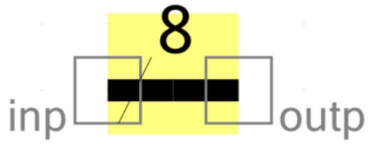
Dummy
0.0

Features

- Empty pass-through component.
- Connects two digital buses of same width.
- Facilitates schematic routing and resolves errors.
- Virtual component – does not consume resources.
- Simplifies schematics and saves space.



General description

The Dummy component is a virtual digital component facilitating wire and bus routing on schematic at the design time. When two buses of different width are joined on schematics, bits correspondence must be explicitly stipulated using standard indexing tool. Unfortunately, in certain cases, the fitter is being confused with design which leads to compilation errors during project build. Inserting the Dummy separator between the buses resolves those issues. The Dummy is an empty (pass-through) component, and is automatically eliminated by the fitter during project build. Using Dummy component simplifies signal routing and saves space on schematics. The Dummy component works only for digital bus types and can't be used with analog signals.

When to use Dummy component

Component can be used to resolve complex routing errors arising when two digital buses of dissimilar width need to be joined by using range indexing tools, and named wires are used for sharing signal between digital components. Demo project is provided.

Functional Description

The Dummy is an empty pass-through component, which seemingly does not do anything: it freely passes across all digital signals and is automatically eliminated during project build (Figure 1). The sole purpose of the component is to facilitate fitter routing and to avoid errors in certain cases, when two digital buses of different width are being joined.

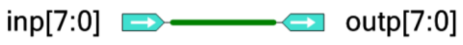


Figure 1. Dummy component schematic.

Below is review of three practical examples, which came from personal experience and various PSoC forums. Here the Control and Status registers are utilized for demonstration purposes only, but same logic applies to any other components, which have digital input or output buses.

Example 1

Consider simple schematic depicted on Figure 2. In this example all bits within the bus are freely passed from one component to another, except for LSB, which is being manipulated through external digital input. Though all range assignments seems to be in place, attempt to build the project terminates with error: ‘Multiple drivers on signal, “Net_…”, bit(s) “0” ‘.

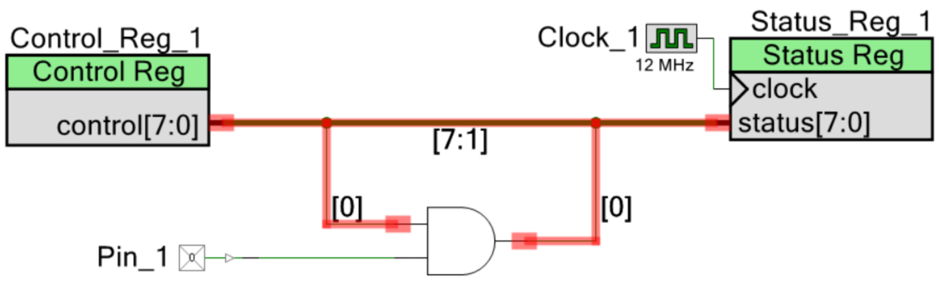


Figure 2. Failed schematic arrangement intended to manipulate the LSB of the bus.

By inserting the Dummy separator to the bus, the issue is resolved and project compiles without errors (Figure 3). Note that the Dummy component does not affect signals in any way and is being eliminated by the fitter during project build.

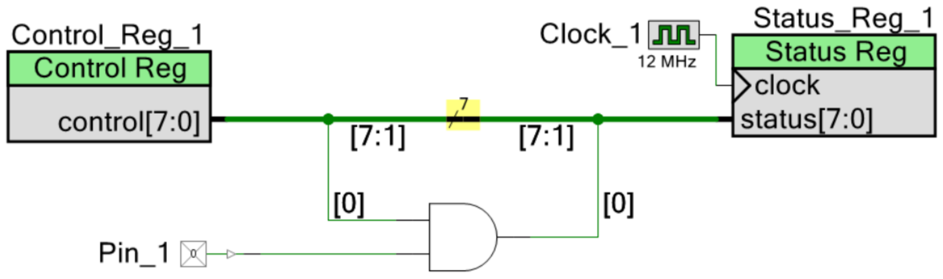


Figure 3. Error-free routing facilitated by the Dummy component.

Example 2

In this example a named wire is being used to connect two components of different bus widths (Figure 4). Such arrangement often happens when only few bits from output bus are being extracted, and signal is shared using named wires. Though wire assignments seems to be correct, the schematic shown shall not compile, and project build terminates with error: 'Terminal "Control_Reg_2_control_bus[7:0]" with width 8 is connected to "mywire_1[0]" with width 1'.

