SEC 130nm Library Usage Guide

For Logic Synthesis Flow

Revision 0

SAMSUNG

ELECTRONICS

Important Notice

L13 Enhancement Library Usage Guide For Logic Synthesis Flow
Revision 0

REVISION HISTORY

Rev. No.	Summary of Revision	Department	Written by	Date
000	The First Edition	Foundation IP CAE Center	Sung Wee CHO Eung Chul JUN	2004. 9.01

Table of Contents

OVERVIEW

For designers to use logic synthesizer with SEC synthesis technology libraries to synthesize modules and integrate them into ASICs, a "seamless" methodology is required. In addition, designers need a structured approach to integrate logic synthesizer and SEC Design Kit to fully achieve functional and performance goals of an ASIC and in a fast turnaround time.

This guideline describes how to use additional special library cell for the logic synthesis tools.

1. PREREQUISITE KNOWLEDGE

1.1 SYNTHESIS LIBRARY

The SEC enhanced 130nm library does not provide the derating factors. Instead, the library is characterized at slow and fast corners.

In general, a synthesis library cell consists of two main parts. The first is an actual gate description: an enumeration of all gates in the library with their input/output pins, gate functionality. The second part contains electrical characteristics (e.g. timing, capacitance). The synthesis libraries include the following data.

- Library name
- Units (timing, capacitance, resistance, power)
- Operating conditions (P, V, T, tree type)
- Default values (maximum transition time, fanout load, input capacitance)
- Wire load model
- Cell model (timing, power, area, pin capacitance, allowable max transition)

1.2 LOGIC SYNTHESIS

Logic synthesis is the basic step that transforms the HDL representation of a design into technology specific logic circuits. As the synthesis tool breaks down high-level HDL statements into more primitive functions, it searches library to find a match between the functions required and those provided in the library.

When a match is found, the synthesis tool copies the function into the design (instantiates the circuit) and gives it a unique name (cell instance name). This process continues until all statements are broken down and mapped (synthesized) to logic circuits. There are potentially hundreds, or even thousands, of different combinations of logic circuit that can implement the same logical function.

The combination chosen by a synthesis tool is determined by the synthesis constraints provided by the designer. These constraints define the design's performance, power and area targets. A design driven primarily by performance criteria may use larger, faster circuits than one driven to minimize area or power consumption.

