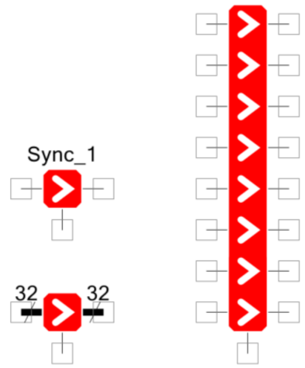
Sync32

0.0

Features



- Compact substitute for standard Sync component
- Synchronizes 1 to 32 input signals to clock
- Can use external or internal clock

General description

The Sync32 component serves the purpose of resynchronizing input signals to the rising edge of the clock. It can directly substitute the Creator standard Sync component while occupying much smaller footprint on the schematic. Component can synchronize 1 to 32 digital signals to the rising edge of the clock, and can be applied to a single line, multiple lines or bus. Component can use internal clock (BUS_CLK or HFCLK), eliminating the need for external clock. Component is compatible with PSoC4 and PSoC5.

When to use Sync32

The component can be used anywhere a standard Sync component is used for synchronizing digital signals across different clock domains. Component is useful for improving readability of the project schematics. Using small Sync32 can shrink the space occupied by a schematic by almost 50%.

Functional Description

The Sync32 component is useful hardware resource, allowing for synchronizing digital signals across different clock domains. It is implemented using standard *cy_sync* primitive, which uses double synchronizer to clock the input signal through two registers in series. This allows to resolve a metastable value that could occur if the incoming signal violates setup or hold on the first register.

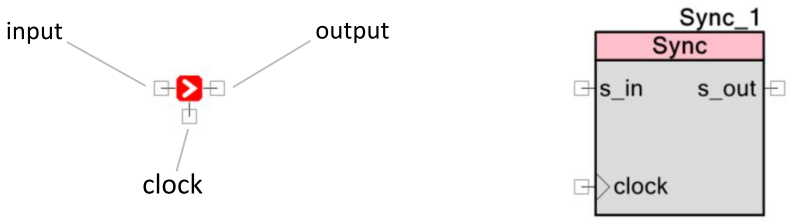


Figure 1. Terminals markup and size comparison of the Sync32 (left), and standard Sync components (right).

Input/Output connections

Input

Signal to be resynchronized. The signal must have a pulse width of at least one clock period(*).

Output

Synchronized output signal.

Clock (Input)

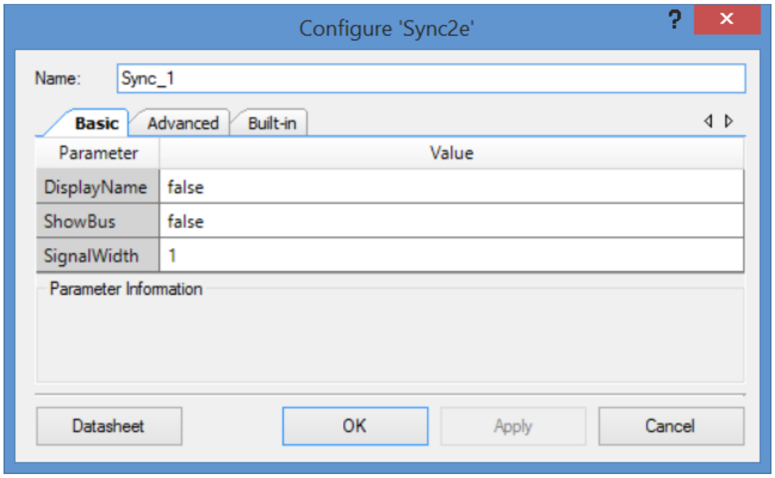
Signal that input is to be synchronized against.

---

[*] Plus 2ns

Parameters and Settings

Basic dialog provides following parameters:



Name: Sync_1

Parameter	Value
DisplayName	false
ShowBus	false
SignalWidth	1

DisplayName (bool)

Sets visibility of the component Instance Name. Default value is False.

ShowBus (bool)

Select the appearance of the input and output terminals as single bus or multiple lines. Default value is False. This parameter has effect only when SignalWidth > 1 (Figure 2).

