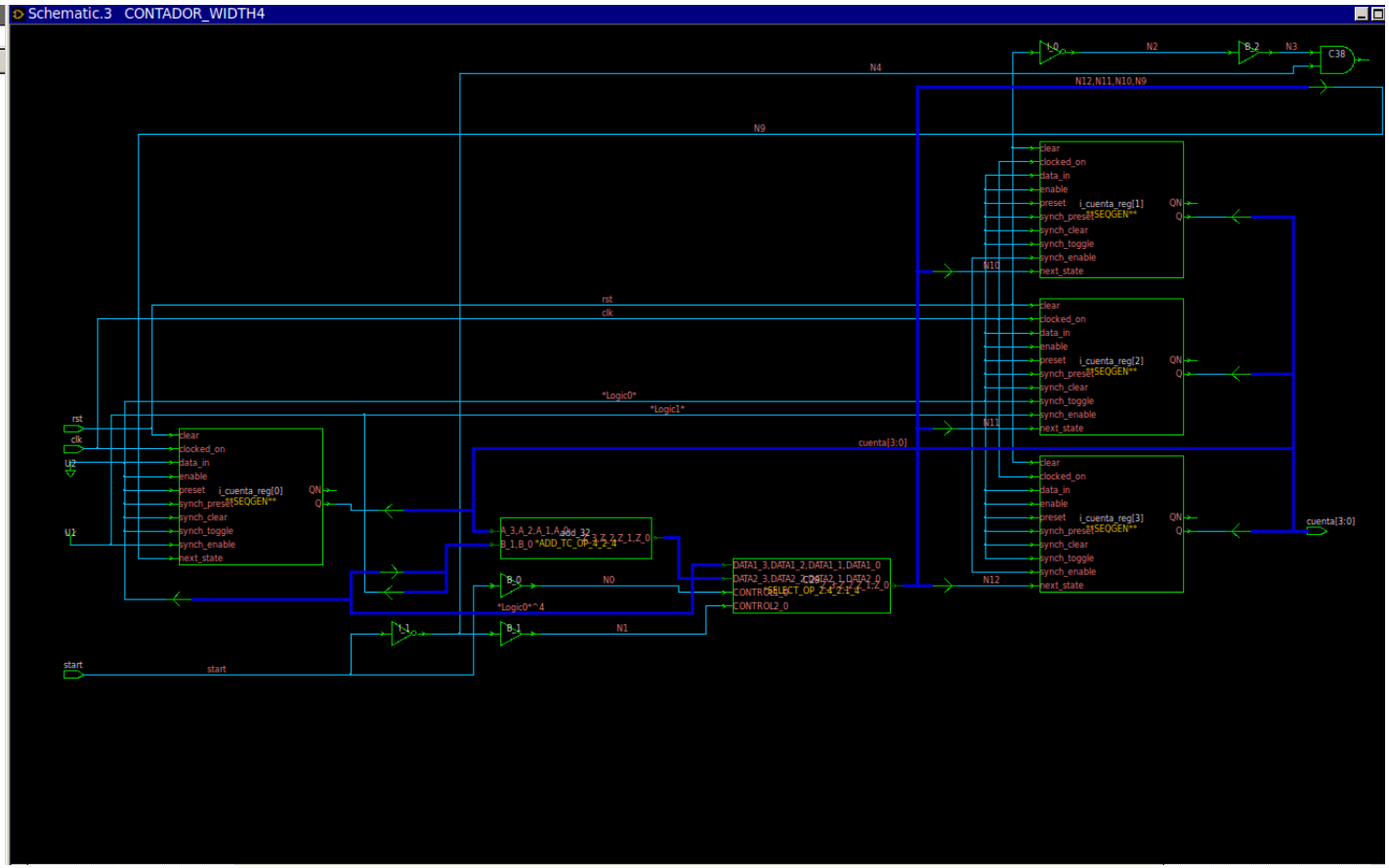
Example: contador.vhdl

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;

entity CONTADOR is
  generic (
    WIDTH     : integer := 8
    );
  port (
    cuenta    : out STD_LOGIC_VECTOR (WIDTH-1
downto 0);
    start     : in  STD_LOGIC;
    rst       : in  STD_LOGIC;
    clk       : in  STD_LOGIC
    );
end CONTADOR;

architecture Behavioral of CONTADOR is
  signal i_cuenta : std_logic_vector (WIDTH-1
downto 0);
begin

  cuenta <= i_cuenta;