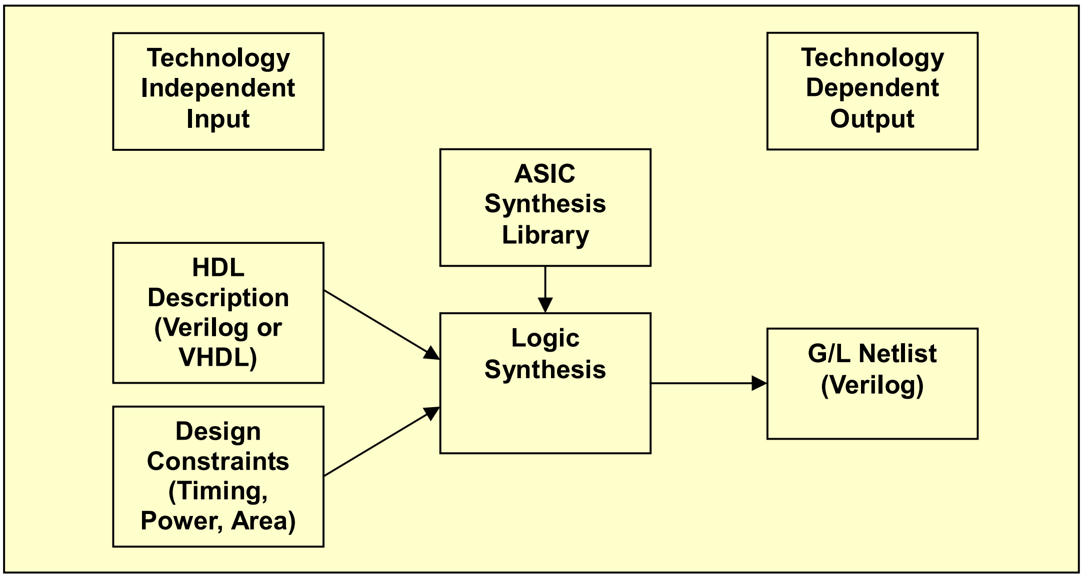

Figure 1. Logic Synthesis Process

2. SEC 130NM ENHANCEMENT LIBRARY COMPOSITION

2.1 COMBINATORIAL LOGIC

SEC 130nm provides various combinatorial logic cells. Every cell has at least five different drive strengths. Especially buffers, inverters, tri-state buffers and tri-state inverters have 11 different drives (DH, D1, D2, D3, D4, D6, D8, D12, D16, D20 and D24).
The combinatorial logic function includes the following:

- **Inverters and Buffers**
- **Tri-state Inverters and Buffers**
- **AND and NAND (1~4 inputs) gates**
- **OR and NOR (1~4 inputs) gates**
- **XOR (2~3 inputs) gates**
- **XNOR (2~3 inputs) gates**
- **MUX (2~4 inputs) gates**
- **Full-Adders and Half-Adders**
- **And-Or and Or-And combination gates**

2.2 SEQUENTIAL CELLS

The sequential cells include latch cells, flip-flop cells and scan-able flip-flop cells.

2.2.1 Latch Cells

SEC 130nm library provides 10 types of latch cells:

1. D Latch with active High. This cell also has output Q only version.
2. D Latch with active High with Reset. This cell also has output Q only version.
3. D Latch with active High with Set.
4. D Latch with active High with Reset and Set.
5. D Latch with active Low. This cell also has output Q only version.
6. D Latch with active Low with Reset. This cell also has output Q only version.

All the latch cells have four different drive strengths (DH, D1, D2, D4).

2.2.2 Flip-Flop Cells

SEC 130nm library provides 33 types of Flip-Flop cells:

1. Positive Edge triggered D Flip-Flop.
2. Positive Edge triggered D Flip-Flop with Reset.
3. Positive Edge triggered D Flip-Flop with Set.
4. Positive Edge triggered D Flip-Flop with Rest and Set.
5. Negative Edge triggered D Flip-Flop.
6. Negative Edge triggered D Flip-Flop with Reset.
7. Negative Edge triggered D Flip-Flop with Set.
8. Negative Edge triggered D Flip-Flop with Rest and Set.
9. Positive Edge triggered D Flip-Flop with Synchronous Reset.
10. Positive Edge triggered D Flip-Flop with Synchronous Set.
11. JK Flip-Flop with Reset.
12. JK Flip-Flop with Reset and Set.
13. Toggle Flip-Flop with Reset.

Each positive edged flip-flop cell has Q / QN and Q only output configuration. All flip-flop cells have four different drive strengths (DH, D1, D2, D4).
Each flip-flop cell, listed in the “Flip-Flop Cells” section has a pair of cells which has a scan controlled input

2.2.3 Newly Added Flip-Flop Cells

SEC 130nm enhancement library has newly added 10 types of flip-flop cells. Newly added flip-flop’s functions are as follows;

1) D FLIP-FLOP WITH SYNCHRONOUS ENABLE

FD1E / FD1EQ

-Function: Positive-edge triggered fd1 with synchronous active-high enable E with output Q / QN and Q only.

-Truth Table

E	D	CK	Q(n+1)	QN(n+1)
0	x	x	Q(n)	QN(n)
1	0	↑	0	1
1	1	↑	1	0
1	x	↓	Q(n)	QN(n)

FD1ES / FD1ESQ

- Function: Positive-edge triggered fd1 with scan input(TI), active-high scan enable(TE) and synchronous active-high enable E with output Q / QN and Q only.
- Truth Table

D	E	TI	TE	CK	Q(n+1)	QN(n+1)
x	x	1	1	↑	1	0
x	x	0	1	↑	0	1
x	0	x	0	↑	Q(n)	QN(n)
0	1	x	0	↑	0	1
1	1	x	0	↑	1	0
x	1	x	x	↓	Q(n)	QN(n)