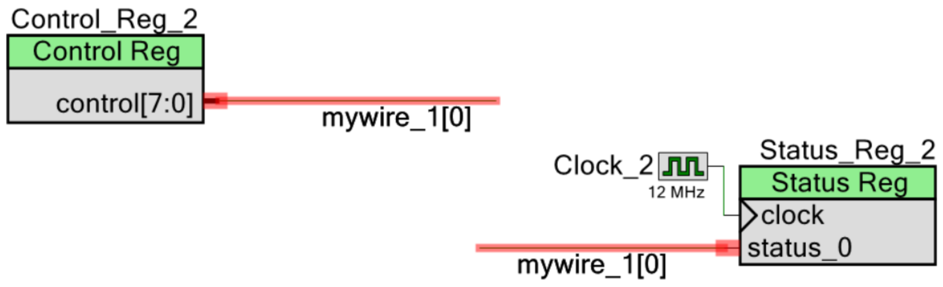


Figure 4. Failed attempt to connect components with different bus width using named wires.

Another attempt to modify schematic by specifying extracted bit number (Figure 5) does not help either, returning several error messages, including: 'Wires "mywire_1" and "mywire_1[0]" have inconsistent subsets'.

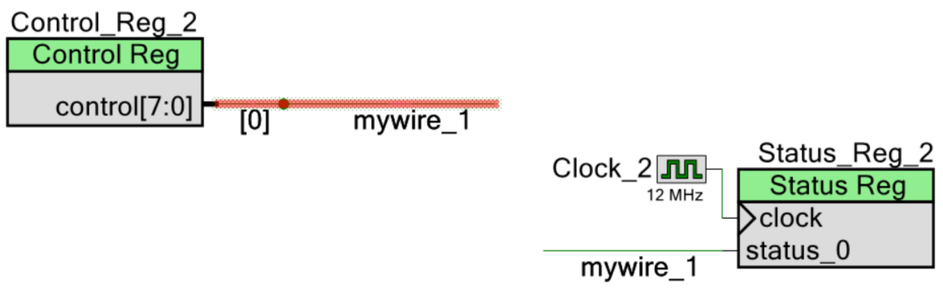


Figure 5. Another failed attempt to use named wires and bits range specifier.

Finally, by adding a Dummy separator between the bit specifier and named wire, the issue is resolved and project compiles without errors (Figure 6). Now two parts of the schematic can be shared across pages without errors. Moreover, should we decide to change extracted bit number (for example from “[0]” to “[1]”), the schematic needs to be updated only in a single place (“mywire_1” no longer need a bit specifier), which is particularly useful when schematic is split among several pages.

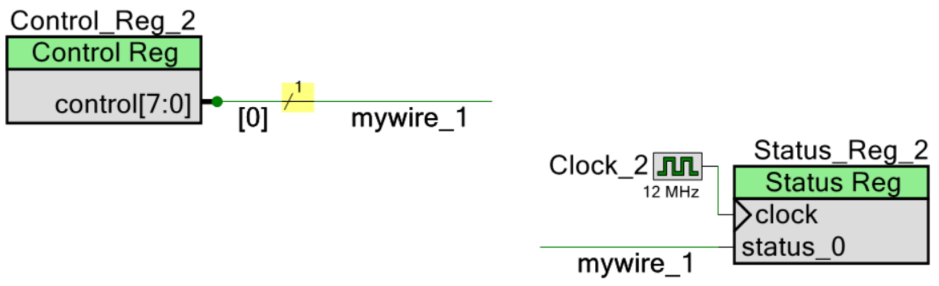


Figure 6. Error-free named wires connection is being facilitated by a Dummy component.

Example 3

This is design example of a custom component schematic, which has output bus width controlled by the **out_width** parameter. Typically, to vary the output bus width a Customizer should be added to the component with a code to modify bus indexes. In simple cases customization can be avoided by splitting output into few buses of different width and assigning visibility parameters to the output terminals* (Figure 7). In the end, only one terminal will appear on the component, while all hidden terminals are automatically eliminated by the fitter. Such arrangement often happens when a stock component is being modified, and intrusion into existing Customizer code is undesirable.

* Cosmetic considerations, like terminals placement and labeling, are being omitted here.

