Vivado Subsite Routing Tool

User Guide and Documentation

Contributors:
Thomas Townsend, Brad White, and Brent Nelson

NSF Center for High Performance Reconfigurable Computing (CHREC) [*]
Department of Electrical and Computer Engineering
Brigham Young University
Provo, UT, 84602

Last Modified: April 27, 2017

[*]This work was supported in part by the I/UCRC program of the National Science Foundation, grant numbers 0801876 and 1265957.

Contents

1 Introduction

The Xilinx Design Language (XDL) is a command line interface into Xilinx's ISE tool suite. Using a single command `xdl`, both device and design information can be extracted from ISE for external use. Device information (the physical components of the FPGA) is extracted via `XDLRC` files, and design information (the logical netlist, placement, and routing) is extracted via `XDL` files. Many FPGA CAD research projects have built upon the XDL interface, resulting in several useful tools capable of targeting commercial devices. With the release of Vivado, however, Xilinx no longer supports XDL. This makes external tools and frameworks that rely on the interface incompatible with next-generation Xilinx devices (Ultrascale, Ultrascale+, and beyond). Instead, Vivado includes a `Tcl` interface that grants access to Xilinx's internal device and design data structures.

Tincr, a Vivado `Tcl` plugin, defines a set of APIs that provide the same functionality as XDL did for ISE. Device and design information can be extracted from Vivado using the following `Tincr` commands:

- *write_xdlrc*: Creates a `XDLRC` file which describes the physical components of a Vivado FPGA part.
- *write_rscp*: Creates a RapidSmith Checkpoint (RSCP) which describes all relevant aspects of a Vivado design.

Due to limitations of Vivado's Tcl API, the `XDLRC` files generated from *write_xdlrc* are incomplete. They are missing the primitive definitions (described in more detail in section 3) of the device. In order to have a complete representation of a device for external tools, these primitive definitions (also called primitive defs) need to be included. Using `Tincr` and Vivado Tcl commands, a set of *partial* primitive defs can be created. These partial primitive defs include everything *except for* the **connections** between primitive def components, which is crucial for any external CAD tool. The Vivado Subsite Routing Tool (VSRT) is a GUI application that helps users add these missing connections to create complete primitive definitions for Vivado devices. The general flow for using the tool is shown in Figure 1.

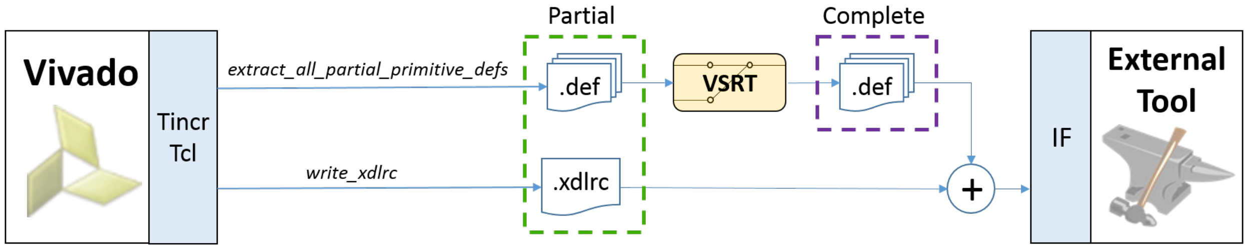


Figure 1: VSRT flow for creating full XDLRC files

As the figure shows, `Tincr` first generates a set of partially complete primitive definitions (one for each site in the device architecture). The partial primitive defs are then loaded into VSRT, where a user processes each site one-by-one. To use the tool, a user brings up a picture of the primitive site in both Vivado's device browser and the VSRT GUI. They then use VSRT to manually draw the connections visible in Vivado's device browser. This may seem like a tedious process, but there are two important observations to note:

1. Primitive definitions are at least identical for parts that have the same architecture. For example, any part within the Kintex UltraScale family will use the same set of primitive definitions. You don't need to create new primitive defs for each part. **NOTE**: This is also most likely true across series (i.e. all UltraScale parts regardless of architecture use the same set of primitive defs)

2. Many primitive definition connections can be inferred automatically in Tcl even though the site wires cannot be explicitly accessed. This is especially true for sites that are dominated by a single BEL. This reduces the workload for VSRT users and makes the process of creating primitive defs for new architectures manageable.

The remainder of this document describes important aspects of VSRT. Section 2 gives installation instructions. Section 3 describes the important terminology that VSRT uses. Section 4 gives a user guide of how to use the tool. Section 5 describes the features of the tool and how to use VSRT most effectively. Section 6 lists all currently known issues.

2 Installation

2.1 Tool Facts

- Written in Java 8
- Built on the QtJambi 4.6 GUI framework
- Supported Operating Systems:
  - Linux 32-bit and 64-bit
  - Windows 32-bit and 64-bit
  - MacOSX
- Tested on:
  - Fedora 20
  - Windows 7/8/10

2.2 Requirements

- Vivado 2016.2
- Tincr: `https://github.com/byuccl/tincr`
- RapidSmith2: `https://github.com/byuccl/RapidSmith2`
- JDK 1.8 (earlier versions may work, but have not been tested)
- Eclipse or IntelliJ IDE (optional)

2.3 Steps

VSRT is a part of the RapidSmith2 repository linked above. In order to install and run VSRT, simply follow the RapidSmith2 installation instructions. The main VSRT java class can be found under the RapidSmith2 package `edu.byu.ece.rapidSmith.device.vsrt.gui.VSRTool`, and can be run via the command line or an IDE (recommended). To generate the partial primitive definitions, you will need to install Tincr (instructions are in the Tincr link above), and Vivado 2016.2. Once you have Vivado, Tincr, and RapidSmith2 setup correctly, you are ready to use VSRT. Section 4 gives more information about how to use the tool.

3 Terminology

This section introduces important terminology used in VSRT (or links to locations that describes the terminology). If you are comfortable with Xilinx FPGA terminology and the contents of XDLRC files, you can skip subsections 3.1 and 3.2.

3.1 Vivado Devices and XDLRC files

VSRT uses the same terminology as Vivado to describe FPGA devices. Therefore it is important to understand Xilinx device terminology to fully utilize VSRT. The "Device" section of the RapidSmith2 Tech Report (found at https://github.com/byuccl/RapidSmith2/tree/master/doc) describes Xilinx FPGA architecture and XDLRC in great detail. Readers are referred to that document for more information.

3.2 Primitive Definitions

A primitive definition is a textual description of all site pins, BELs, and routing muxes inside of a primitive site, and how they are connected through site wires. The remainder of this section details how the components of a primitive site are represented in a primitive def file. It is important to note that primitive defs that enter VSRT do not include the connections ("conn" keyword). The purpose of VSRT is to generates these.

3.2.1 Header

```
(primitive_def BUFHCE 3 8
```

Every primitive definition starts with a header line (as shown above). This line tells three important things about the primitive def: (1) the name of the primitive site that it represents, (2) the number of site pins on the corresponding primitive site, and (3) the number of elements defined in the primitive def. Element is a generic term used to represent a "component" of the primitive site. Essentially, everything within the primitive site ends up being an element in the resulting primitive def (this will be shown in the following subsections).

3.2.2 Site Pins

```
// Site pins first show with the "pin" keyword
(pin O O output)
(pin I I input)
(pin CE CE input)
...
// They also show up later in the primitive def as "elements"
(element O 1
  (pin O input)
  (conn O O <== BUFHCE O)
)
(element I 1
  (pin I output)
  (conn I I ==> BUFHCE I)
)
(element CE 1
  (pin CE output)
  (conn CE CE ==> CEINV CE)
  (conn CE CE ==> CEINV CE_B)
)
```

The pins of a site show up in two different places in the primitive def file. The first is the pin section, which appears directly after the header. Each pin is defined with a name and a direction. Site pins are also defined as elements, where the connections of the pin are included. It is important to note that the direction of the "element" pin is the opposite of the direction found in the pin section. The element pin direction represents the pin's direction viewed from the internal components of the site, while the other direction represents the pin's direction external to the site.

3.2.3 BELs

```
  (element BUFHCE 3 # BEL
    (pin O output)
    (pin I input)
    (pin CE input)
    (conn BUFHCE O ==> O O)
    (conn BUFHCE I <== I I)
    (conn BUFHCE CE <== CEINV OUT)
  )
```

BELs within a primitive site are represented as elements in the primitive def with the special token "# BEL" included after the name of the BEL. The BEL pins and their connections are defined within the element.

3.2.4 Site Pips (Routing Muxes)

```
  (element CEINV 3
    (pin OUT output)
    (pin CE input)
    (pin CE_B input)
    (cfg CE CE_B)
    (conn CEINV OUT ==> BUFHCE CE)
    (conn CEINV CE <== CE CE)
    (conn CEINV CE_B <== CE CE)
  )
```

Site Pips (or routing muxes) are defined exactly the same way as BELs, but they **do not** have the "# BEL" token after the name. Elements that do not have the BEL token, and do not have a corresponding primitive def pin with the same name are distinguished as routing muxes.