SignalWidth (uint8)

This parameter configures the number of signals that will be synchronized to the associated clock. Valid range is [1 to 32] for bus mode, and [1 to 8] for multi-line mode. Default value is 1.

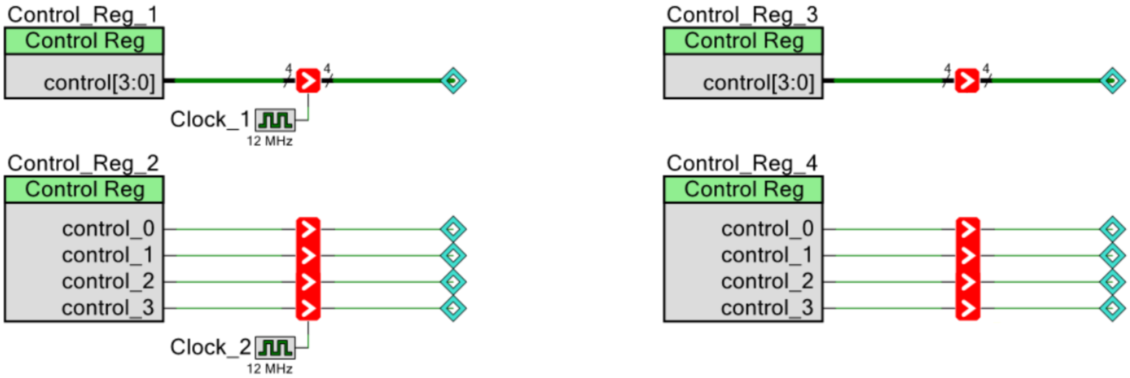
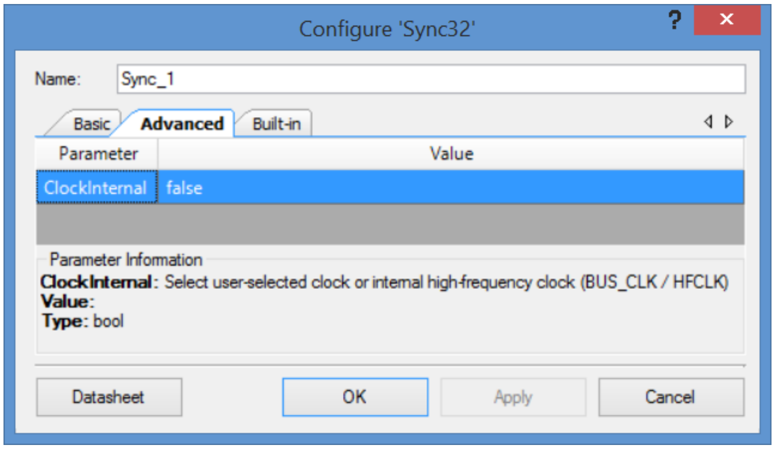


Figure 2. Component appearance with bus enabled (Top) and disabled (Bottom). Left – sync clock external, Right – synch clock internal.

Advanced dialog provides following parameters:



Name: Sync_1

Parameter	Value
ClockInternal	false

Parameter Information

ClockInternal: Select user-selected clock or internal high-frequency clock (BUS_CLK / HFCLK)
Value:
Type: bool

ClockInternal (bool)

Select external clock for synchronization or use internal high-frequency clock. Default is false. When internal clock is selected the clock terminal becomes hidden and internally generated high-frequency clock is used for synchronization (BUS_CLK for PSoC5, HFCLK for PSoC4).

Application Programming Interface

The component does not have associated API.

Resources

Component utilizes hardware resources of the Status Register. Up to four digital signals can be synchronized per each Status Register consumed.

Application examples

See **Appendix 1** for component application examples.

Component Changes

Version	Description of changes	Reason for changes/impact
0.0	First beta release.	

Appendix 1

Component application examples

Presented below are typical applications demonstrating effect of Sync32 small size, multi-line capability and internal clocking on schematic appearance.