  process (rst, clk)
  begin
    if rst='1' then
      i_cuenta <= (others => '0');
    elsif clk='1' and clk'event then
      if start='1' then
        i_cuenta <= (others => '0');
      else
        i_cuenta <= i_cuenta + '1';
      end if;
    end if;
  end process;
end Behavioral;
```

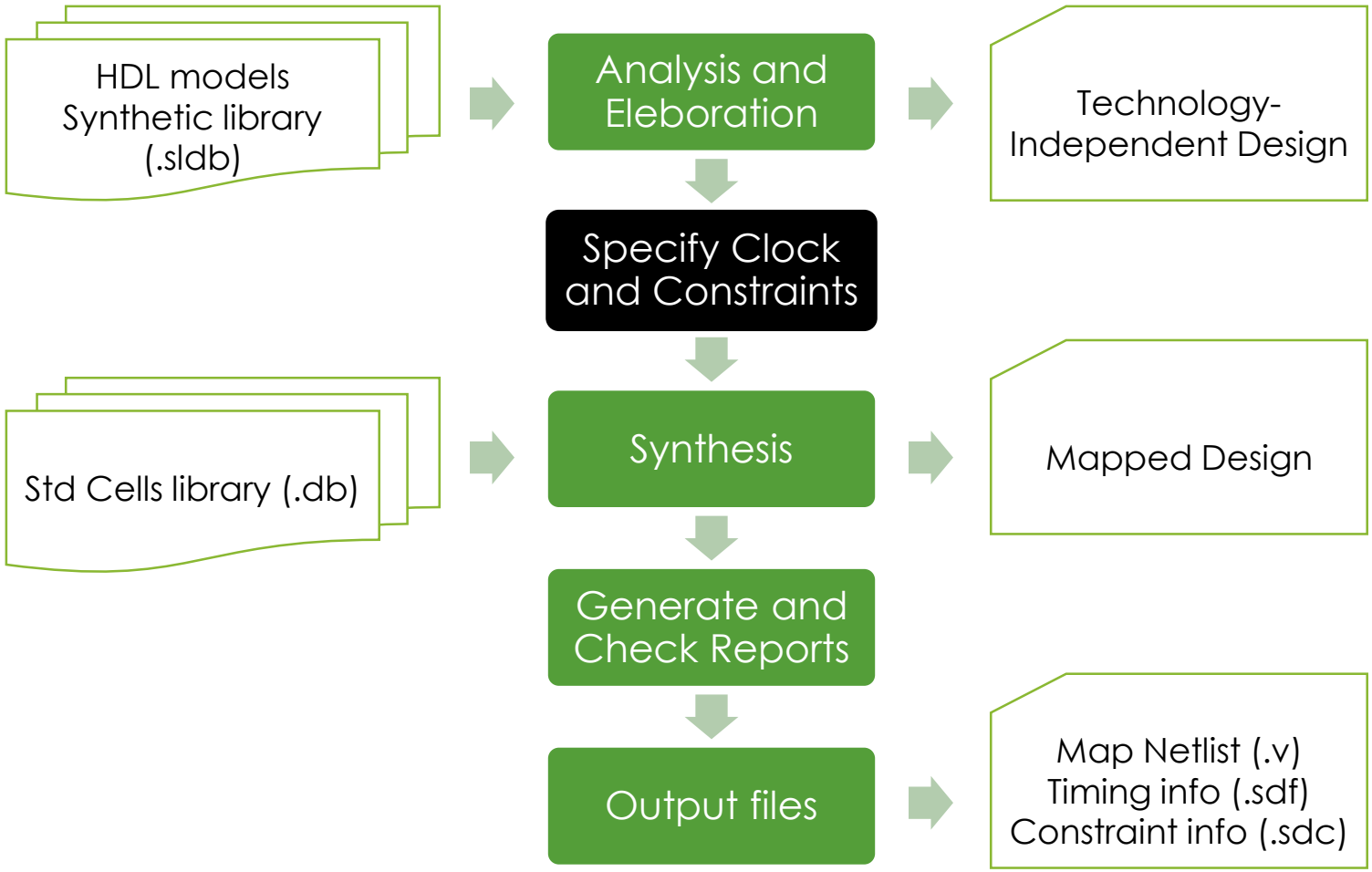
Elaborated Design

Cells (All)

Cell Name	Ref Name	Cell Path
i_cuenta_reg[3]	**SEQGEN**	i_cuenta_reg[3]
i_cuenta_reg[2]	**SEQGEN**	i_cuenta_reg[2]
i_cuenta_reg[1]	**SEQGEN**	i_cuenta_reg[1]
i_cuenta_reg[0]	**SEQGEN**	i_cuenta_reg[0]
add_32	*ADD_TC_OP_4_2_4	add_32
C29	*SELECT_OP_2.4_2.1_4	C29
B_0	GTECH_BUF	B_0
B_1	GTECH_BUF	B_1
U1	**logic_1**	U1
U2	**logic_0**	U2
I_0	GTECH_NOT	I_0
B_2	GTECH_BUF	B_2
I_1	GTECH_NOT	I_1
C38	GTECH_AND2	C38



Synthesis Flow



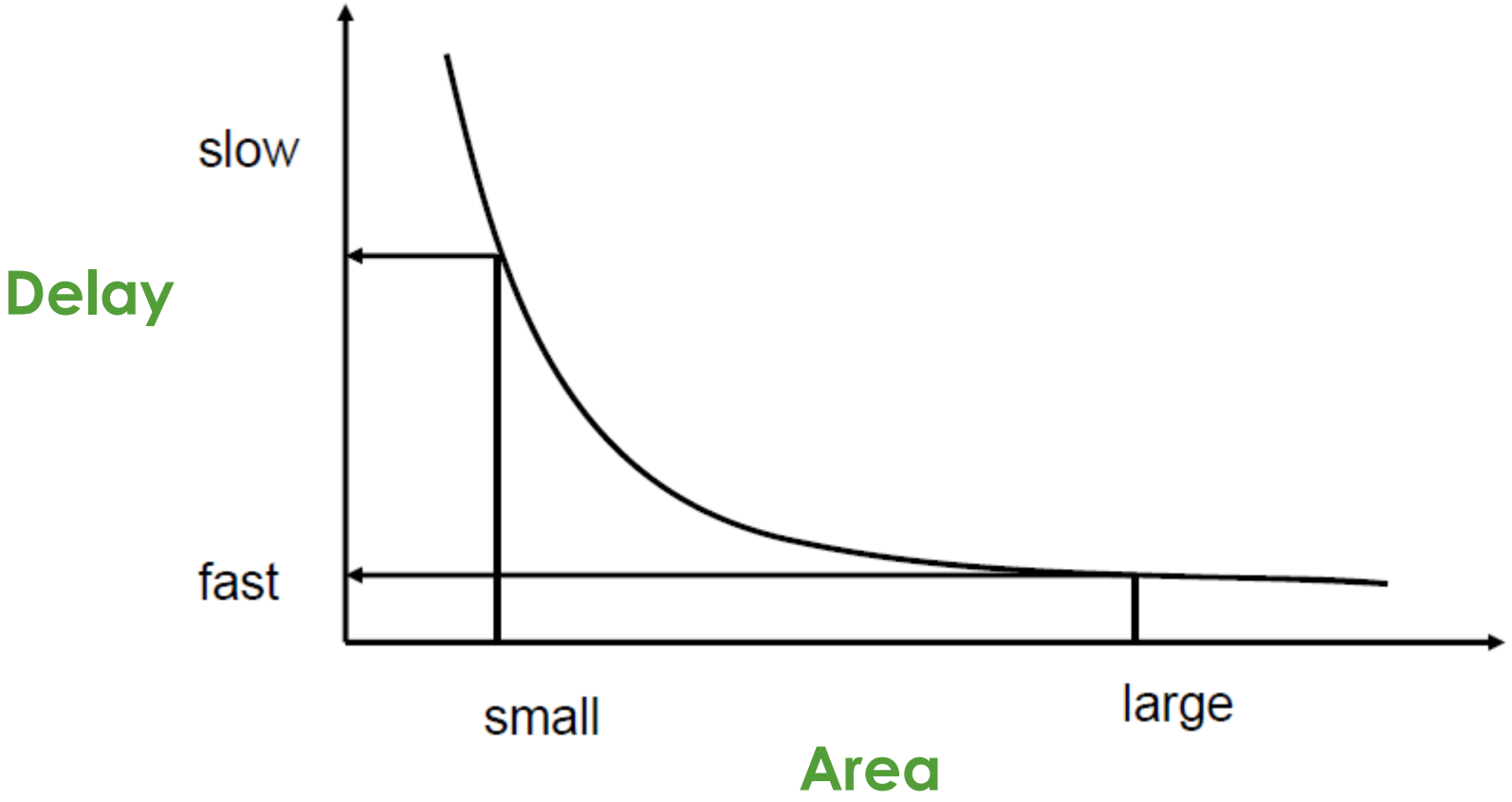
Constraints

- Many different constraints available to apply to design:
  - Timing Constraints
    - Max clock period
    - Maximum slack
    - Max delay (non-clocked logic)
  - Area Constraints
    - Max area
  - Power Constraints
    - Max Dynamic Power

Clock Specification

- Sets the speed goal for synthesis. Big impact on synthesis time.
- The period is the maximum combinational delay allowed in your design.
- If your design is purely combinational, you can set a virtual clock to set the speed goal.

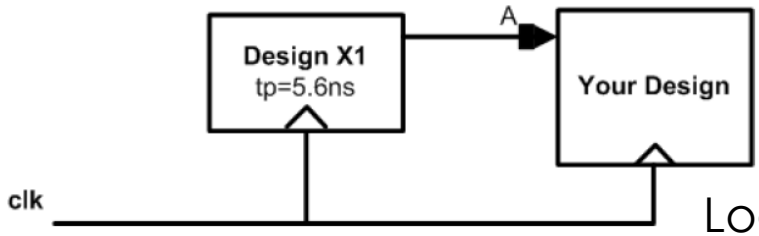
Area vs. Speed Trade-off



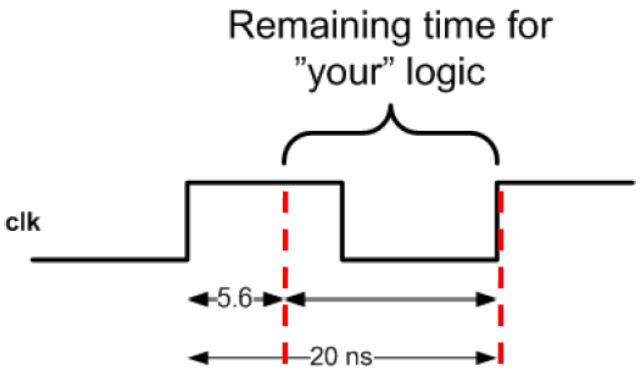
Area vs. Speed