3.2.5 Configuration Options

```
  (element CE_TYPE 0
    (cfg SYNC ASYNC)
  )
```

Site-level and BEL-level configuration parameters can also be included in the the primitive def file. They are also represented as elements, but they only have a single "cfg" child, which defines the possible values of the configurable parameter. These elements are generally not important for VSRT, and can be safely ignored.

3.2.6 Routethroughs

```
  (element _ROUTETHROUGH-I-O 2
    (pin I input)
    (pin O output)
  )
```

Routethrough elements in the primitive def represent a configurable connection between an input site pin and an output site pin. These connections can be used in external tools instead of trying to configure the necessary routing muxes of the site to create the routethrough. For VSRT, these elements can be ignored because they can be automatically generated from Vivado's Tcl interface.

3.3 VSRT GUI

Figure 2 shows the terminology used to describe the VSRT GUI. This terminology will be used through the the rest of the document, so it is important to briefly review the figure before continuing.

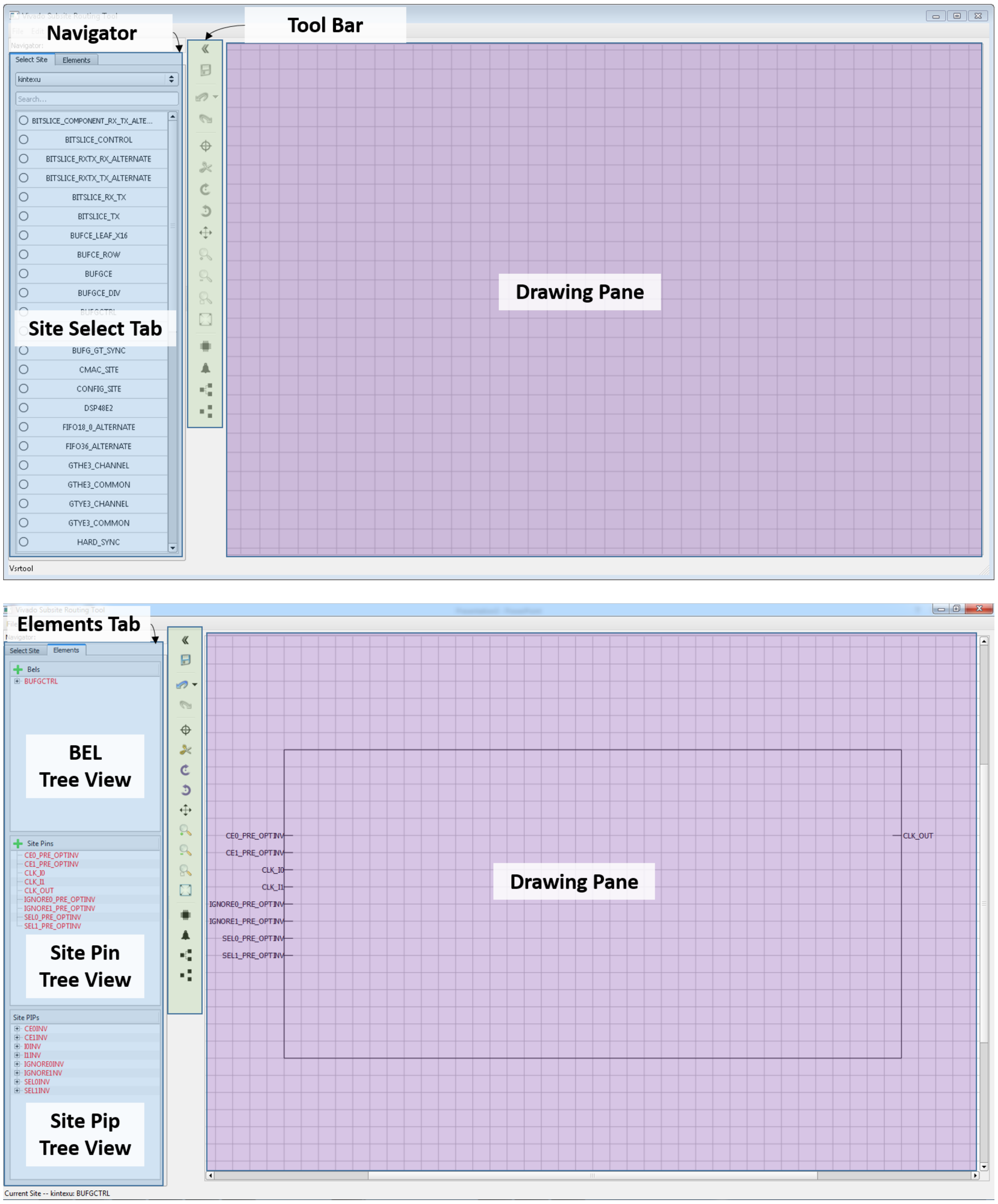


Figure 2: Parts of the VSRT GUI. (Top) Start view. (Bottom) Primitive site view

4 User Guide

Before running VSRT, you will need to generate the partial primitive definitions from Vivado. To do this, open a new terminal and start Vivado in "Tcl" mode. Make sure the Vivado executable is on your PATH.

```
[ttown523@CB461-EE09968:~] vivado -mode tcl

****** Vivado v2014.2 (64-bit)
  **** SW Build 928826 on Thu Jun 5 17:55:10 MDT 2014
  **** IP Build 924643 on Fri May 30 09:20:16 MDT 2014
    ** Copyright 1986-2014 Xilinx, Inc. All Rights Reserved.

Vivado%
```

Next, run the `Tincr` command *get_all_partial_primitive_defs* in the Vivado prompt. The first argument to this command is the directory to store the partial primitive defs (if it does not yet exist, it will be created). The second (optional) argument is the architecture to generate primitive defs for. Possible architectures for Vivado 2016.2 are: `artix7`, `kintex7`, `virtex7`, `zynq`, `kintexu`, `virtexu`, `kintexuplus`, and `virtexuplus`. If no architecture is specified, then the primitive defs for all architectures will be generated. An example usage of this command with sample output is shown below. Replace "primitiveDefs/" with the directory of your choice and "kintexu" with the architecture of your choice. **NOTE**: This command may take a few hours to run if you are generating partial primitive defs for all architectures.

```
Vivado% tincr::extract_all_partial_primitive_defs primitiveDefs/ kintexu
...
Extracting Primitive Sites from part xqku060-rfa1156-1-i (28 out of 31)...
Extracting Primitive Sites from part xqku095-rfa1156-1-i (29 out of 31)...
Extracting Primitive Sites from part xqku115-rld1517-1-i (30 out of 31)...
Extracting Primitive Sites from part xqku115-rlf1924-1-i (31 out of 31)...
Successfully created all .def files in directory primitiveDefs/
Vivado%
```

Once the command is finished running, navigate to the generated partial primitive def directory and view the newly created files. If you generated the primitive defs for all architectures, it will look something like the directory structure shown below. There is a folder for each architecture that holds all of the partial primitive defs for that architecture.

```
[ttown523@CB461-EE09968:primitiveDefs] ls
artix7 kintex7 kintexu kintexuplus log.txt virtex7 virtexu virtexuplus zynq
```

Also included in the directory is a *log.txt* file. An example log file is shown below.

```
Extracting Primitive Sites from part xcku025-ffva1156-1-c (1 out of 31)...
  kintexu-SLICEL -> SLICE_X0Y179...
  kintexu-SLICEM -> SLICE_X1Y179...
  kintexu-RAMB181 -> RAMB18_X0Y71...
  ...
  ALTERNATE: kintexu-FIFO18_0 -> RAMB18_X1Y70...
  ALTERNATE: kintexu-RAMB180 -> RAMB18_X2Y70...
```

This file is important to VSRT users because it gives a mapping between a site type, and a specific instance of that site type. This makes it easy to bring up a site in the Vivado device browser when drawing the corresponding connections in VSRT. The *log.txt* file also shows what site types are *alternate-only* sites (they include the keyword ALTERNATE). For these sites, an additional step must be taken in Vivado to show the correct connections. The Tcl commands in Listing 1 can be used to visualize an instance of any site in Vivado's device browser.

Listing 1: Tcl commands for viewing a site in Vivado's device browser

```
// open the part and start the device browser (only needs to happen once)
```

```
Vivado% link_design -part xcku025-ffva1156-1-c
Vivado% start_gui

// gain a handle to the site, and zoom to its view
Vivado% set site [get_sites RAMB18_X1Y70]
Vivado% select $site (you may have to press F9 to zoom to the site)

// set the site type (FOR ALTERNATE-ONLY SITES!)
Vivado% set_property MANUAL_ROUTING FIFO18_0 $site
```

Now that the partial primitive defs directory has been created, you are ready to use VSRT. To run VSRT, simply execute the Java class `edu.byu.ece.rapidSmith.device.vsrt.gui.VSRTool` found in the RapidSmith2 repository. VSRT accepts a single command line argument: the partial primitive def directory generated above. You can choose to run the tool on the command line, or in an IDE (which is recommended). Figure 3 shows the VSRT GUI on startup if everything was setup correctly.