HDL CODING DESCRIPTION FOR LOGICAL MAPPING OF D F/F WITH SYNC ENABLE

```
// Verilog Example of D Flip-Flop with synchronous enable

      module dff_s_load (sload , sdata, clk , q) ;
      input sload, sdata , clk;
      output q ;
      reg q ;
          always @ (posedge clk)
           begin
           if ( sload )
                       q = sdata;
           end
      endmodule
```

```
-- VHDL Example of D Flip-Flop synchronous enable

      library IEEE;
      use IEEE.std_logic_1164.all;
      entity dff_s_load is
      port( sload, sdata , clk : in std_logic;
      q : out std_logic );
      end dff_ s_load;
      architecture rtl of dff_ s_load is
          begin
          infer: process (clk)
          begin
           if (clk'event and clk = ‘1') then
              if (sload = ‘1') then
              q <= sdata;
              end if;
          end if;
      end process infer;
      end rtl;
```

2) D FLIP-FLOP WITH TWO INPUT MUX

FD1MQ

- Function : Positive-edge triggered fd1q with muxed two inputs with output Q only.
- Truth Table

S	D1	D0	CK	Q(n+1)
1	0	x	↑	0
1	1	x	↑	1
0	x	0	↑	0
0	x	1	↑	1
x	x	x	↓	Q(n)

FD1MSQ

- Function : Positive-edge triggered fd1q with scan input(TI), active-high scan enable(TE) and muxed two inputs with output Q only.
- Truth Table

S	D1	D0	TI	TE	CK	Q(n+1)
x	x	x	0	1	↑	0
x	x	x	1	1	↑	1
1	0	x	x	0	↑	0
1	1	x	x	0	↑	1
0	x	0	x	0	↑	0
0	x	1	x	0	↑	1
x	x	x	x	x	↓	Q(n)

HDL CODING DESCRIPTION FOR LOGICAL MAPPING OF D F/F WITH TWO INPUT MUX

```
// Verilog Example of D Flip-Flop with two input mux

    module dff_mux2 (clk , sel, D1, D2, q) ;
    input clk, sel, D1, D2;
    output q ;
    reg q ;
        always @ (posedge clk)
        if (sel)
        q= D1;
        else
        q = D0;
    endmodule
```

```
-- VHDL Example of D Flip-Flop with two input mux

      library IEEE;
      use IEEE.std_logic_1164.all;
      entity dff_mux2 is
          port(clk, sel, D1, D0: in std_logic;
          q : out std_logic );
      end dff_mux2;
      architecture rtl of dff_mux2 is
          begin
          infer: process (clk)
          begin
           if (clk'event and clk = ‘1') then
            if (sel = ‘1') then
              q <= D1;
           else
           q <= D0;
              end if;
          end if;
      end process infer;
     end rtl;
```

3) D FLIP-FLOP WITH SYNCHRONOUS RESET/ENABLE

FDS2E /FDS2EQ

- Function: Positive-edge triggered fd2 with synchronous active-high enable(E) and synchronous active-low reset(CRN) with output Q / QN and Q only.
- Truth Table

CRN	E	D	CK	Q(n+1)	QN(n+1)
0	x	x	↑	0	1
x	x	x	↓	Q(n)	QN(n)
1	0	x	↑	Q(n)	QN(n)
1	1	0	↑	0	1
1	1	1	↑	1	0

FDS2ES / FDS2ESQ

- Function: Positive-edge triggered ff with scan input(TI), active-high scan enable(TE), synchronous active-high enable(E) and synchronous active-low reset(CRN). Scan enable TE dominates CRN and E with output Q / QN and Q only.

