6Peter the Great St. Petersburg Polytechnic University
Institute of Computer Science and Cybersecurity
Graduate School of Computer Technologies and Information Systems

Lecture: Creating a System Design with Platform Designer

Subject: Automation of discrete device design (in English)

Completed by student of group 5130901/10101 _____ Nepomnyaschiy M.T.
(signature)

Lecturer _____ Antonov A.P.
(signature)

Saint Petersburg
2024

Оглавление

List of illustrations

Creating a System Design with Platform Designer

Introduction to SystemVerilog Complexity

This lecture provides information about the usage of Platform Designer user interface to create a system design for implementation in an Intel FPGA device. Also, here explains how Platform Designer fits into the Intel Quartus Prime software design flow as another method of design entry.

4. Part 4 – Connection Details

In this part some information on establishing system connections through various interface types is provided. It covers the methods for making these connections, issues related to unconnected interfaces, and details about specific connections such as clocks, resets, conduits, and memory-mapped interface addressing.

4.1. Clocks

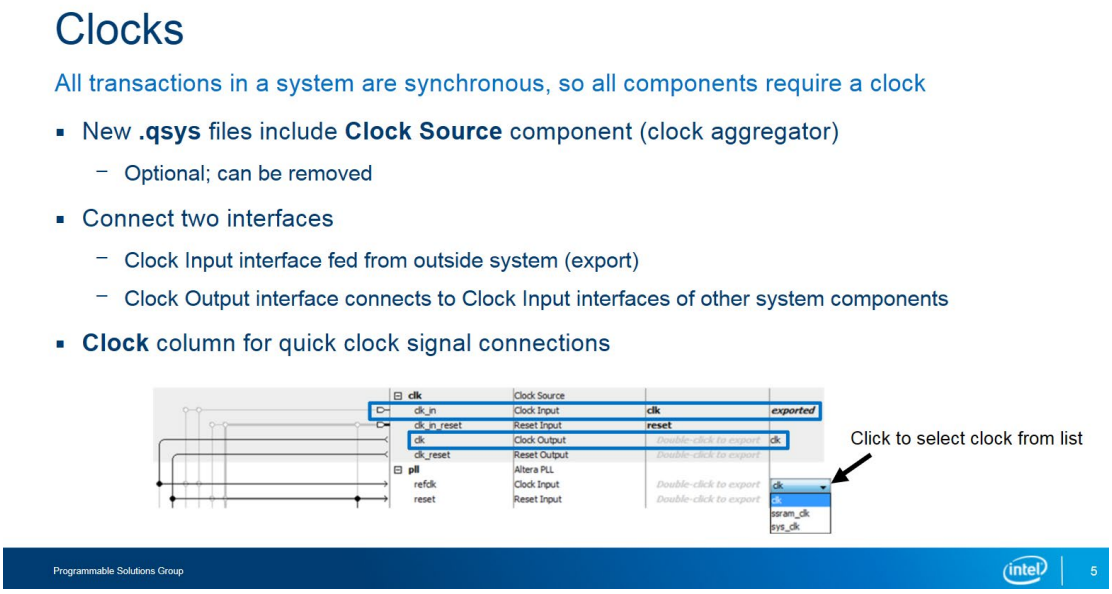

Figure 1 – Application of clock

In a system, each component needs a clock for synchronized operations. To streamline this process, we use a Clock Source component, which acts as a hub for distributing a single external clock to all components, eliminating the need to individually connect each one to an external source. You can customize the Clock Source by specifying the clock frequency for timing considerations. If your system has multiple incoming clocks, you can add more Clock Source components. However, if a component already directly connects to an external clock, you can skip using the Clock Source. Connecting clocks can be done through various strategies or by selecting options from a drop-down list in the System Contents tab for efficiency. This approach simplifies the management of clock connections in the system.

4.2. Clock Domains and Domain Filtering

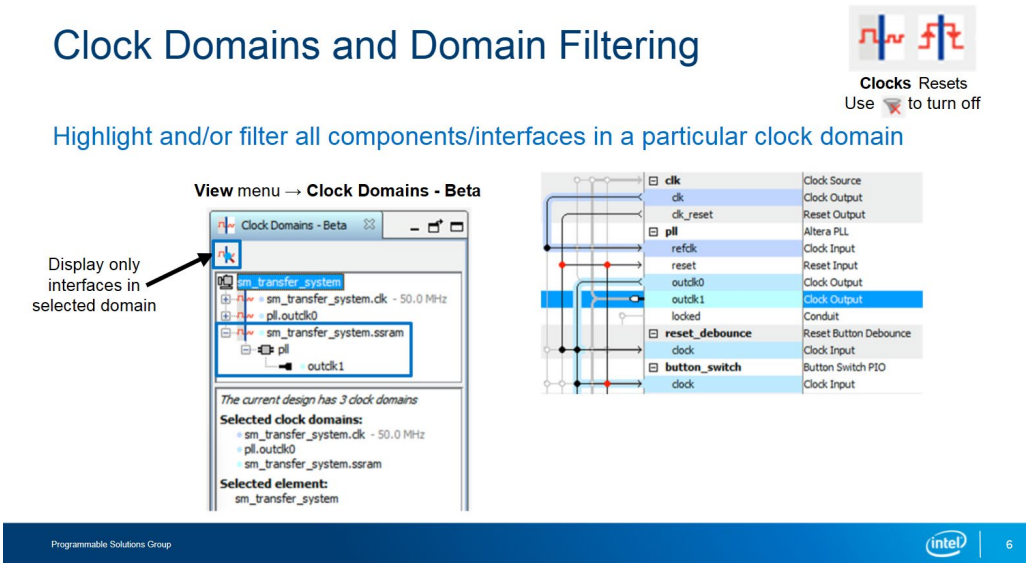

Figure 2 – Clock Domains and Domain Filtering

To manage different clock domains in your system, access the Clock Domains tab through the View menu. This tab organizes clock domains hierarchically, presenting each component and its interfaces within the respective clock region. By selecting specific clock domains in this tab, you can filter the System Contents tab to display only the components associated with the chosen domains. The System Contents tab will also use colors to indicate which interfaces belong to each clock domain. Additionally, you can refine the display further to show only the component interfaces within a particular domain, useful when a single component has interfaces across multiple domains.

