Stratix II EDA and Academic Developer Functional Description

Version 1.3

March 2, 2006

By

Altera Corporation

Table of Contents:

1. Overview

This document is intended for developers working with the Stratix II™ architecture. It is a companion to the WYSIWYG Device Primitive Guide for Stratix II and should be used in conjunction with that document. This document provides information about the Stratix 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 Stratix 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 the Stratix II family, 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). Note that the terms LE (logic element) and LC (logic cell or lcell) are used interchangeably in this document, as are the terms flip-flop and register.

Figure 1 shows the full functionality of the Stratix II register logic cell. It has eight inputs of **datain, adatasdata, clk, ena, sload, sclr, aload** and **aclr**. It has one output of **regout**.

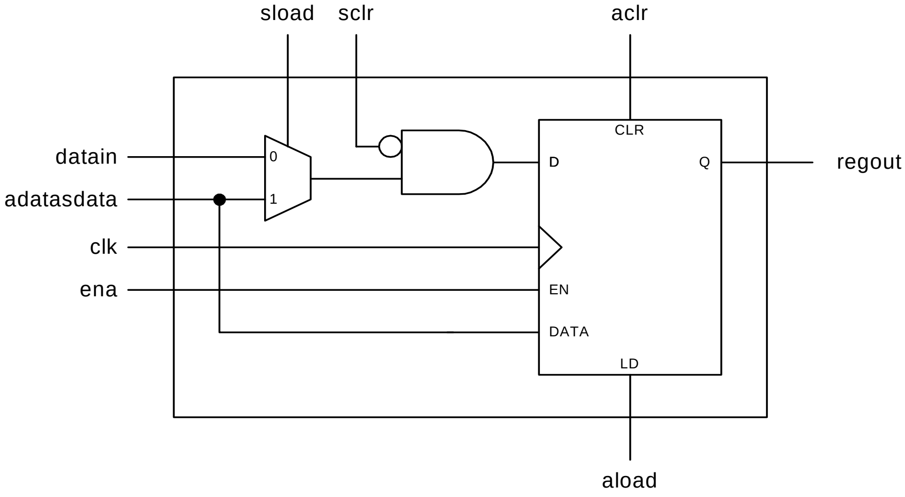


Figure 1 – Stratix II Register Logic Cell (lcell_ff)

Figure 2 shows the full functionality of the Stratix II combinational logic cell (now referred to as an lcell_comb). It has eight inputs of **dataa, datab, datac, datad, datae, dataf, sharein** and **cin**. It has four outputs of **shareout, cout, sumout** and **combout**.

