Cyclone II EDA and Academic Developer Functional Description

Version 1.6

By
Altera Corporation

Table of Contents:

1. Overview

This document is intended for developers working with the Cyclone II™ architecture. It is a companion to the WYSIWYG Device Primitive Guide for Cyclone II and should be used in conjunction with that document. This document provides information about the Cyclone II architecture to allow CAD tool developers to synthesize designs to the architecture which will not have fitting problems. It also enables CAD tool developers to create placements for Cyclone II which will be routable and will not violate any constraints on what constitutes a legal LAB or DSP block.

2. Logic Cell Descriptions

In previous architectures, a single LE contained combinational logic and a register. For Cyclone II devices, the combinational logic and register have been split apart into separate blocks, creating two different types of logic cells - the lcell_comb (combinational) and the lcell_ff (flip-flop/register). This is the same concept as the Stratix® II architecture. Note that the terms flip-flop and register are used interchangeably in this document.

Figure 1 shows the full functionality of the Cyclone II register logic cell. It has seven inputs of **datain, sdata, clk, ena, sload, sclr,** and **aclr**. It has one output of **regout**.

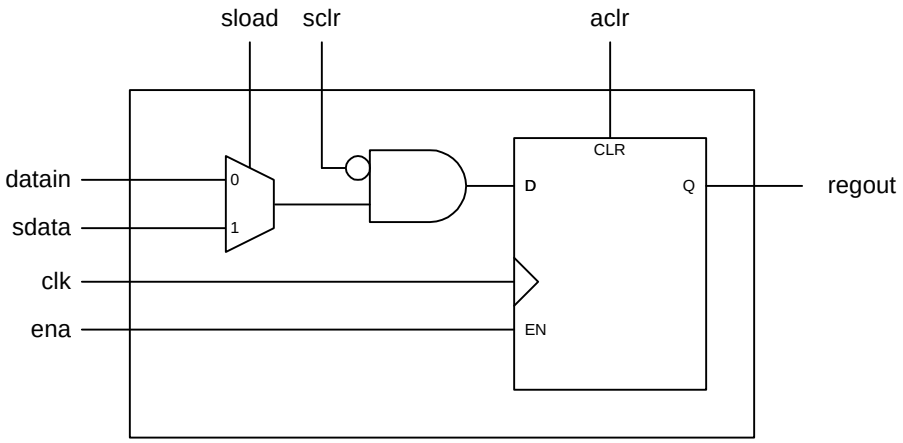

Figure 1 – Cyclone II Register Logic Cell (lcell_ff)

Figure 2 shows the full functionality of the Cyclone II combinational logic cell (now referred to as an lcell_comb). It has five inputs of **dataa, datab, datac, datad** and **cin**. It has two outputs of **cout** and **combout**.