4.3. Clock Domain Crossing

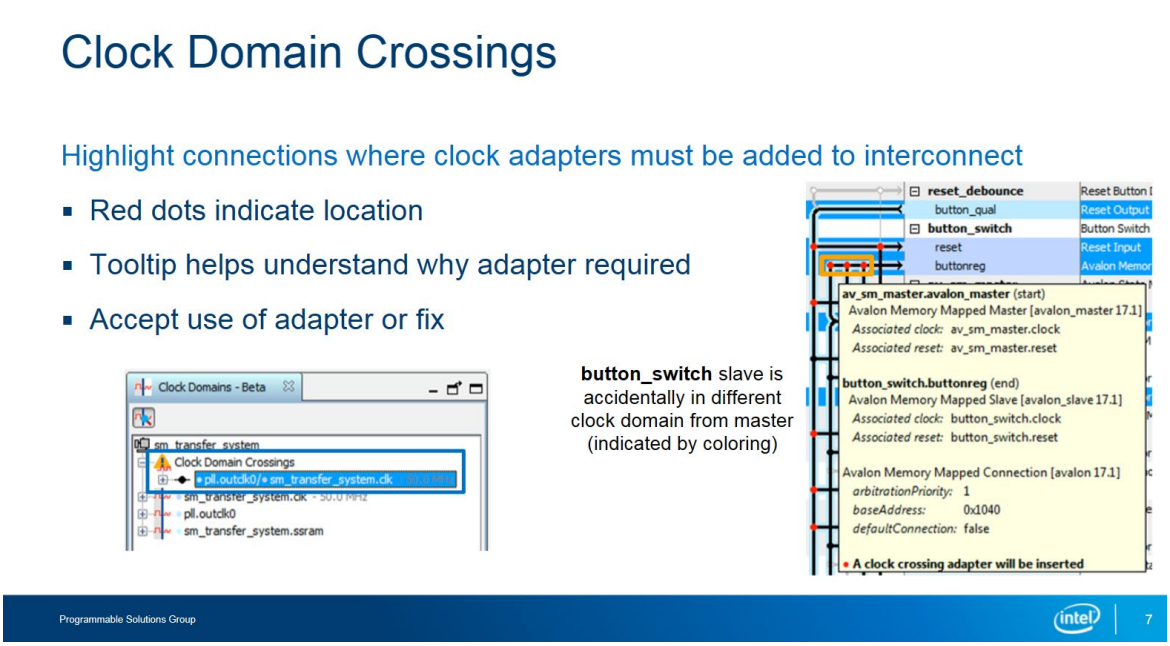

Figure 3 – Clock Domain Crossing

1. Clock Domains Tab Overview:
   - If your system has multiple clock domains (different timing zones), you can check them out in the Clock Domains tab. Here, you'll also see where any special clock crossing logic is placed. This logic helps data move smoothly between areas with different timing.
2. Automatic Clock Crossing Adapter:
   - Platform Designer does some clever work for you by automatically adding this clock crossing logic where necessary.
   - If you select a clock domain crossing from the top of the Clock Domains tab, red dots appear in the System Contents tab. These dots show where the tool plans to add the adapter.
3. Making Decisions:
   - When you hover over these red dots, you'll get little tooltips that explain why the adapter is needed. This helps you understand what's going on.
   - Now, you have a choice: you can let the tool insert the adapter as planned, or you can tweak your design to avoid needing it altogether.
4. Example Situation:
   - Let's say you have a component with two interfaces: one controlled by one clock (let's call it "buttonreg"), and another by a different clock ("avalon_master").
   - This setup might work fine, but if you're facing timing issues in this clock crossing scenario, switching both interfaces to use the same clock might solve the problem.

So, in a nutshell, the Clock Domains tab helps you understand and manage the timing differences in your system, while the tool takes care of adapting data flow between different clock domains automatically, giving you control over the process based on your design needs.

4.4. Reset Control Options

Figure 4 - Reset Control Options

In Platform Designer, reset interfaces are distinct from clock interfaces but are always linked to a specific clock domain. External resets can enter the system directly or through a Clock Source component, with control over their synchronization to the component's clock signal. Resets can also be generated internally by other components, providing reset control through an interface. Two methods exist for connecting resets: a global reset, automatically managed by Platform Designer, or manual connections for more precise control. Care must be taken to avoid reset loops or system lockups. If multiple external resets are present, a reset controller or reset sequencer can be used for efficient management.

In summary, Platform Designer provides options for managing reset interfaces, allowing you to synchronize resets with clock signals and control reset behavior either automatically or manually, depending on your system's requirements.

4.5. Reset Connection Points

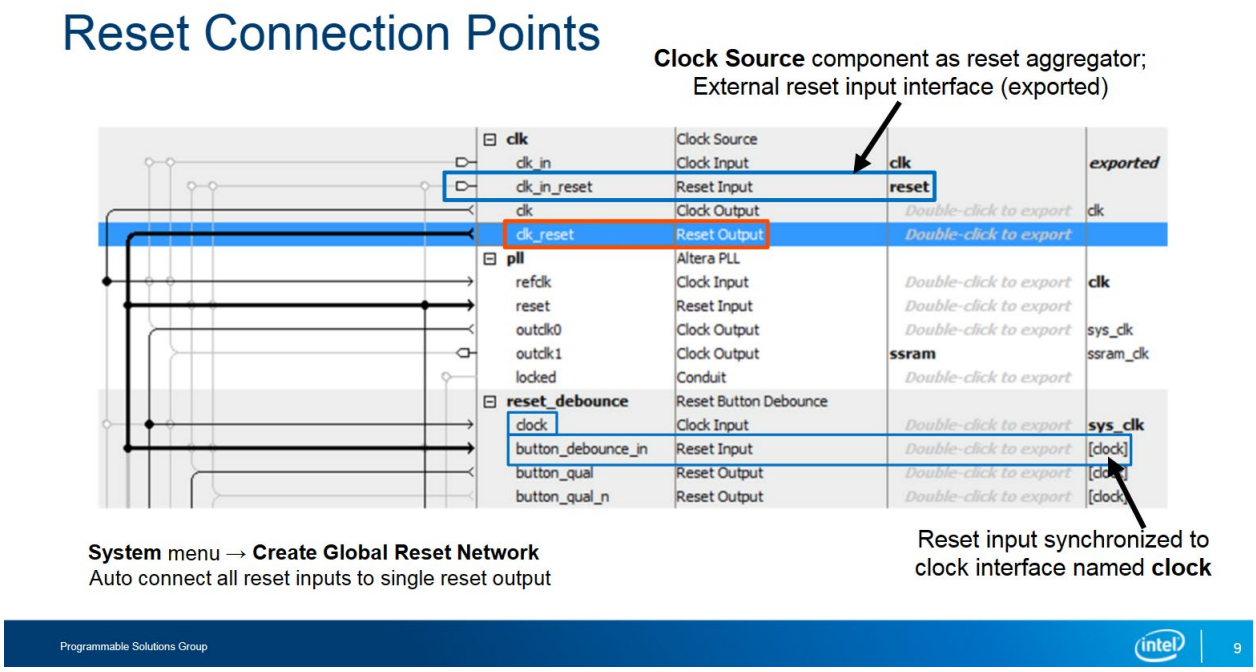

Figure 5 – Reset connect points

1. Manual Reset Connection:
   - In this scenario, resets are connected manually.
   - First, the reset input interface of the Clock Source component is exported to link with an external reset signal.
   - Then, the reset output interface of the Clock Source component connects to the reset input interfaces of other components across the system.
2. Clock Synchronization:
   - In the system, each interface, including reset interfaces, is synchronized by a specific clock interface.
   - The Clock column in the system indicates which clock interface is responsible for synchronizing each interface.
3. Example Highlight:
   - For example, let's consider the highlighted "button_debounce_in" reset input interface.
   - This interface's synchronization is managed by the clock input interface named "clock" on the "reset_debounce" component.

By manually connecting resets and ensuring synchronization with specific clock interfaces, the system maintains proper timing coordination between different components.