The major feature of the Sync32 is its tiny size, which is fully utilized on the schematic makeover shown on Figure 3. The original project uses multiple Sync components, which are cluttering the view and obscuring the functionality of the design. Replacing the Sync component with Sync32 shrinks the schematic space almost by half and makes it more readable.

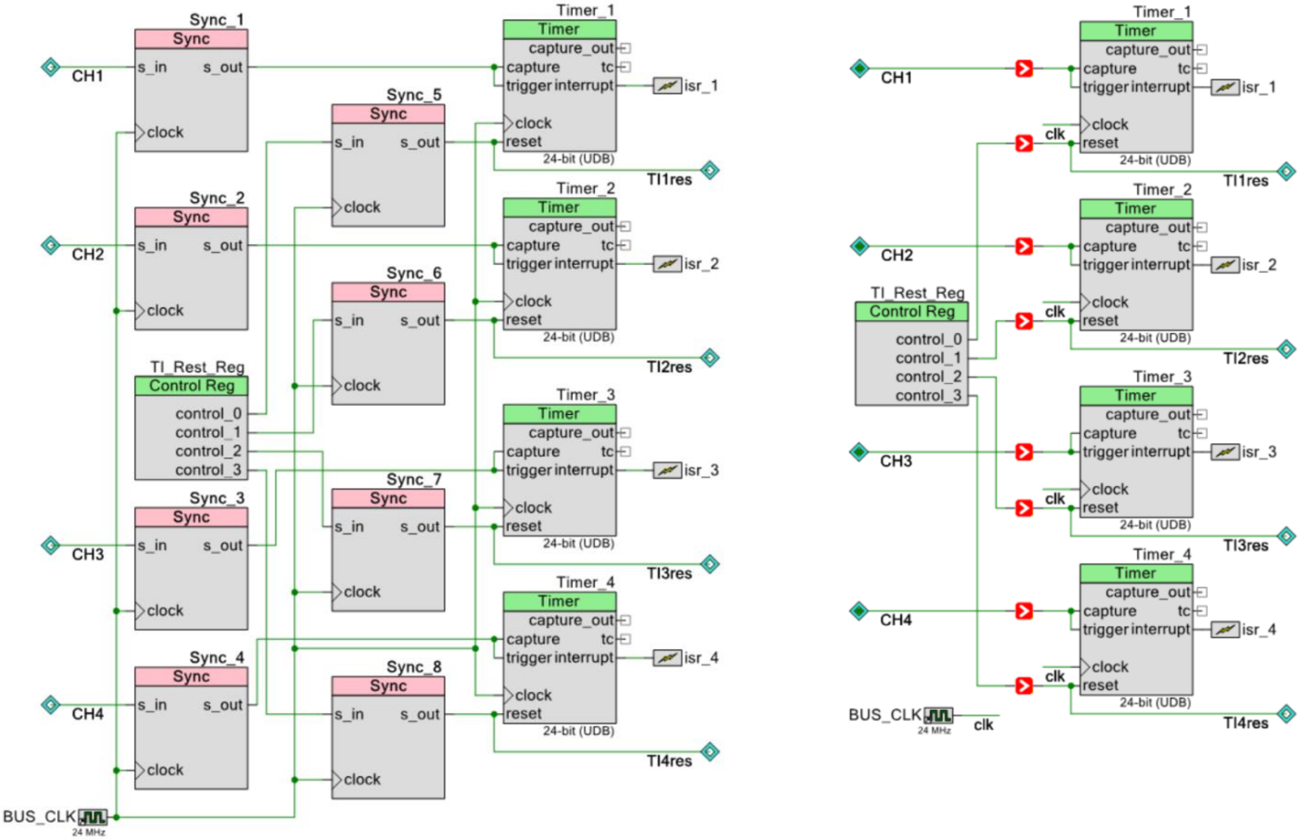


Figure 3. Left - original schematic using standard Sync, Right - modified schematic using custom Sync32. The original schematic is taken from elsewhere. The Sync32 is configured for internal clock, eliminating external clock routing.