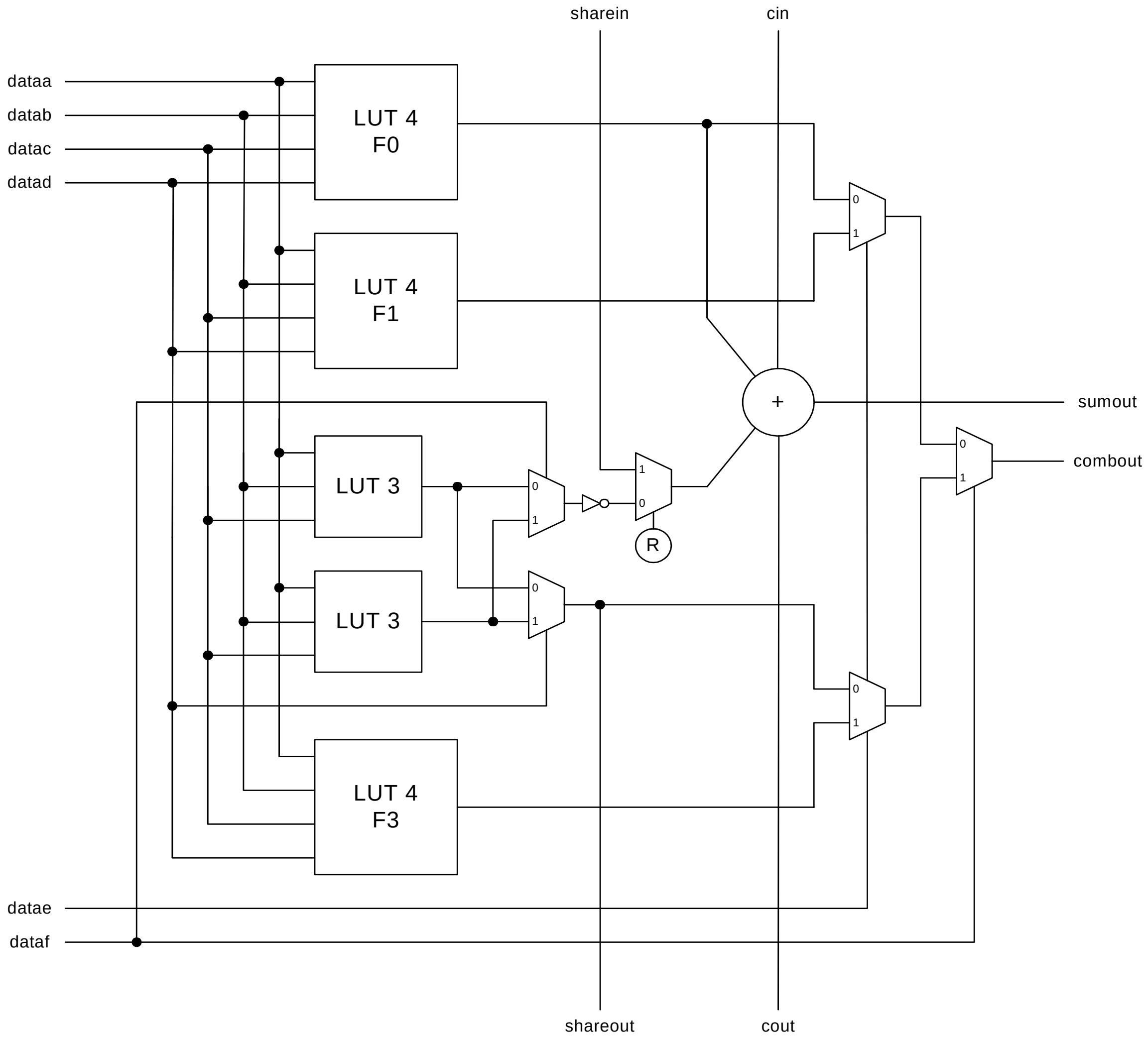


Figure 2 – Stratix II Combinational Logic Cell (lcell_comb)

3. Coordinate System and Location Assignments

The diagram below shows the Stratix II coordinate system.

