AN81623

PSoC® 3, PSoC 4, and PSoC 5LP Digital Design Best Practices

Authors: Todd Dust, Mark Ainsworth

Associated Part Family: All PSoC 3 and PSoC 5LP parts, and PSoC 4 4200 family

Software Version: PSoC Creator™ 2.2 SP1 and higher

For a complete list of the application notes, click here.

To get the latest version of this application note, or the associated project file, please visit http://www.cypress.com/go/AN81623.

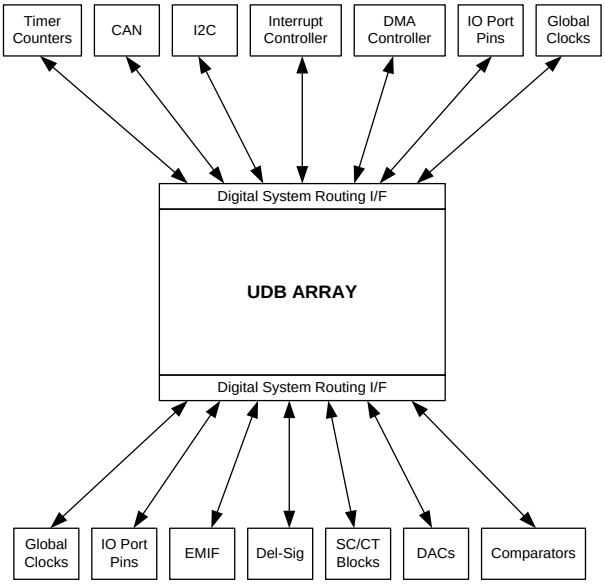
AN81623 gives a brief introduction to the digital hardware design theory and then describes the powerful and highly flexible digital subsystems in PSoC 3, PSoC 4 (4200 family), and PSoC 5LP. It describes best practices for digital design using PSoC Creator, and shows how to use static timing analysis (STA) report files.

Contents

1 Introduction

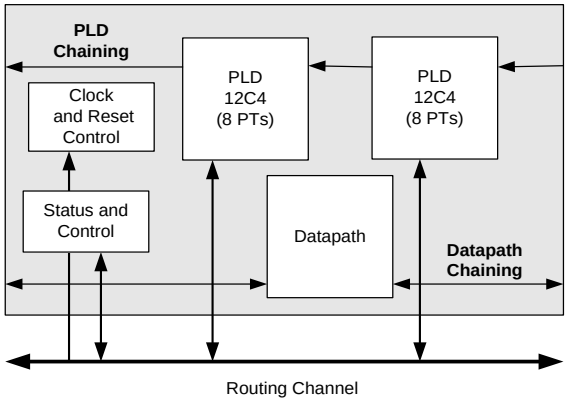
PSoC 3, PSoC 4, and PSoC 5LP have a powerful and flexible programmable digital peripheral system. In addition to a set of fixed function blocks (4 timers, I2C, USB, CAN), they offer as many as 24 programmable Universal Digital Blocks (UDBs) and an extensive signal routing system called the Digital System Interconnect (DSI). Figure 1 shows how these systems are located and used in some of the PSoC devices.

Figure 1. PSoC 3 and PSoC 5LP Programmable Digital Architecture



Each UDB contains two small programmable logic devices (PLDs), a datapath module containing a programmable 8-bit ALU, and other registers and functions, as Figure 2 shows.

Figure 2. PSoC 3 and PSoC 5LP UDB Block Diagram



By programming the UDB PLDs and datapaths, and routing signals within and between UDBs, you can make complex custom peripherals that significantly de-burden the CPU. Additional routing in the DSI allows these peripherals to interface with other PSoC elements such as I/O pins, DMA and the analog system, as shown in Figure 1.

The PSoC Creator IDE offers a large library of preprogrammed peripherals and enables you to create your own custom designs.

The configurability of PSoC allows unprecedented opportunities to optimize your design at the system level. However, it also exposes a whole new set of design considerations that a traditional MCU user may not be aware of. This application note provides information on a variety of digital design topics for PSoC 3 and PSoC 5LP.

This application note assumes that you are familiar with developing applications using PSoC Creator for PSoC 3, PSoC 4, or PSoC 5LP. If you are new to these products, introductions can be found in AN54181, Getting Started with PSoC 3, AN79953, Getting Started with PSoC 4, and AN77759, Getting Started with PSoC 5LP. If you are new to PSoC Creator, see the PSoC Creator home page.

If you are new to digital design principles in general, basic concepts are explained in the section What is Digital Design?

If you are familiar with digital design principles and want to see how they are addressed in PSoC devices using PSoC Creator, see the sections Digital Design in PSoC and PSoC Digital Design Considerations.

For specific information on how to read a PSoC Creator STA report file, see Using the PSoC Creator STA Report.

A list of more advanced PSoC digital design resources is given in Related Application Notes.

Note: The examples shown in this application note are designed to produce, under most conditions, the static timing analysis (STA) warnings indicated. However, in some cases, depending on routing and placement, and clock frequencies, a warning may not always be generated. This is especially true for PSoC Creator 2.1, which adds a timing-driven placement (TDP) feature.

2 What is Digital Design?

If you have worked only in the MCU domain, you may not have needed to deal directly with low-level digital design problems. This is because today's MCUs are highly integrated – most, if not all, of the memory and peripherals are located inside the chip with the CPU. The only thing that you have to do is write the code for the CPU to access the memory or peripheral register addresses in the correct manner.

Different from traditional MCUs, PSoC 3 and PSoC 5LP enable you to design your own custom peripherals out of general-purpose digital (and analog) building blocks. However, before you do this, it is helpful to know some digital design basic concepts.

All of the discussions in the rest of this application note are ultimately based on the simple concepts described in this section.

2.1 Digital Design Basic Concepts

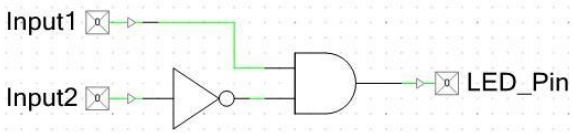
The digital design process is actually similar to writing firmware for CPUs, with the advantage that digital designs can operate much faster than CPUs. For example, consider a case where you want to turn on an indicator LED when one condition is true (logic high or "1") and another is false (logic low or "0"). You can write C code to do the function:

```
if (GetInput1() && !GetInput2())
{
    SetOutput(1); // turn on the LED
}
else
{
    SetOutput(0); // turn off the LED
}
```

which may take many CPU cycles to execute.

However, instead of writing CPU code, you can simply build the function using low-level logic gates, as Figure 3 shows:

Figure 3. Example Digital Logic Function

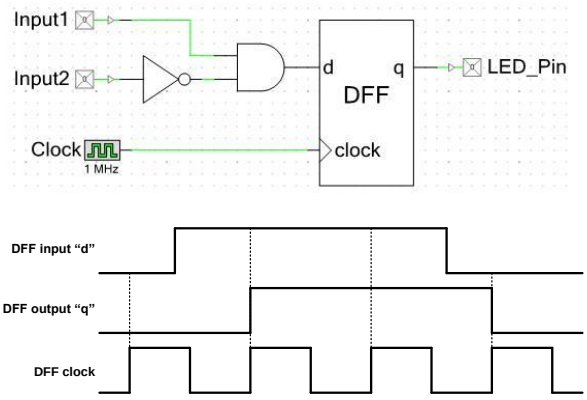


Unlike other MCUs, you can draw the design shown in Figure 3 in a PSoC Creator project schematic, and then program and directly implement it in a PSoC device. The only delays are from the pins, gates and routing, which are in the order of nanoseconds. Of course, no CPU cycles are required.

2.1.1 Registers and DFFs

There is a problem with the design in Figure 3: if the inputs change quickly or simultaneously, you may get unwanted glitching on the output. To reduce the possibility of that happening, we use a clocked register (also known as a “D flip-flop”, or “DFF”), as Figure 4 shows:

Figure 4. Example Clocked Digital Logic Function



The output “q” of the register changes only when the “clock” input transitions from 0 to 1, that is, at the “rising edge”. The output becomes the same as the value at the input “d”, and stays at that value until the next rising edge of the clock.

A clock is usually a continually running square wave of a fixed frequency and 50% duty cycle. PSoC Creator lets you implement in PSoC multiple clocks of any desired frequency up to the device limit – see PSoC 3 and PSoC 5LP Clocks on page 9.

Using a clocked register, you can better control the timing of event detection and response. This is also known as “synchronizing” an asynchronous input to a known clock signal.

If the input changes at the exact same time as the clock edge then the output may be indeterminate for a period of time. This condition is called metastability – more information on this topic can be found in the section Metastability, and Register Timing on page 7.

2.2 Methods of Digital Design

Users of traditional MCUs may program the MCUs at multiple levels:

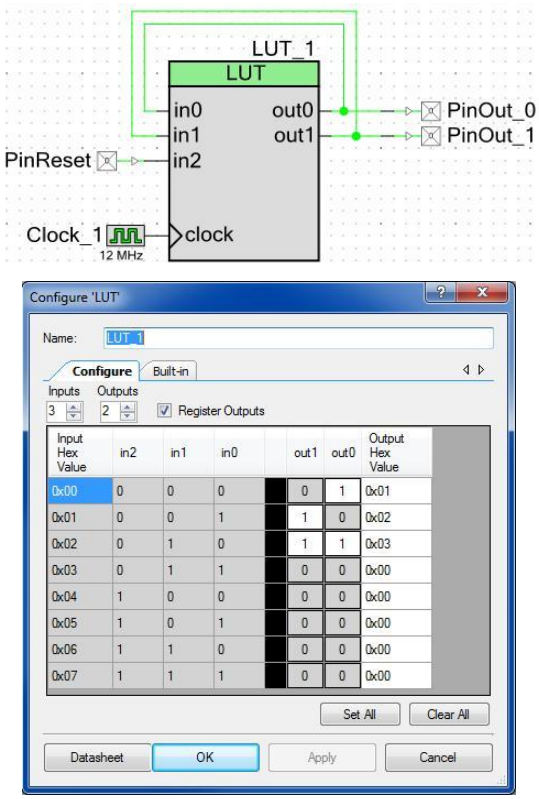
- **Assembler:** highly optimized code at the cost of low coding efficiency and non-portability. Usually done only for specific modules.
- **C:** good coding efficiency and portability. The most frequently used language for coding MCUs.
- **Object-oriented language (Java, C++, etc):** allows definition of custom objects for easy reuse and high coding efficiency. Usually available only on higher-end MCUs, such as 32-bit, with at least 256 K of memory.