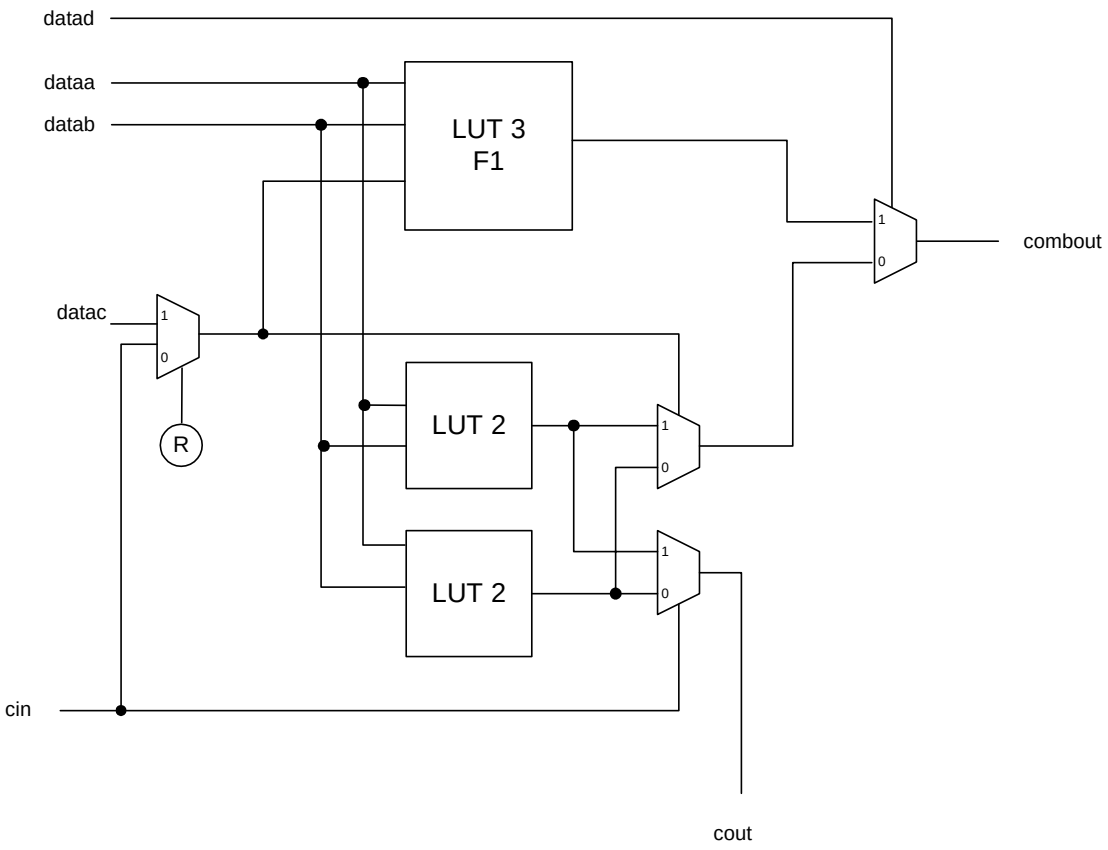

Figure 2 – Cyclone II Combinational Logic Cell (lcell_comb)

3. Coordinate System and Location Assignments

The diagram below shows the Cyclone II coordinate system. Note that the device floorplan implied by this figure is not the final floorplan and may be modified later. The coordinate system will still follow the same overall philosophy regardless of the final floorplan.