- Truth Table

CRN	D	E	TI	TE	CK	Q(n+1)	QN(n+1)
x	x	x	0	1	↑	0	1
x	x	x	1	1	↑	1	0
1	x	0	x	0	↑	Q(n)	QN(n)
0	x	x	x	0	↑	0	1
1	1	1	x	0	↑	1	0
1	0	1	x	0	↑	0	1
x	x	x	x	x	↓	Q(n)	QN(n)

HDL CODING DESCRIPTION FOR LOGICAL MAPPING OF D F/F WITH SYNC RESET/ENABLE

```
// Verilog Example of D Flip-Flop with synchronous reset/load

    module dff_s_rst_ld (sload , sdata, clk , sreset, q) ;
    input sreset , sload, sdata , clk;
    output q ;
    reg q ;
        always @ (posedge clk)
         begin
         if ( !sreset )
            q = 1'b0;
         else if (sload)
         q = sdata;
        end
    endmodule
```

```
-- VHDL Example of D Flip-Flop synchronous reset/load

    library IEEE;
    use IEEE.std_logic_1164.all;
    entity dff_s_rst_ld is
    port( sload, sreset, sdata , clk : in std_logic;
    q : out std_logic );
    end dff_s_rst_ld;
    architecture rtl of dff_s_rst_ld is
        begin
        infer: process (clk)
            begin
             if (clk'event and clk = '1') then
                if (sreset = '0') then
                    q <= '0';
                elsif (sload = '1') then
                    q <= sdata;
                end if;
            end if;
        end process infer;
    end rtl;
```

4) SCAN-ABLE FLIP-FLOP CELLS

Each flip-flop cell, listed in the “Flip-Flop Cells” section has a version that can be scanned.

2.3 SPECIAL CELLS

SEC 130nm library provides the following special cells:

- **Gated Clock Cells**
- **Balanced Clock Cells**
- **Hold Buffer Cells for Fixing Hold Violation easily**
- **Antenna fix Cell**
- **Tie-High and Tie-Low Cells**

2.3.1 Clock Gating Cells

The clock gating cell provides a power-efficient implementation of register banks that are disabled during some clock cycles. The SEC 130nm library provides two types of clock gating cells, CGLN and CGLP. You can use the HDL Compiler tool and Synopsys Power Compiler to perform clock gating at the register transfer level (RTL). Refer to the *Synopsys Power Compiler Reference Manual* for details.

- **CREATION OF CLOCK GATING CELL**

In the dc_shell environment, HDL Compiler automatically gates clocks in RTL design when designer elaborates the design with the –gate_clock option or by insert_clock_gating after elaboration.
The clock gating cells have “clock_gating_integrated_cell” attribute. This attribute identifies cells that are used for clock gating. Design Compiler uses this attribute when it compiles a design containing gated clocks. Therefore, although these clock gating cells have a dont_use attribute, the Power Compiler automatically removes dont_use attribute, and mapped these cells for power optimization.
If your design is at post-layout stage, the cell attributes of dont_use and dont_touch should be removed for design optimization.

CLOCK GATING INSERTION EXAMPLE IN DC_SHELL ENVIRONMENTS

```
dc_shell-t> analyze –f verilog PROJ.v
dc_shell-t> set_clock_gating_style –sequential_cell latch \
                -positive_edge_logic {integrated} \
                -control_point before –control_signal scan_enable \
                -max_fanout 32
dc_shell-t> elaborate –update –gate_clock PROJ
```

2.3.2 Balanced Clock Cells

SEC 130nm Library provides 6 balanced cells for clock network, named clk*.   These cells have various driver sizes. Especially clkiv and clknid have 13 strengths and have equal rise and fall time to keep the duty cycle in the clock tree.

Six balanced cells are as follows

1. Inverter : clkiv with 13 strengths
2. Buffer : clknid with 13 strengths
3. 2-input NAND : clknd2 with 6 strengths
4. 2-input AND : clkad2 with 6 strengths
5. 2-input XOR : clkxo2 with 6 strengths
6. 2->1 MUX : clkmx2 with 6 strengths

You should use these cells to make balanced rise and fall time on the clock network at clock tree synthesis step. You can use these cells in CTS step, and clkiv and clknid cell have dont_use attributes in synthesis library.
If normal inverters and buffers are used in the clock tree, the design cannot maintain the duty cycle because these cells have unbalanced rise and fall time.