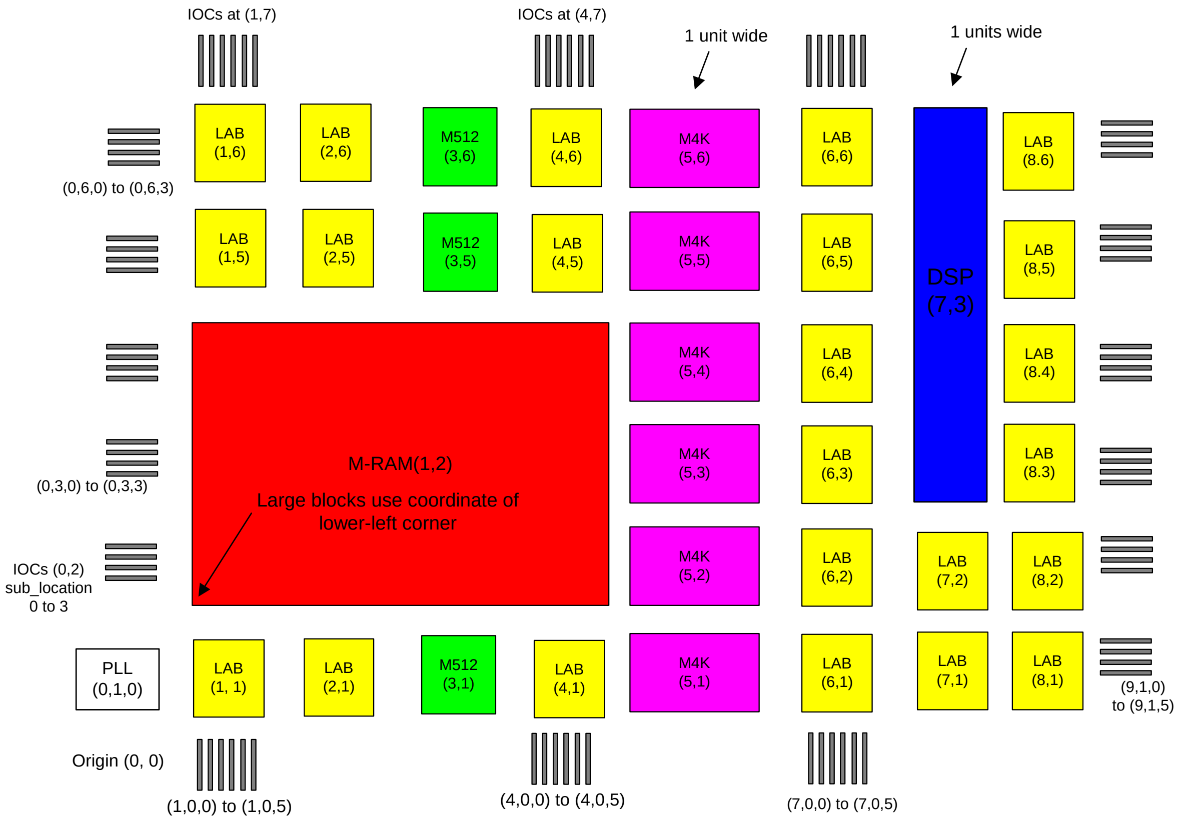


Figure 3 - The Stratix II Coordinate System

Stratix 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 shorthand for 2D regions that are the same size as the block to which the assignment is being made.

You can also make 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. For example, a region assignment of custom_region_X4_Y4_X7_Y4 would be considered to include the M-RAM block at (1, 2), the M4K RAM block at (5, 4) the LAB at (6, 4), and the DSP block at (7,3).

4. Routing Delay vs. Distance & Routing Rules

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

The speed of connections in the Stratix 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. Horizontal routing is also approximately 30% faster than vertical routing for long distances.

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 stratixii_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.

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

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

stratixii_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 Stratix II Family" 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.

```
stratixii_lcell_ff aload_using_preset {
      .aload(aloadsig),
      .adatasdata(VCC), // OK, but can't connect to GND directly
      ...
```

Logic Cell	Input Port	Can Connect to VCC	Can Connect to GND
lcell_comb	dataa-dataf	Yes	Yes
	cin	**No**	Yes
	sharein	**No**	Yes
lcell_ff	datain	Yes	**No**
	clk	Yes	Yes
	aclr	Yes	Yes
	aload	Yes	Yes
	sclr	Yes	Yes
	sload	Yes	Yes
	adatasdata	Yes	**No**
	ena	Yes	Yes

6. Synthesis Overview and Optimization

6.1 Description of ALMs and LABs

Section 2 described the basic logic cells that are used in the Stratix II devices. The Quartus II software groups these logic cells into two structures called ALMs 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 ALM ("Adaptive Logic Module") is a grouping of up to 2 lcell_comb blocks and up to 2 lcell_ff blocks. Figure 4 shows the ALM. Note that not all positions in an ALM need necessarily be populated with a logic cell for the ALM to be valid. For more details on the Stratix II logic cell detail refer to the "LCELL WYSIWYG Description for the Stratix II Family" document.

lcell_comb_0	lcell_ff_0
lcell_comb_1	lcell_ff_1

Figure 4 – Logic cells in one ALM

A LAB is a grouping of 8 ALMs, where each ALM contains up to 4 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 ALM. Note that the locations shown are *possible* locations but may not necessarily all be used in the fitting of a user circuit.