4.6. Reset Domain Filtering

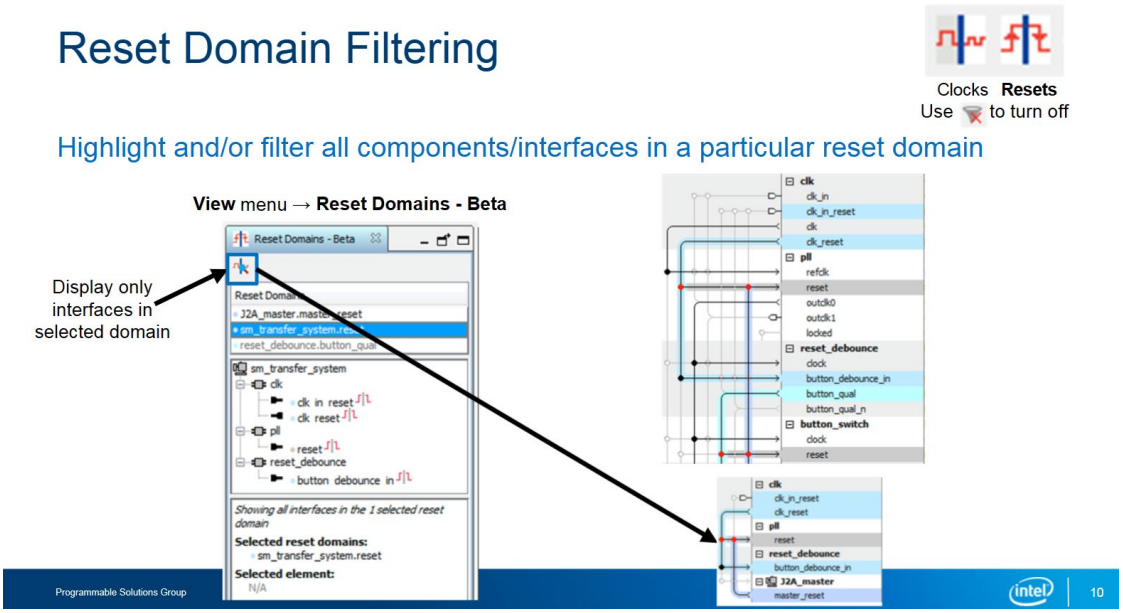

Figure 6 – Reset Domain Filtering

Just like the Clock Domains tab, you can access the Reset Domains tab from the View menu to view components and interfaces grouped into reset domains. This is useful, especially if you've made custom reset connections instead of using the global reset option. Additionally, you can apply further filtering to show only reset interfaces, simplifying the view. One distinction from the Clock Domains tab is that when selecting multiple reset domains, the System Contents tab displays components and interfaces connected to both domains, rather than just those connected to either one.

4.7. Reset Adapter Insertion

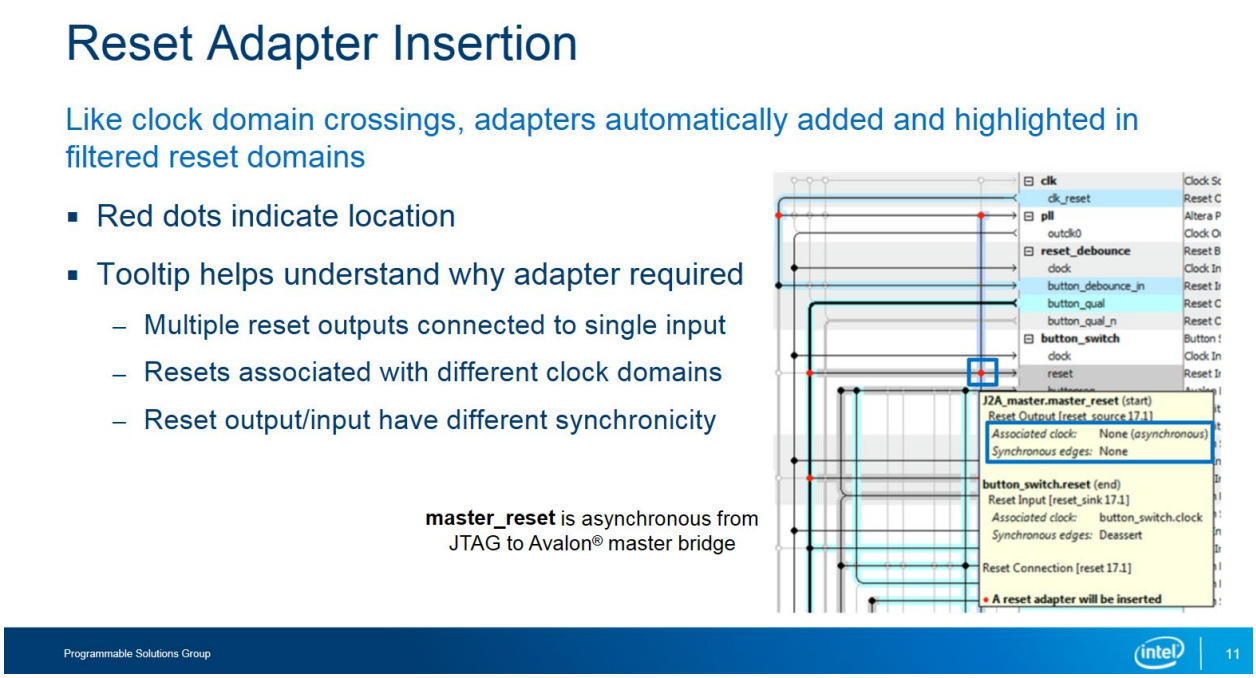

Figure 7 – Reset Adapter Insertion

1. Reset Synchronizers:
   - Like clock crossing adapters, you can view where reset synchronizers are inserted in the system by selecting a domain crossing from the Reset Domains tab.
   - Red dots on the System Contents tab show where these synchronizers are placed, with tooltips explaining why they're necessary.
2. Reasons for Adapter Insertion:
   - Three common situations prompt the insertion of an adapter.
   - Like clock domain crossings, you can choose to keep the adapter or adjust the design to avoid needing it, although sometimes it's unavoidable.
3. Example Scenario:
   - In the provided example, the system features a JTAG to Avalon master bridge, a special component allowing JTAG access for hardware debugging.
   - This bridge uses an asynchronous reset and is connected to the reset input of the button_switch component.
   - An adapter is required to synchronize the asynchronous reset output of the bridge with the synchronous reset input of the button_switch component.

In summary, reset synchronizers are inserted similarly to clock crossing adapters, with red dots indicating their placement. Understanding when and why adapters are needed helps ensure proper system functionality, even in scenarios involving components with different reset types.

4.8. Conduit Interfaces

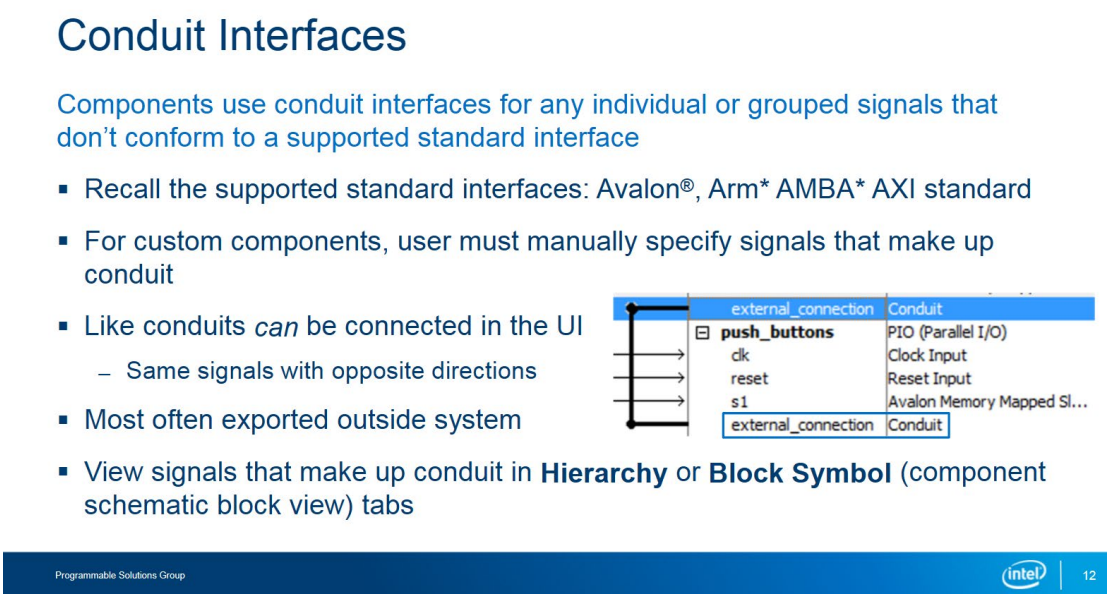

Figure 8 – Conduit Interfaces

Conduit interfaces serve as communication channels between components within a system. They facilitate the exchange of data, signals, or control information between different modules or blocks. These interfaces act as pipelines, allowing seamless transmission of data streams or signals from one component to another. Conduit interfaces play a crucial role in enabling interaction and coordination between various elements of a system, enhancing its functionality and efficiency.