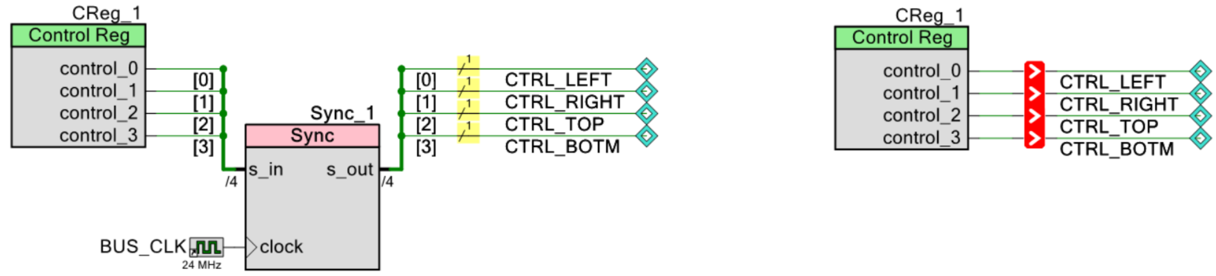


Figure 4. Left - original schematic using standard Sync, Right - modified schematic using custom Sync32. A custom component (the Dummy) had to be used in the original schematic to convert the indexed lines into the named wired (w/o index). Schematic using Sync32 needs no extra components and more clear.

Extra features of the Sync32 are demonstrated on Figure 4. The original schematic uses Sync component to synchronize several control lines to the system clock. The lines are combined into a bus, connected to the *s_in*. To fan out the *s_out* bus into named wires an additional custom component, the Dummy$^{(*)}$, had to be added to the project. Replacing the standard Sync with Sync32 in multi-line mode simplifies the schematic, eliminating the need for the bus and the Dummy. Using Sync32 internal clock obsoletes the BUS_CLK, further simplifying schematic.

The last example of the schematic overhaul using Sync32, is shown on Figure 5. As in previous examples, using the component shrinks schematic by about 50% and reveals its functionality.

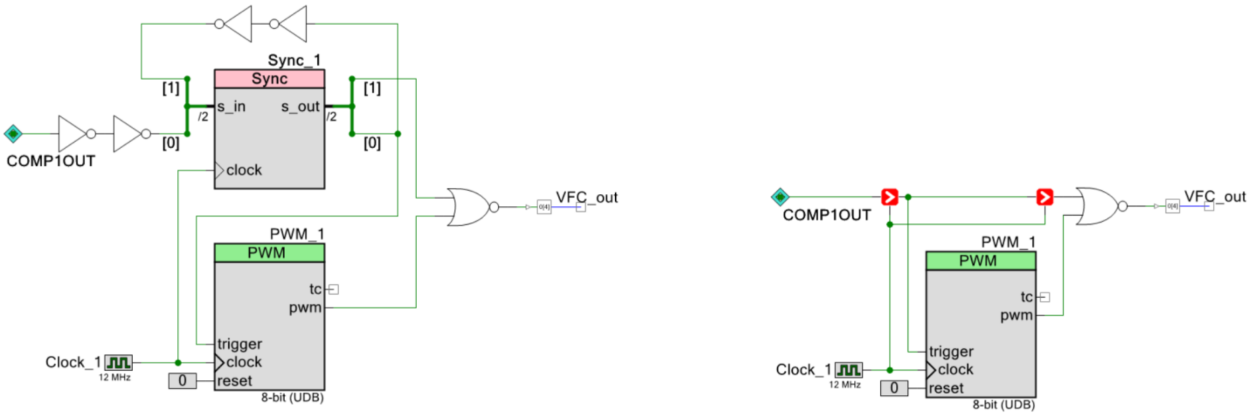


Figure 5. Left - original schematic using standard Sync, Right - modified schematic using custom Sync32. The original schematic was taken from elsewhere. Schematic using Sync32 is more readable than the original, revealing its functionality: PWM serves as a cut-off delay, while second Sync32 compensates for the PWM propagation delay. The double-NOT blocks, which purpose was to separate the named wire COMP1OUT and indexed lines [0] and [1], are entirely eliminated upon schematic makeover.

---

[*] The Dummy: empty component for digital bus routing, https://community.cypress.com/thread/13152