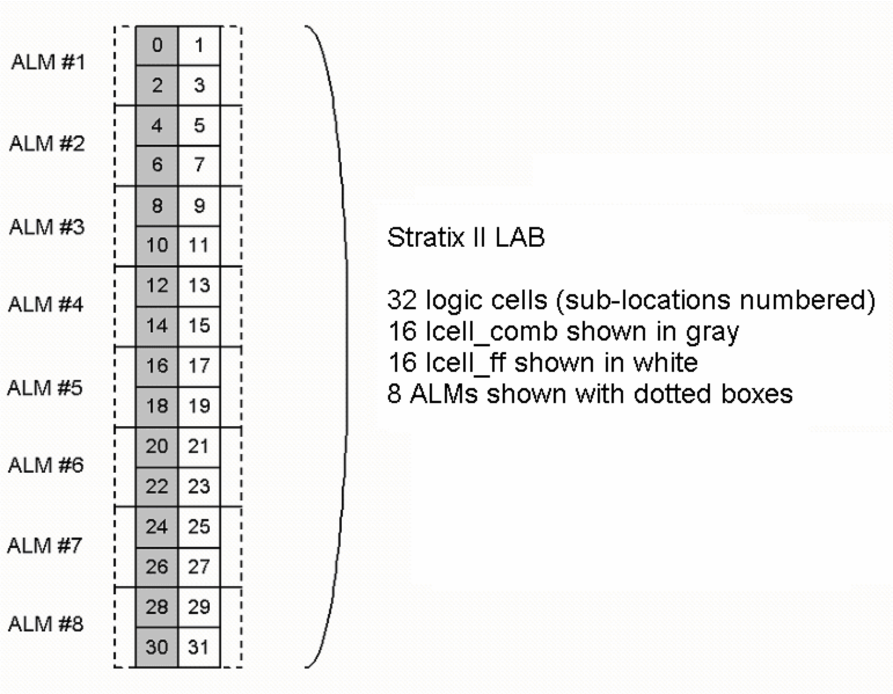


Figure 5 – Stratix II LAB with 32 Logic Cells

6.2 Measuring Synthesis Area Quality

Because lcell_comb elements can combine in different ways to form ALMs, measuring synthesis area is not as simple as counting the number of logic cells created. Sometimes a smaller number of logic cells will require more area in terms of ALMs (and hence LABs). As a simple motivating example, consider that 80 6-input lcell_comb cells could use up to 80 ALMs (since each ALM can only contain two 6-input functions if certain restrictions are met in the LUT masks for those functions). One hundred 4-input lcell_comb cells, on the other hand, would only take up 50 ALMs (since each ALM can always contain two 4-input functions). From an area standpoint, synthesis that creates the fewest number of ALMs is best.

We have also provided a quartus.ini variable, “fit_pack_for_density_light=on”, that will allow for accurate area estimation. When this variable is used, the Quartus II software will try to minimize circuit area when packing logic cells into ALMs and LABs, without undue f_{max} degradation. This emulates the behaviour a designer will naturally see when they have a fairly full device. When tuning synthesis algorithms for performance, the Quartus II software should also be run without this variable. When tuning for density, you should also set fit_report_lab_usage_stats=on. This causes the Quartus II software to print a message in the fitter report file that states the number of ALMs and LABs used by the Quartus software II for a circuit, before any spreading out of the design to improve routability is performed.

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

```
set_global_assignment -name INI_VARS "fit_pack_for_density_light=on;
                     fit_report_lab_usage_stats=on"
```

The messages printed that give the LAB and ALM counts required by the design are:

Info: Number of LABs at the end of packing: <number>
Info: Number of ALMs at the end of packing: <number>

6.3 Recommendations for Good Area and Performance

Generally, it is good to use 6-input functions to minimize circuit depth and thus improve the performance. In cases that don’t affect depth, it is best to use 6-input functions only if they either save more than 2 5-or-fewer-input functions, or their LUT mask allows an ALM to hold 2 such 6-input functions.

6.4 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 Stratix 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 only to implement real synchronous clear signals in a user circuit, since there tends 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.

2. The same restrictions for **sclr** as described above exist for **sload**. The **sload** port should only be used for high fanout signals.
3. Clock enables (**.ena**) can be used more liberally, since three 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 in APEX devices 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% in APEX devices. Hence some care to limit the number of distinct **.ena** signals (particularly low-fanout signals) is merited. *Stratix devices have 2 CLK/CE pairs in a LAB while Stratix II devices have 2 clocks and 3 clock enables. Stratix II devices are slightly more flexible with clock enables than Stratix devices.*
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 44. For good routability, however, it is best not to go above 38 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 "ALMs" 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 ALM.
6. An ALM (see Figure 4) can contain up to two lcell_comb cells. Table 1 shows the restriction (in addition to the individual logic cell restrictions) that clustering has on which lcell_comb cells can be clustered together into a single ALM. This table does not show every possible combination of lcell_comb types. See the "Lcell WYSIWYG Description for the Stratix II Family" document for more details on how lcell_comb cells combine in ALMs.