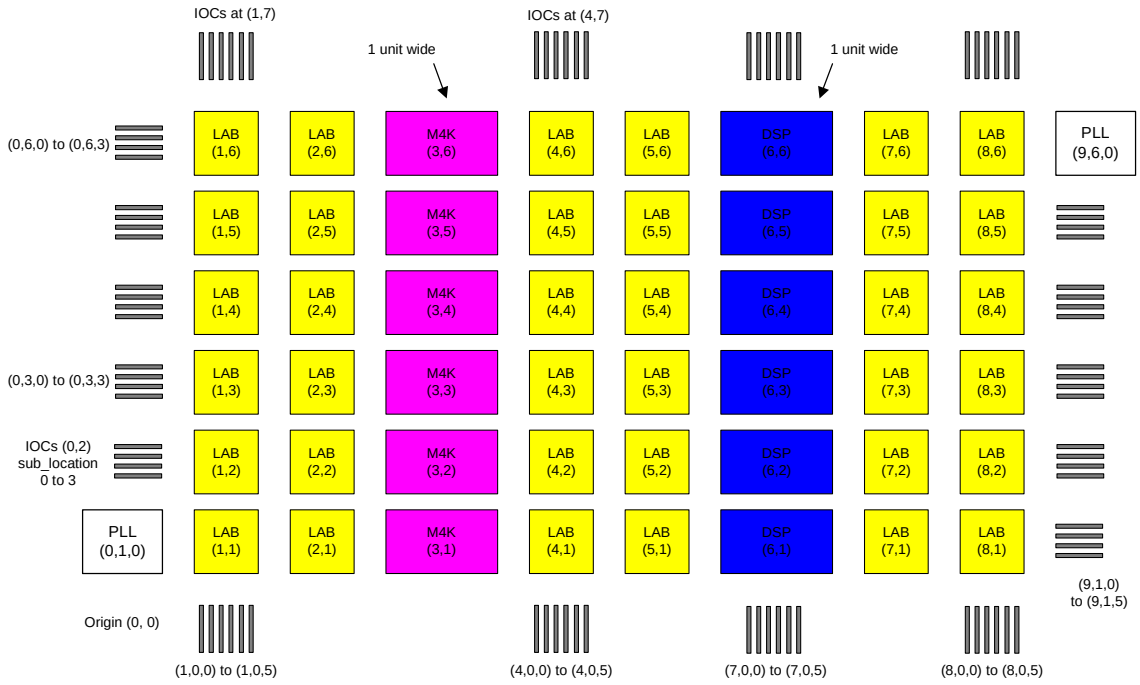

Figure 3 – Cyclone II Coordinate System

Cyclone II devices use a flat coordinate system, with the origin (0, 0) in the **lower-left** corner. The routing channels define an (x,y) grid, and every block is identified by its (x,y) location. Each row is represented by a y-value, e.g. Y6, starting at 0 for the bottom-most row. Each column is represented by an x-value, e.g. X4, starting at 0 for the left-most row. Large blocks that span multiple grid points are referred to by the coordinate of their lower-left corner.

To the fitter, most location assignments are 2D region assignments. For example, custom_region_X4_Y3_X8_Y7 constrains a cell to the region between coordinates (4, 3) and (8, 7), inclusive. Assignments to specific LABs and ESBs (e.g. LAB_X7_Y3) are simply a shorthand for 2D regions that are the same size as the block to which the assignment is being made.

The other type of location assignment you can make are assignments to individual logic cells. For example, lcell_comb_X4_Y6_N0 assigns a circuit element to the lcell_comb at x = 4, y = 6 and number = 0. Number is simply an index that is used to differentiate between cells when there are multiple cells at a certain (x, y) location. Note that generally it is a bad idea to constrain logic cells all the way down to the number, or logic cell level. Constraining cells all the way down to the logic cell level doesn't provide any additional timing predictability vs. constraining to the LAB level, and it constrains the router more, causing more fitting problems.

If a region assignment includes any portion of a large block within it, the entire large block is considered to be a legal placement location.

4. Routing Delay vs. Distance & Routing Rules

Cyclone II delays follow similar patterns to those of the Cyclone and Stratix architectures. The routing delay increases roughly linearly with Manhattan distance.

The speed of connections in Cyclone II devices, from fastest to slowest is:

- Source and destination in the same LAB

    - Connections within the same LAB are fastest since they can use either (i) direct lcell_comb to lcell_ff connections, (ii) lcell_ff to lcell_comb connections via qfbk, (iii) lcell_ff to lcell_ff connections via register cascade, or (iv) local line connections.

- Destination in the LAB immediately to the left or right of the source, since lcell_comb and lcell_ff atoms in a LAB can directly drive some of the LAB lines in these immediately adjacent LABs.
- Delay increases roughly linearly with Manhattan distance. There is a slight delay advantage (for equal Manhattan distances) when the source and destination are in the same row or column, since no "elbow" switching from vertical to horizontal or vice versa is required in this case.

5. Netlist Recommendations

Most input ports on WYSIWYG primitives have programmable inversion built into their hardware. To take advantage of this programmable inversion, *netlists should directly connect the complement of a signal to an input port if that is the circuit behavior desired*. For example, if it was desired to make a negative-edge triggered register in an lcell_ff cell, the netlist should connect **!clock** to the .clk port on the cycloneii_lcell primitive. The programmable inverter hardware in the LAB will be used to invert the clock in this case. Consider what would have happened if instead the netlist had used an lcell_comb cell to invert the clock and create a new signal, nclock, which was then connected to the .clk port of an lcell_ff cell to make a negative-edge triggered register. This netlist will result in one extra lcell_comb cell being created, and lead to a larger circuit with much worse clock skew.

```
cycloneii_lcell_ff good_cell {
      .clk(!clock),  // good way to make an inverted clock
      ...

cycloneii_lcell_comb unneeded_inverter {
      .dataa(clock),
      .combout(nclock)  // half of bad way to make an inverted clock
}

cycloneii_lcell_ff bad_cell {
      .clk(nclock),  // bad way to make an inverted clock
      ...
```

A few input ports on some WYSIWYG primitives do not have programmable inversion. For such ports, a complemented signal (e.g. !clock) cannot be connected, and instead an explicit inversion using an lcell_comb must be used. See the "LCELL WYSIWYG Description for the Cyclone II Architecture" document for a listing of such ports.