2.3.3 Antenna Fix Cell

SEC 130nm cell library provides antenna fix cells which are named diode_cell. Usually, two approaches have been used to solve antenna violations on a chip. One is a metal change; the other is diode addition. Most Place and Route (P&R) tools support antenna-fix features and SEC recommends metal change on a chip level.

But on a Hard-Core design, such as ARM, SEC recommends using the both metal change and diode insertion. In other words, a designer should fix all the antenna violations using the metal change first, and add the diode_cell on the primary input and output pins. This method guarantees that there will be no antenna violation at the chip level implementation.

This diode_cell is supposed to be used only at the layout stage, so this cell should have dont_use and dont_touch attributes at the logic synthesis stage.

2.3.4 Hold Buffer Cells

SEC 130nm cell library has the following 2 types of special buffers to help hold violation fix.

1 Holdbuf {a,b,c,d} : Hold buffer cells for ECO with 3 strengths.
2 Holdbuf {1,2,3,4} : Area optimized hold buffer cells with 3 strengths.

The ECO type hold buffers, named holdbuf{a,b,c,d} have the same footprint except physical layout shape, so you can easily adjust hold margin with changing the lower delay hold buffer to the higher delay hold buffer in the post optimization step without any routing changes. Area optimized hold buffers, named holdbuf{1,2,3,4} are for optimized area, so holdbuf1 is smaller than holdbufa but each cell has similar delay.

@ Best Condition	std150e			std150hse			stdh150e		
	tphl [ns]	tplh [ns]	avg. [ns]	tphl [ns]	tplh [ns]	avg. [ns]	tphl [ns]	tplh [ns]	avg. [ns]
holdbuf1d1	0.13861	0.12245	0.13053	0.13654	0.12089	0.12872	0.07530	0.07140	0.07335
holdbuf2d1	0.20130	0.18861	0.19496	0.19990	0.18770	0.19380	0.11510	0.11410	0.11460
holdbuf3d1	0.27324	0.27937	0.27631	0.27190	0.27850	0.27520	0.15480	0.16100	0.15790
holdbuf4d1	0.34862	0.38985	0.36924	0.34730	0.38870	0.36800	0.20180	0.21940	0.21060
holdbufad1	0.14105	0.12459	0.13282	0.14050	0.12420	0.13235	0.07570	0.07180	0.07375
holdbufbd1	0.20524	0.19229	0.19877	0.20440	0.19160	0.19800	0.11530	0.11420	0.11475
holdbufcd1	0.27398	0.28097	0.27748	0.27310	0.28010	0.27660	0.15540	0.16170	0.15855
holdbufdd1	0.34857	0.38981	0.36919	0.34730	0.38870	0.36800	0.20180	0.21940	0.21060
nid1	0.09079	0.09374	0.09227	0.09420	0.08060	0.08740	0.06020	0.05070	0.05545
input slope [ns]	0.088 [ns]								
output load cap [pF]	0.010 [pF]								

2.3.5 Tie-High and Tie-Low Cells

SEC 130nm cell library provides tie-high cells to tie any inputs to logic level of high and tie-low cells to tie any inputs to a logic level of low. The tie-high/tie-low cells have dont_use, and dont_touch attributes in synthesis library. If above tie-high/tie low cells are used at synthesis stage, there would be many problems because too many tie-high/tie-low cells are replaced at logic0 and logic1 input ports. Please do not change default library dont_use attributes. If your design has an ESD weakness point, choose the tie gate and carefully connect it to a tie cell.

2.3.6 Half Drive Cells

SEC 130nm cell library provides half-drive cells, named *dh. These cells have dont_use attributes in synthesis library. If half drive cells are used at initial synthesis stage, it would be down the performance.
If your design is at post-layout stage, the cell attributes of dont_use could be removed as occasion demands for design optimization.

3. DONT_USE AND DONT_TOUCH CELL

To get a good synthesis result, a designer MUST aware of dont_use and dont_touch attributes. Please ensure that dont_use and dont_touch list of target synthesis library.