- Synthesis tool prioritizes total negative slack over area
- A design that does not meet timing will not work
- For a high-speed circuit do not set any area constraint and specify a high clock frequency
- For an area optimized circuit set area to 0 and specify a low clock frequency

Constraining Inputs



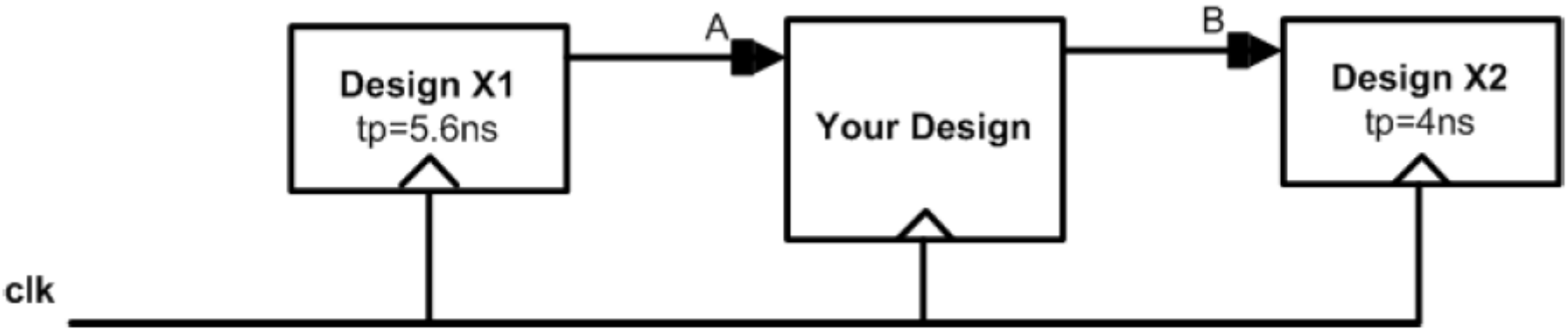
Logic is triggered by the clock



You need to specify propagation delay of external the logic that drives your logic

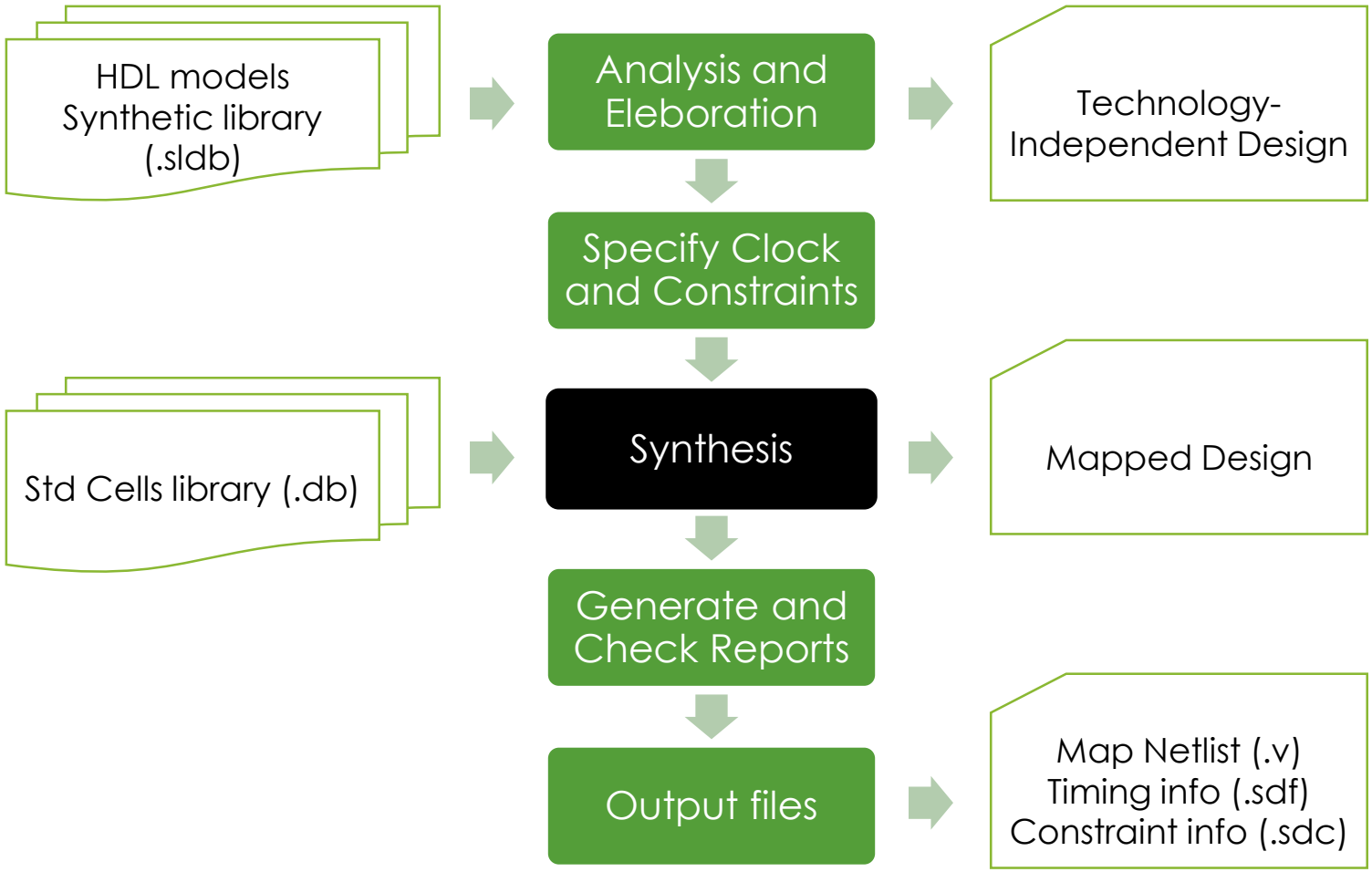
- Also, you need to specify the drive strength (eq. resistance) of the driving logic. Normally you set the cell that drives the input.

Constraining Output Paths



- You need to specify propagation delay of external logic that is driven by your logic.
- Also, you need to specify the capacitance of the driven logic. Normally you set the pin of the cell that drives the input.

Synthesis Flow



Synthesis and Optimization

- Maps the design to an optimal combination of specific target library cells, based on the design constraints.
- Standard cell library (.db), Synopsys compiled version of (.lib)
- Optimization at three levels:
  - Architectural
  - Logic
  - Gate

Target library

- Contains information about the characteristics and functions of each cell
  - Cell names
  - Pin names
  - Area
  - Delay
  - Pin loading
  - Maximum transition time for nets
  - Operating conditions
- Different libraries for each process corner

Cell definition example (.lib)