Similarly, digital design can be done at multiple levels:

- **Gate-level:** wiring individual gates (AND, OR, XOR, NOT) and DFFs to perform logic functions. PSoC Creator offers gate symbols for all of the logic functions, and also, at a slightly higher level, offers a Lookup Table (LUT) Component.

  Using a LUT makes it easier to design complex logic functions without wiring individual gates. You can also easily design state machines without programming an MCU. Figure 5 shows a 2-bit counter with reset, as an example:

Figure 5. LUT-Based State Machine Design



For more information on designing state machines in PSoC 3 and PSoC 5LP, see AN62510 – Implementing State Machines with PSoC 3 and PSoC 5LP

- **Code-level:** PSoC Creator supports a hardware design language called Verilog, which has a structure and syntax that is similar to C.
- **Datapath programming:** gate, LUT, and Verilog-level designs are usually implemented in the PLDs in the PSoC UDBs (see Figure 2). To program the other portion of the UDBs, such as the datapath, a Datapath Configuration Tool is offered by PSoC Creator. Most UDB-based designs use both datapaths and PLDs.
- **Components:** similar to object-oriented programming, PSoC Creator lets you define custom Components that are easy to implement and reuse. Components can use a variety of PSoC resources including UDB datapaths and PLDs.

Note: Detailed instructions for Verilog and datapath programming, and Component development, are beyond the scope of this application note. For more information on these topics see Related Application Notes.

2.2.1 Digital Design Debugging

Firmware development must include debugging tools and techniques. Some good debugging techniques for firmware are as follows:

- Step through each line of code, and observe its operation, at least once under each possible condition.
- Debug as you develop, that is, write and test a small block of code before going on to the next block.

Similarly, in digital design you may want to look at gate, module, or Component-level performance. The easiest way to do this with PSoC is to route signals of interest to test pins and then observe the pin activity using an oscilloscope or a logic analyzer. Similar to firmware, when debugging digital designs you should do the following:

- Observe each signal at least once, under each possible condition.

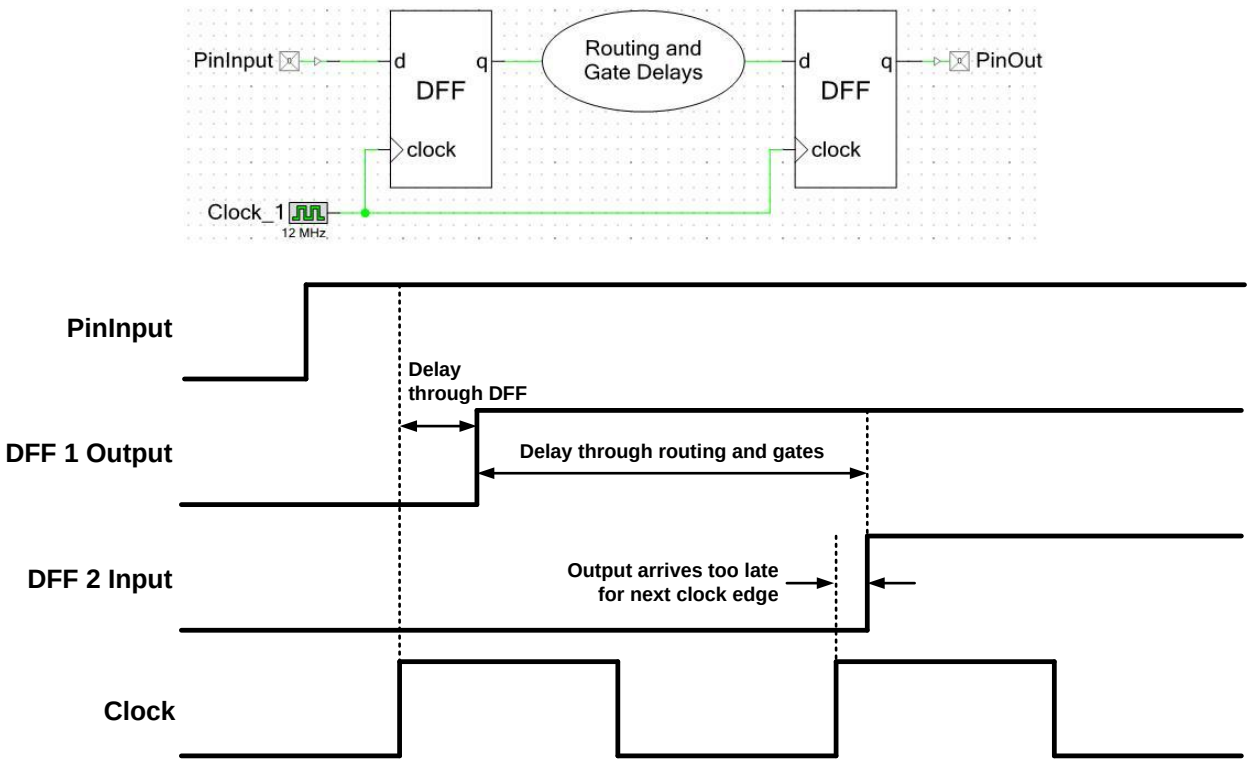
- Debug as you develop. In this case, get a small portion of the design, at the logic or Component level, tested and working before going on to the next portion of the design.

2.3 Timing Issues in Digital Design

The most common problem in digital designs is timing. There are always delays through DFFs, gates, and other logic elements, as well as delays through pins and routing. A timing problem becomes apparent when the delays are long enough to cause a signal to arrive too late (or sometimes too early) for correct processing, as Figure 6 shows.

In Figure 6, the delay through the DFF 1 plus the delay through the logic and routing between the DFFs make the signal arrive too late at the DFF 2 input. To correct it, you must either slow down the clock or reduce the delays between the DFFs.

Figure 6. Delays in Multi-Register Designs

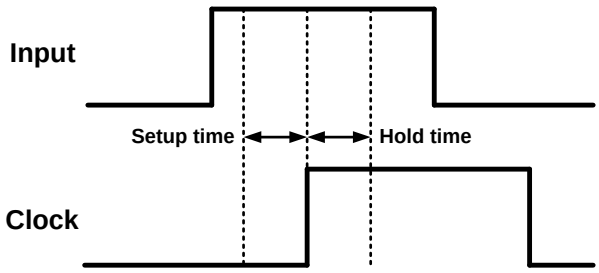


2.3.1 Metastability, and Register Timing

As stated previously, if a register input changes exactly at the same time as the clock edge, then the output may be indeterminate, a condition called metastability.

More precisely, there is a period of time before and after the clock edge during which the input must be stable. That time period is defined by the register's specified setup and hold times, as Figure 7 shows.

Figure 7. Register Setup and Hold Times

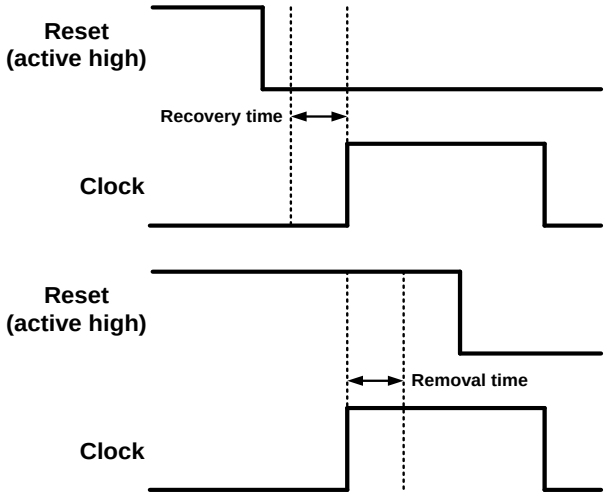


In general, an input should maintain its state throughout the setup and hold times, or the output may be indeterminate - a metastable condition.

Recovery and removal times are similar to setup and hold times, but instead of register inputs, they describe the timing of asynchronous controls relative to the clock. An example of an asynchronous control is the DFF reset input - see Figure 30 and related text on page 20.

Recovery time is the minimum time that an asynchronous control must be inactive before a clock edge. Removal time is the minimum time that the control must remain active after the clock edge. See Figure 8.