3.1 DONT_USE ATTRIBUTE

The dont_use attribute with a true value indicates that a cell should not be added to a design during optimization. If you do not want a cell added to a design during optimization, give this attribute a true value.

In addition to defining dont_use in a cell group or a model group, you can also apply the dont_use attribute to a cell, by using the set_dont_use command at dc_shell.

3.2 DONT_TOUCH ATTRIBUTE

The dont_touch attribute with a true value indicates that all instances of the cell must remain in the design during optimization.

In addition to defining dont_touch in a cell group or a model group, you can also apply the dont_touch attribute to a cell, by using the set_dont_touch command at dc_shell.

3.3 DONT_USE, DONT_TOUCH ATTRIBUTE CONTROL

To get a list of dont_use, dont_touch cell, use
dc_shell-t> report_lib std150e
or
dc_shell-t> filter_collection [get_lib_cells std150e/*] “dont_use == true”
dc_shell-t> filter_collection [get_lib_cells std150e/*] “dont_touch == true”
or
dc_shell-t> get_attribute [get_lib_cells std150e/holddbuf1] dont_use

To set dont_use, dont_touch attributes on library cell, use
dc_shell-t> set_dont_use [get_lib_cells std150e/holdbuf*]
dc_shell-t> set_dont_touch [get_lib_cells std150e/holdbuf*]

To remove dont_use, dont_touch attributes on library cell, use
dc_shell-t> remove_attribute [get_lib_cells std150e/holdbuf*] dont_use
dc_shell-t> remove_attribute [get_lib_cells std150e/holdbuf*] dont_touch

To get better results, some cells already have dont_use and dont_touch attributes implicitly. The list of cells which has dont_use or dont_touch attributes is as follows;

Default dont_use and dont_touch list according to design stage.

SYNTHESIS					
DONT_USE_List					DONT_TOUCH_List
acdly	clknid20	fd3sqd2	fj2sd1	nd3dh	acdly
ad2bdh	clknid24	fd3sqd4	fj2sd2	nd4dh	busholder
ad2dh	clknid2	fd3sqdh	fj2sd4	nidh	diode_cell
ad3dh	clknid32	fd4dh	fj2sdh	nitdh	holdbuf1d1
ad4dh	clknid3	fd4qdh	fj4dh	nr2bdh	holdbuf1d2
ao211dh	clknid40	fd4sd1	fj4sd1	nr2dh	holdbuf1d4
ao21dh	clknid48	fd4sd2	fj4sd2	nr3dh	holdbuf2d1
ao221dh	clknid4	fd4sd4	fj4sd4	nr4dh	holdbuf2d2
ao222adh	clknid6	fd4sdh	fj4sdh	oa211dh	holdbuf2d4
ao222dh	clknid8	fd4sqd1	ft2dh	oa21dh	holdbuf3d1
ao22adh	diode_cell	fd4sqd2	hadh	oa221dh	holdbuf3d2
ao22dh	fadh	fd4sqd4	holdbuf1d1	oa222dh	holdbuf3d4
ao31dh	fd1dh	fd4sqdh	holdbuf1d2	oa22adh	holdbuf4d1
ao32dh	fd1edh	fd5dh	holdbuf1d4	oa22dh	holdbuf4d2
ao33dh	fd1eqdh	fd5sd1	holdbuf2d1	oa31dh	holdbuf4d4
busholder	fd1esd1	fd5sd2	holdbuf2d2	oa32dh	holdbufad1
cglnd12	fd1esd2	fd5sd4	holdbuf2d4	oa33dh	holdbufad2
cglnd16	fd1esd4	fd5sdh	holdbuf3d1	or2bdh	holdbufad4
cglnd1	fd1esdh	fd6dh	holdbuf3d2	or2dh	holdbufbd1
cglnd20	fd1esqd1	fd6sd1	holdbuf3d4	or3dh	holdbufbd2
cglnd24	fd1esqd2	fd6sd2	holdbuf4d1	or4dh	holdbufbd4
cglnd2	fd1esqd4	fd6sd4	holdbuf4d2	rtc_etc	holdbufcd1
cglnd32	fd1esqdh	fd6sdh	holdbuf4d4	rtc_osc	holdbufcd2
cglnd3	fd1mqdh	fd7dh	holdbufad1	scg10dh	holdbufcd4
cglnd4	fd1msqd1	fd7sd1	holdbufad2	scg11dh	holdbufdd1
cglnd6	fd1msqd2	fd7sd2	holdbufad4	scg12오	holdbufdd2
cglnd8	fd1msqd4	fd7sd4	holdbufbd1	scg13dh	holdbufdd4
cglpd12	fd1msqdh	fd7sdh	holdbufbd2	scg14dh	rtc_etc
cglpd16	fd1qdh	fd8dh	holdbufbd4	scg15dh	rtc_osc
cglpd1	fd1sd1	fd8sd1	holdbufcd1	scg16dh	tie0
cglpd20	fd1sd2	fd8sd2	holdbufcd2	scg17dh	tie1