```
cell(INV1) {
        area : 36.4;
        cell_footprint : "CORE_INV";
        cell_leakage_power : 0.288043 ;
        pin(A) {
                direction : input;
                max_transition : 4;
                capacitance : 0.007;
                fanout_load : 6.606;
        }
        pin(Q) {
                direction : output;
                max_capacitance : 0.32;
                internal_power() {
                rise_power(power_outputs_0) {
                values( " 0.060, 0.072, 0.085, 0.089, 0.090",\
                        " 0.088, 0.087, 0.085, 0.079, 0.075",\
                        " 0.126, 0.121, 0.108, 0.086, 0.074",\
                        " 0.167, 0.161, 0.147, 0.121, 0.099");
                }
                fall_power(power_outputs_0) {
                values( " 0.008, 0.010, 0.010, 0.008, 0.008",\
                        " 0.029, 0.026, 0.020, 0.014, 0.011",\
                        " 0.066, 0.060, 0.046, 0.028, 0.021",\
                        " 0.106, 0.097, 0.077, 0.048, 0.035");
                }
                related_pin  : "A";
                }
                function  : "(A)'";
                timing() {
                related_pin     : "A";
                timing_sense    : negative_unate;
                cell_fall(del_0_4_5) {
                values( " 0.030, 0.046, 0.110, 0.321, 0.587",\
                        " 0.039, 0.072, 0.152, 0.375, 0.642",\
                        " 0.040, 0.073, 0.176, 0.443, 0.706",\
                        " 0.041, 0.074, 0.184, 0.498, 0.779");
                }
                fall_transition(del_0_4_5)  {
                values( " 0.054, 0.104, 0.244, 0.656, 1.240",\
                        " 0.159, 0.184, 0.289, 0.680, 1.258",\
                        " 0.225, 0.275, 0.398, 0.747, 1.264",\
                        " 0.300, 0.351, 0.493, 0.868, 1.357");
                }
                cell_rise(del_0_4_5) {
                values( " 0.039, 0.071, 0.171, 0.491, 0.913",\
                        " 0.110, 0.142, 0.235, 0.559, 0.977",\
                        " 0.168, 0.211, 0.327, 0.649, 1.063",\
                        " 0.221, 0.271, 0.404, 0.751, 1.174");
                }
                rise_transition(del_0_4_5)  {
                values( " 0.110, 0.192, 0.383, 1.140, 2.200",\
                        " 0.201, 0.235, 0.418, 1.163, 2.202",\
                        " 0.255, 0.309, 0.486, 1.168, 2.225",\
                        " 0.309, 0.380, 0.569, 1.246, 2.227");