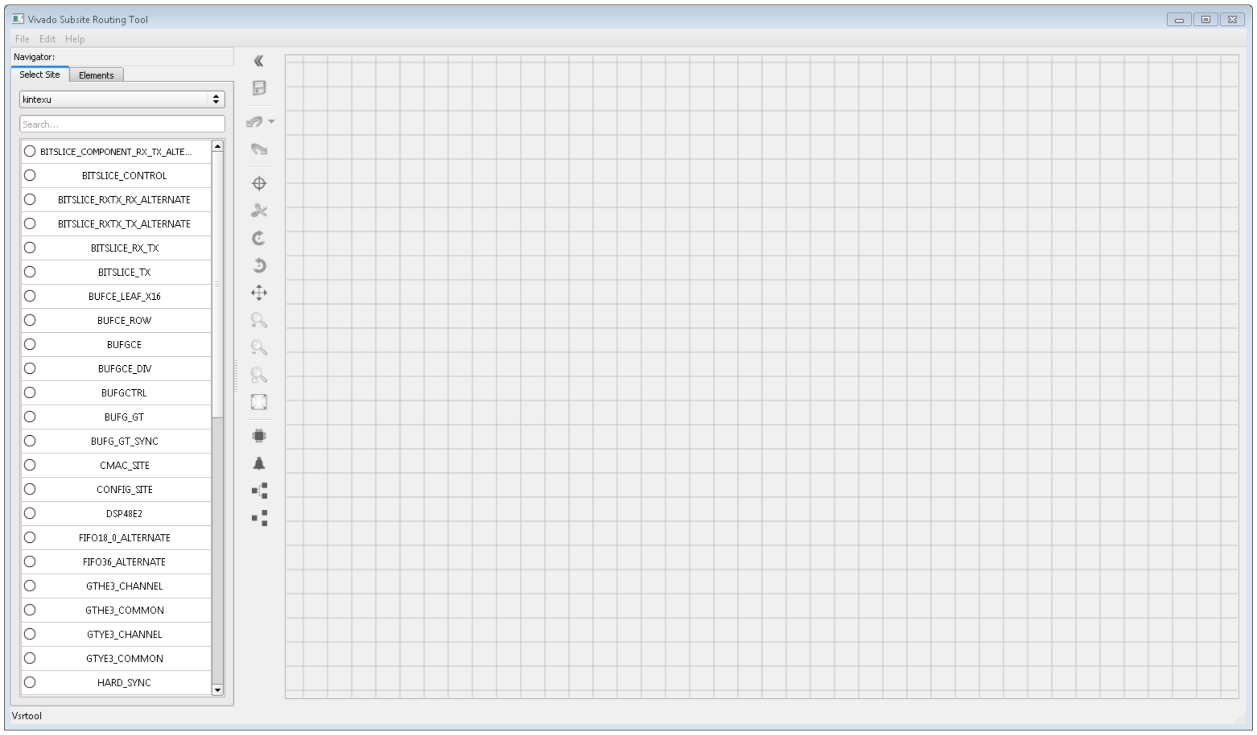


Figure 3: VSRT GUI start screen

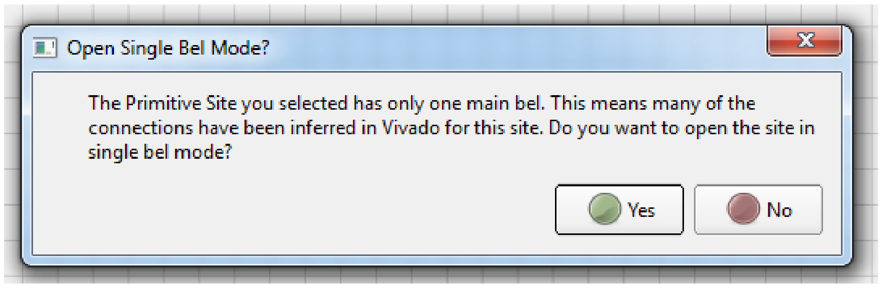
The panel on the left has a drop-down box showing the available architectures, and the table view lists all primitive sites in that architecture. VSRT can be run in two different modes: regular mode and single bel mode. The remainder of this section gives a walkthrough of how to use both modes.

4.1 Regular Mode

Regular mode is the default mode for VSRT, and is intended for two use cases in particular:

1. Sites where connections cannot be automatically inferred in Tcl (see subsection 4.2 for these sites).
2. Small sites that can be wired up very quickly (even if some connections can be automatically generated.)

In this mode, the user is required to create all of the connections of the primitive site by manually drawing wires between components. This is best shown through an example. In the VSRT GUI start screen, select the "kintexu" architecture in the dropdown box and find the BUFGCTRL site in the table. Double click the site, and you should see the following prompt:



The Primitive Site you selected has only one main bel. This means many of the connections have been inferred in Vivado for this site. Do you want to open the site in single bel mode?

Figure 4: User mode prompt

Since we want to run the tool in regular mode, select "No" and the BUFGCTRL site will load in VSRT.

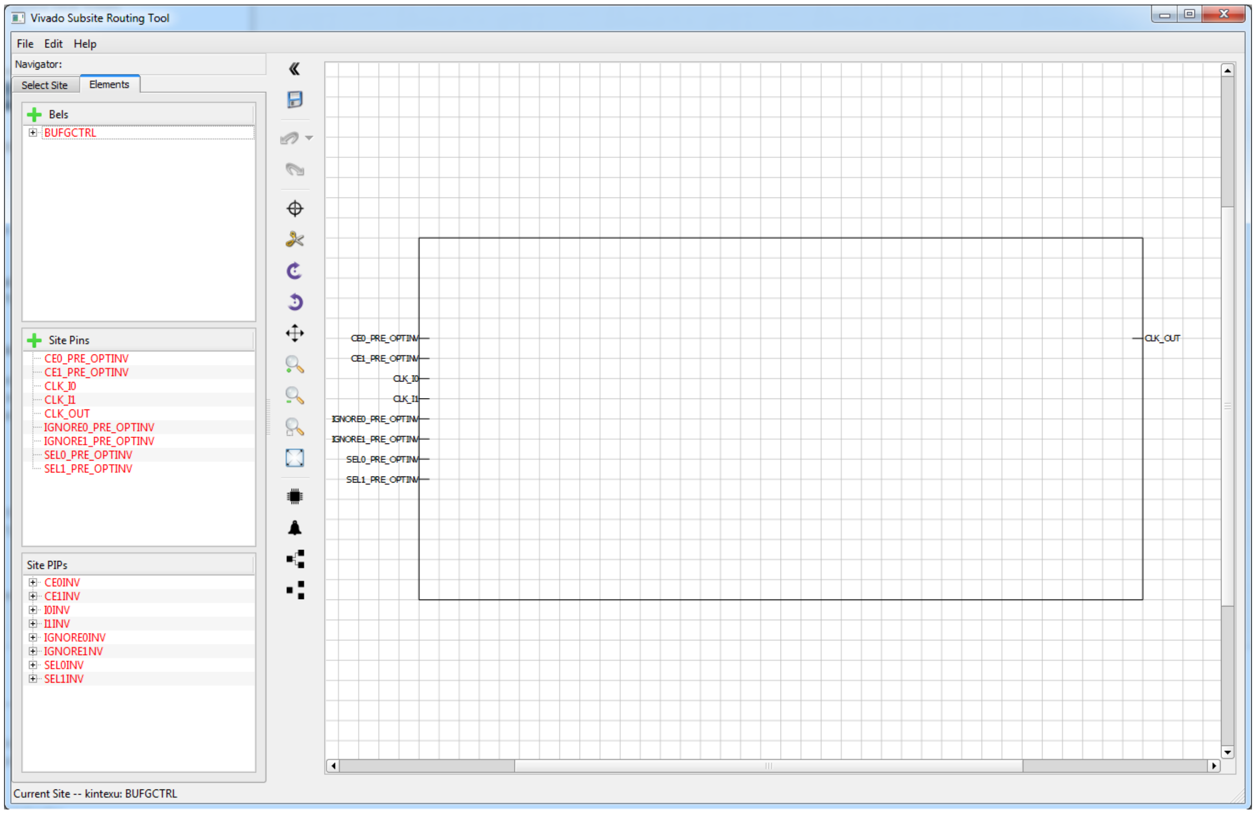


Figure 5: BUFGCTRL site in VSRT

As the screenshot above shows, all of the BELs, site pins, and site pips (routing muxes) of the site are displayed in the "Elements" tab on the left. The red font color indicates that at least one pin attached to the element is unconnected. The drawing pane (gridded area) is initialized to include all of the site pins attached to the site, and a rectangle representing the site's boundaries. In this case, there is 8 input site pins and 1 output site pin. To add elements to the drawing pane, simply double click the item in the Elements tab. The item will appear in the top-left corner of the drawing pane, where you can then drag it to the location of your choice (shown in Figure 6).