SYNTHESIS				
DONT_USE_List				
cglpd24	fd1sd4	fd8sd4	holdbufcd4	scg18dh
cglpd2	fd1sdh	fd8sdh	holdbufdd1	scg1dh
cglpd32	fd1sqd1	fds2dh	holdbufdd2	scg20dh
cglpd3	fd1sqd2	fds2edh	holdbufdd4	scg21dh
cglpd4	fd1sqd4	fds2eqdh	ivdh	scg22dh
cglpd6	fd1sqdh	fds2esd1	ivtdh	scg2dh
cglpd8	fd2dh	fds2esd2	ld1dh	scg3dh
clkivd12	fd2qdh	fds2esd4	ld1qdh	scg4dh
clkivd16	fd2sd1	fds2esdh	ld2dh	scg5dh
clkivd1	fd2sd2	fds2esqd1	ld2qdh	scg6dh
clkivd20	fd2sd4	fds2esqd2	ld3dh	scg7dh
clkivd24	fd2sdh	fds2esqd4	ld4dh	scg8dh
clkivd2	fd2sqd1	fds2esqdh	ld5dh	scg9dh
clkivd32	fd2sqd2	fds2sd1	ld5qdh	tie0
clkivd3	fd2sqd4	fds2sd2	ld6dh	tie1
clkivd40	fd2sqdh	fds2sd4	ld6qdh	xn2dh
clkivd48	fd3dh	fds2sdh	mx2dh	xn3dh
clkivd4	fd3qdh	fds3dh	mx2idh	xo2dh
clkivd6	fd3sd1	fds3sd1	mx3dh	xo3dh
clkivd8	fd3sd2	fds3sd2	mx4dh	
clknid12	fd3sd4	fds3sd4	nd2bdh	
clknid16	fd3sdh	fds3sdh	nd2dh	
clknid1	fd3sqd1	fj2dh	nd3bdh	

P&R					
DONT_USE_List					DONT_TOUCH_List
acdly	clknid20	fd3sqd4	fj2sd2	nd4dh	acdly
ad2bdh	clknid24	fd3sqdh	fj2sd4	nidh	busholder
ad2dh	clknid2	fd4dh	fj2sdh	nitdh	diode_cell
ad3dh	clknid32	fd4qdh	fj4dh	nr2bdh	holdbuf1d1
ad4dh	clknid3	fd4sd1	fj4sd1	nr2dh	holdbuf1d2
ao211dh	clknid40	fd4sd2	fj4sd2	nr3dh	holdbuf1d4
ao21dh	clknid48	fd4sd4	fj4sd4	nr4dh	holdbuf2d1
ao221dh	clknid4	fd4sdh	fj4sdh	oa211dh	holdbuf2d2
ao222adh	clknid6	fd4sqd1	ft2dh	oa21dh	holdbuf2d4
ao222dh	clknid8	fd4sqd2	hadh	oa221dh	holdbuf3d1
ao22adh	fadh	fd4sqd4	holdbuf1d1	oa222dh	holdbuf3d2
ao22dh	fd1dh	fd4sqdh	holdbuf1d2	oa22adh	holdbuf3d4
ao31dh	fd1edh	fd5dh	holdbuf1d4	oa22dh	holdbuf4d1
ao32dh	fd1eqdh	fd5sd1	holdbuf2d1	oa31dh	holdbuf4d2
ao33dh	fd1esd1	fd5sd2	holdbuf2d2	oa32dh	holdbuf4d4
busholder	fd1esd2	fd5sd4	holdbuf2d4	oa33dh	holdbufad1
cglnd12	fd1esd4	fd5sdh	holdbuf3d1	or2bdh	holdbufad2
cglnd16	fd1esdh	fd6dh	holdbuf3d2	or2dh	holdbufad4
cglnd1	fd1esqd1	fd6sd1	holdbuf3d4	or3dh	holdbufbd1
cglnd20	fd1esqd2	fd6sd2	holdbuf4d1	or4dh	holdbufbd2
cglnd24	fd1esqd4	fd6sd4	holdbuf4d2	rtc_etc	holdbufbd4