                }
                }
        }
}       /* cell(INV1) */
/*
```

Architectural Optimizations

- Works on the HDL description.
- Arithmetic Optimizations. Rules of algebra. Rearrange operations.
- Resource Sharing. E.g. several addition operators can be implemented with just one adder component.
- Choose a specific implementation for a module. E.g. Ripple carry, carry look ahead, fast carry look ahead, etc.
- The output is a generic technology independent net list.

Logic Optimizations

- Structuring: Adds intermediate variables and logic structures to a design, which can result in reduced design area. (true by default)

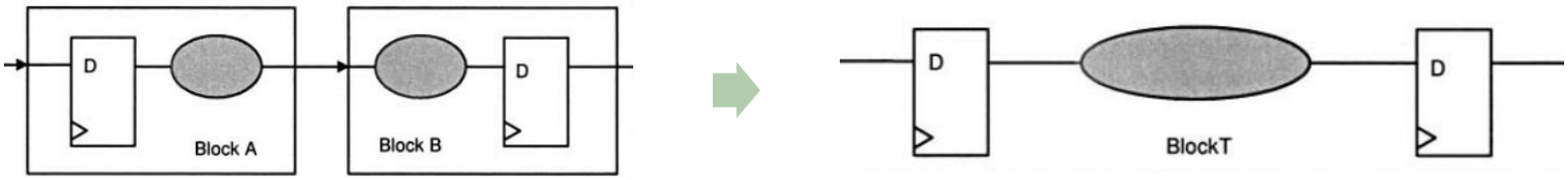
$$X1 = ab + ad \quad X2 = bc + cd \Rightarrow Y1 = b + d \quad X1 = a \cdot Y1 \quad X2 = c \cdot Y2$$

- Flattening: Converts logic into two-level, Sum-of-Products representation. Fast logic, area increase. (false by default)

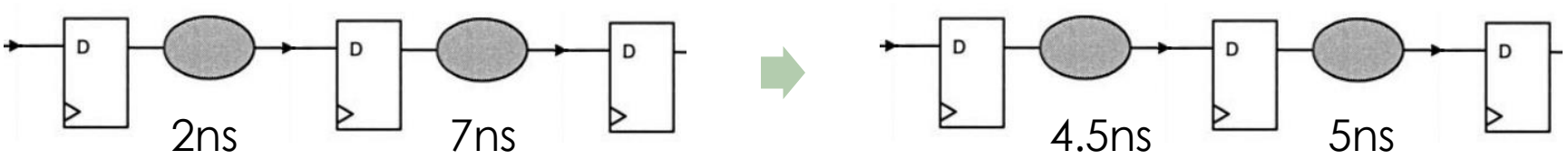
$$X1 = a + b \quad X2 = X1 \cdot c \quad X3 = X2 + d \Rightarrow Y1 = ac \quad X1 = bc \quad X2 = Y1 + Y2 + d$$

Logic Optimizations

- Hierarchy Removal. (false by default)



- Register Balancing. (false by default)



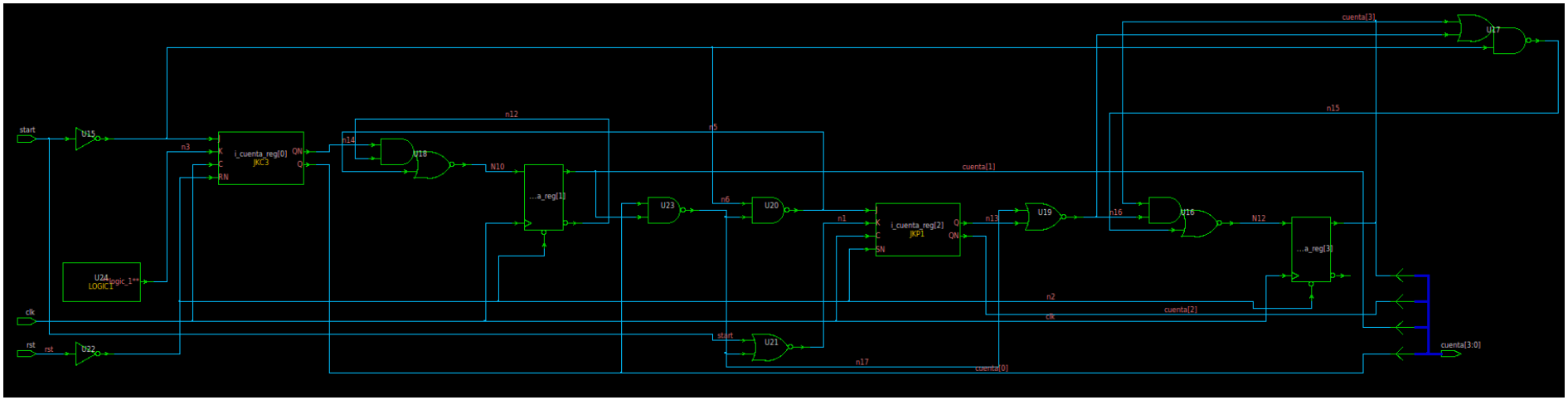
Logic Optimizations

- FSM Encoding
  - **Adjacent encoding** assigns states by the minimum logic difference in the state transition graph. Gray codes. This normally reduces the amount of logic needed to decode each state. An FSM with *n* states requires at least log 2 *n* bits.
  - **One-hot encoding** sets one bit in the state register for each state. An FSM with *n* states requires n bits. Simplifies the logic and also the interconnect between the logic. One-hot encoding often results in smaller and faster FSMs.
  - **Random encoding** assigns a random code for each state.
  - **User-specified encoding** keeps the explicit state assignment from the HDL.
  - **Moore encoding** is useful for FSMs that require fast outputs. A Moore state machine has outputs that depend only on the current state (Mealy state machine outputs depend on the current state and the inputs).

Gate-level Optimizations

- Mapping. Maps from a technology-independent netlist to cells in the target library.
- Delay Optimization. Fixes timing violations.
- Design Rule Fixing. Buffer insertion and cell resizing to fix rules violations.
- Area Optimization.

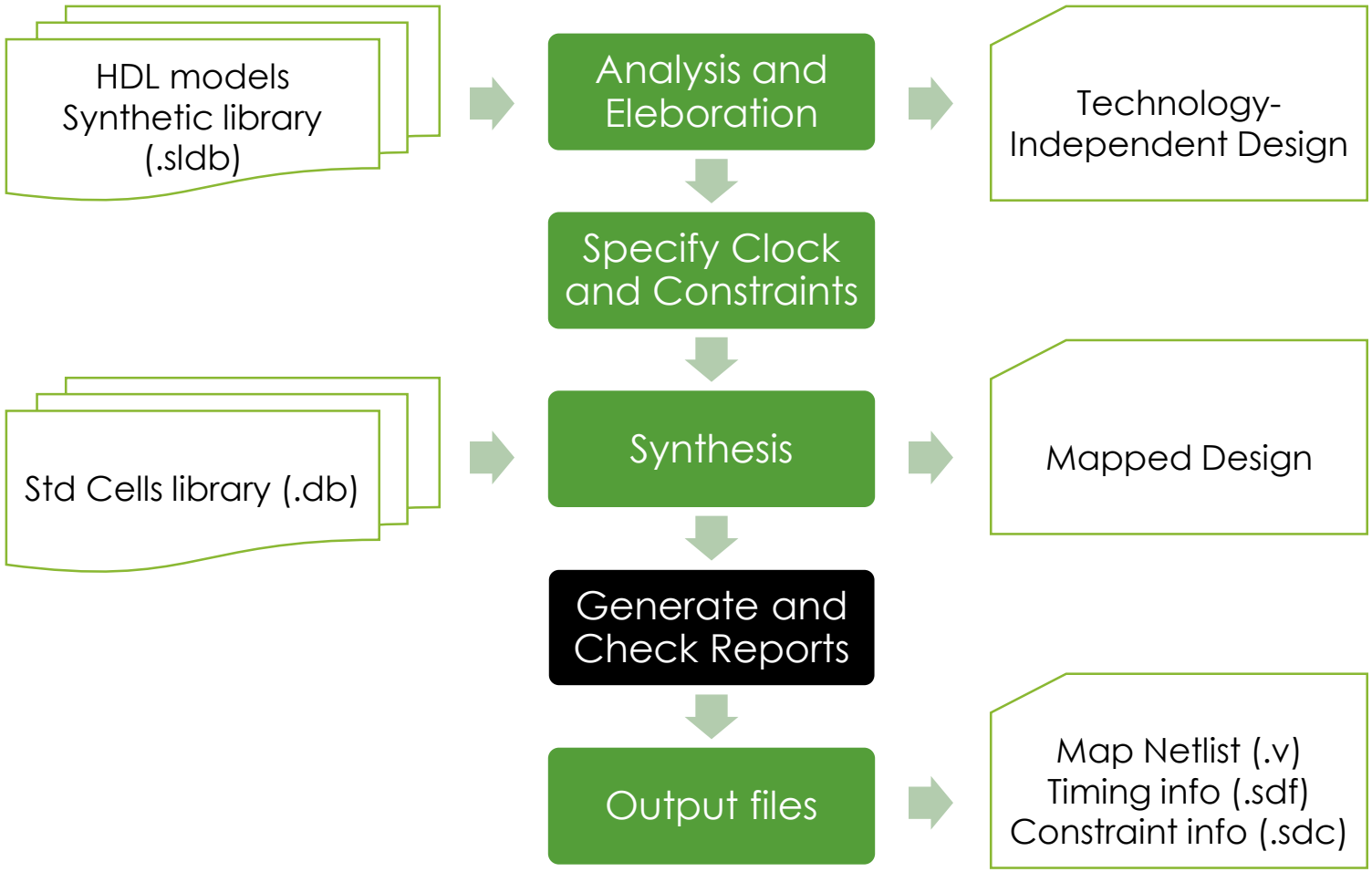
Synthesis Result



Synthesis result (.v)