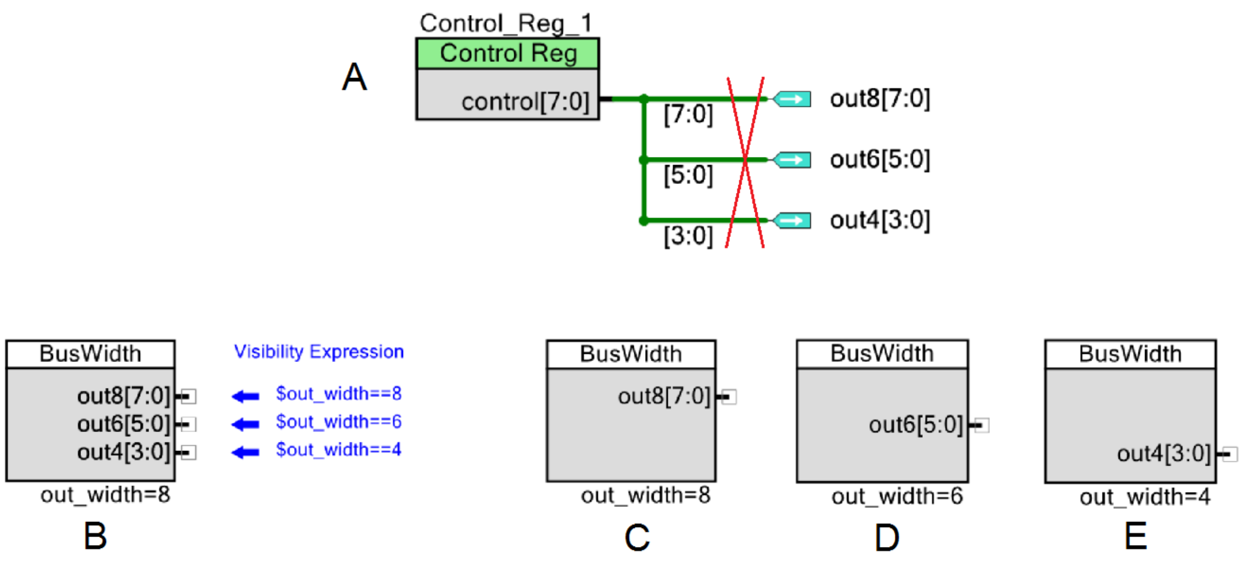


Figure 7. A- seemingly correct component schematics that shall not compile. B- component symbol with three output terminals; each terminal assigned own visibility expression. C,D,E- final component outlook depending on the control parameter out_width = 8, 6 or 4 correspondingly.

Despite seemingly correct routing, the component schematics shall not compile, producing error message: 'The background elaborator has encountered a problem. The given key was not present in the dictionary.' No further clues are provided. Apparently, the fitter was confused with schematics and aborted the build. Adding the Dummy separator between range specifiers and output terminals resolves the issue and project compiles without errors (Figure 8). As in the previous examples, the Dummy component, which has been added to schematic at design time, is automatically eliminated by the fitter during project build and does not affect project size and performance.

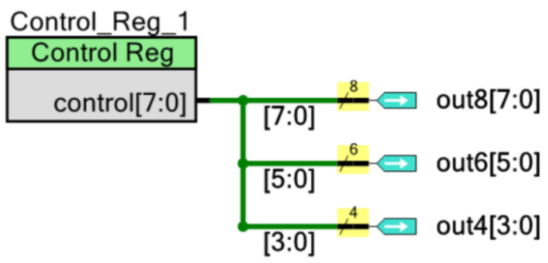


Figure 8. Error-free component schematic is being facilitated by the Dummy component.

Input-output connections

inp[N-1: 0] – input bus

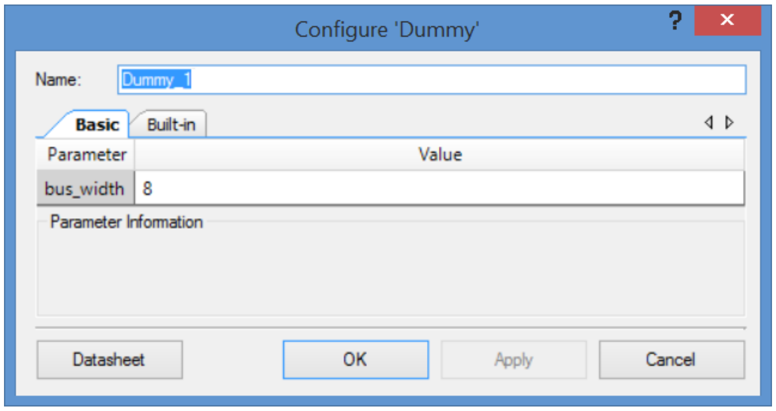
This pin is digital input bus. The bus width N is user-selectable, highest is 32. The pin is always visible. The pin must be connected to valid digital source.

outp[N-1: 0] – output bus

This pin is digital output bus. Output width is equal to the input width. This pin is always visible. The pin doesn't have to be connected.

Parameters and Settings

Basic dialog provides following parameters:



Name: Dummy_1

Parameter	Value
bus_width	8

bus_width [1…32]

Component bus width. Valid range is from 1 to 32.

Application Programming Interface

The component does not have associated API.

Resources

The component doesn’t consume UDB. The component is not device-specific and works with any of PSoC4 or PSoC5. Component does not use any clocks.

Performance

The component is virtual, it does not affect performance of the design.

Sample Firmware Source Code

A demo project containing examples 1, 2 and 3 is provided.

Component Changes

Version	Description of changes	Reason for changes/impact
0.0	Version 0.0 is the first beta release of the Dummy component	