Most input ports on WYSIWYG primitives can also be directly connected to GND and VCC. For such ports it is always best to make such direct connections, rather than creating a new logic cell whose output is always 0 or 1 and connecting this signal to ports. A few ports cannot be directly connected to VCC and/or GND; in such cases a logic cell whose output is always 0 or 1 must be created and that signal connected to the desired input ports.

```
cycloneii_lcell_ff aload_using_preset {
      .sload(sloadsig),
      .sdata(VCC), // OK, but can't connect to GND directly
```

. . .

Logic Cell	Input Port	Can Connect to VCC	Can Connect to GND
lcell_comb	dataa-datad	Yes*	Yes*
	cin	Yes*	Yes*
lcell_ff	datain	Yes	**No**
	clk	Yes	Yes
	aclr	Yes	Yes
	sclr	Yes	Yes
	sload	Yes	Yes
	sdata	Yes	**No**
	ena	Yes	Yes

* The LUT mask will be modified to account for constant signal inputs.

6. Synthesis Overview and Optimization

6.1 Description of LEs and LABs

Section 2 described the basic logic cells that are used in Cyclone II devices. The Quartus® II development software groups these logic cells into two structures called LEs and LABs. Both these structures have legality restrictions in addition to the restrictions on the individual logic cells. These restrictions will be discussed in later sections.

An LE ("Logic Element") is a grouping of up to 1 lcell_comb block and up to 1 lcell_ff block. Figure 4 shows the LE. Note that not all positions in a LE need necessarily be populated with a logic cell for the LE to be valid. For more details on the Cyclone II logic cell detail refer to the "LCELL WYSIWYG Description for the Cyclone II Architecture" document.

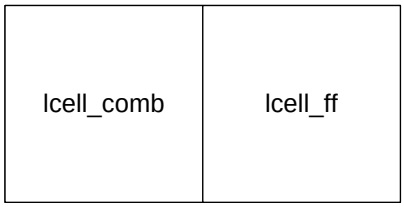

Figure 4 – Logic Cells in One LE

A LAB is a grouping of 16 LEs, where each LE contains up to 2 logic cells. Thus, each LAB contains up to 32 logic cells (16 lcell_comb cells and 16 lcell_ff cells). Figure 5 shows the 32 cells with the lcell_comb cells in gray and the lcell_ff cells in white. The dotted boxes show the groupings of blocks making an LE. Note that the locations shown are *possible* locations but may not necessarily all be used in the fitting of a user circuit.

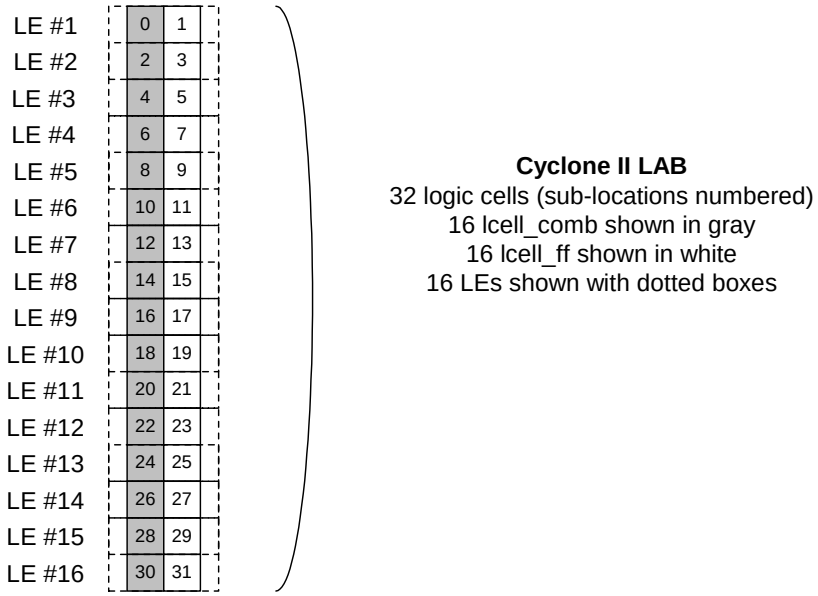

Figure 5 – Cyclone II LAB with 32 Logic Cells

6.2 Measuring Synthesis Area Quality

Because logic cells can combine in different ways to form LEs, measuring synthesis area is not as simple as counting the number of logic cells created. From an area standpoint, synthesis that creates fewest LEs and LABs when the fitter is striving for maximum density is best.