4.9. Memory-Mapped Addressing

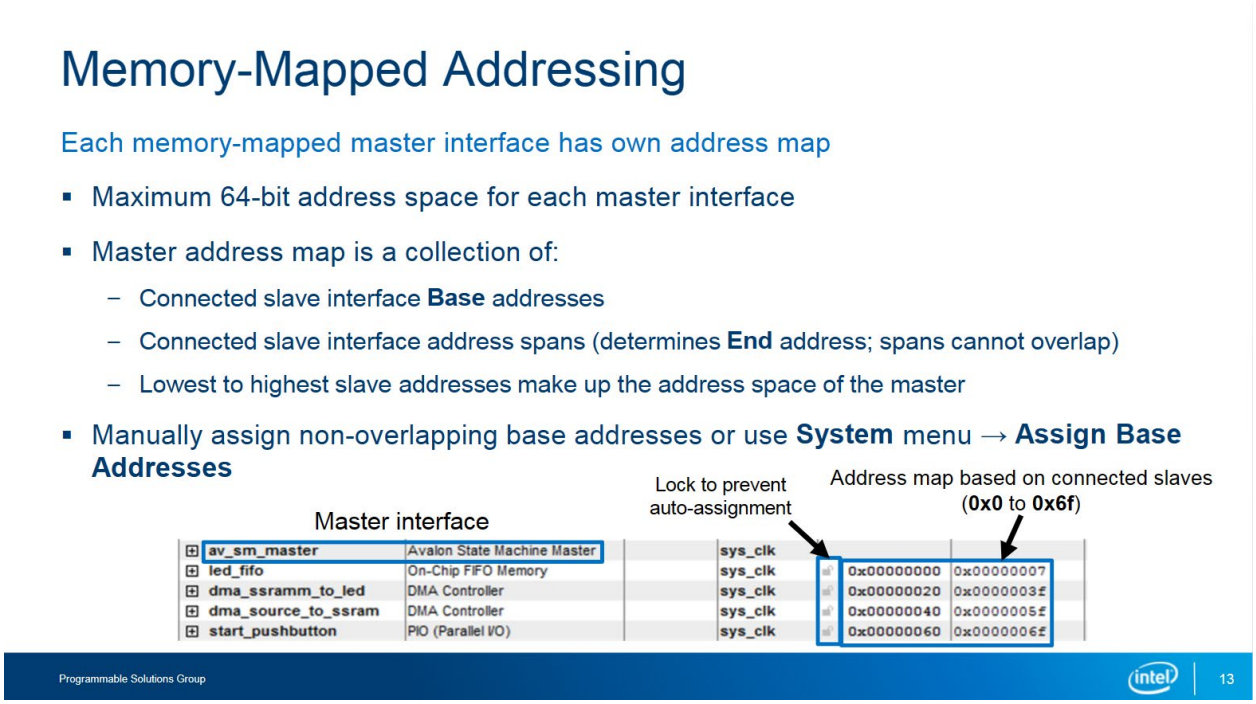

Figure 9 – Memory-Mapped Addressing

The final item to configurate on the System Contents tab is memory-mapped addressing. It allows you to define addressing for the memory-mapped section of your system, supporting up to 64-bit addressing. Each memory-mapped master interface has its unique address map to access connected slave interfaces. The address span of a master interface is determined by the collective address spans of its connected slave interfaces. If there are gaps in the address map, the interconnect automatically handles accesses to them. If multiple master interfaces connect to the same slave interface, their address maps converge. Platform Designer automatically assigns base and ending addresses for slave interfaces as connections are made, but you can manually change these addresses if needed. This can simplify decoding logic, resolve conflicts, or accommodate pre-written software or hardware logic. Slave address spans must not overlap, and auto-assignment of non-overlapping addresses is available. You can prevent auto-assigned addresses from changing by locking them manually.

4.10. Address Map Tab

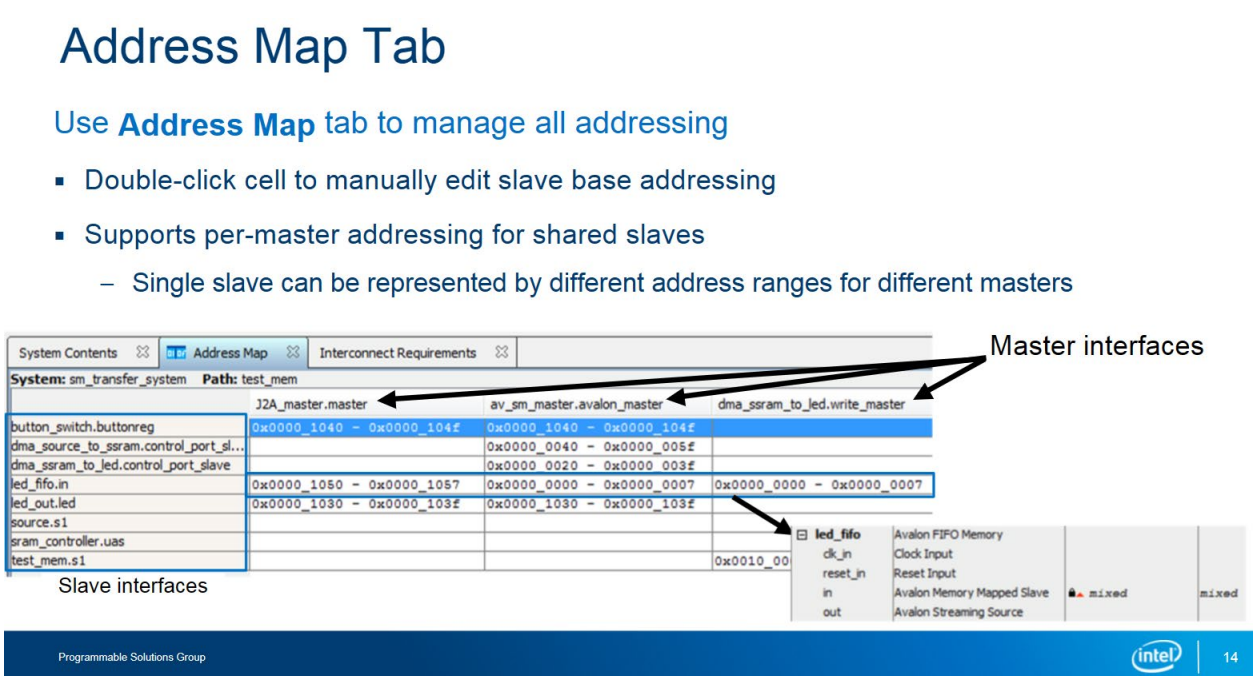

Figure 10 – Address Map Tab

In Platform Designer, there's a detailed Address Map tab in addition to the System Contents tab. This tab offers a comprehensive table of all memory-mapped device addresses, simplifying address mapping and configuration. Each column represents a memory-mapped master interface, while each row represents a slave interface. When a slave is connected to a master, the corresponding cell in the table displays the slave's address span for that master. You can edit these addresses directly in the table. If a slave is connected to multiple masters, its address may vary for each master, and the Address Map tab allows you to manage these variations efficiently.