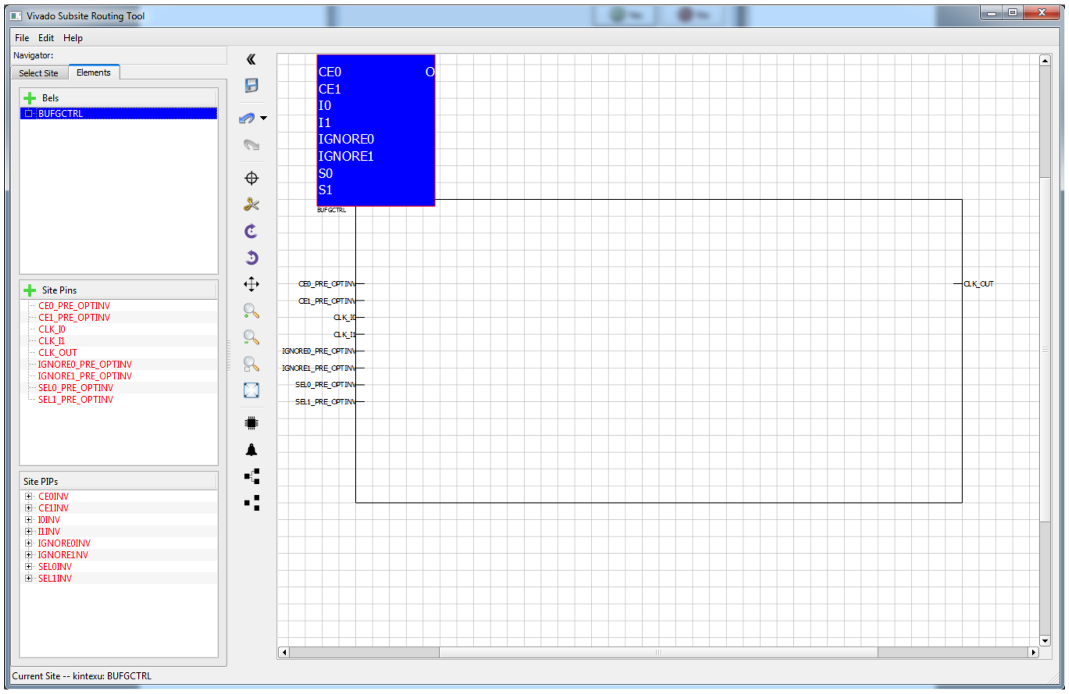


Figure 6: Adding a BEL to the drawing pane

To create the internal site connections, all you have to do is add elements to the drawing pane, and connect elements by drawing wires between their pins. To add a new wire, you can either press "w" (recommended), or click the draw wire button [ ] on the tool bar. This will replace the cursor with crosshairs and put the tool in wire drawing mode.

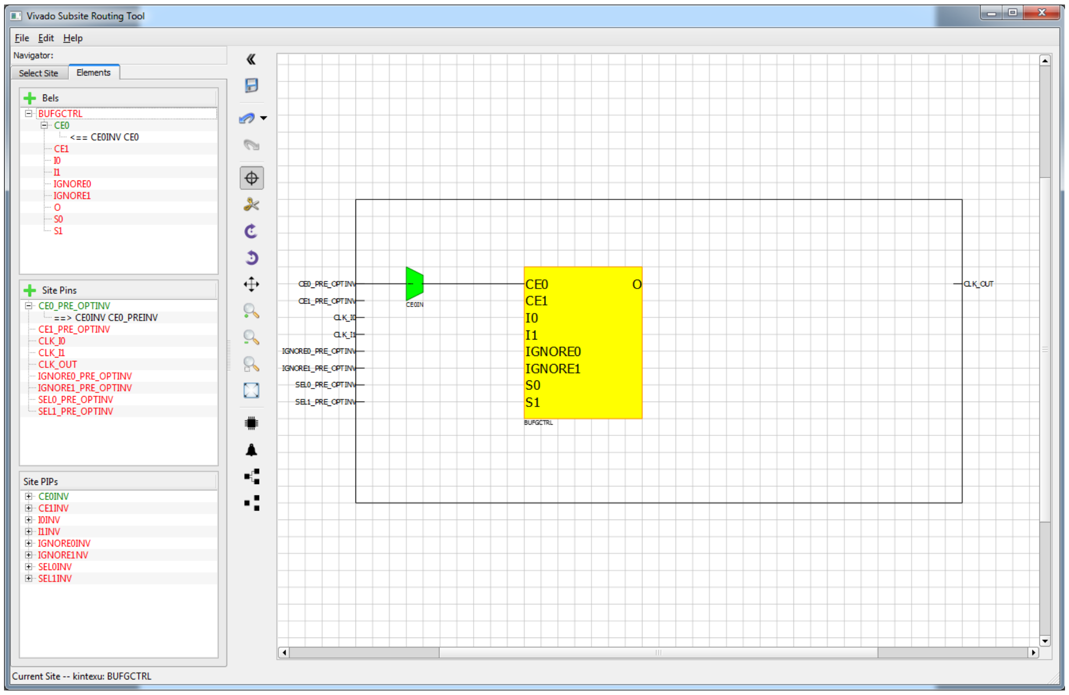


Figure 7: Wire connections in VSRT

As shown in Figure 7, several things happen when two pins are connected with a wire. First, the tree view of the pin is updated to show the connection using the same format as it will appear in the primitive def file. Second, the pin in the element tab changes color from red to green (which signifies a connection has been made to the pin). A BEL or site pip element will turn green once all of its pins have been connected. Third, the outline color of the shapes in the drawing pane will change. A green outline represents a fully connected element, orange represents an element that is partially connected, and red represents an element that is completely unconnected. This color coding makes it easier to determine what remaining connections need to be generated.

Now that you know that basics of creating connections in VSRT, let's go back to the BUFGCTRL example. You will first need to open an instance of a BUFGCTRL site in Vivado's device browser using the Tcl commands shown in Listing 1 (replace RAMB18_X1Y70 with the appropriate value found in the *log.txt* file for BUFGCTRL sites). After running the commands, the Vivado device browser should now display a BUFGCTRL site as shown in Figure 8.

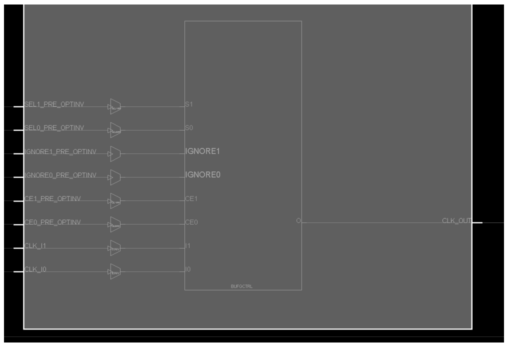


Figure 8: BUFGCTRL site in Vivado's device browser

The site in the device browser is interactive, so you can click-on or hover-over elements to view their names. The next step is to recreate the connections you see in Figure 8 in the VSRT GUI. As described above, you can do this by adding elements to the drawing pane, rearranging the elements, and drawing wires between element pins. Once you have generated all of the connections in VSRT for the BUFGCTRL site, it should look similar to Figure 9.