We have also provided quartus.ini variables, “fit_pack_for_density=on” and “fit_report_lab_usage_stats=on”, that will allow for accurate area estimation. When these variables are used, the Quartus II software will try to minimize circuit area when packing logic cells into LEs and LABs, typically with a small Fmax degradation. The Quartus software will also print info messages like:

Info: Number of LABs at the end of packing: 100
Info: Number of LEs at the end of packing: 1600

stating the minimum number of LEs and LABs into which it could fit the circuit. When tuning synthesis algorithms for performance (maximum circuit speed), the Quartus II software should be run without these variables.

The best way to set these variables is to add the following line to your <circuit>.qsf file in the compiler settings section:

set_global_assignment -name INI_VARS
"fit_pack_for_density=on;fit_report_lab_usage_stats=on”

It is important that there be no spaces in the quoted string.

6.3 Recommendations for Good Fitting Results

This section presents a list of rules that are not necessary for a working design but will aid in improving the quality of the design and maximizing the number of designs that will fit onto the device.

1. The **sclr** port on an lcell_ff should be used with discretion. The Cyclone II architecture supports a maximum of 1 **sclr** per LAB. Improper use of this port could result in a severe degradation in logic utilization, fitting, and circuit speed. For example, in the past **sclr** has been used to generate a 5-input function in one logic cell. This reduces the number of logic cells needed to implement a circuit, and hence may appear to be a good idea from a synthesis perspective. However, once an lcell_ff that uses **sclr** is placed into a LAB, that LAB typically cannot have any other registered lcell_ff cells packed into it unless they use the same **sclr** value or have **sclr** unconnected. This can result in a large reduction in logic density in the worst case, since only one lcell_ff cell can be placed in a LAB, rather than 16. Hence undisciplined use of **sclr** will lead to very bad fitting and speed, as the fitter will have little flexibility on which lcell_ff cells can be grouped together into LABs.

   The **sclr** port should be used to implement real synchronous clear signals in a user circuit, since there tend to be a small number of high-fanout synchronous clear signals in this case. Hence the fitter will not be unduly constrained by the synchronous clear signals. The **sclr** port can also be used judiciously to implement other moderate fanout signals to reduce logic cell count, but the techniques of Section 6.2 should be used to ensure that logic cell count *and* minimum LAB count do indeed improve when measured by the fitter in its “pack densely” mode.

2. The same restrictions for **sclr** as described above exist for **sload**. The **sload** port should only be used for high fanout signals. The **sload** port can be used to implement 3:1 multiplexers in one logic cell, where several multiplexers share the same select signals and hence **sload** will have reasonable fanout, successfully.

3. Clock enables (**ena**) can be used more liberally, since two distinct **ena** values are allowed per LAB, and this greatly improves the fitter’s ability to cope with a large number of **ena** values vs. having one per LAB. Nonetheless, an excessive number of clock enables (**ena**) can make it difficult to achieve high levels of logic utilization. We found that circuits that use thousands of distinct **ena** signals typically can achieve only 90% or so logic utilization before the fitter can no longer divide the logic cells into legal LABs. Circuits that use **ena** more sparingly routinely achieve logic utilizations over 99%. Hence some care to limit the number of distinct **ena** signals (particularly for low-fanout signals) is merited.

4. The total number of signals that can be routed into a LAB (not including cin or signals that are generated by logic cells inside the LAB) is 38. For good routability, however, it is best not to go above 36 distinct signals that need to be routed into a LAB (not including clk and aclr, which will usually be routed on special, global interconnect).

5. The fitter will be responsible for trying to “pack” lcell_comb cells and lcell_ff cells together into LEs when necessary. Because of this, the synthesis step no longer needs to put lcell_comb and lcell_ff cells together. However, care should be taken to make sure that cells are not created in a way that adversely affects the fitter’s ability to put lcell_comb and lcell_ff cells in the same LE.

7. Logic Cell Rules

7.1 Constraints for Individual Register logic cells (lcell_ff)

Refer to Figure 1 – Cyclone II Register Logic Cell (lcell_ff) for a diagram of the input/output ports.