5. Part 5 – Review Connections

Next step in Basic Platform Designer System Building Flow after adding all components to the system and after this connection to each other is to review their connections.

5.1. System Review and Modification

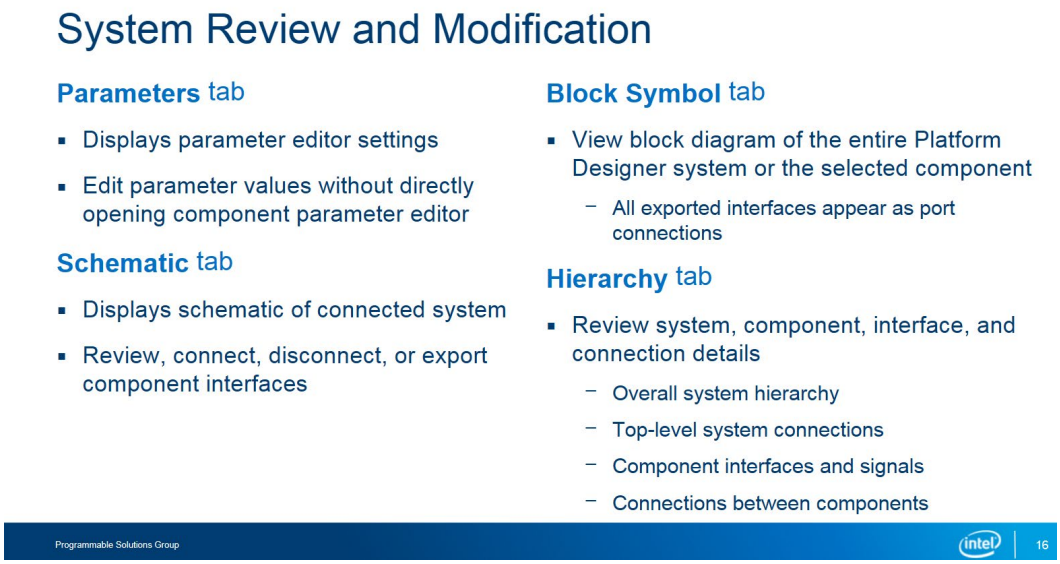

Figure 11 – System Review and Modification

Use tabs like System Contents, Clock, or Reset Domains to check your progress in building a Platform Designer system. You can access these tabs from the View menu.

1. Additional Tabs for Assistance:
   - Apart from the mentioned tabs, there are others to assist you.
   - The Parameters tab allows you to review and edit component parameters without reopening the editor.
   - Simply select a component in any tab and switch to the Parameters tab to make changes.
2. Schematic tab:
   - The Schematic tab offers a visual representation of your system, helping you understand data flow and control logic.
   - You can filter the display and make or break interface connections directly in this tab, providing a bird's eye view of complex designs.
3. Block Symbol View:
   - The Block Symbol tab displays a block diagram of selected components or the entire system.
   - It highlights external connections and shows exported interfaces.
   - Enabling the Show Signals option reveals individual signals within each interface.
4. Hierarchical Review:
   - The Hierarchy tab presents system connections hierarchically, similar to the Project Navigator in Intel® Quartus® Prime software.

- It allows you to examine the entire system, its components, exported interfaces, component interfaces, signals, and connections between interfaces easily.

These tabs help you navigate, understand, and make adjustments to your Platform Designer system efficiently.

5.2. System Editing on System Contents tab

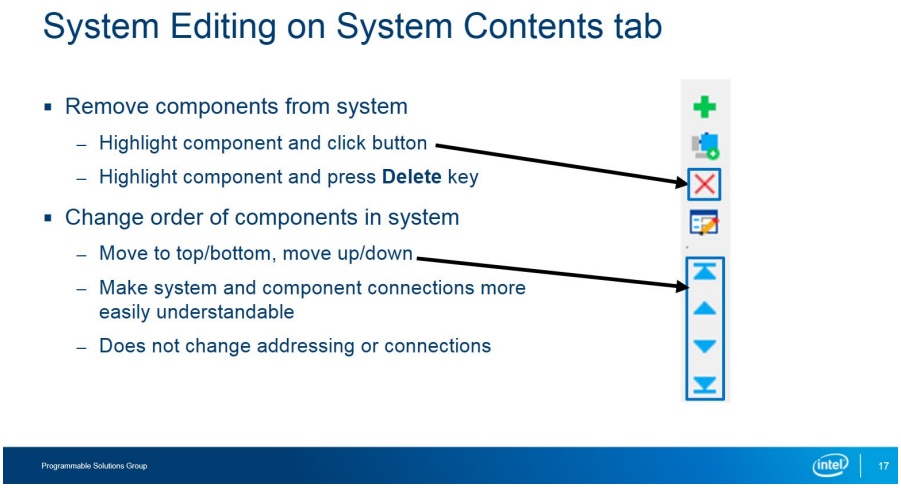

Figure 12 – System Editing on System Contents tab

In Platform Designer, there's a toolbar for editing systems. It lets you remove or edit components and change their order. Rearranging components mainly helps with organization and readability. For instance, you can group components with similar interfaces together. Importantly, changing the order doesn't affect interface connections or memory-mapped slave addresses. It's all about making your system easier to understand and manage.