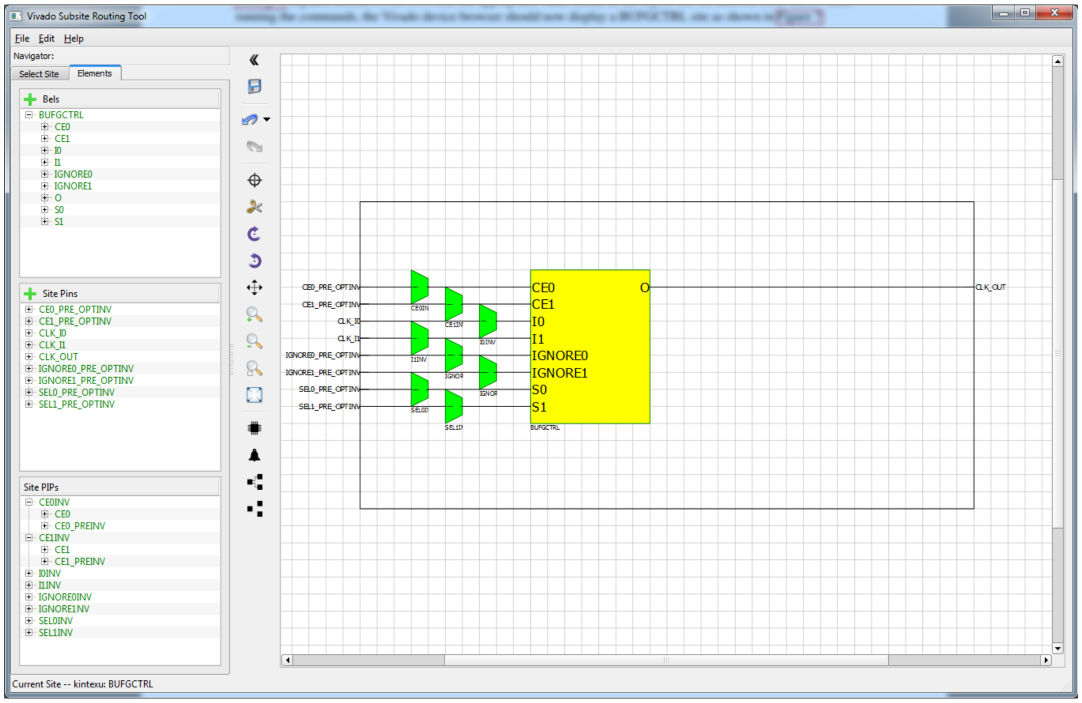


Figure 9: Completed BUFGCTRL in VSRT

Notice that all items in the "Elements" tab are green, meaning that every pin has at least one connection. This is a good indicator that we are done wiring up the BUFGCTRL site. To generate the completed primitive def file, click File → Save, and the prompt shown in Figure 10 will appear.

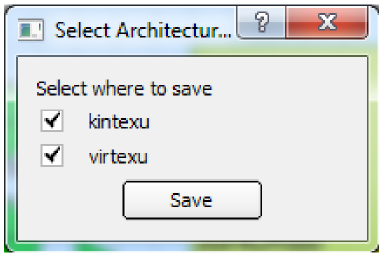


Figure 10: VSRT save prompt

The prompt will display all available architectures to the tool (in this example, there is only two). Since we know that the BUFGCTRL site for `kintexu` and `virtexu` architectures are identical, we can save the completed primitive def to both of those architectures. Click "Save" and navigate back to the partial primitive defs directory that was created from Tincr. You will see a new folder has been created called "CompletePrimitiveDefinitions".

```
[ttown523@CB461-EE09968:primitiveDefs] ls
artix7     CompletePrimitiveDefinitions     kintex7   kintexu      kintexuplus
log.txt    virtex7                          virtexu   virtexuplus  zynq
```

All completed primitive definitions will appear in this folder (separated by architecture sub-folders). After saving the primitive def, you will be returned to the VSRT start screen. The final step in completing the primitive site is to right click on the BUFGCTRL site you just completed, and tag it as complete.

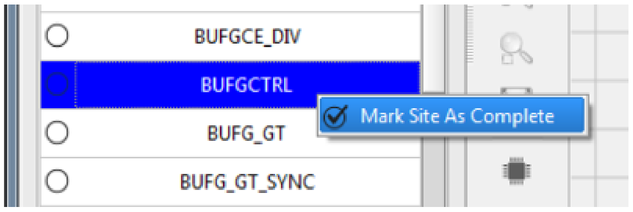


Figure 11: How to tag a site as complete in VSRT

4.2 Single-Bel Mode

For primitive sites with a single BEL (or one large BEL that takes up most of the site), many of the connections can be automatically inferred in Tcl. However, not all connections can be generated and some may be generated incorrectly. VSRT offers a "Single-Bel Mode" for these sites so that instead of creating every single connection by hand (which could be thousands of connections for large sites), only the connections that Tincr could not infer need to be generated. This is a *significant* time-saver for large primitive sites.

The biggest difference between single-bel mode and regular mode, is that BEL pins can be individually added to the drawing pane instead of an entire BEL (BELs can still be added if desired). Users can then add the BEL pins that are unconnected, and generate connections for those pins only. To show an example of single-bel mode, let's generate connections for a GTHE3_CHANNEL site. Double-click the GTHE3_CHANNEL site in the "Select Site" tab, and click "Yes" in the popup prompt (Figure 4) that appears. Figure 12 shows the initial view of the GTHE3_CHANNEL site in VSRT.

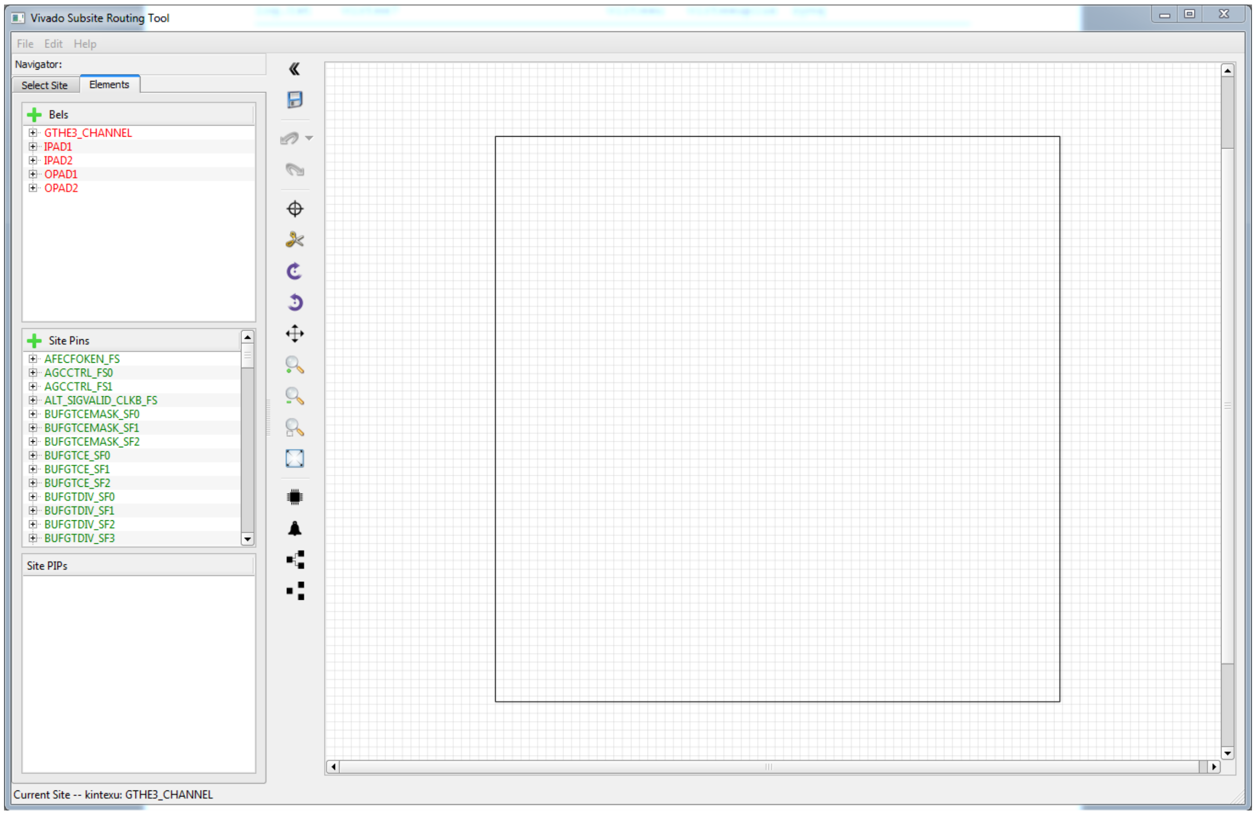


Figure 12: GTHE3_CHANNEL in VSRT

There are a few important things to note about sites that have been loaded in single-bel mode:

1. Site pins are not, by default, added to the drawing pane. Instead, only a generic site boundary is shown. Site pins can be added to the drawing pane by double clicking the site pin in the "Elements" tab. Bel pins can also be added to the drawing pane by double-clicking the corresponding bel-pin.

2. Several site and bel pins show as green in the "Elements" tab. This indicates that these pins have connections that were automatically generated in Tcl, and usually don't have to be modified in VSRT.

3. There is more than one BEL in this site, which is normal. The "GTHE3_CHANNEL" BEL is the one where most connections have been automatically generated. The other BELs can be added to the scene like they would in regular mode, where their connections can be generated.

4. If a BEL or pin is added to the drawing pane, any existing connections **will be removed**. This is because the tool will assume that the automatically created connections for those elements are incorrect (why else would the user add the elements).

To generate the remaining connections, simply add the unconnected pins to the drawing pane and wire them together (based on the equivalent Vivado GTHE3_CHANNEL image which is not shown in this document). The following Tcl commands may be useful to isolate a specific bel-pin in the device browser.

Listing 2: Tcl command to select a bel pin in Vivado's device browser

```
// First click on the Bel you want in the device browser!
Vivado% set bel [get_selected]
Vivado% select_objects [get_bel_pins $bel/GTHRXN]
```

Once you have connected all of the unconnected bel-pins, the drawing pane should look similar to Figure 13.