Figure 8. Register Recovery and Removal Times



2.3.2 Static Timing Analysis (STA)

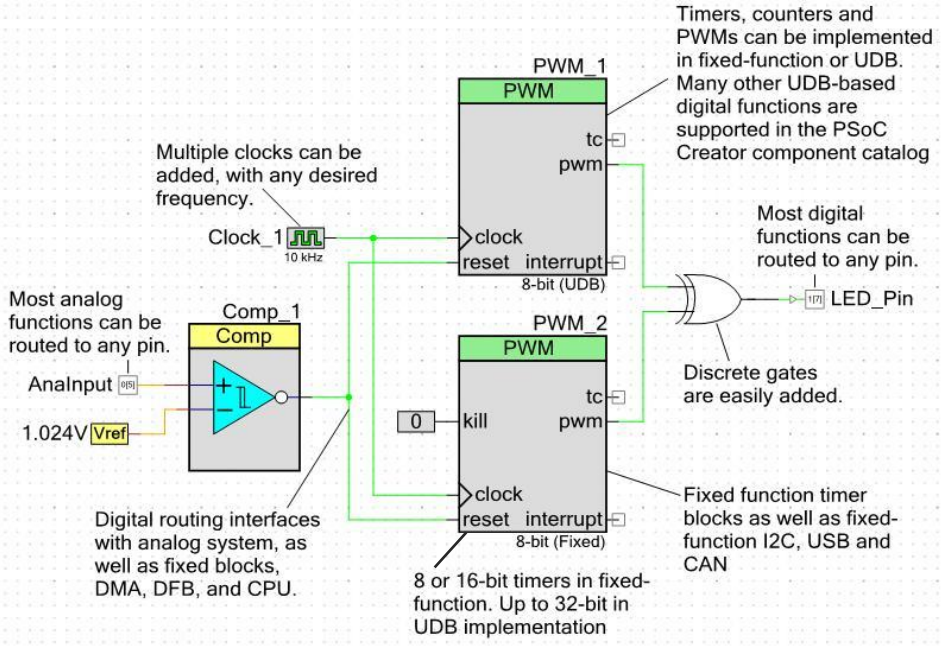
An important part of debugging digital designs is static timing analysis (STA). STA evaluates a digital design and calculates delays between signal outputs and inputs. From those delays it computes the maximum allowable frequency of each clock used in the design. For more information see PSoC Creator Static Timing Analysis.

3 Digital Design in PSoC

Although not described in detail in this application note, it is useful to note how the many flexible features of the PSoC digital subsystem are expressed by PSoC Creator, and how easy it is to quickly create even complex designs.

For example, Figure 9 shows a design for PSoC 3 and PSoC 5LP, where an analog voltage level enables or disables a “breathing” LED. The two PWMs have slightly different periods, which make the LED gradually become dimmer and brighter. With a little knowledge of PSoC Creator, you can build and run a design such as this on a PSoC device in under half an hour.

Figure 9. PSoC Creator Digital Design Example, for PSoC 3 and PSoC 5LP (PSoC 4 is Similar)



In another example, a digital system can be designed to control multiplexed ADC inputs, and interface with DMA to save the data in SRAM, to create an advanced analog data collection system with zero usage of the CPU.

For more information see the PSoC Creator web page, or any example design in PSoC Creator.

3.1 PSoC Digital Subsystem

As mentioned previously, the heart of the PSoC digital subsystem is an array of UDBs (see Figure 1). When you place a digital Component on a PSoC Creator schematic, a set of datapaths, PLDs, and other UDB registers (see Figure 2) are configured to implement the desired functions.

Within and surrounding the UDBs is the DSI – an extensive fabric of programmable switches which connect signals within a UDB, between pairs of UDBs, throughout the array of UDBs, and between the UDB array and many other blocks in PSoC. This is what gives PSoC its tremendous flexibility in constructing intelligent custom peripherals.

When you draw a wire between digital Components on a PSoC Creator schematic, a set of DSI switches between the source and destination points are turned on to implement that connection. Each switch has a delay of 1 to 2 nanoseconds. This is small but if a signal is routed through multiple switches, then the total delay can become significant.

To manage this, PSoC Creator has the following features:

- Timing driven routing (TDR), where signals with critical timing are routed to minimize DSI switch and routing delays
- Timing driven placement (TDP), where UDB resources are selected to further optimize Component timing

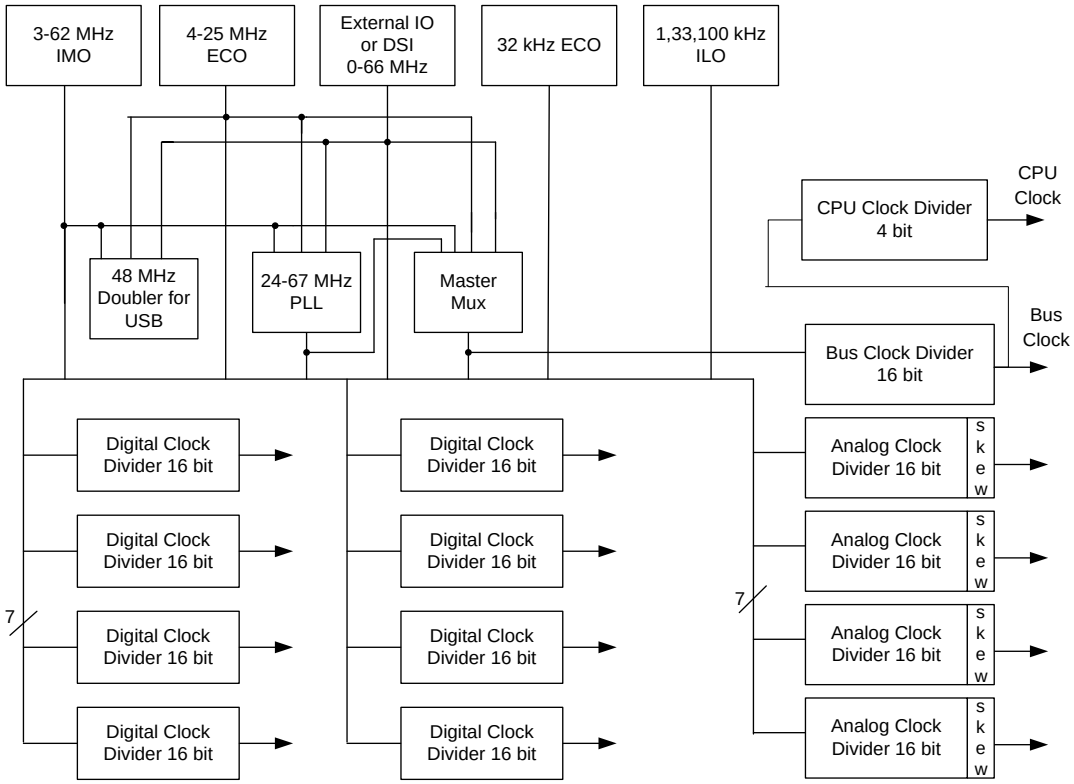
For more information on the PSoC digital subsystem, see one of the device datasheets or Technical Reference Manual (TRM).

3.2 PSoC 3 and PSoC 5LP Clocks

Because a lot of the discussion in this application note centers around synchronizing signals to clocks, let us take a brief look at the PSoC clock system and how it is expressed in PSoC Creator.

PSoC 3 and PSoC 5LP have a highly flexible clock generation and distribution system. Figure 10 shows PSoC 3 as an example; PSoC 5LP is similar. PSoC 4 has a much simpler clock system; for details see the PSoC 4 device datasheet.

Figure 10. PSoC 3 Clocking Subsystem

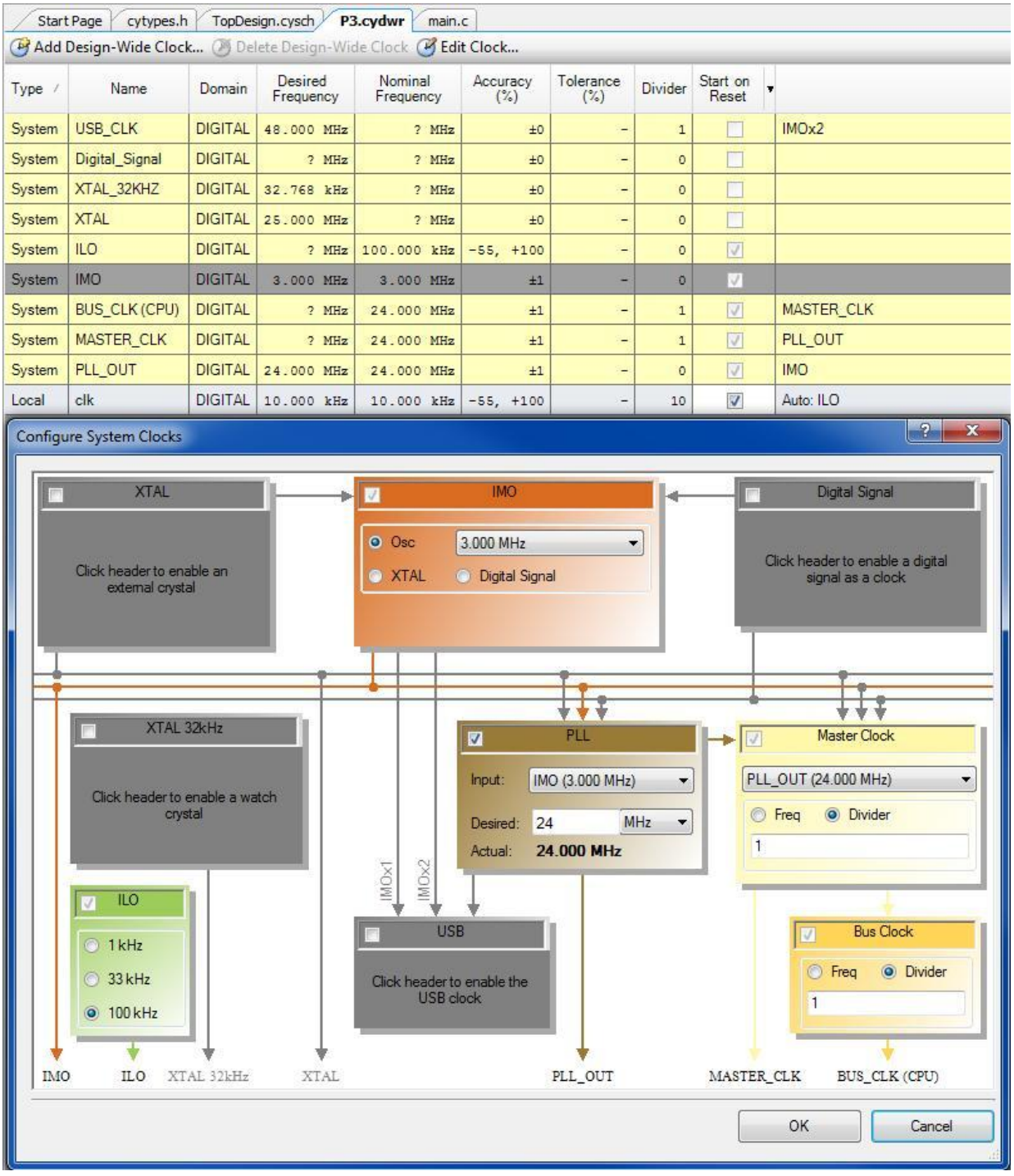


Many different clock sources are available, with different frequency ranges, accuracies, and power requirements. These are routed to the rest of the system, that is, to generate the bus, digital, and analog clocks. For the digital system, eight clock dividers provide clocks of any desired frequency.