5.3. Basic Filtering

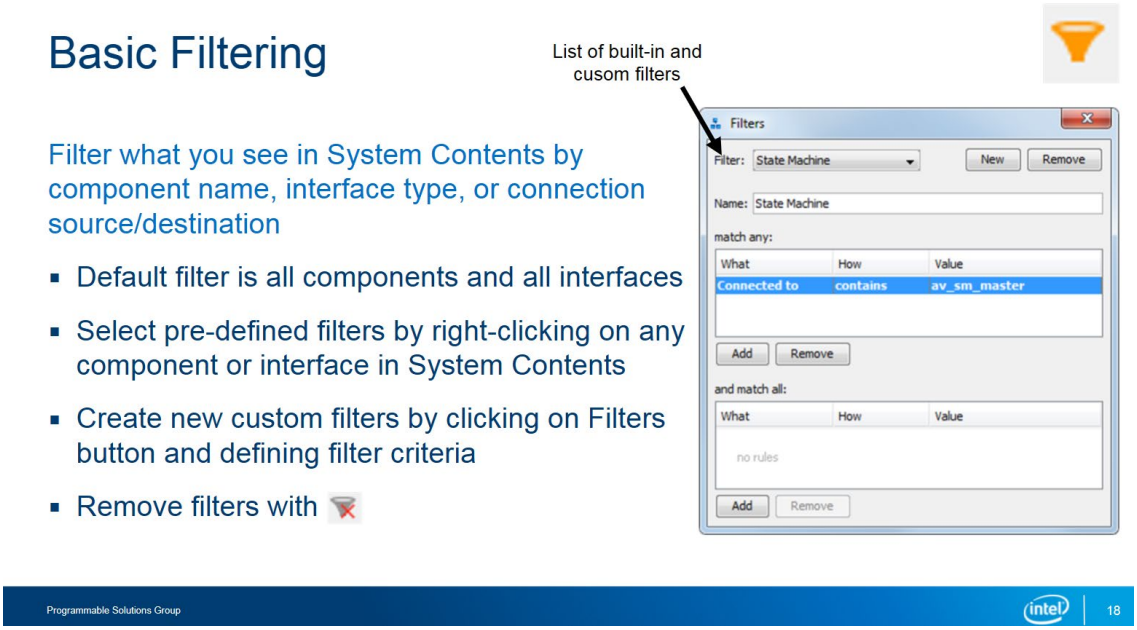

Figure 13 – Basic Filtering

In Platform Designer, you can filter the System Contents tab by clocks and resets, which helps in understanding your system's contents and connections. For more detailed filtering, you can open the Filters dialog box from the toolbar. This allows you to filter components and connections based on criteria such as interface type and component name. You can choose from pre-defined filters or create custom ones, making it easier to focus on specific aspects of your design and store them for future use.

5.4. Device Family and Interconnected Requirements Tab

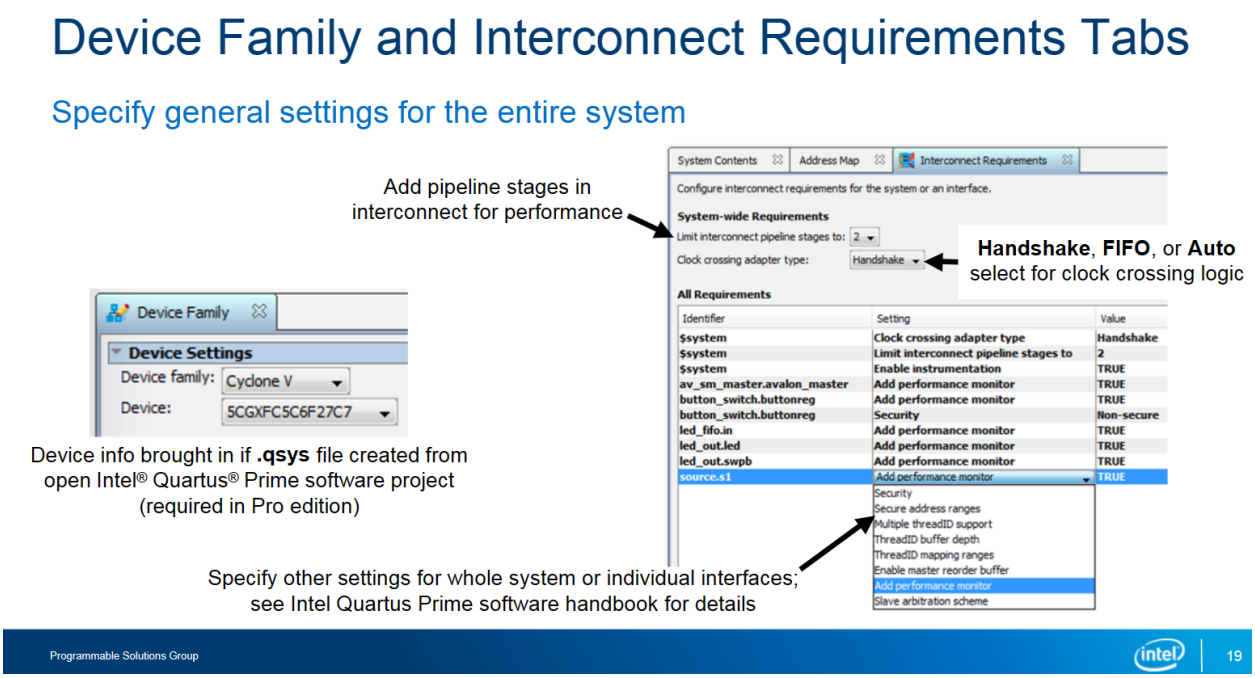

Figure 14 – Device Family Tab

Once your system is ready, you'll want to generate it with its custom interconnect. To do this, you need to set up some important details in two tabs: the Device Family tab and the Interconnect Requirements tab.

- On the Device Family tab, you need to make sure you've picked the right type of device you're targeting, like which family it belongs to and its specific part number. This helps the tool understand what kind of hardware it's working with.
- Then, on the Interconnect Requirements tab, you can tweak settings that affect how the connections between different parts of your system are handled. For example, you can decide how much extra processing the interconnect should do and how it should handle timing differences between different parts of your system. It's like giving instructions on how you want things to be connected inside your system.
- These tabs might seem a bit complicated, but they're important to make sure everything works smoothly. If you're not sure about something, you can always check the handbook for more details.

5.5. Messages Tab

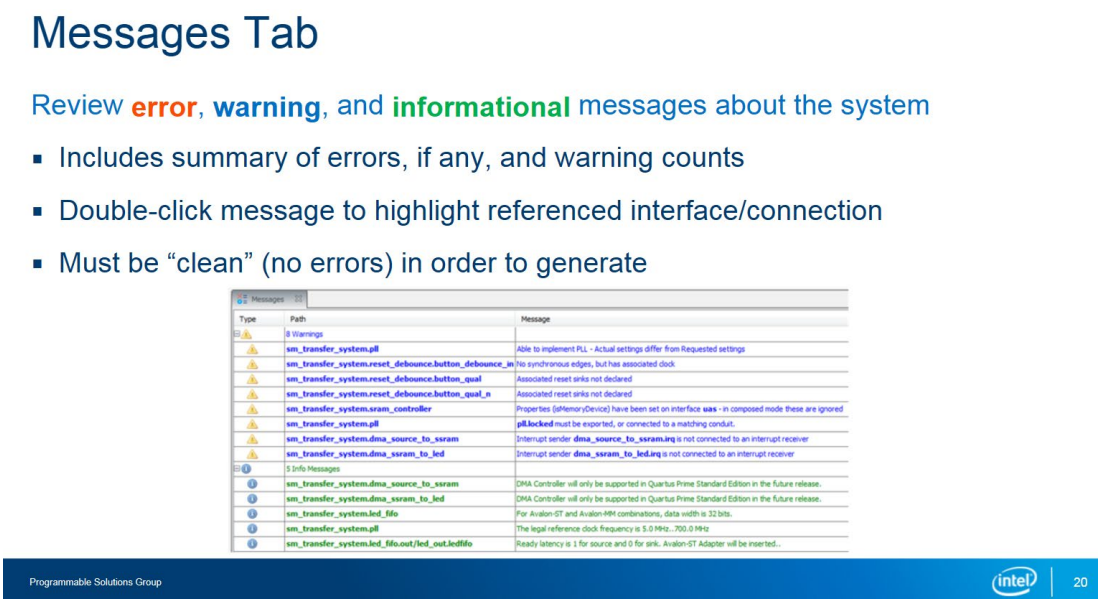

Figure 15 – Messages Tab

At the bottom of Platform Designer, there's a Messages tab. It shows different types of messages: info, warnings, and errors. These messages tell you about things you've done or haven't done in your system. If you click on a message, it will show you what it's talking about in your system. Red errors need fixing before you can finish your work, while blue warnings are just reminders to double-check things. It's like a little helper that tells you if something needs attention before you're done.

6. Part 6 – Generate HDL for System

The final step is to generate the HDL code for the system.