lcell_comb_0	lcell_comb_1	Restrictions
4-input lcell_comb	4-input lcell_comb	None
5-input lcell_comb	5-input lcell_comb	- 2 of the inputs must be common
6-input lcell_comb	6-input lcell_comb	- 4 of the inputs must be common - the LUT masks of both lcell_comb must be equivalent (LUT inputs can be rotated to make the mask identical)

Table 1 – Restrictions on clustering combinations of lcell_comb cells into an ALM

Choosing the type of lcell_comb to make is not trivial. A general rule of thumb is to only create 6-input functions when there is a clear advantage to doing so (such as if it reduces the depth of the circuit or decreases the area). When there is no compelling reason to create a 6-input function, it is better to create smaller functions (since, as shown in the table, there are less clustering restrictions on functions with less than 6 inputs).

7. Logic Cell Rules

7.1 Constraints for Individual Register logic cells (lcell_ff)

Refer to Figure 1 – Stratix 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 **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 – Stratix II Combinational Logic Cell (lcell_comb) for a diagram of the input/output ports.

1. The **cin** port must be unconnected, connected to a **cout** port of another lcell_comb cell or connected to GND. If connected to the cout port of another cell, this cell must be using the cout or sumout port for the cin port 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. The **sharein** port can only be connected if the **cin** port is connected and only on lcell_comb cells with parameter **shared_arith** = **on**.
4. When the **sharein** port is connected it can either be connected to GND or must be fed by the **shareout** port of the same lcell_comb cell which feeds the **cin** port.
5. The **shareout** port can only be connected when the **cout** port is connected.
6. If the **shareout** port is connected, it must feed the **sharein** port of the same lcell_comb cell being fed by the **cout** port.
7. If cout/cin is connected, but not sharein/shareout, then only the **dataa**, **datab**, **datac**, **datad** and **dataf** input ports on the lcell_comb can be used.
8. If parameter **shared_arith** = **on** (i.e. sharein/shareout is connected) then only the **dataa**, **datab**, **datac**, **datad**, and **dataf** input ports on the lcell_comb can be used. The dataf port can only affect the combinational output (combout) of the lcell_comb.
9. 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, the Quartus II software will disconnect this input).
10. On an lcell comb, it is illegal for a LUT mask to be dependent on an unconnected input.

8. ALM 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 ALMs 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 lcell_comb cell (with optional **sharein/shareout** connections). In contrast to previous architectures, it is impossible to create an illegal carry chain because the lcell_ff and lcell_comb cells aren’t combined into one LE in the Stratix II devices.

The start of a carry chain always has a constant **carryin**/**sharein** connection. The fitter can assign a carry chain to start at either sub-location 0 or 16 in the lab (only these sub-locations have tie-offs for the carry chain input). If this is too restrictive for a particular design, the fitter can add an lcell_comb cell to generate a constant **carryin**/**sharein** signal to the beginning of the chain. This added cell is has all inputs unconnected (or only connected to GND/VCC) but has the **carryout** (and possibly **shareout**) connected to another lcell_comb cell’s **carryin**. By using this technique, the fitter can put a carry-chain at any sub-location.

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.

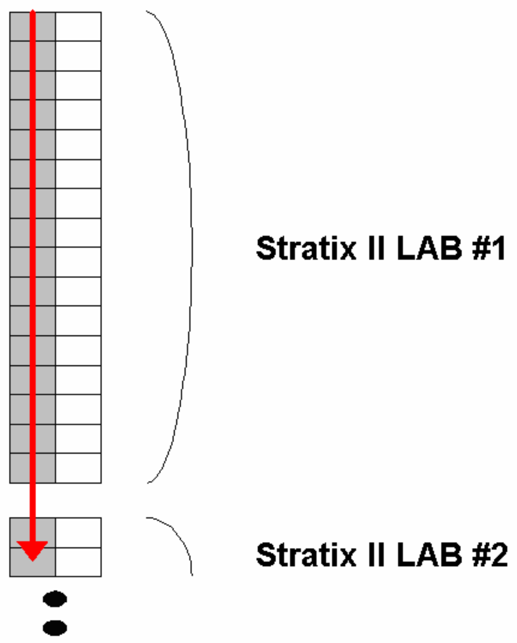


Figure 6 - Long carry chain spanning multiple LABs

The Stratix II ALMs can implement arithmetic functions that require up to 8 inputs per ALM. If such chains were to use all 8 ALMs in a LAB, there would not be enough LAB inputs to bring those signals to the chain. In these cases, the fitter automatically splits these chains into smaller chains that each fits into half of a LAB. These chains are then connected between LABs by using the new “early exit” and “late entry” modes described below.