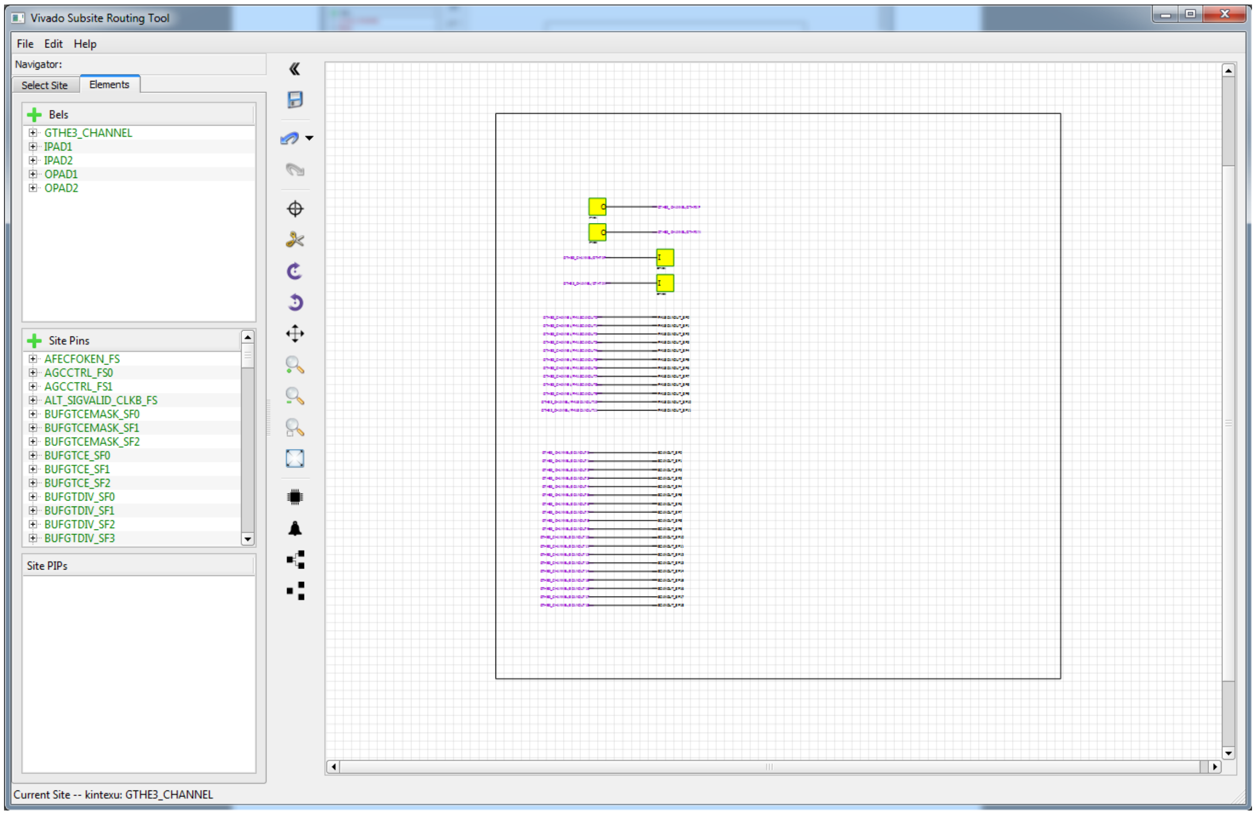


Figure 13: Complete GTHE3_CHANNEL in VSRT

Of the original 959 bel-pins for the GTHE3_CHANNEL site, only 35 of them (3.65%) need to be connected using VSRT. This demonstrates the productivity advantages of using single-bel mode. It is *strongly encouraged* for VSRT users to run the tool in single-bel mode for large sites where applicable.

We are not quite done with the GTHE3_CHANNEL site however. As you scroll through the site pins in the "Elements" tab, you will see several pins that are still unconnected. This may seem like a problem, but it is not. If you view the offending pins in Vivado's device browser, you will see that they do not connect to any elements inside of the site. Therefore, the final step to complete the GTHE3_CHANNEL site in VSRT is to mark the remaining site pins as unconnected. To do this, right click the unconnected site pins and select "Mark As Unconnected" (Figure 14). The pin colors will change to gray indicating that they no longer need to be connected. After all unconnected pins are marked as unconnected, the complete primitive definition can be generated using the steps introduced in subsection 4.1.

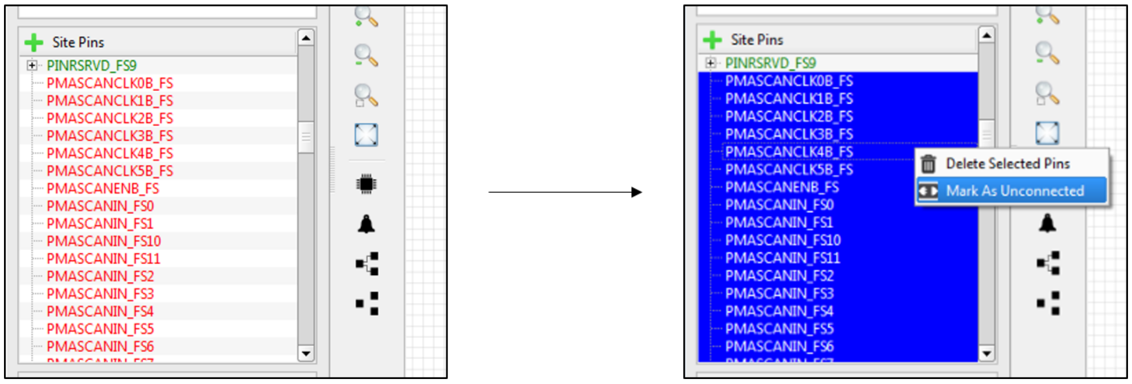


Figure 14: How to mark site pins as unconnected in VSRT

5 Features

5.1 Wire Hiding

The current version of VSRT does not support drawing wires along the grid system. Wires always appear as straight lines between two connected pins. For complex primitive sites (such as SLICEL or DSP48E2), this can quickly clutter up the drawing pane and make it difficult to see what pins are connected to where. To alleviate this, VSRT gives the users the option to hide wires. The toolbar has two buttons for wire hiding:

- Hide *all* wires in the current drawing pane
- Show *all* wires in the current drawing pane

You can also hide and show wires for individual BELs and Site Pips. To do this, right-click on the element in the drawing pane and select "Hide/Show Wires".

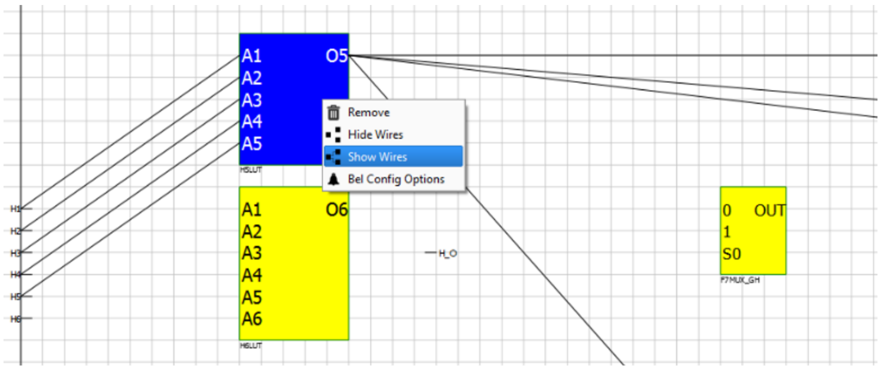


Figure 15: Popup menu for BELs/Site Pips showing wiring hiding options

The suggested flow for VSRT users is to draw the connections for one BEL at time, while hiding all other wires. Even if an element's wires are hidden, the border color around the shape in the drawing pane indicates how many pins of the element have been connected (see next section).

5.2 Shape Border Colors

Each element in the drawing pane has a border color based on how many of its pins are connected. Figure 16 shows the possible colors. Green means all of the pins have been connected, orange means some (but not all) pins have been connected, and red means no pins have been connected. This color coding makes it easier to identify which elements in the drawing pane have unconnected pins when wire hiding is enabled.