6.1. System Review and Modification

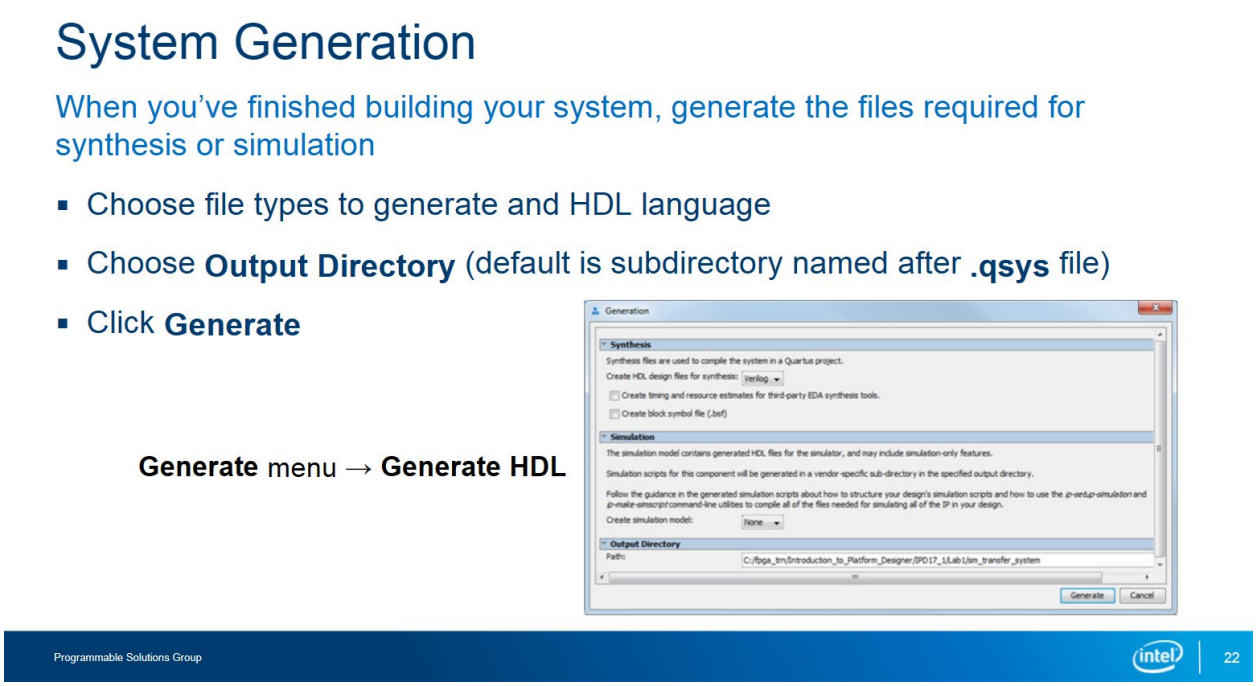

Figure 16 – System Generation

Imagine you've just finished building your system using Platform Designer, and now it's time to turn it into real computer code. Here's what you need to do:

1. Go to the menu and select "Generate HDL" – this tells the computer to create the code for your system.
2. Decide what kind of files you want the computer to make. If you're planning to make your system real and run it on actual hardware, you'll want "synthesis optimized" files. If you just want to test it out on the computer, you'll need "simulation model" files. You can also get a "symbol" file to use in a schematic if you want.
3. Choose which computer language you want the code to be written in – it's like picking the language you speak best.
4. Tell the computer where to save all these new files. You can choose a main folder, and the computer will make subfolders inside it for the different types of files.
5. Finally, click "Generate" and watch the computer work its magic – it'll create all the code you need to make your system come to life!

6.2. HDL Instantiation Template

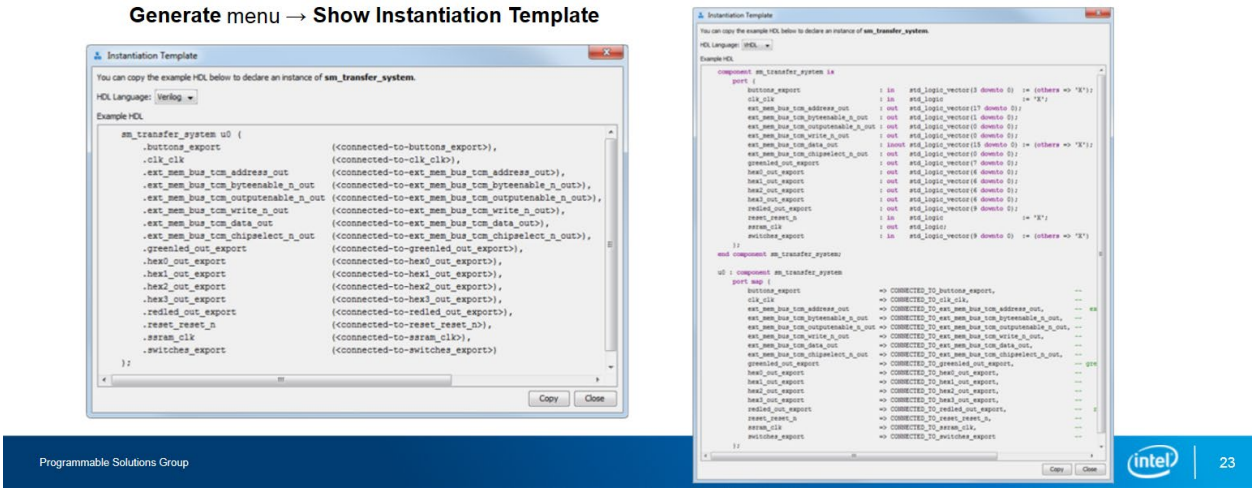

Figure 17 – HDL Instantiation Template

If you want to add another design file, Platform Designer makes it super easy. Here's how:

1. Go to the menu and pick "Show Instantiation Template" – this gives you a special kind of template, like a guide, to help you put your system into the other design file.
2. Choose whether you want the guide to be in VHDL or Verilog – it's like picking the language you want the guide to be written in.
3. Copy the code into your design file.
4. You'll see some parts in the guide that you need to change to fit your design. Editing the port mapping sections in the template involves specifying how each part of your system connects to other components in the overall design.

7. Part 7 - Generalizing Basic Platform Designer system building

Here are the key steps for creating a system design in Platform Designer:

1. First, choose the components you need from the IP Catalog and configure them using their parameter editors.
2. Then, utilize the Platform Designer interface to connect or export interfaces.
3. Next, review your connections to ensure they align with your design's goals and functionality.
4. And lastly, generate Verilog or VHDL files to represent your system, ready to integrate into your Intel® Quartus® Prime software project for compilation.