```
module CONTADOR_WIDTH12 ( cuenta, start, rst, clk );
  output [11:0] cuenta;
  input start, rst, clk;
  wire   N18, N19, N20, N21, N22, N23, N24, N25, N26, N27, N28, n1, n13, n14,
         n15, n16, n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27, n28,
         n29, n30, n31, n32, n33, n34, n35;

  DFC1 \i_cuenta_reg[11]  ( .D(N28), .C(clk), .RN(n1), .Q(cuenta[11]) );
  DFC1 \i_cuenta_reg[10]  ( .D(N27), .C(clk), .RN(n1), .Q(cuenta[10]), .QN(n23) );
  DFC1 \i_cuenta_reg[9]  ( .D(N26), .C(clk), .RN(n1), .Q(cuenta[9]) );
  DFC1 \i_cuenta_reg[8]  ( .D(N25), .C(clk), .RN(n1), .Q(cuenta[8]), .QN(n22) );
  DFC1 \i_cuenta_reg[7]  ( .D(N24), .C(clk), .RN(n1), .Q(cuenta[7]) );
  DFC1 \i_cuenta_reg[6]  ( .D(N23), .C(clk), .RN(n1), .Q(cuenta[6]), .QN(n21));
  DFC1 \i_cuenta_reg[5]  ( .D(N22), .C(clk), .RN(n1), .Q(cuenta[5]) );
  DFC1 \i_cuenta_reg[4]  ( .D(N21), .C(clk), .RN(n1), .Q(cuenta[4]), .QN(n20));
  DFC1 \i_cuenta_reg[3]  ( .D(N20), .C(clk), .RN(n1), .Q(cuenta[3]) );
  DFC1 \i_cuenta_reg[2]  ( .D(N19), .C(clk), .RN(n1), .Q(cuenta[2]), .QN(n19) );
  DFC1 \i_cuenta_reg[1]  ( .D(N18), .C(clk), .RN(n1), .Q(cuenta[1]) );
  JKC1 \i_cuenta_reg[0]  ( .J(n24), .K(n18), .C(clk), .RN(n1), .Q(cuenta[0]));
  NOR21 U27 ( .A(n26), .B(n19), .Q(n27) );
  NAND22 U28 ( .A(n27), .B(cuenta[3]), .Q(n28) );
  NOR21 U29 ( .A(n28), .B(n20), .Q(n29) );
  NAND22 U30 ( .A(n29), .B(cuenta[5]), .Q(n30) );
  NOR21 U31 ( .A(n30), .B(n21), .Q(n31) );
  NAND22 U32 ( .A(n31), .B(cuenta[7]), .Q(n32) );
  NOR21 U33 ( .A(n32), .B(n22), .Q(n33) );
  NAND22 U34 ( .A(n33), .B(cuenta[9]), .Q(n34) );
  NOR21 U35 ( .A(n34), .B(n23), .Q(n35) );
  OAI210 U36 ( .A(n29), .B(cuenta[5]), .C(n30), .Q(n13) );
  NOR20 U37 ( .A(start), .B(n13), .Q(N22) );
  OAI210 U38 ( .A(n33), .B(cuenta[9]), .C(n34), .Q(n14) );
  NOR20 U39 ( .A(start), .B(n14), .Q(N26) );
  OAI210 U40 ( .A(n27), .B(cuenta[3]), .C(n28), .Q(n15) );
  NOR20 U41 ( .A(start), .B(n15), .Q(N20) );
  OAI210 U42 ( .A(n31), .B(cuenta[7]), .C(n32), .Q(n16) );
  NOR20 U43 ( .A(start), .B(n16), .Q(N24) );
  NOR20 U44 ( .A(n35), .B(cuenta[11]), .Q(n17) );
  AOI2110 U45 ( .A(n35), .B(cuenta[11]), .C(start), .D(n17), .Q(N28) );
  LOGIC1 U46 ( .Q(n18) );
  AOI2110 U47 ( .A(n30), .B(n21), .C(start), .D(n31), .Q(N23) );
  AOI2110 U48 ( .A(n28), .B(n20), .C(start), .D(n29), .Q(N21) );
  INV2 U49 ( .A(n25), .Q(N18) );
  CLKIN3 U50 ( .A(start), .Q(n24) );
  OAI2111 U51 ( .A(cuenta[0]), .B(cuenta[1]), .C(n24), .D(n26), .Q(n25) );
  AOI2111 U52 ( .A(n26), .B(n19), .C(start), .D(n27), .Q(N19) );
  AOI2111 U53 ( .A(n32), .B(n22), .C(start), .D(n33), .Q(N25) );
  AOI2111 U54 ( .A(n34), .B(n23), .C(start), .D(n35), .Q(N27) );
  NAND22 U55 ( .A(cuenta[0]), .B(cuenta[1]), .Q(n26) );
  INV3 U56 ( .A(rst), .Q(n1) );
endmodule
```

Synthesis Flow



Area Report