1. If the **clk** port is connected the **regout** port must be connected.
2. The **clk** port must be connected when the **sclr** port is connected.
3. The **clk** port must be connected when the **sload** port is connected.
4. The **sdata** port must be connected when the **sload** port is connected.
5. The **clk** port must be connected when the **ena** port is connected.

7.2 Constraints for Individual Combinational Logic Cells (lcell_comb)

Refer to Figure 2 – Cyclone II Combinational Logic Cell **(lcell_comb)** for a diagram of the input/output ports.

1. The **cin** port must be either unconnected or connected to a **cout** port of another lcell_comb cell. If connected to the cout port of another cell this cell must be using the parameter “sum_lutc_input = cin” to have any effect.
2. The **cout** port must either be connected to exactly 1 **cin** port of another lcell_comb cell or be unconnected.
3. If **cout**/**cin** is connected then only the **dataa**, **datab**, **datac** (if **cin** is not used), input ports on the lcell_comb can be used.
4. On an lcell_comb, it is illegal to connect an input, which does not affect at least one of its outputs according to the lcell_comb’s LUT mask. (For this rule, Quartus will disconnect this input).
5. On an lcell_comb, it is illegal for a LUT mask to be dependent on an unconnected input.

8. LE and LAB Rules

This section describes the rules that should be obeyed by synthesis tools in order to ensure that (i) all logic cells are electrically valid, and (ii) when the logic cells are grouped into LEs and LABs, the resulting LABs are legal and routable.

8.1 Constraints for Carry Chains in LABs

A “carry chain” is a group of lcell_comb cells where the **cout** port of one cell feeds the **cin** port of the next lcell_comb cell.

The logic cells in a chain must go into adjacent logic cells. For example, if the first cell in a carry chain is placed at sublocation 2 in a LAB, the second cell in the chain **must** be placed in sublocation 4 of the same LAB.

Very long carry chains can span across multiple LABs. The carry chain must exit the LAB and continue onto the LAB immediately below it (carry chains propagate downwards). Figure 6 illustrates a carry chain that spans multiple LABs.

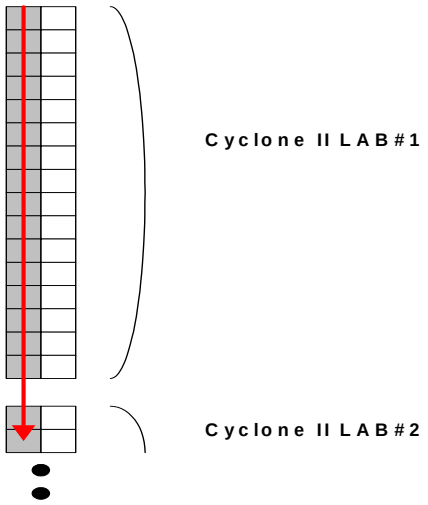

Figure 6 - Long Carry Chain Spanning Multiple LABs

8.2 Constraints for LAB-Wide Signals

Many of the signals connected to the logic cells are “LAB-wide” signals. Such signals are generated once per LAB and are shared by all the logic cells placed within the LAB. Hence, having logic cells that require too many distinct LAB-wide signals placed within one LAB is illegal.

Section 8.1 discussed the constraints of carry-chains in LABs. Although carry-chains themselves cannot be made illegal, poor quality fitting may result from having carry-chains that feed register cells where the register cells use an excessive number of distinct LAB-wide signals. This is best shown in an example. Suppose we have a carry chain of 6 logic cells (spanning 6 LEs) and each of these logic cells feed a register. If each register contains a unique clock signal, a LAB can only contain two of these registers (since each LAB can have a maximum of 2 unique clock signals). This results in a sub-optimal placement if the registers really don’t need unique clock signals.