Chains that are placed at sub-locations 0, 4, 8 or 12 have the option to leave the LAB in the middle and continue into sub-location 0 of the next LAB. This is referred to as “early exit”. Chains that leave the LAB at the end (sub-location 30) have the option of entering directly into the middle of the next lab (sub-location 16). This is referred to as “late entry”. In Stratix II devices, columns alternate supporting early exit chains or late entry chains (i.e. each LAB only supports one of these depending on which column it is in). These concepts are illustrated in Figure 7.

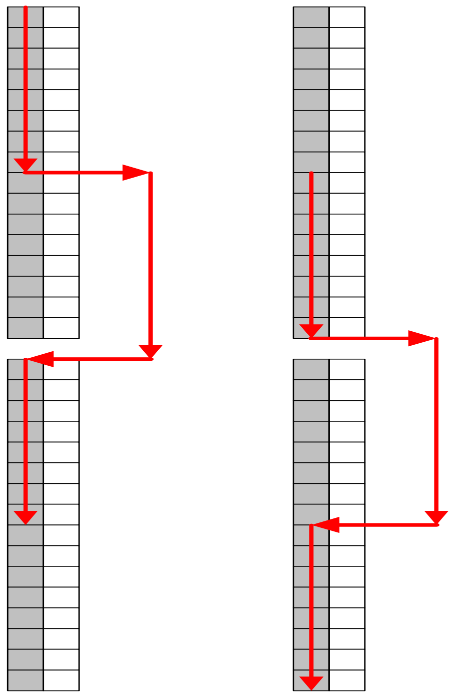


Figure 7 - Early Exit and Late Entry carry chains across LABs

In addition to splitting carry chains and using “early exit” or “late entry” modes, the fitter may split illegal carry chains and use “regular” routing resources (i.e. not the dedicated carryout to carryin connections) to continue the carry chain from one LAB to another. This is avoided if at all possible. When this type of splitting is done, a warning message is output to indicate the chain that required this form of splitting.

8.2 Constraints for Carry Chains in ALMs

Carry chains imply a specific ALM packing for adjacent cells, and so require certain ALM constraints to be satisfied. For example, the first and second lcell_comb of a carry chain will always be placed into one ALM, the third and fourth lcell_comb of that carry chain will always be placed in the next ALM, etc. Figure 8 illustrates the pairings of the carry cells into ALMs.