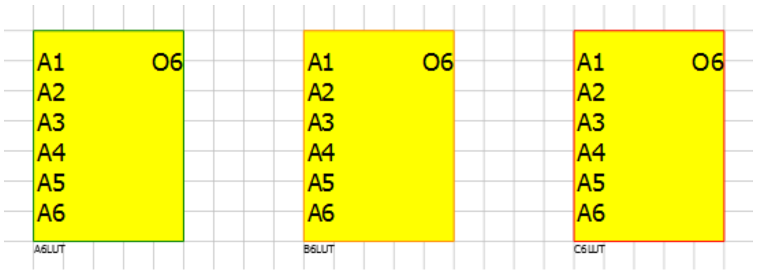
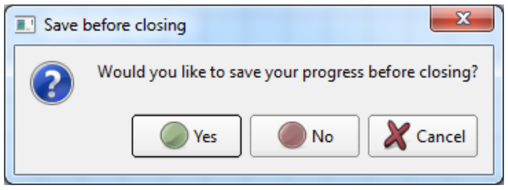


Figure 16: Border colors for shapes on the drawing pane

5.3 Saving

VSRT supports saving and restoring the progress of a primitive site. To save your progress, click File→Close Site. The prompt shown in Figure 17 will appear. Click “Yes” and your site progress will be saved to an XML file. To restore the site progress and continue working, simply open the site again. A site is automatically saved when a complete primitive definition is created. **NOTE:** If you save a site open in single-bel mode, the connections will be saved but the drawing pane will not.


Save before closing

Would you like to save your progress before closing?

Figure 17: Save progress prompt

5.4 INOUT Pins

Bel-pins in Xilinx FPGAs have three possible directions: INPUT, OUTPUT, and INOUT. INPUT pins are signal sinks, OUTPUT pins are signal sources, and INOUT pins are bidirectional, meaning the pin can act as either a source or sink. Because of this behavior INOUT pins can connect to pins of any direction. Table 1 shows the connections that should be created when an INOUT pin is connected to another pin.

Pin 1 Type	**Pin 2 Type**	**Connection(s) Generated**
INOUT	INPUT	Pin1 → Pin2
INOUT	OUTPUT	Pin1 ← Pin2
INOUT	INOUT	Pin1 ← Pin2 Pin1 → Pin2

Table 1: INOUT pin connections

VSRT automatically generates the correct connections for INOUT pins according to the table. Also, VSRT displays the color of INOUT pins as gray instead of black so they can be easily identified in the drawing pane.

5.5 Undo/Redo Functionality

Most actions in VSRT can be undone. So, if you make a mistake don’t worry! Click “Undo” and continue working. Two buttons in the tool bar allow actions to be undone or redone (shortcut keys are shown in parenthesis):

Undo the last action (Ctrl+Z)

Redo the last action (Ctrl+Y)

The “Undo” button on the tool bar also has a dropdown menu to undo multiple actions at once.

5.6 Deleting Elements

You can remove items from the drawing pane by entering “delete mode.” To toggle delete mode, click the toolbar button shown below or click the hotkey shortcut “d.”

Enter delete mode (d)

The cursor will now have a small red “x” in the bottom right-hand corner to signify that you can remove elements. To remove a set of elements, click on the drawing pane and draw a bounding box around the elements you want to delete. All elements within the bounding box will be deleted (including wires).

5.7 Adding VCC/GND BELs

In some device architectures, the Tcl API into Vivado *cannot* access VCC or GND BELs (i.e. the Tcl function *get_bels* will omit VCC and GND). An example GND BEL is shown in Figure 18.

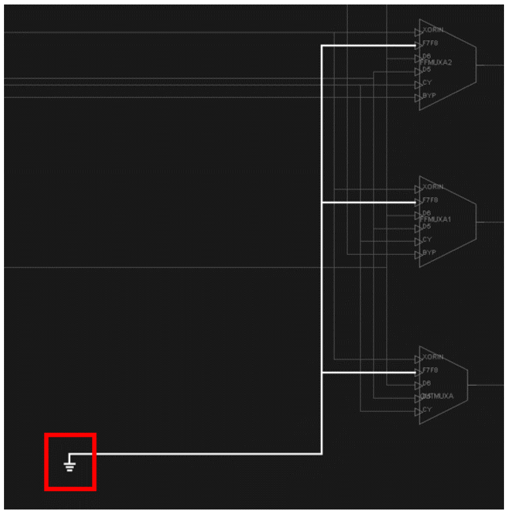


Figure 18: Vivado GND BEL that connects to other BELs

This indicates that some partial primitive defs generated from Vivado are missing required BEL elements to generate all connections. VSRT allows users to add these missing VCC and GND BELs to a site by clicking the add icon next to the "Bels" header. Figure 19 demonstrates the complete process.

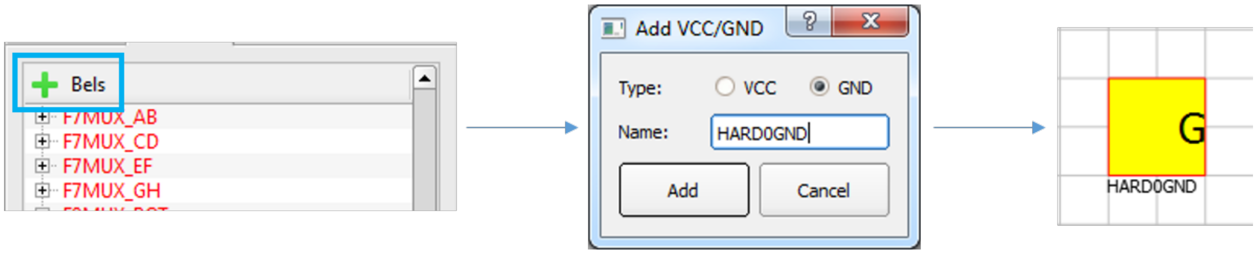


Figure 19: How to add new VCC/GND BELs to a site

5.8 Manipulating Site Pins

As you are using VSRT to create complete primitive definitions, you may notice three issues with site pins:

1. Site Pins that appear in Vivado, but not VSRT
2. Site Pins that appear in VSRT, but not Vivado.
3. Unconnected Site Pins

VSRT provides solutions to handle each of these issues. For site pins that appear in Vivado but not VSRT (1), they can be added by clicking the add icon next to the "Site Pins" header. Figure 20 demonstrates the complete flow. As the figure shows, after you click the add icon a popup window appears. Enter the pin name, number of pins to create, and the pin direction in the window and click "Add." The pins will then be created and added to the scene in the top-left corner of the drawing pane. The "Count" field is useful for creating many site pins with the same name. For example, if the pin name entered was "Test" and the count value entered was "5", the pins Test0, Test1, Test2, Test3, and Test4 will be created. **NOTE:** To create a single site pin with no number on the end of the name, simply set the count field to 1.

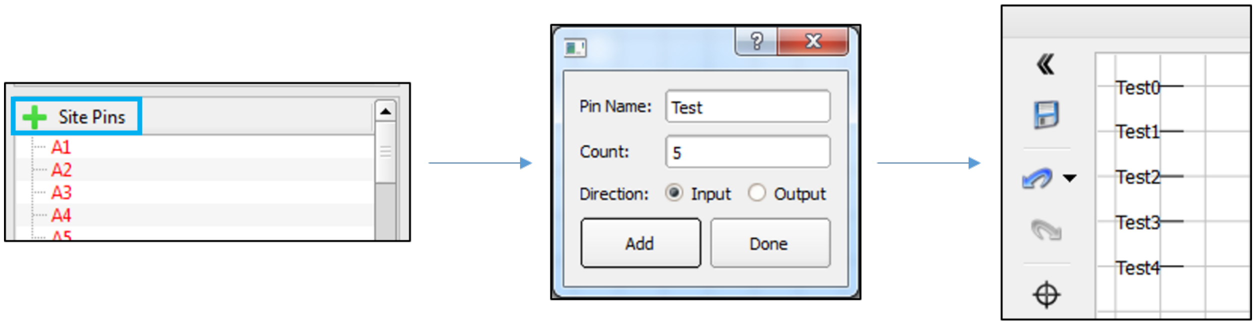


Figure 20: How to add new site pins

For site pins that appear in VSRT but not Vivado (2), you can simply remove them. To do this, right-click the site pins you want to remove in the "Site Pins" section of the "Elements" tab and select "Delete Selected Pins." For site pins that are unconnected (3), you can mark them as unconnected by using the same steps but selecting "Mark As Unconnected" instead. Figure 21 shows what these options look like in the GUI. **Do NOT delete unconnected site pins! They still have external connections in the Tile, so need to be included in the final primitive def file**.