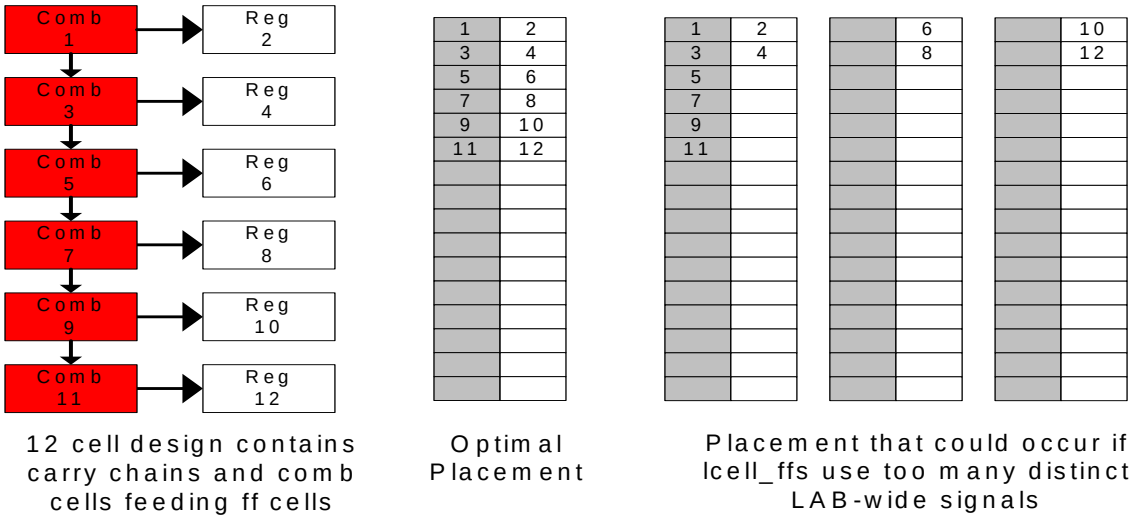

Figure 7 – Effect on Placement When LAB-Wide Signals Are Used Excessively

In Cyclone II devices, the signals connected to the **clk**, **ena**, **aclr**, **sclr** and **sload** ports on lcell_ff cells are all LAB wide. An lcell_ff port connected to a "regular" signal consumes one LAB-wide signal of that type. The same signal used in inverted form counts as a separate LAB-wide signal. So, if we have clock used by an lcell_ff cell in a LAB and !clock used by another lcell_ff cell, we require two LAB-wide clock lines. VCC and GND connected to a logic cell port also require a LAB-wide signal line (set to GND or VCC). There are situations where some ports on a logic cell are left unconnected. If this is the case, then the unconnected may or may not count as a use of a LAB-wide signal line, depending on the port and the exact logic cell configuration. Table 1 below shows how unconnected ports on an lcell_ff are handled.

Unconnected Port	Register Used (.regout connected)	Register Unused (.regout unconnected)
aclr	Used for GND	Not Used
sload	Not Used	Not Used
sclr	Not Used	Not Used
ena	Used for VCC	Not Used
clk	Used for GND	Not Used

Table 1 – Whether or not ports are considered "used" when unconnected

In all the rules below, it is important to count unconnected signals that are considered "used" Register ports, as defined by Table 1.

1. All **clk** and **ena** signals on an lcell_ff are paired to form a register clock (see Figure 8). In any LAB, there can be no more than 2 distinct clock pairs.
    - The same **clk** signal with 2 different **ena** signals counts as 2 register clock pairs.

- The same **ena** signal with 2 different **clk** signals counts as 2 register clock pairs.

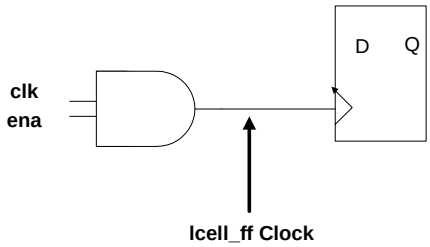

Figure 8 - Grouping of Clock and Clock Enable Pairs to Form lcell_ff Clocks

2. A maximum of 2 distinct clock signals can be connected to the **clk** ports.

Table 2 provides some examples of how clock + clock enable pairs are created. The examples show 2 sets of clocks and 2 sets of clock enables corresponding to any 2 register logic cells. clk (#1) and ena (#1) belong to register cell #1, etc. Unconnected **ena** ports are set to VCC, as Table 1 specified. A, B, C and D correspond to signal nets.

clk (#1)	ena (#1)	clk (#2)	ena (#2)	Number of Reg Clock pairs	Pairs
A	B	C	D	2	(A, B) (C, D)
A	B	A	C	2	(A, B) (A, C)
A	C	B	C	2	(A, C) (B, C)
A	B	A	B	1	(A, B)
A	-	B	-	2	(A, VCC) (B, VCC)
A	-	A	-	1	(A, VCC)
A	-	A	B	2	(A, VCC) (A, B)
A	B	-	-	2	(A, B) (GND, VCC)