```
****************************************
Report : area
Design : CONTADOR_WIDTH12
Version: G-2012.06-SP4
Date   : Thu May  9 13:38:23 2013
****************************************


Library(s) Used:


    c35_CORELIB_TYP (File:
/usr/local/ams/ams_v3.80/synopsys/c35_3.3V/c35_CORE
LIB_TYP.db)


    Number of ports:                          15
    Number of nets:                           50
    Number of cells:                          42
    Number of combinational cells:            30
    Number of sequential cells:               12
```

```
Number of macros:                          0
Number of buf/inv:                         3
Number of references:                     13

Combinational area:       1892.799995
Buf/Inv area:              109.200005
Noncombinational area:    3767.399933
Net Interconnect area:    1098.000000

Total cell area:          5660.199928
Total area:               6758.199928

***** End Of Report *****
```

Timing Report

```
****************************************
Report : timing
-path full
-delay max
-max_paths 1
-sort_by group
Design : CONTADOR_WIDTH12
Version: G-2012.06-SP4
Date   : Thu May  9 14:08:02 2013
****************************************

Operating Conditions: TYPICAL   Library: c35_CORELIB_TYP
Wire Load Model Mode: enclosed

  Startpoint: i_count_reg[1]
  (rising edge-triggered flip-flop clocked by clk)
  Endpoint: i_count_reg[11]
  (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  COUNTER_WIDTH12    10k                   c35_CORELIB_TYP

  Point                                    Incr       Path
  ------------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  i_count_reg[1]/C (DFC1)                  0.00       0.00 r
  i_count_reg[1]/Q (DFC1)                  0.80       0.80 f
  U55/Q (NAND22)                           0.29       1.10 r
(…)
  U35/Q (NOR21)                            0.30       3.70 f
  U44/Q (NOR20)                            0.30       4.00 r
  U45/Q (AOI2110)                          0.24       4.24 f
  i_count_reg[11]/D (DFC1)                 0.00       4.24 f
  data arrival time                                   4.24

  clock clk (rise edge)                   10.00      10.00
  clock network delay (ideal)              0.00      10.00
  i_count_reg[11]/C (DFC1)                 0.00      10.00 r
  library setup time                       0.00      10.00
  data required time                                 10.00
  ------------------------------------------------------------
  data required time                                 10.00
  data arrival time                                  -4.24
  ------------------------------------------------------------
  slack (MET)                                         5.76


  ***** End Of Report *****
```

Power Report

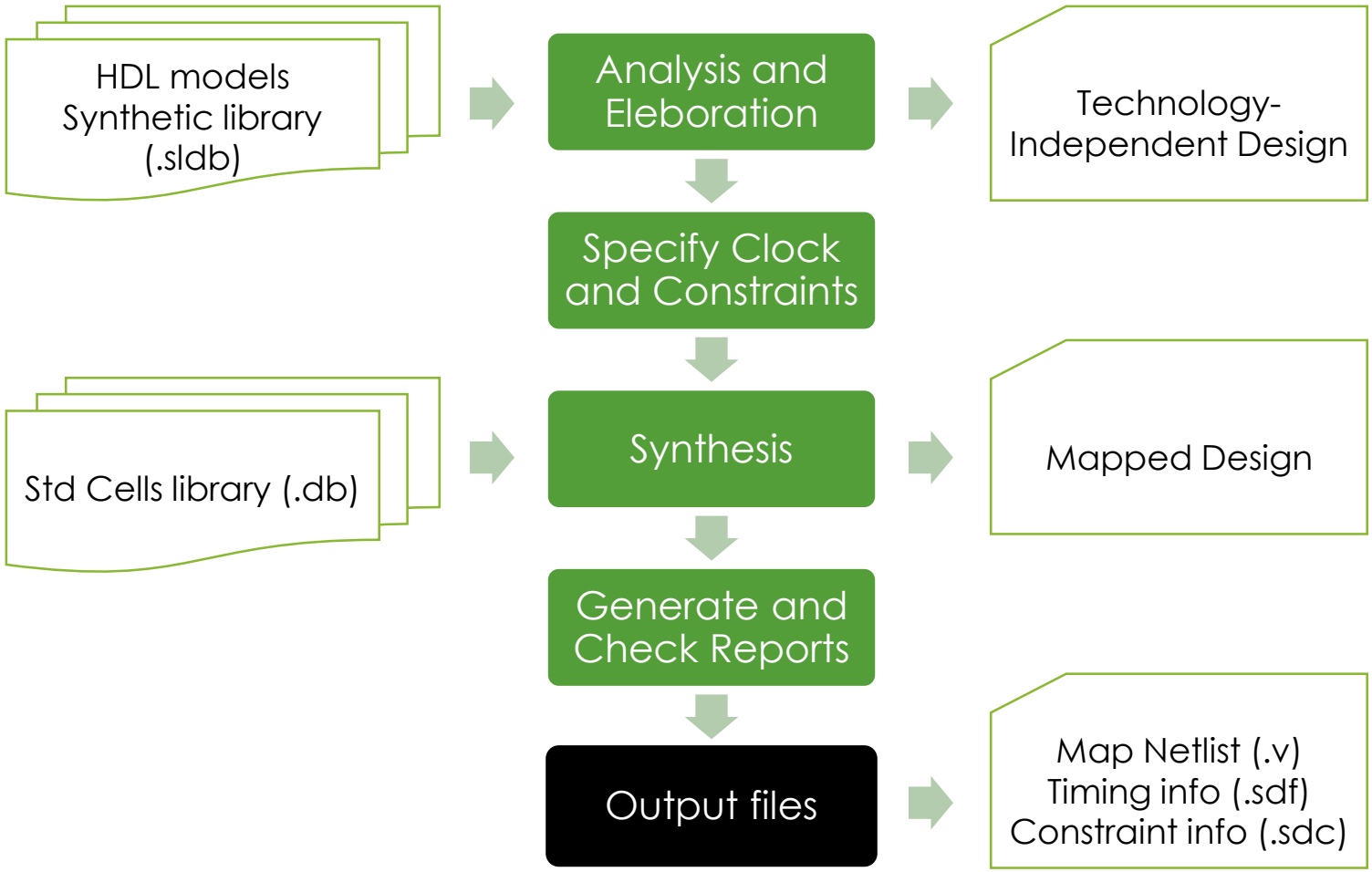
```
****************************************
Report : power
-analysis_effort low
Design : CONTADOR_WIDTH12
Version: G-2012.06-SP4
Date   : Fri May 10 16:39:55 2013
****************************************

 Library(s) Used:

 c35_CORELIB_TYP (File:
/usr/local/ams/ams_v3.80/synopsys/c35_3.3V/c35_CORELIB_TYP.db)


 Operating Conditions: TYPICAL   Library: c35_CORELIB_TYP
 Wire Load Model Mode: enclosed

 Design        Wire Load Model            Library
 ------------------------------------------------
 CONTADOR_WIDTH12        10k               c35_CORELIB_TYP


 Global Operating Voltage = 3.3
 Power-specific unit information :
 Voltage Units = 1V
 Capacitance Units = 1.000000pf
 Time Units = 1ns
 Dynamic Power Units = 1mW    (derived from V,C,T units)
 Leakage Power Units = 1pW
```