The clocking system is set up using PSoC Creator. The user-defined clocks are placed on the schematic page (see Figure 9 on page 8) and the system clock settings are shown on the Design-Wide Resources (DWR) page. Figure 11 on page 10 shows the clock configuration for the example in Figure 9.

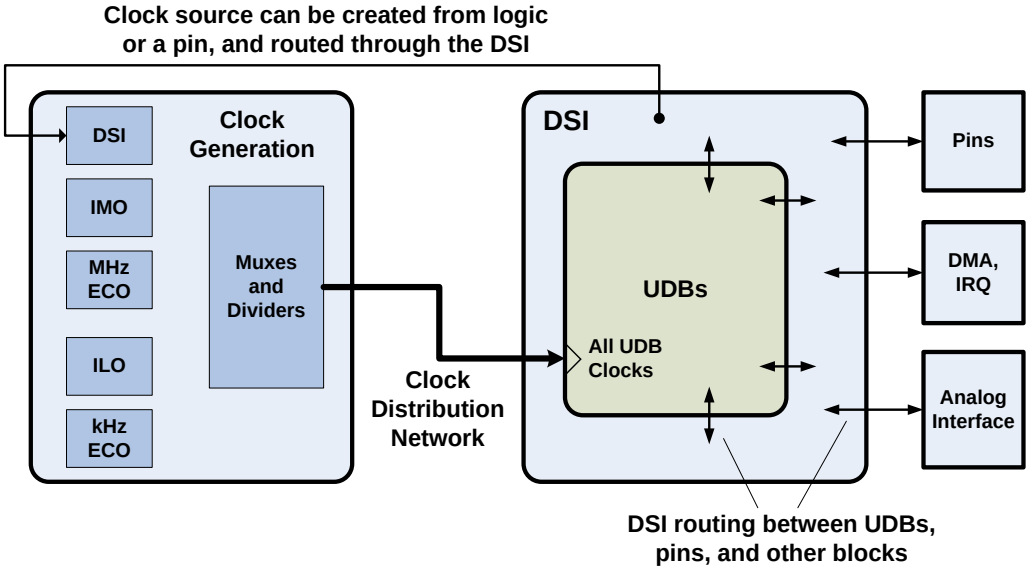
One important point to note is the existence of a master clock and a bus clock, and their use. Because bus clock is usually the same frequency as master clock, the two terms are often used interchangeably. However, they have different functions. The master clock is a source for all other PSoC clocks, including bus clock, using the dividers shown in Figure 10. Its frequency is equal to or greater than all other clocks in the PSoC. The bus clock is the clock source for the CPU, DMA, DFB, and other major blocks.

Figure 11. Example PSoC Creator Clock Configuration for PSoC 3 and PSoC 5LP



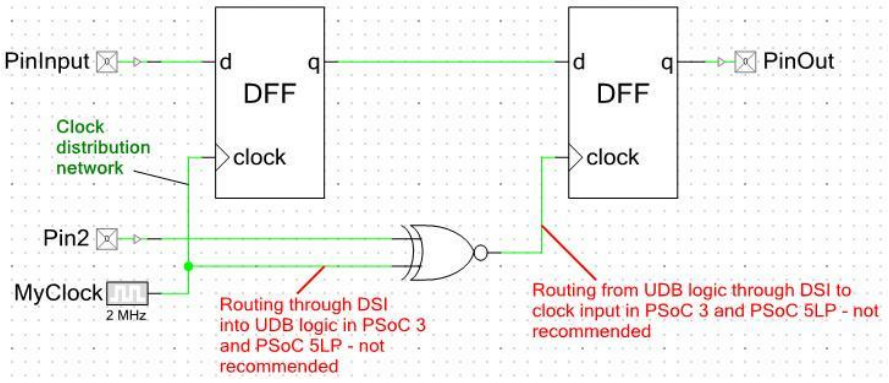
Any clock source can be routed into the UDB clock inputs. The clocks are distributed on a dedicated network (separate from the DSI) to the clock inputs in the UDBs and other blocks, as Figure 12 shows.

Figure 12. PSoC 3 and PSoC 5LP Clock Distribution and DSI



In PSoC 3 and PSoC 5LP, the clocks can also be routed to logic in the UDBs, as Figure 13 shows. However, in general, it is a good idea to tie clock sources, such as clock Components, directly to clock inputs, which takes advantage of the clock distribution network. It is also a good idea to avoid routing clocks into gate or data inputs, and routing gate or data outputs to clock inputs. For details, see Topic #2: Using Clocks on page 15, and Don't Gate Clocks, Use Enable Component on page 17.

Figure 13. Clock Routing Paths



For more information on PSoC clocks see AN60631, PSoC 3 and PSoC 5LP Clocking Resources, or the Clocking chapter in the PSoC Creator System Reference Guide.

3.3 Synchronization in PSoC

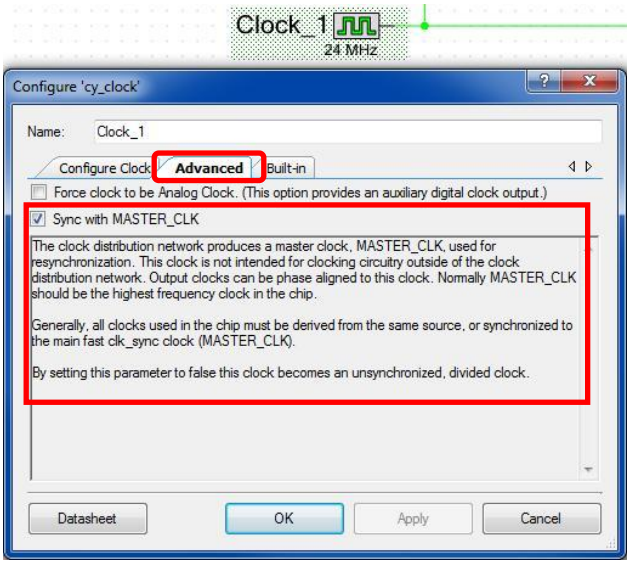
As noted previously, an asynchronous digital system can lead to unwanted glitches in signals. PSoC is designed to operate as a synchronous system, to enable communication between the digital subsystem and the CPU and DMA. Generally, asynchronous signals are not supported except for PLD-based logic that does not interact with the CPU or DMA.

Following are several instances of synchronization in PSoC that you should note.

3.3.1 PSoC 3 and PSoC 5LP Clocks

As noted previously, eight clock dividers are available in PSoC 3 and PSoC 5LP to provide to the digital subsystem clocks of any desired frequency. As a default, these clocks are synchronized to the master clock. This is controlled through the PSoC Creator Clock Component configuration dialog, as Figure 14 shows.

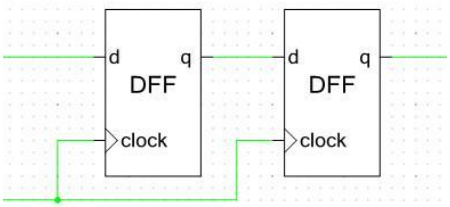
Figure 14. Clock Synchronization Settings for PSoC 3 and PSoC 5LP



3.3.2 Pins (all PSoC Devices)

Signals coming into an input pin are considered to be asynchronous, and as a default are double-synchronized before being routed to the DSI. Double-synchronization, as shown in Figure 15, further reduces the chance of having metastability and unexpected system behavior.

Figure 15. Double Synchronization

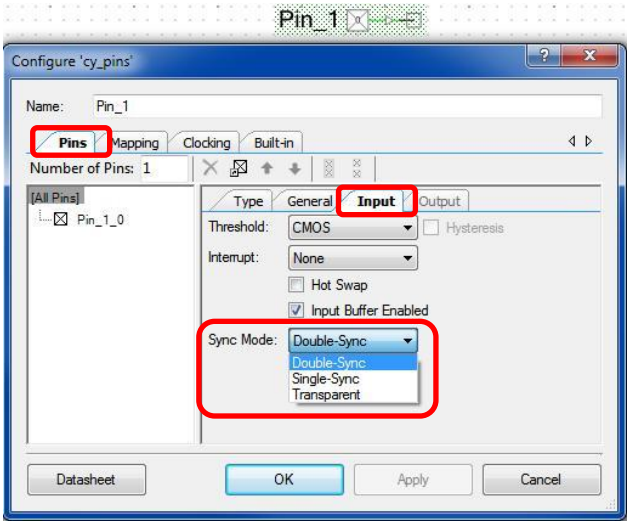


Note that the delay through a double synchronizer is 1 to 2 cycles of the synchronizing clock. In PSoC 3 and PSoC 5LP, the synchronizing clock is bus clock, which is usually the same frequency as the master clock. In PSoC 4, the synchronizing clock is HFCLK.