8. Intel Quartus Prime Design Flow

8.1. FPGA Hardware Design Flow

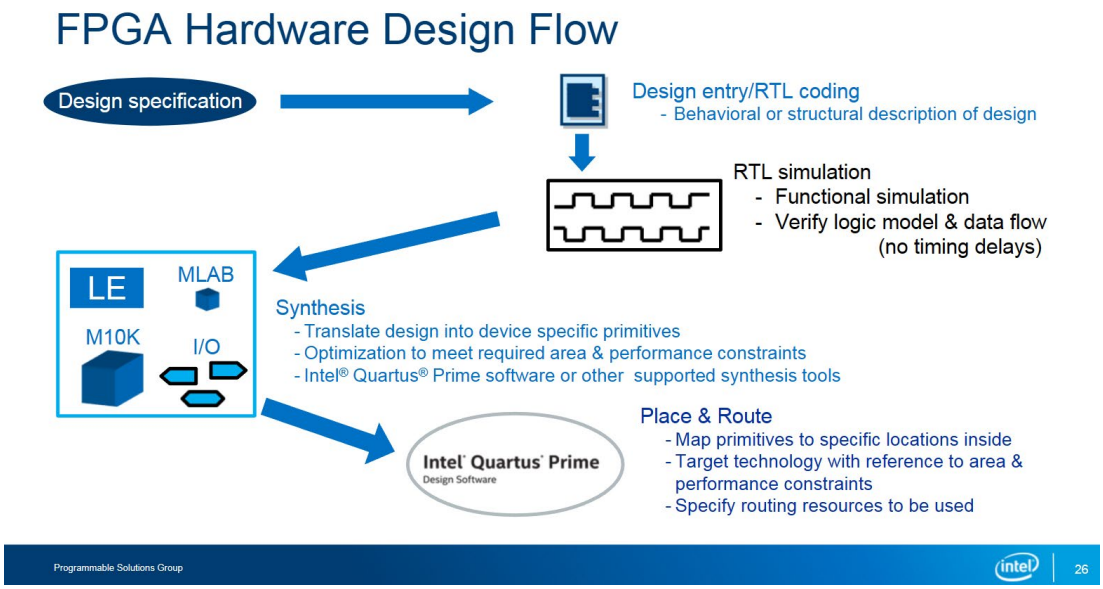

Figure 18 – Simple QP Design Flow

On the figure above we can see the typical FPGA design process:

1. First, you write code that describes how your design should work.
2. Then, you run a simulation to make sure it behaves as expected.
3. Next, synthesis transforms your code into specific instructions for the FPGA.
4. After that, the Intel Quartus Prime software figures out where to put these instructions and connects them together. Platform Designer fits into this process as a tool for building systems, creating code that fits into this flow just like any other method of design entry.

8.2. Platform Designer System Design Flow

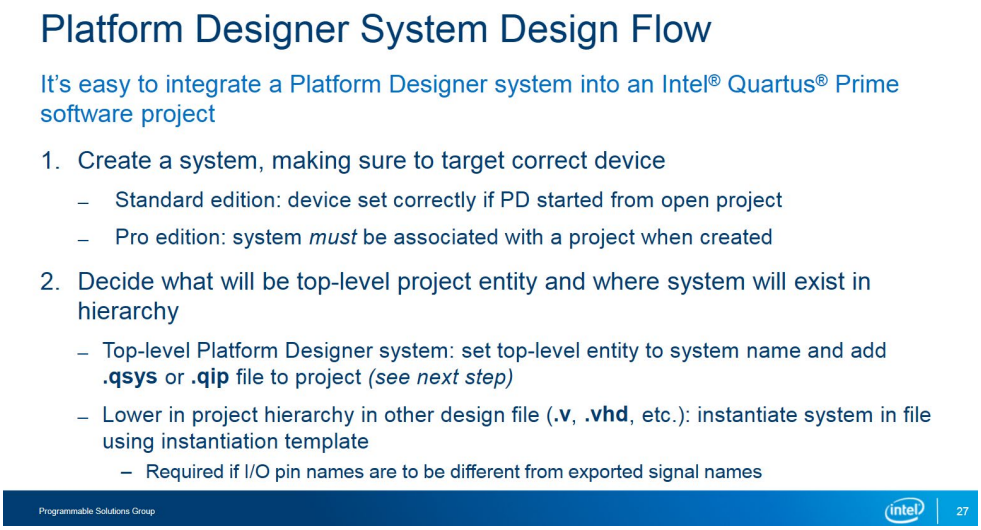

Figure 19 – PD System Design Flow (1)

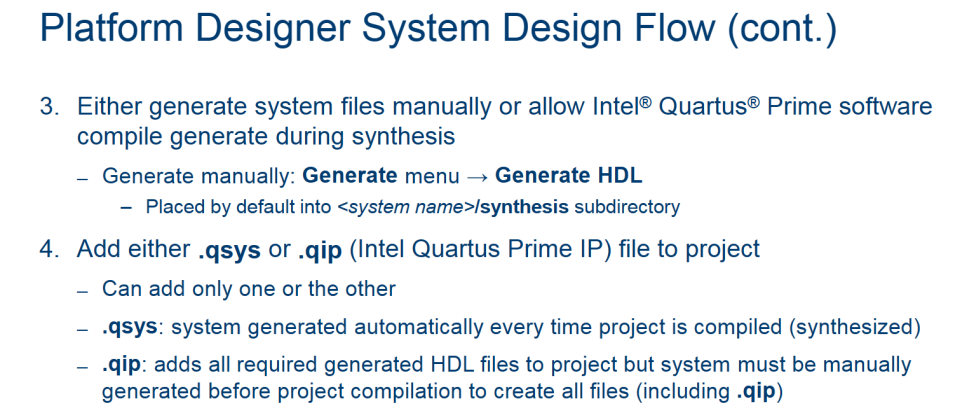

Figure 20 – PD System Design Flow (2)

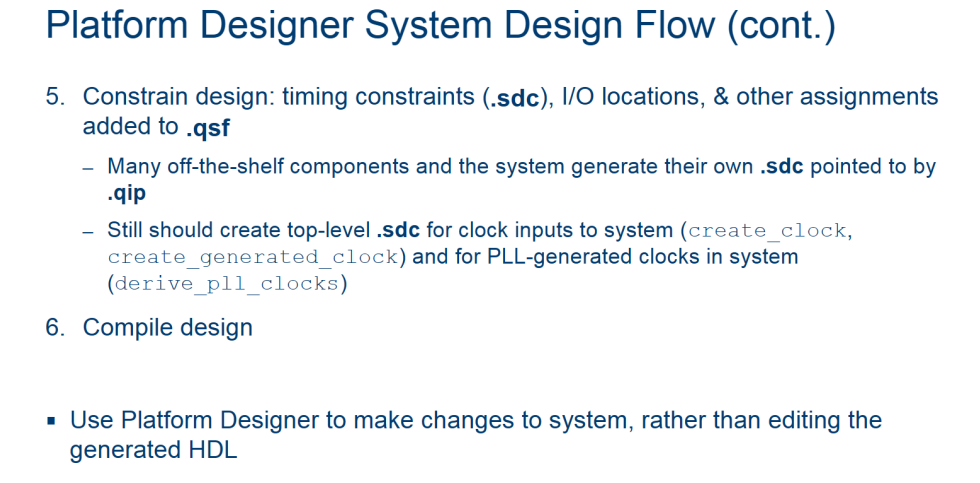

Figure 21 – PD System Design Flow (3)

To integrate a Platform Designer system into the usual Intel Quartus Prime software design process, follow these steps:

1. Build your system using Platform Designer and set the target device.
2. Decide if your system will be the main part of your project or a part of it.
3. If your system is the main part, specify its name as the top-level design entity in the Intel Quartus Prime project. If it's lower down, use the instantiation template to place it correctly.
4. If you want different names for your system's signals and the I/O pins, create a wrapper to translate them.
5. Set up your project to reference the generated system. You can add the .qip file manually or include the .qsys file and let the project compile it.
6. Add project constraints like I/O assignments and timing constraints.
7. Compile your project.

Remember, if you need to change your Platform Designer system, do it in Platform Designer itself. Don't edit the generated HDL directly, as your changes won't be saved back to the .qsys file and will be overwritten when the system is regenerated.