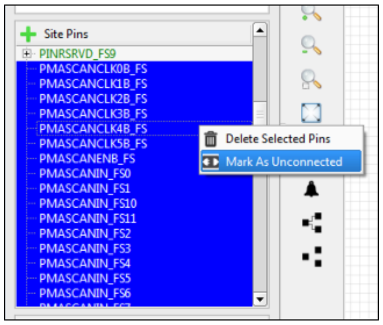
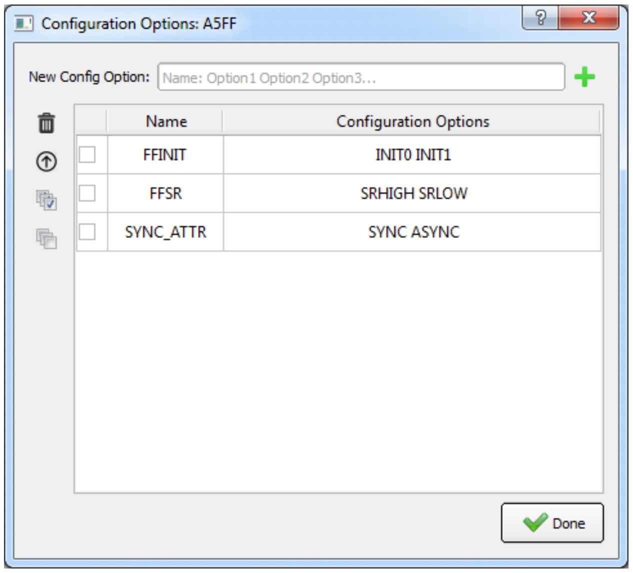


Figure 21: How to mark site pins as unconnected or remove them in VSRT

5.9 Configuration Options

If you generated the partial primitive defs from Vivado with the option "includeConfigs" enabled, then the primitive defs will include configurable properties for each BEL. To view a BELs configurable properties in VSRT, right-click the BEL in the drawing pane and select "Bel Config Options." Alternatively, you can select the BEL in the drawing pane and click [ ] on the tool bar. Figure 22 shows the popup window that is displayed when you view a BEL's configuration options for a SLICEL A5FF. As the figure shows, the window allows you to delete configurations [ ], add new configurations [ ], and promote configurations to the site-level [ ]. Adding and deleting configurations is fairly straightforward, but what does it mean to promote configurations to the site-level? The "FFSR" property shown in Figure 22 is a good example of why promoting properties is required. FFSR determines if the reset on the corresponding Flip Flop should be high or low-asserted. However, every Flip Flop in a SLICEL **must have the same FFSR property**. This means that FFSR is really a property on the site, and not on the individual BEL. VSRT allows you to "promote" these types of properties to the site-level. Specifically, the promoted property will be removed from all BELs and added to the site's configuration properties instead. A site's configuration properties can be viewed by pressing [ ] on the tool bar.



New Config Option: Name: Option1 Option2 Option3...

	Name	Configuration Options
☐	FFINIT	INIT0 INIT1
☐	FFSR	SRHIGH SRLOW
☐	SYNC_ATTR	SYNC ASYNC

Figure 22: BEL configuration popup window

5.10 Moving Shapes

All shapes in the drawing pane can be moved to another location (even site pins) by dragging and dropping them. Multiple shapes can be moved at a time by selecting multiple items at once. Also, the site boundary rectangle can be resized if the initial site estimate was too large or small.

5.11 Tool Bar Buttons and Shortcuts

This section contains a description of all of the options on the VSRT tool bar, and keyboard shortcuts if they exist.

Hide the navigator pane

Show the navigator pane

Write the completed primitive definition (Ctrl + S)

Undo the last action (Ctrl + Z)

Redo the last action (Ctrl + Y)

Toggle wire drawing mode (w)

Toggle delete mode (d)

Rotate selected elements clockwise (Ctrl + R)

Rotate selected elements counterclockwise (Ctrl + Shift + R)

Drag View (Ctrl + D)

Zoom in (Ctrl + mouse forward scroll)

Zoom out (Ctrl + mouse backward scroll)

Zoom to selection

Zoom Best Fit: Set the zoom level on the drawing pane to show the entire primitive site.

Show site configuration properties

Show BEL configuration properties for selected BEL.

Show all wires on the drawing pane

Hide all wires on the drawing pane

6 Known Issues

This section contains a list of known issues with both Vivado 2016.2 and VSRT when trying to generate complete primitive definitions. If you find any other bugs with VSRT, please create a new issue at `https://github.com/byuccl/RapidSmith2` to report it.

- When moving multiple site pins at once, occasionally not all site pins will snap to the grid (see Figure 23). To fix this, simply select all of the site pins that are not on the grid, and drag-and-drop them in place. This should snap the pins to the grid and upate their locations.

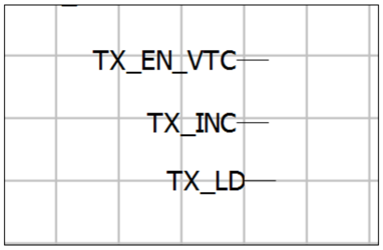


Figure 23: Site Pin moving bug

- Sometimes the VSRT drawing pane gets stuck in zooming mode so that the mouse scroll wheel will zoom in and out when "Ctrl" is not pressed. To fix this press "Ctrl", zoom in and out slightly (by scrolling the mouse wheel), and release "Ctrl." The scroll wheel should work normally after doing this.
- **Do not select a group of items while scrolling the drawing pane**. The items may end up in the top-left corner of the drawing pane (or some other random location) where they can be hard to find. Instead of scrolling while selecting items, zoom out enough so you can see all the items you want to move, and then select all of the items at once.
- When you resize the left side of a primitive site on the drawing pane, all input site pins will move as well. Similarly, when you resize the right side of a primitive site, all output site pins will move. This can be useful if you haven't moved any site pins yet, but otherwise it can disturb the current state of the drawing pane. **Best practice is to resize the site ONCE before you start working on a primitive site, and then do not resize it again**.
- There are some site pins in Vivado whose names do not show up in the device browser (see Figure 24). To determine the names of these pins, click on the **Tile Wire** connected to the pin (highlighted as white in the figure) and run the following Tcl command:

```
get_site_pins -of [get_selected]
```

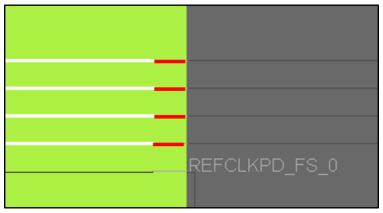


Figure 24: Missing site pin names in Vivado (shown in red)

- In the Vivado device browser, there appears to be some wires that can have multiple drivers but are not connected to any bidirectional pins. Figure 25 shows an example for an UltraScale BITSLICE_COMPONENT_RX_TX site. This is most likely not what the site physically looks like (they are leaving out a tri-state buffer). In VSRT **do not connect the two output pins together**. Only connect output pins to input pins and vice versa.

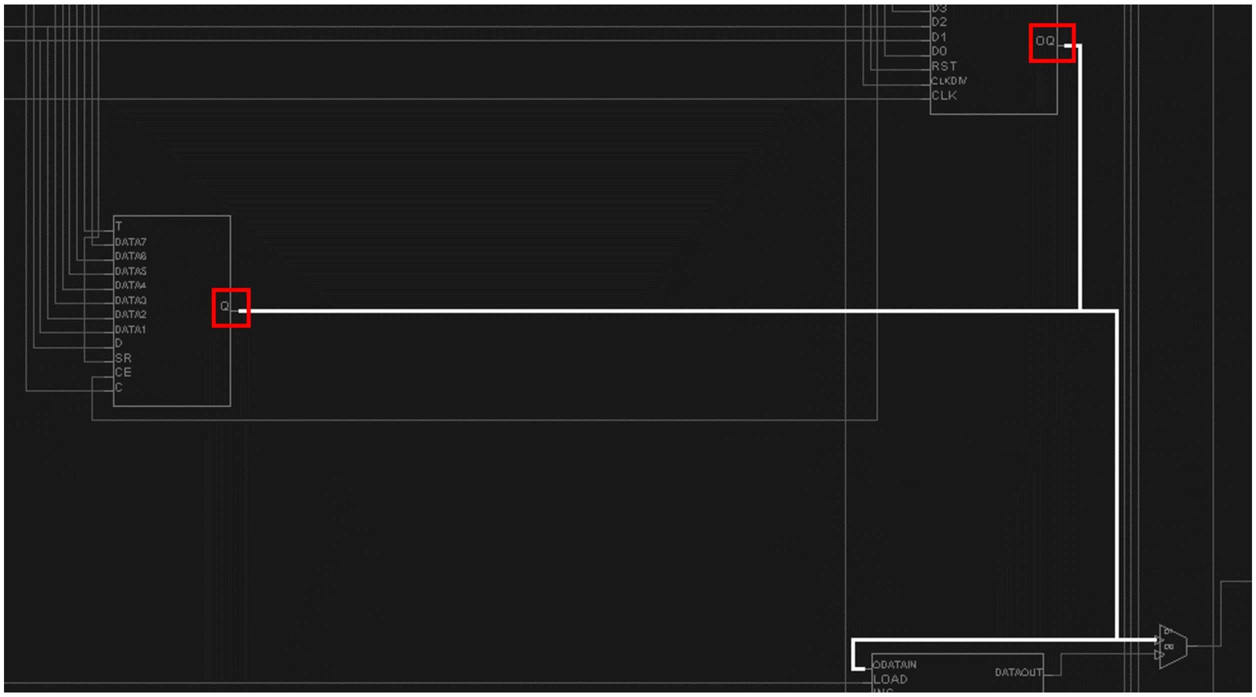


Figure 25: Example of multiple drivers on a site wire