If the synchronizing clock is less than or equal to 33 MHz, dedicated circuitry within the pin block is used for double synchronization. If the bus clock is greater than 33 MHz, the pin input is routed through the DSI to UDB circuitry and the double synchronization is done in the UDBs. This can cause routing and STA results to vary when master clock, bus clock, or HFCLK are changed to be above or below 33 MHz. See, for example, the discussion around Figure 50 on page 36.

As noted previously, a clock source can be external to PSoC, coming in through an I/O pin. If that input is a source for bus clock, then input synchronization must be turned off. This is done in PSoC Creator through the Pins Component configuration dialog, as Figure 16 shows – set the Sync Mode to Transparent.

Figure 16. Input Pin Synchronization



In general, pin inputs should be synchronized; they must be synchronized to be read correctly by the CPU or DMA. However, there are some cases when timing with systems external to PSoC is critical. In these cases, synchronizing delays may be unacceptable and input pin synchronization should be turned off. For more information see Topic #6: Interface with Pins on page 22.

Pin outputs also have a synchronization option; they can be (single) synchronized to bus clock. This option exists mainly to reduce skew, or relative delay, between multiple outputs. It is disabled as a default.

3.3.3 UDB Control and Status Registers

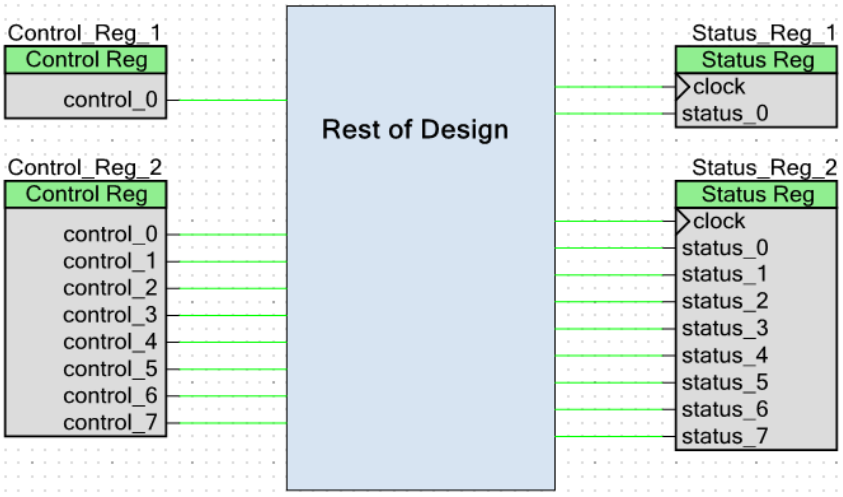
Two other major UDB Components are a control register and a status register – a UDB has one of each. A control register lets the CPU / DMA drive signals into the digital subsystem, and a status register lets the CPU / DMA read signals from the digital subsystem. PSoC Creator Components exist for both registers. For more information on these registers see the TRM.

In addition to their normal modes of operation, these registers have special modes that can be used to reduce timing restrictions. For example, a status register can be reconfigured to be a 4-bit double synchronizer – to take advantage of this feature, use the PSoC Creator Sync Component, as shown in Figure 22 on page 17.

In another example, the control register can operate in three different modes – direct, sync, and pulse. The sync and pulse modes allow input from the CPU / DMA, which is synchronized to the bus clock, to be resynchronized to another clock. For details, see Topic #3: Control Registers on page 19.

Note: The PSoC Creator Control and Status Register Components can be configured as one to eight control outputs or status inputs, as Figure 17 shows. Although it is possible to create multiple single Control or Status Components, each Component uses an entire UDB register. A better method is to use Control or Status Components with multiple outputs or inputs. This is especially true for status inputs which have a common clock.

Figure 17. Single and Multiple-Wire Control and Status Registers



3.4 PSoC Creator Static Timing Analysis (STA)

Whenever a PSoC Creator project is built, an STA is done automatically. A report is generated showing the critical paths in the design that limit the frequency of each clock. If an actual clock frequency exceeds the calculated maximum frequency, a warning is generated indicating that a timing violation exists in the design.

It is important to know about synchronization and other digital features in PSoC, to avoid STA warnings, as detailed in the next section.

4 PSoC Digital Design Considerations

This section contains a series of topics that describe considerations and best practices for doing digital designs in PSoC 3, PSoC 4, and PSoC 5LP with PSoC Creator.

4.1 Topic #1: Component Datasheet Specifications

When you use one of the digital Components offered in the PSoC Creator Component Catalog that are built from UDBs (for example, counter or SPI), it is important to check the Component's timing specifications. These are available in the Component datasheets, as Figure 18 shows.

Figure 18. UDB Counter Timing Specifications

Table 2. AC Characteristics

Parameter	Description	Min	Typ	Max[2]	Units
f_{CLOCK}	Component clock frequency				
	8-bits UDB Up Counter	–	–	39	MHz
	8-bits UDB Counter with direction	–	–	39	MHz
	16-bits Up Counter	–	–	33	MHz
	16-bits Counter with direction	–	–	33	MHz
	24-bits UDB Up Counter	–	–	29	MHz
	32-bits UDB Up Counter	–	–	26	MHz

Depending on the Component type, the clock frequency for a UDB-based Component is limited due to the routing delays mentioned previously. It is important that the frequencies of your clocks do not exceed the datasheet maxima.

For more information, check any of the digital Component datasheets.

4.2 Topic #2: Using Clocks

4.2.1 Use the Slowest Possible Clock

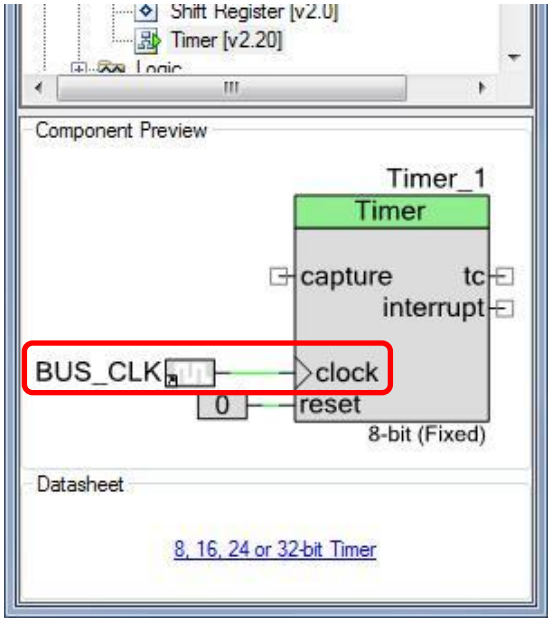
As mentioned in the previous topic, the frequency of clocks for PSoC Creator Components should be limited according to the Component datasheet specifications.

In general, the speed of all clocks should be kept as low as possible while still meeting your application requirements. This gives PSoC Creator's TDR system more freedom to optimize routing of the critical portions of your design. It may also reduce system noise and power.

4.2.2 Clock Settings in Macros

In some cases, inclusion of a high-frequency clock is not immediately apparent, as Figure 19 shows. Some macros in the PSoC Creator Component Catalog include Clock Components, which are set to bus clock as a default. The clock frequency should be changed if bus clock speed is not needed.

Figure 19. Timer Macro Includes Bus Clock



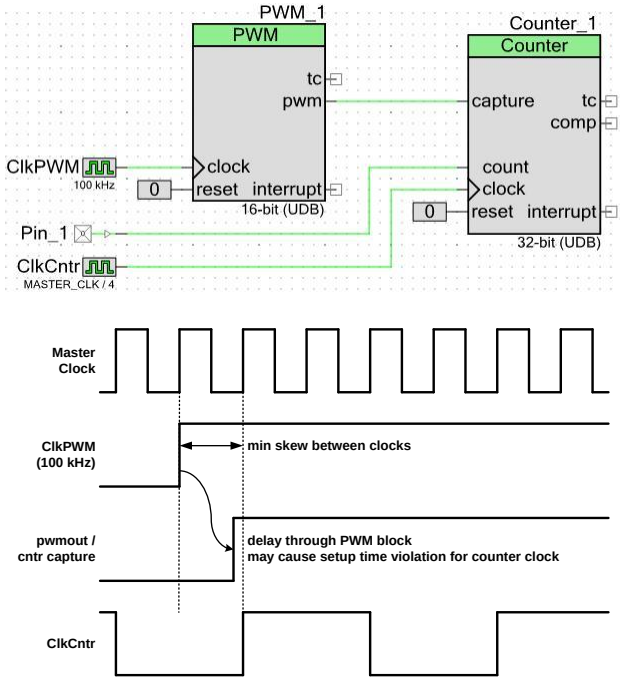
4.2.3 Multiple Clocks

As mentioned previously, most digital signals in PSoC are synchronized. Even most clocks are synchronized, to other higher frequency clocks. Clocks are usually, but not always, synchronized to the master clock; see Figure 10.

Although multiple clocks may be synchronized to the same source, Figure 20 shows that they are not necessarily synchronized to each other. In this example, which shows a frequency counter with a periodic capture from the PWM, the two clocks are synchronized to the master clock (HFCLK in PSoC 4), but not necessarily to each other. Thus, if the master clock frequency is high enough, delays through the PWM Component may cause a setup time violation at the counter Component.