P&R					
DONT_USE_List					DONT_TOUCH_List
cglnd2	fd1esqdh	fd6sdh	holdbuf4d4	rtc_osc	holdbufcd1
cglnd32	fd1mqdh	fd7dh	holdbufad1	scg10dh	holdbufcd2
cglnd3	fd1msqd1	fd7sd1	holdbufad2	scg11dh	holdbufcd4
cglnd4	fd1msqd2	fd7sd2	holdbufad4	scg12오	holdbufdd1
cglnd6	fd1msqd4	fd7sd4	holdbufbd1	scg13dh	holdbufdd2
cglnd8	fd1msqdh	fd7sdh	holdbufbd2	scg14dh	holdbufdd4
cglpd12	fd1qdh	fd8dh	holdbufbd4	scg15dh	rtc_etc
cglpd16	fd1sd1	fd8sd1	holdbufcd1	scg16dh	rtc_osc
cglpd1	fd1sd2	fd8sd2	holdbufcd2	scg17dh	tie0
cglpd20	fd1sd4	fd8sd4	holdbufcd4	scg18dh	tie1
cglpd24	fd1sdh	fd8sdh	holdbufdd1	scg1dh	
cglpd2	fd1sqd1	fds2dh	holdbufdd2	scg20dh	
cglpd32	fd1sqd2	fds2edh	holdbufdd4	scg21dh	
cglpd3	fd1sqd4	fds2eqdh	ivdh	scg22dh	
cglpd4	fd1sqdh	fds2esd1	ivtdh	scg2dh	
cglpd6	fd2dh	fds2esd2	ld1dh	scg3dh	
cglpd8	fd2qdh	fds2esd4	ld1qdh	scg4dh	
clkivd12	fd2sd1	fds2esdh	ld2dh	scg5dh	
clkivd16	fd2sd2	fds2esqd1	ld2qdh	scg6dh	
clkivd1	fd2sd4	fds2esqd2	ld3dh	scg7dh	
clkivd20	fd2sdh	fds2esqd4	ld4dh	scg8dh	
clkivd24	fd2sqd1	fds2esqdh	ld5dh	scg9dh	
clkivd2	fd2sqd2	fds2sd1	ld5qdh	tie0	
clkivd32	fd2sqd4	fds2sd2	ld6dh	tie1	
clkivd3	fd2sqdh	fds2sd4	ld6qdh	xn2dh	
clkivd40	fd3dh	fds2sdh	mx2dh	xn3dh	
clkivd48	fd3qdh	fds3dh	mx2idh	xo2dh	
clkivd4	fd3sd1	fds3sd1	mx3dh	xo3dh	
clkivd6	fd3sd2	fds3sd2	mx4dh		
clkivd8	fd3sd4	fds3sd4	nd2bdh		
clknid12	fd3sdh	fds3sdh	nd2dh		
clknid16	fd3sqd1	fj2dh	nd3bdh		
clknid1	fd3sqd2	fj2sd1	nd3dh		