```
Cell Internal Power  = 550.2869 uW   (84%)
Net Switching Power  = 102.8524 uW   (16%)
---------
Total Dynamic Power    = 653.1393 uW  (100%)

Cell Leakage Power     =  40.0220 pW
```

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)
clock_network	0.0000	0.0000	0.0000	0.0000	(0.00%)
register	0.5073	4.7922e-02	29.3524	0.5552	(85.01%)
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)
combinational	4.3000e-02	5.4931e-02	10.6696	9.7931e-02	(14.99%)
Total	0.5503 mW	0.1029 mW	40.0220 pW	0.6531 mW	

```
 ***** End Of Report *****
```

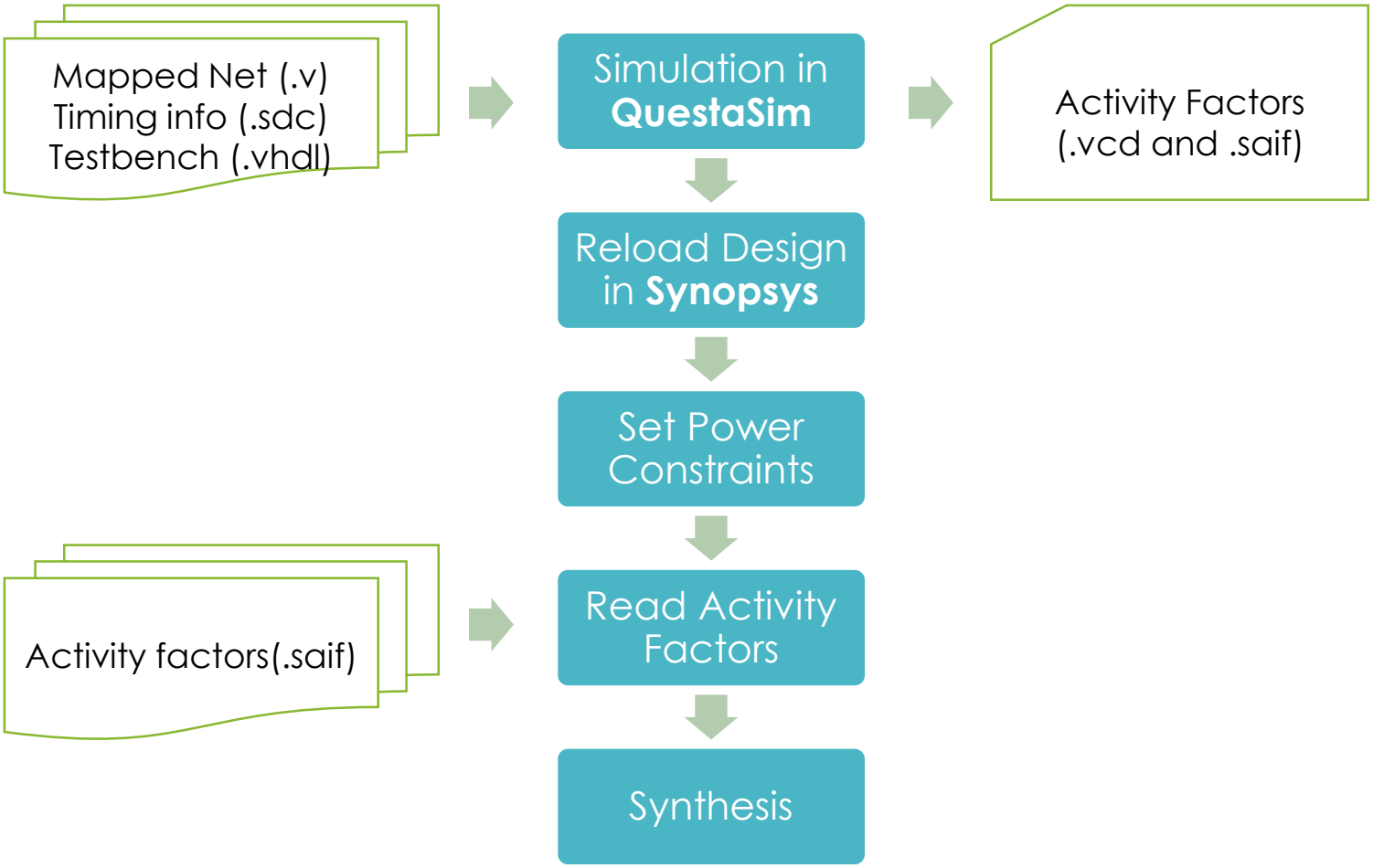
Synthesis Flow



Output Files

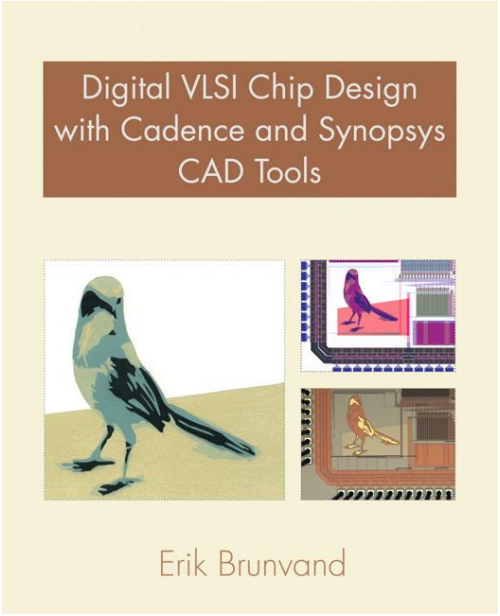
- Map Netlist (.v)
- Timing Information (.sdf). Standard delay format. Includes delays, timing checks, timing constraints, timing environment and technology parameters.
- Constraints Information (.sdc). Synopsys design constraints. Stores constraints for synthesis, clocking, timing, power, test and environmental and operating conditions.

Power Optimization Flow



Bibliografía

- Notas prácticas de la asignatura.
- Manuales de las herramientas.
- *Digital VLSI Chip Design with Cadence and Synopsys CAD Tools*. Erik Brunvand. Addison-Wesley. 2010.
- *Application-Specific Integrated Circuits.* Michael John Sebastian Smith. Addison-Wesley. 1997.



Slides Credits

- Digital IC-Proyect and Verification. ASIC Synthesis. Deepak Dasalukunte & Joachim Rodriges. Lund University.
- Design Compiler Tutorial. http://www.tkt.cs.tut.fi/tools/public/tutorials/synopsys/design_compiler/gsdc.html Tampere University of Technology.
- *Application-Specific Integrated Circuits.* Michael John Sebastian Smith. Addison-Wesley. 1997.