Note: The examples shown in this and other sections are designed to produce, under most conditions, the STA warnings indicated. However in some cases, depending on routing and placement, and clock frequencies, a warning may not always be generated.

Figure 20. Multi-Clock Example



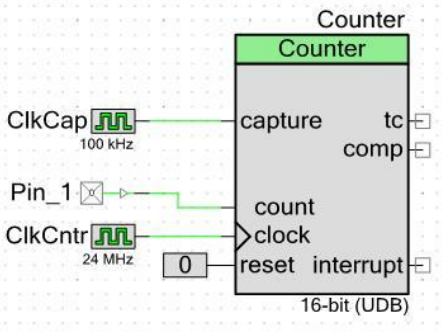

```
Warning-1366: Setup time violation found in a path from clock (ClkPWM) to clock
(ClkCntr).
```

In this case, decreasing the ClkCntr frequency does not correct the problem, because the minimum time between clocks is one master clock period. To correct the problem, either the master clock frequency must be reduced, or the PWM output must be synchronized to ClkCntr, as discussed in the next section. You can also refer to Using the STA Report to Remove Warnings on page 33.

4.2.4 Synchronizing Clocks with Sync Component

In a simpler example, shown in Figure 21, a low-frequency clock is used to directly control the counter capture frequency. This can cause a synchronization warning from the STA.

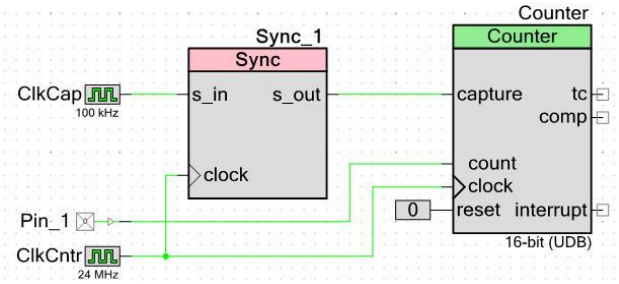
Figure 21. Another Multi-Clock Example




```
Warning-1350: Path(s) exist between clocks ClkCap(routed) and ClkCntr, but the clocks
are not synchronous to each other
```

To clear this warning, synchronize the low-frequency clock to the counter clock, using a PSoC Creator Sync Component, as Figure 22 shows.

Figure 22. Synchronizing Clocks



Note that because the Sync Component does double-synchronization (see Figure 15 on page 12 and UDB Control and Status Registers on page 13), the duty cycle of the sync output is never exactly 50%. This is acceptable when the synchronizing clock is much faster than the synchronized clock. However, when the two clocks are nearly the same frequency, the sync output may not generate a clock at all – it may just stay at a DC level. The synchronizing clock must be more than two times faster than the synchronized clock; four times faster is preferred.

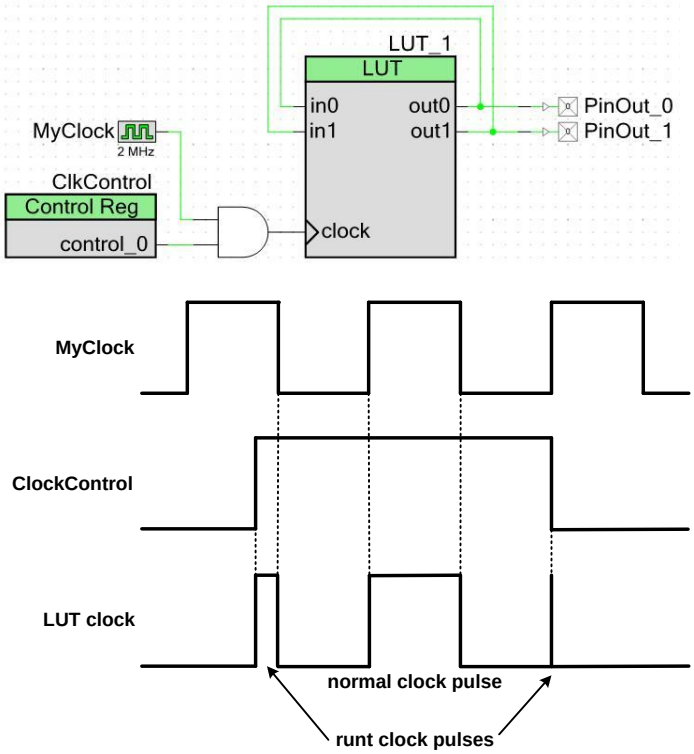
For more information on the Sync Component, see the Sync Component datasheet.

Note: In Figure 21 and Figure 22, with PSoC 4 ClkCap cannot be directly tied to the Counter 'capture' terminal or the Sync 's_in' terminal. A good workaround is to double the frequency of ClkCap (200 kHz) and run it through a toggle flip-flop (TFF) Component first.

4.2.5 Don't Gate Clocks, Use Enable Component

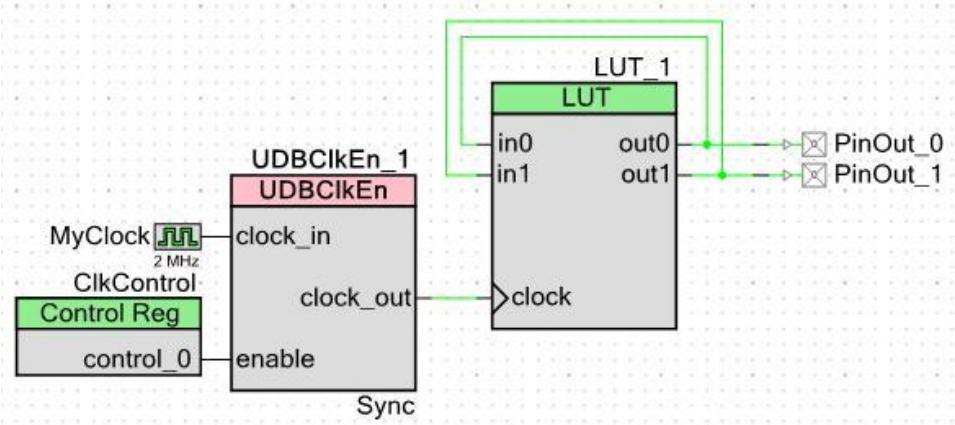
In many designs, a function is controlled by simply turning its clock on or off, by gating the clock. However, the example in Figure 23 shows how gating a clock with an unsynchronized control can cause runt clock pulses and yield unpredictable results.

Figure 23. Gating Clocks Example



If your function is UDB-based (as is the example in Figure 23), a better method is to use the UDBClkEn Component, as Figure 24 shows. This Component offers a controlled, level-sensitive enable for a clock, and can optionally synchronize the input clock to bus clock. For more information on the UDBClkEn Component, see the UDBClkEn Component datasheet. With PSoC 4, a clock can be directly routed to a UDBClkEn 'clock_in' terminal.

Figure 24. Using the UDBClkEn Component



Note: The Digital Function Components (see Topic #5: Using Digital Function Components on page 21) all have a UDBClkEn Component embedded in them. Therefore, a UDBClkEn Component usually only needs to be used with a DFF or LUT Component. A UDBClkEn Component cannot drive another UDBClkEn Component; attempting to do so gives one or more instances of the following error:

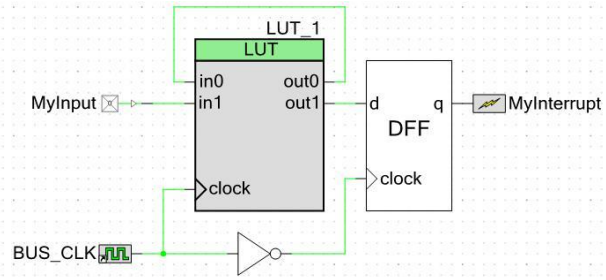
```
Error: mpr.M0096: The UDB Clock/Enable components may only drive clocks in UDB content
and \UDBClkEn_1:udbclkenable\ is connected to clock_in on ... which is not a clock
input to the UDB. (App=cydsfit)
```

4.2.5.1 Inverted clocks are acceptable.

This topic does not usually apply to using inverted clocks, as Figure 25 shows. This is because each UDB has a clock control module, which allows use of a noninverted or an inverted clock – the inverter symbol simply causes the UDB to use its inverted clock input.

Using inverted clocks may allow you to use a slower clock because both edges of the clock are used. However, conversely, timing problems may be more prevalent because clocking happens at twice the frequency.

Figure 25. Example Design with Inverted Clock

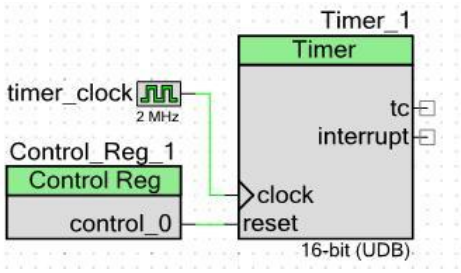


4.3 Topic #3: Control Registers

As noted in UDB Control and Status Registers on page 13, the standard way for firmware to interact with the digital system is through control registers. Each UDB has one 8-bit control register; with 24 UDBs as many as 192 bits of control are available. Control register outputs can be routed through the DSI to any desired point in the digital system.