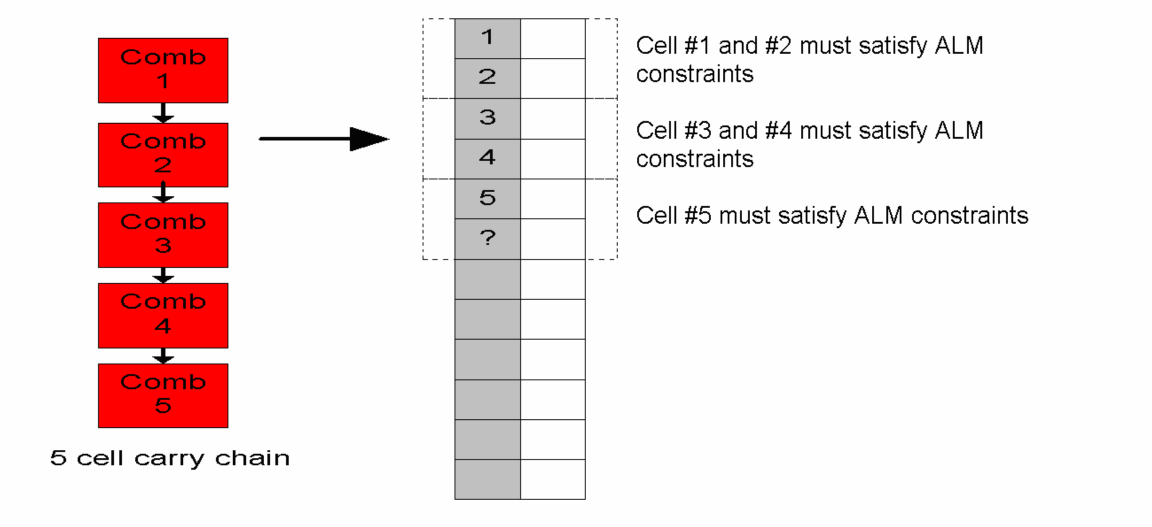

Comb 1
Comb 2
Comb 3
Comb 4
Comb 5
5 cell carry chain
1
2
3
4
5
?
Cell #1 and #2 must satisfy ALM constraints
Cell #3 and #4 must satisfy ALM constraints
Cell #5 must satisfy ALM constraints

Figure 8 - Clustering of a carry-chain into ALMs

The following ALM legality rules must be obeyed by all adjacent lcell_comb cells in a carry chain (such as cell #1 and #2 in Figure 8).

1. They must be in the same shared arithmetic mode (i.e. either both lcell_comb have connected **sharein**/**shareout** ports or neither has connected **sharein**/**shareout** ports). Note that the LUTs of *different* ALMs in the chain can use either shared or non-shared mode without any restriction as long as the two LUTs of a chain in a given ALM use the same mode.
2. The number of distinct signals connected to the lcell_comb ports **dataa**, **datab** and **datac** (of both logic cells) can be no more than 4. This is because two of the three ports (**dataa**, **datab**, **datac**) must be shared between the two logic cells.

Figure 9 shows an example of rule #2 for an ALM that obeys the rule and an ALM that violates the rule. The legal ALM has signals s1, s2, s3, and s6 on the dataa-datac ports of both its logic cells, which is the limit of 4 signals. The illegal ALM has signals s1, s2, s3, s6, s7, and s8 on those ports – a total of 6 signals.

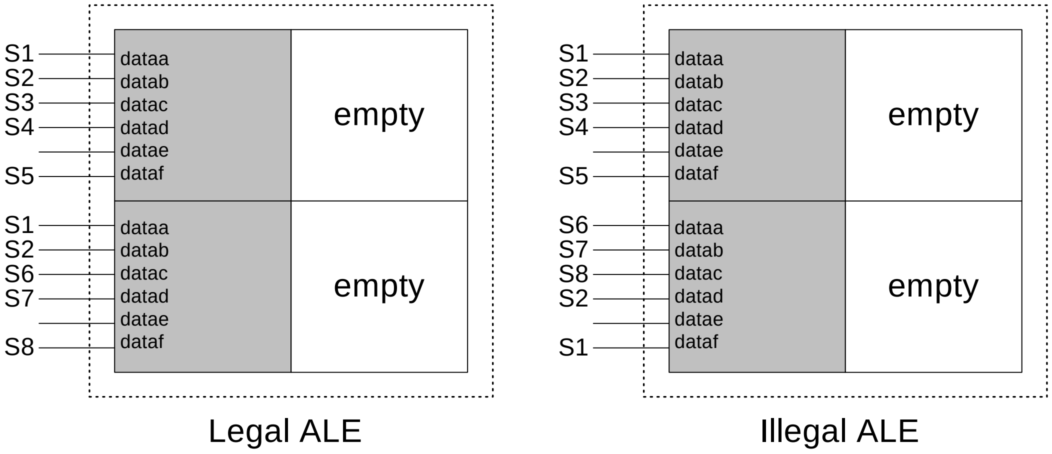
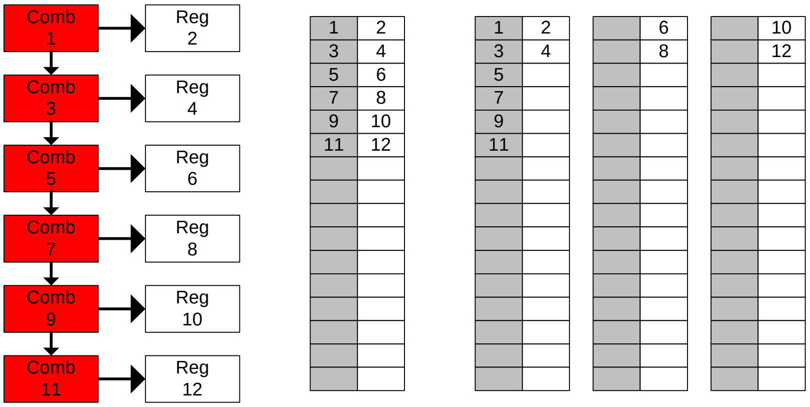


Figure 9 - Legal and Illegal Example of Rule #2

8.3 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 3 ALMs) 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.



