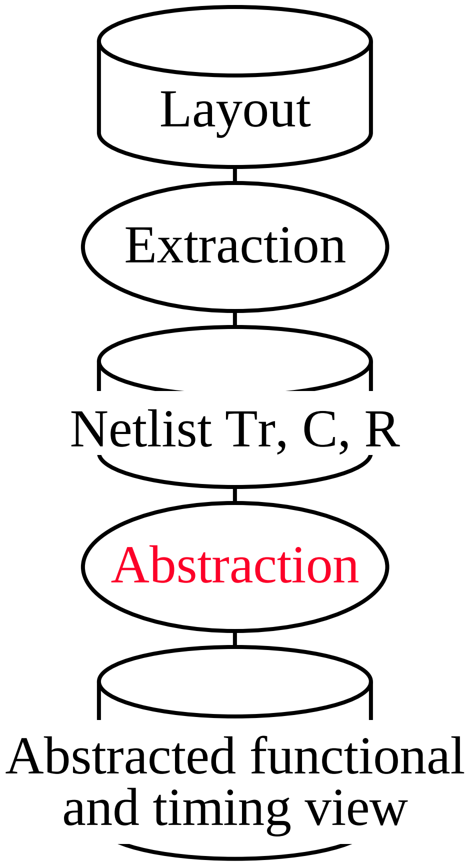
Functional and Timing Abstraction

Pirouz Bazargan Sabet

Patricia Renault

Dominique Le Dû

Introduction



The input netlist contains :

- Transistors
- Capacitors
- Resistors

Functional Abstraction

Formal Method

Build a gate (list of branches)

If not dual : Color simulation

Remove non functional branches

Analyze conflict and tri-state condition

If loop : Build a list of conflicts

Analyze transient conflicts

If dual : Loop analysis (memory ?)

System of Boolean equations
Signal correlation

Functional Abstraction

Analyzing transient conflicts

F_{cfl} is the expression of a conflict on gate x

$$F_{cfl} = x \cdot F_{x} + \overline{x} \cdot F_{\overline{x}}$$

$$F_{cfl} = F_{x} F_{\overline{x}} + x \cdot \frac{\partial F}{\partial x^{+}} + \overline{x} \cdot \frac{\partial F}{\partial x^{-}}$$

Permanent conflict

Transient conflict $\longrightarrow 0$

Transient conflict $\longrightarrow 1$

Functional Abstraction

Formal Method : Identifying a memory point

F is the Boolean expression of the gate *x*

$$F = x \cdot F_x + \bar{x} \cdot F_{\bar{x}}$$

Memory point

$$\boxed{\begin{array}{l} \dfrac{\partial F}{\partial x^+} = F_x \, \overline{F_{\bar{x}}} \neq 0 \\[2ex] \dfrac{\partial F}{\partial x^-} = \overline{F_x} \, F_{\bar{x}} = 0 \end{array}}$$

$\overline{\dfrac{\partial F}{\partial x^+}}$ Defines the write condition

$F_x . F_{\bar{x}} / \overline{\dfrac{\partial F}{\partial x^+}}$ Defines the written data

SpSmall

SpSmall_1x2 : 2s

SpSmall_2x2 : 5s

SpSmall_3x2 : 10s

Functional Abstraction

```
mygal <option> <desc>

options :
-c    <file>  : use the configuration file <file>. Default value is `config.cnf`
-d    <char>  : use <char> as index delimiter for vectors. Default value is '('
-h            : print this help
-l    <file>  : save a log file in <file>. This file can be used as input parameters
              : for a further run. By default the log is printed on stdout
-o    <file>  : save the abstracted description in <file>
-p            : load the user's parameters from the description that has the
              : same name has the input netlist
-P    <name>  : load the user's parameters from the descrpition <name>
-vdd  <name>  : define the signal named <name> as vdd. Default value is 'vdd'
-vss  <name>  : define the signal named <name> as vss. Default value is 'vss'
```

Functional Abstraction

```
configuration root is
inputs
 (
 sp_1x2   $<.spi netlist    format = spice                  ;
 sp_2x2   $<.spi netlist    format = spice                  ;
 sp_3x2   $<.spi netlist    format = spice                  ;

 nlvtlp    $<.vbe extern   format = vhdl  leaf = ntransistor;
 nsvtlp    $<.vbe extern   format = vhdl  leaf = ntransistor;

 plvtlp    $<.vbe extern   format = vhdl  leaf = ptransistor;
 psvtlp    $<.vbe extern   format = vhdl  leaf = ptransistor;
 );
end;
```

Functional Abstraction

```
-- log file : functional abstrcation
-- log file : sp_1x2
-- log file : Mon Jun  8 12:28:01 2009

-- conflicts resolved by boolean analysis :

19/9/pass_h  <= '0' when ((row<0> and (19/9/pass_l and (not b_<1>))) = '1');
18/e/net27   <= '1' when ((not b<0> and (18/e/bit_latched_h)) = '1');
16/13/a1     <= '0' when ((not 16/13/a1inv and (dec<0>)) = '1');

-- conflicts resolved by correlation analysis :

assert ((row<0> and (19/9/pass_l and (not 18/f/clk_local_h))) = '0');
assert ((19/8/pass_l and (row<0> and (not 18/e/clk_local_h))) = '0');
```

Functional Abstraction

```
entity sp_3x2 is                                     -- sp_3x2
port
(
  signal a    : in  bit_vector (1 downto 0) ;-- a
  signal ck   : in  bit                     ;-- ck
  signal csn  : in  bit                     ;-- csn
  signal d    : in  bit_vector (1 downto 0) ;-- d
  signal oen  : in  bit                     ;-- oen
  signal q    : out bit_vector (1 downto 0) ;-- q
  signal wen  : in  bit                      -- wen
);
end;
```

```
v_16_13_a1 <= not dec (0);

process (ck, v_17_12_10_ext_cs_h)
begin
  if (not ck) = '1' then
    v_17_12_10_ext_cs_n <= not v_17_12_10_ext_cs_h;
  end if;
end process;

process (v_18_e_i47_out_p_drive_l, v_18_e_i47_out_n_drive_h)
begin
  if (not v_18_e_i47_out_p_drive_l) = '1' then
    q (0) <= '1';
  end if;
  if (v_18_e_i47_out_n_drive_h) = '1' then
    q (0) <= '0';
  end if;
end process;

end;
```