Table 2- Examples of How the Clock and Clock Enable Are Paired to Form lcell_ff Clocks (regout is connected for all cells)

3. A maximum of 2 distinct signals can be connected to the **aclr** ports. (Recall from Table 1 that an unconnected **aclr** on an lcell_ff counts as a GND signal.)
4. A maximum of 1 distinct signal can be connected to the **sload** ports and a maximum of 1 distinct signal to the **sclr** ports.
    - An lcell_ff that uses **sload** and/or **sclr** can only be placed in a LAB that uses the same **sload** and/or **sclr** or in a LAB that does not use its **sclr** and **sload**.

Table 3 shows combinations of **sload** and **sclr** and whether or not **sload** and **sclr** in this combination are considered “used”. Table 4 shows examples of which combinations of **sload** and **sclr** would be legal to place within a single LAB. In both tables, A and B are signal nets (non-VCC, non-GND).

	sload signal	sclr signal	sload used?	sclr used?
Case #1	Unconnected	Unconnected	Not Used	Not Used
Case #2	GND	GND	Not Used	Not Used
Case #3	A	Unconnected	Used	Used (set to GND)
Case #4	Unconnected	A	Used (set to GND)	Used
Case #5	A	B	Used	Used
Case #6	VCC	Unconnected	Used	Used (set to GND)

Table 3 - Examples of When sload and sclr Are Considered Used/Free

sload (lcell_ff #1)	sclr (lcell_ff #1)	sload (lcell_ff #2)	sclr (lcell_ff #2)	Form legal LAB?
A	B	A	B	Legal
A	B	A	C	Illegal
A	B	B	A	Illegal
A	B	Unconnect or GND	Unconnect or GND	Legal
A	B	A	Unconnect or GND	Illegal
A	B	Unconnect or GND	B	Illegal

Table 4 - Examples of Legal and Illegal Combinations for lcell_ff Cells Using sclr and sload

8.3 Constraints for LAB Control Signal Routability

The following rules ensure that all the control signals required within a LAB can be routed into the LAB. Any logic cell port that is connected to a signal net (non-VCC, non-GND) requires that signal to be routed into the LAB, to the appropriate LAB-wide line. If a signal is required in true and complemented form in a LAB, it must be routed in twice. If one of the logic cell ports is connected (or considered connected by Table 1) to either VCC or GND then it may or may not require a VCC or GND signal to be routed into the LAB to the appropriate LAB-wide line. This depends on whether or not there exists a LAB-wide tie-off for that signal. Table 5 below shows which lcell_ff ports require signals to be routed into the LAB when the port is connected (or considered to be connected by Table 1) to either VCC or GND.

Unconnected Signal	VCC	GND
aclr	Must be routed	No routing
clk	Must be routed	Must be routed
ena	No routing	Must be routed
sload	No routing	No routing
sclr	Must be routed	No routing

Table 5 - When VCC/GND Require Routing into the LAB

In all the rules below, it is important to count VCC/GND signals that require routing resources, as defined by Table 5.

1. The sum of the following signals (only counting distinct signals) must be at most 4:
    - **clk** (non-global)
    - **ena**
    - **sload**
    - **aclr** (non-global)
    - **sclr**

Since one register can contain all 5 of the above signals such a register will not be able to fit in an empty LAB. This will cause a fitter error. Synthesis should avoid creating registers with all five control inputs when **clk** and **aclr** are both known to be non-global. The Quartus II software will promote **clk** and **aclr** signals to global networks automatically, until it runs out of global networks (there are 16 global networks in Cyclone II). The best heuristic then is that synthesis should assume that at least one of **clk** or **aclr** will be globally routed, and hence using all 5 register control signals on a single register is valid.

2. The sum of the following signals (only counting distinct signals) must be at most 2: This is due to the fact that **sload** steals a non-global **clk** line.
    - **clk** (non-global)
    - **sload**
3. The sum of the following signals (only counting distinct signals) must be at most 3. This is due to the fact that non-clk and non-aclr global signals must enter the LAB via regular LAB lines and the connections between global lines and LAB lines are limited.
    - **ena** (global)
    - **aload** (global)
    - **sload** (global)
    - **sclr** (global)