12 cell design contains carry chains and comb cells feeding ff cells

Optimal Placement

Placement that could occur if lcell_ffs use too many distinct LAB-wide signals

Figure 10 – Effect on placement when LAB-wide signals are used excessively

In the Stratix II family, the signals connected to the **clk**, **ena**, **aclr**, **aload**, **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 2 below shows how unconnected ports on an lcell_ff are handled.

Unconnected Port	Register Used (.regout connected)	Register Unused (.regout unconnected)
aload	Not Used	Not Used
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 2 –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 2.

1. All **clk** and **ena** signals on an lcell_ff are paired to form an register clock (see Figure 11). In any LAB, there can be no more than 3 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.

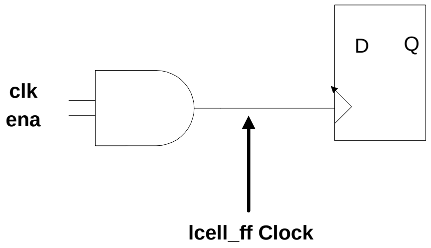


Figure 11 - Grouping of clock and clock enable pairs to form lcell_ff clocks

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

Table 3 provides some examples of how clock + clock enable pairs are created. The examples show 3 sets of clocks and 3 sets of clock enables corresponding to any 3 register logic cells. clk (#1) and ena (#1) belong to register cell #1, etc. Unconnected **ena** ports are set to VCC, as Table 2 specified. A, B, C, D, E and F correspond to signal nets.

clk (#1)	ena (#1)	clk (#2)	ena (#2)	clk (#3)	ena (#3)	Number of Reg Clock pairs	Pairs
A	B	A	C	A	D	3	(A, B) (A, C) (A, D)
A	C	B	C	B	C	2	(A, C) (B, C)
A	B	A	B	A	B	1	(A, B)
A	B	A	B	!A	B	2	(A, B) (!A, B))
A	-	B	-	B	C	3	(A, VCC) (B, VCC) (B, C)
A	-	A	-	A	-	1	(A, VCC)
A	-	A	B	-	-	3	(A, VCC) (A, B) (GND, VCC)
A	B	-	-	-	-	2	(A, B) (GND, VCC)
A	B	C	D	E	F	*illegal combination (see rule #2)*	*N/A*

Table 3- 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 2 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 **aload** ports.
5. If any lcell_ff in a LAB uses **aload**:
    - All lcell_ff cells in that LAB that use **aload** must use the same **aclr** signal as the first cell
    - All lcell_ff cells in that LAB that do not use **aload** must use the same **aclr** signal.
6. 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 has that uses **sloads** and **sclrs** can only be placed in a LAB that uses the same sload or sclr or in a LAB that does not use its sclr and sload.

Table 4 shows combinations of sload and sclr and whether or not sload and sclr in this combination are considered "used". Table 5 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 net signals (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 4 - 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 5 - Examples of legal and illegal combinations for lcell_ff cells using sclr and sload

8.4 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 2) 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 6 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 2) to either VCC or GND.

Unconnected Signal	VCC	GND
aclr	No routing *****	No routing
aload	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 6 - When VCC/GND require routing into the LAB (* = Different from Stratix!)

In all the rules below, it is important to count VCC/GND signals that require routing resources, as defined by Table 6.

1. The sum of the following signals (only counting distinct signals) must be at most 6:
    - **clk** (non-global)
    - **ena**
    - **aload**
    - **sload**
    - **aclr** (non-global)
    - **sclr**
2. The sum of the following signals (only counting distinct signals) must be at most 5:

- **ena**
- **aload**
- **sload**
- **aclr** (non-global)
- **sclr**

3. The sum of the following signals (only counting distinct signals) must be at most 3. This is due to the fact that **aload** steals an **ena** line but we can still use that **ena** signal if it is VCC.
    - **ena**
    - **aload**
4. 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 but we can still use that **clk** signal if it is global.
    - **clk** (non-global)
    - **sload**
5. 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)