However, there are some clocking considerations with control registers that may lead to STA warnings. For example, in Figure 26 a control register is driving the reset input of a UDB-based timer. What is not apparent is that by default the control register is clocked by bus clock. With a high frequency bus clock you can get setup time violations between bus clock and the timer clock:

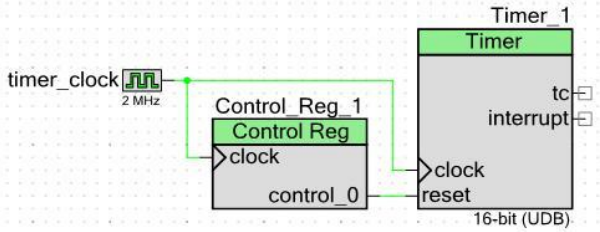
Figure 26. STA Error With Control Register




```
Warning-1366: Setup time violation found in a path from clock (CyBUS_CLK) to clock
(timer_clock).
```

This is easy to fix by putting the control register in Sync mode, as Figure 27 shows:

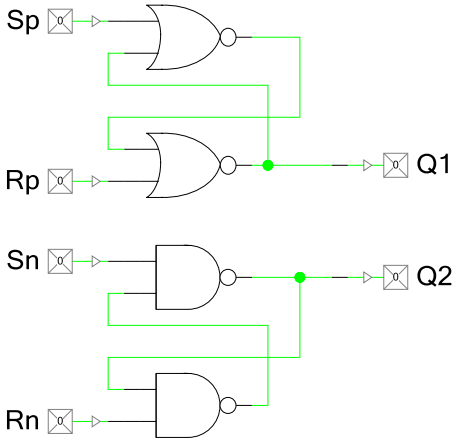
Figure 27. Control Register Sync Mode



4.4 Topic #4: Don’t Make Latches

Figure 28 shows classic set-reset (S-R) latch designs. (In the top circuit, S and R are active high and in the bottom circuit, S and R are active low.) A latch is similar to a register or DFF but is less stable – a glitch pulse on an input may cause the output to change unexpectedly. Also, it has an unstable state if both inputs are active at the same time. It is good practice to use registers instead of latches.

Figure 28. Classic S-R Latch Designs



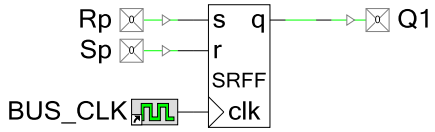
If you try to build one of the above designs, PSoC Creator gives the following warning:

```
Warning-1361: The design contains a combinational loop. Check the design for
unintentional latches. Breaking the loop at ...
```

which means that STA does not include the loop in its analysis.

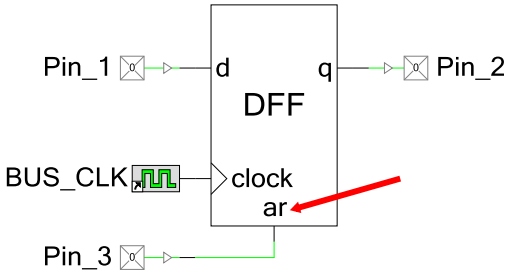
The designs in Figure 28 have other problems, in that they are not synchronized (see discussion around Figure 3 and Figure 4), and they use two UDB PLD macrocells for each latch. A better alternative is to use the PSoC Creator SRFF Component, as Figure 29 shows. This design uses only one macrocell per latch.

Figure 29. Synchronized S-R Latch Component



Note that the latch in Figure 29 can be designed using the LUT Component; see Figure 5. It can also be designed with a DFF Component, and the PSoC Creator DFF Component can be configured for reset or preset, as Figure 30 shows.

Figure 30. DFFs With Resets

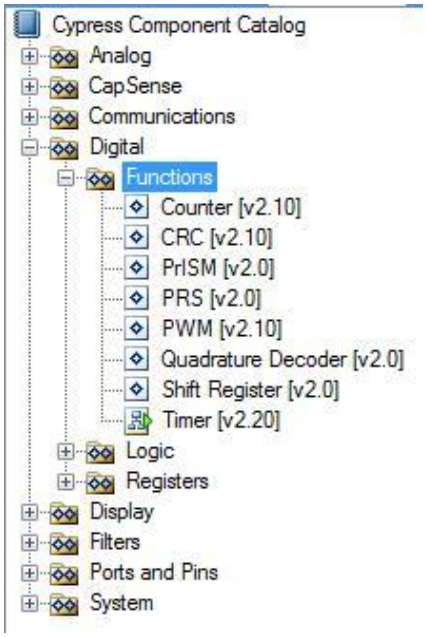


In the configuration dialog box, you can select asynchronous or synchronous preset or reset. The asynchronous inputs take effect as soon as they are asserted, which can cause metastability problems – see recovery and removal times discussion in Metastability, and Register Timing on page 7. The synchronous inputs take effect at the next clock edge and thus have a predictable behavior. For more information, see the DFF Component datasheet.

4.5 Topic #5: Using Digital Function Components

Most of the STA warnings generated by PSoC Creator are related to use of Components in the Digital Functions folder in the Component Catalog; see Figure 31. These Components are unique in that they all have clock inputs.

Figure 31. Digital Function Components



UDB datapaths, PLDs, and other blocks are clocked by multiple sources. The digital function Components take advantage of this feature, and are synchronized to their clock inputs as opposed to bus clock. This can cause STA warnings when unsynchronized signals are connected to the other inputs, as mentioned in Synchronizing Clocks – see Figure 21 and Figure 22 on page 16.

This topic does not apply to the fixed-function timer blocks, even though they have clocks and are supported by the Counter, Timer, and PWM Components. For more information, see the TRM or the datasheets for these Components.

4.6 Topic #6: Interface with Pins

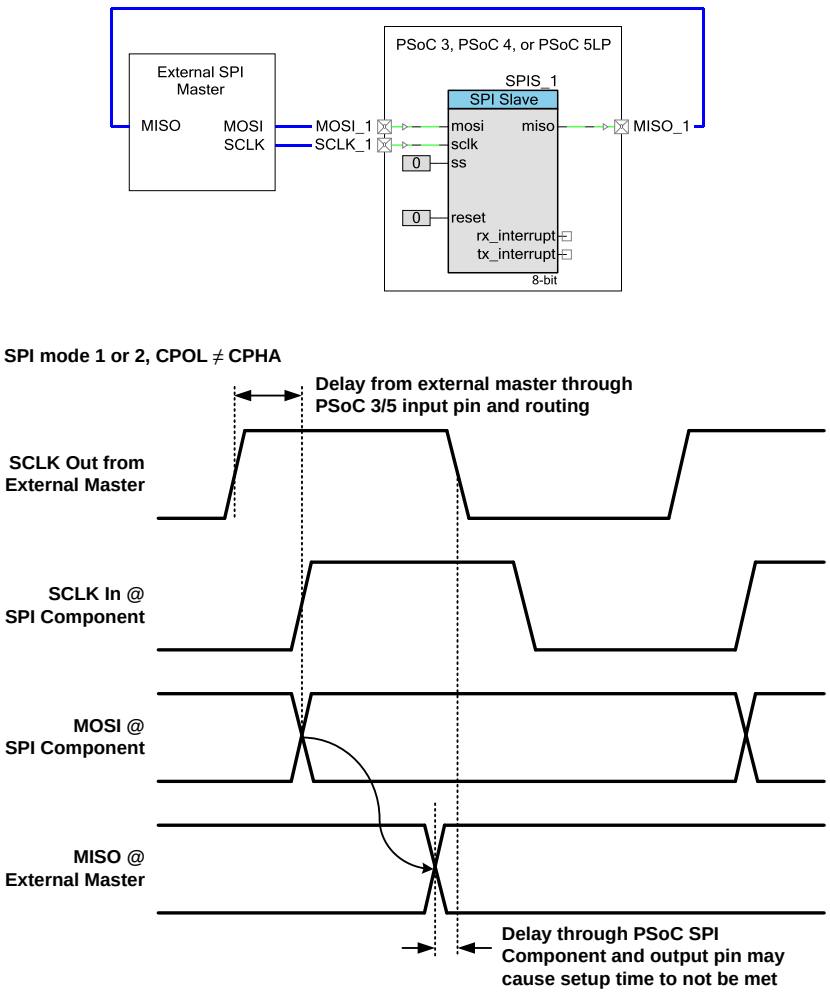
The PSoC 3 and PSoC 5LP I/O pins can be connected to many sources / sinks within PSoC, including CPU, DMA, analogs, and LCD. They can also be connected to the DSI and thus their signals can be routed throughout the digital fabric. This makes pins subject to the same synchronization issues as all other Components. Pin inputs and outputs can be synchronized or not, independent of each other.

As noted in Pins on page 12, pin input signals can be double-synchronized to the bus clock. The default is synchronization enabled however there are special cases where input synchronization should be turned off.

The best example of this is with the PSoC Creator SPI Slave (SPIS) Component, as Figure 32 shows. Delays in the PCB routing and PSoC pins, DSI routing, and UDBs cause an overall delay from the master's SCLK and MOSI outputs to its MISO input. This, in turn, may cause the master's MISO input to be read incorrectly.

A similar issue exists with the PSoC Creator SPI master (SPIM) Component.

Figure 32. SPI Timing with SPIS Component



Little can be done about the delays within the PSoC Creator SPIS Component. However, synchronizing the MISO, SCLK, and MOSI pins can add significant delay. This is especially true for the input pins – as noted in Pins on page 12 the input pins are double-synchronized, which can cause a lot of delay. In general, SPI pins should have synchronization turned off.

All of the PSoC Creator SPI Components are available as macros, which include Pin Components where synchronization is already turned off. If you do not use the macros, that is, if you use the SPIM or SPIS Components directly, you should make sure that the pins connected to these Components have synchronization turned off.

4.7 Topic #7: Interface with Fixed Blocks

As Figure 1 shows, PSoC 3 and PSoC 5LP have several interfaces between the UDB system and the fixed digital and analog subsystems. Figure 33 shows some examples of interfaces between fixed blocks and UDB-based Components (in this case Counters). Note that the Timer Component is based on a fixed-function Timer / Counter / PWM (TCPWM) block.

At this time PSoC Creator's STA feature does not include signals routed between the analog and digital subsystems. Moreover, due to a possible timing problem, the comparator and ADC outputs may not be captured properly at the counter inputs.

Also, you may get STA warnings for an asynchronous path from a fixed-block clock to a UDB clock, for example:

```
Asynchronous path(s) exist from "Clock_1(fixed-function)" to "Clock_1".
```

This is due to the timing differences between the clocking network and the DSI; for more information see Figure 12 and related text on page 11. In general you should add Sync Components to fixed block outputs that are routed to UDB-based Components, as Figure 34 shows.

Figure 33. Fixed Block / UDB Interfaces

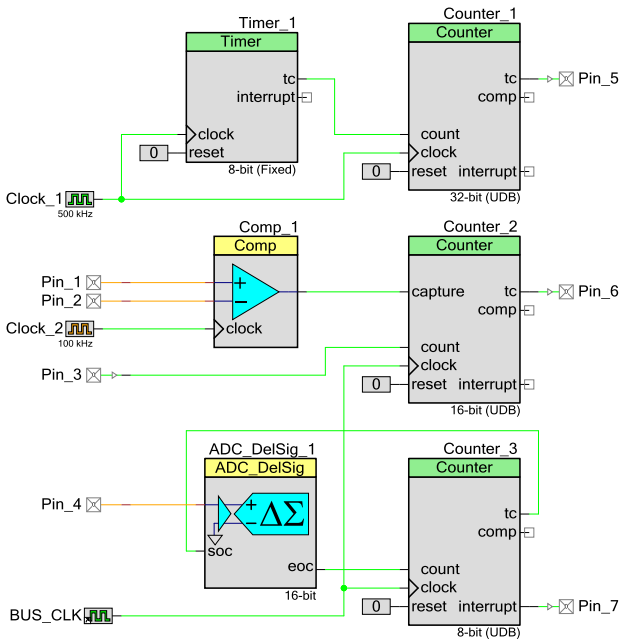
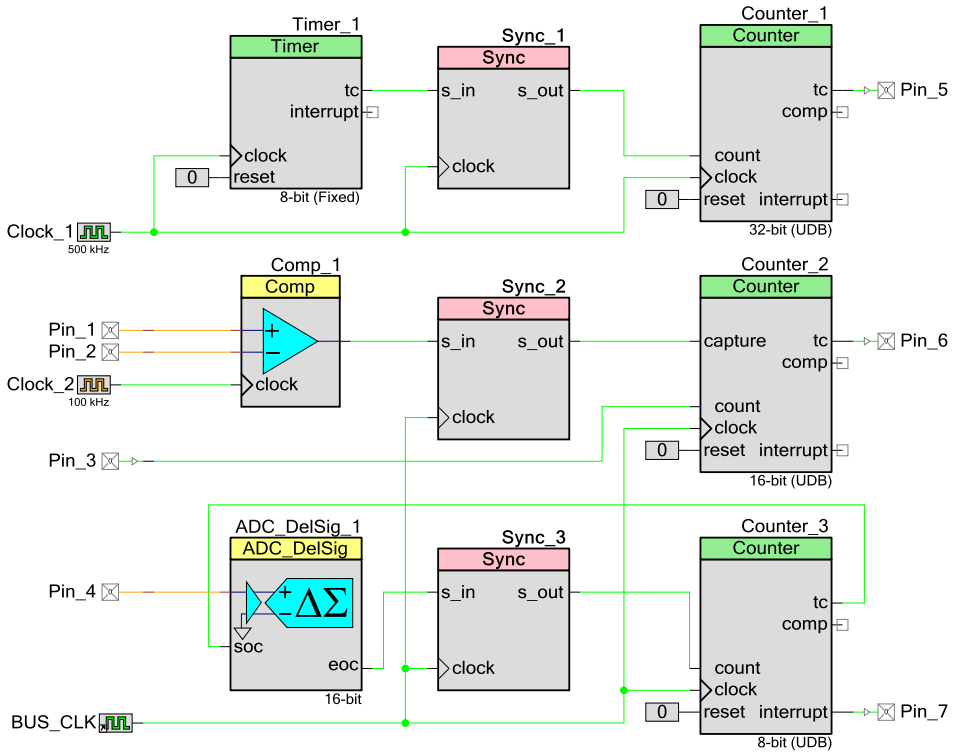


Figure 34. Fixed Block Interfaces with Sync Components



5 Using the PSoC Creator STA Report

As mentioned previously, the PSoC Creator STA feature evaluates a digital design and generates a report on the maximum allowable frequency of each clock and whether any potential timing problems exist. If you get an STA warning during a project build, you may need to review the STA report to fully understand and resolve the warning.

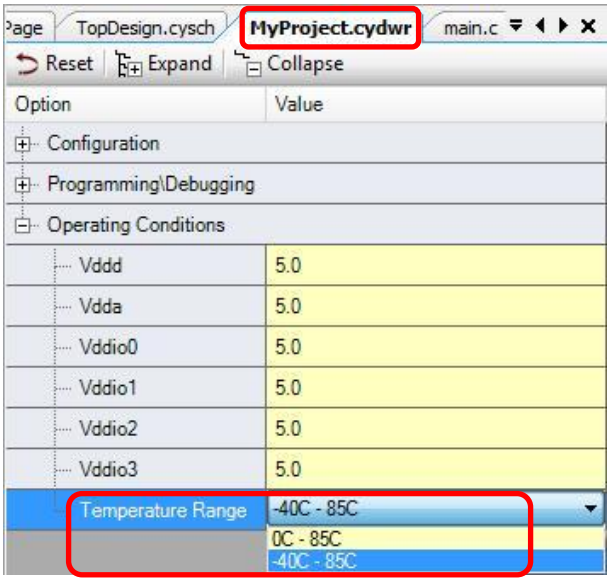
This section discusses strategies and best practices for using the STA report.

Note: Although it is possible for PSoC 4 based designs to produce STA warnings, PSoC 3 and PSoC 5LP are more likely to do so due to their more complex digital and clocking systems. Therefore this section is oriented toward PSoC 3 and PSoC 5LP based designs.

5.1 Setting STA Temperature Conditions

PSoC devices operate across a wide range of supply voltages and temperatures, and TDR constraints are selected with the full range of temperature specifications in mind. However, if your design runs in a more controlled temperature environment, the TDR constraints are more easily satisfied and more flexible routing may be achieved. The temperature range for your design is a PSoC Creator project system setting, as Figure 35 shows.

Figure 35. Device Temperature Range Setting

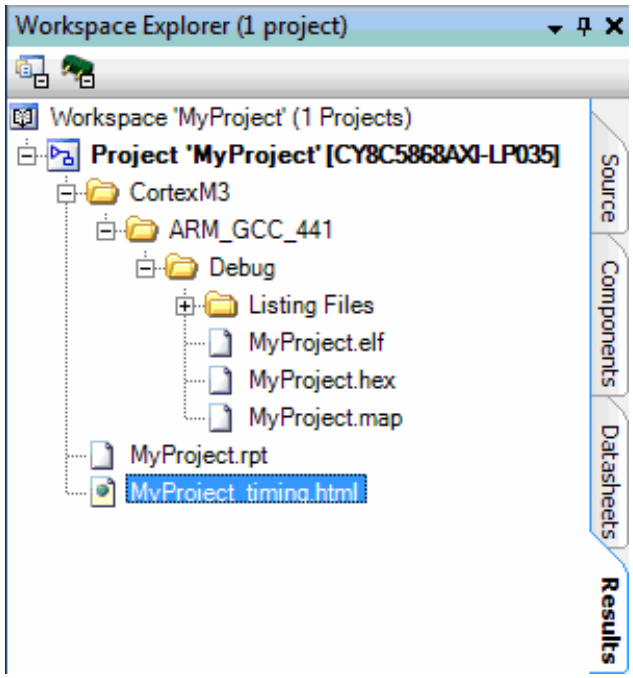


If you select the narrower temperature range for your project, the TDR tool can more easily find timing-compliant routing solutions. This may in turn affect whether or not you get STA warnings when the project is built.

5.2 Finding the STA Report

The STA report is an auto-generated HTML file, and can be found in the PSoC Creator project Results tab, as Figure 36 shows.

Figure 36. PSoC Creator Project STA Report File



5.3 Elements of an STA Report

In addition to timing violation and clock summaries, the STA report has several sections that show detailed analyses of setup, hold, recovery, and removal times between clocks and other signals. For more detailed information on sections of the STA report, see the PSoC Creator Help article “Static Timing Analysis”.

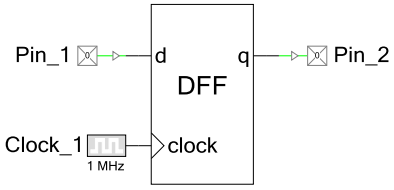
Following are some examples that show in detail the relationship between the design and the STA report data.

5.3.1 Example #1

Let us start with a very simple example, shown inFigure 37 on page 26. In this example, all Component and clock settings are the defaults for PSoC 3 and PSoC 5LP:

- Master clock and bus clock are 24 MHz
- Pin_1 has input synchronization turned on (see Pins on page 12)
- Pin_2 has output synchronization turned off
- Clock_1 is synchronized to master clock

Figure 37. STA Report Example #1



Static Timing Analysis

Project : Fig36
Build Time : 07/22/13 17:10:53
Device : CY8C3866AXI-040
Temperature : -40C - 85/125C
Vdda : 5.00
Vddd : 5.00
Vio0 : 5.00
Vio1 : 5.00
Vio2 : 5.00
Vio3 : 5.00
Voltage : 5.0
Vusb : 5.00

Expand All | Collapse All | Show All Paths | Hide All Paths

+ Timing Violation Section

No Timing Violations

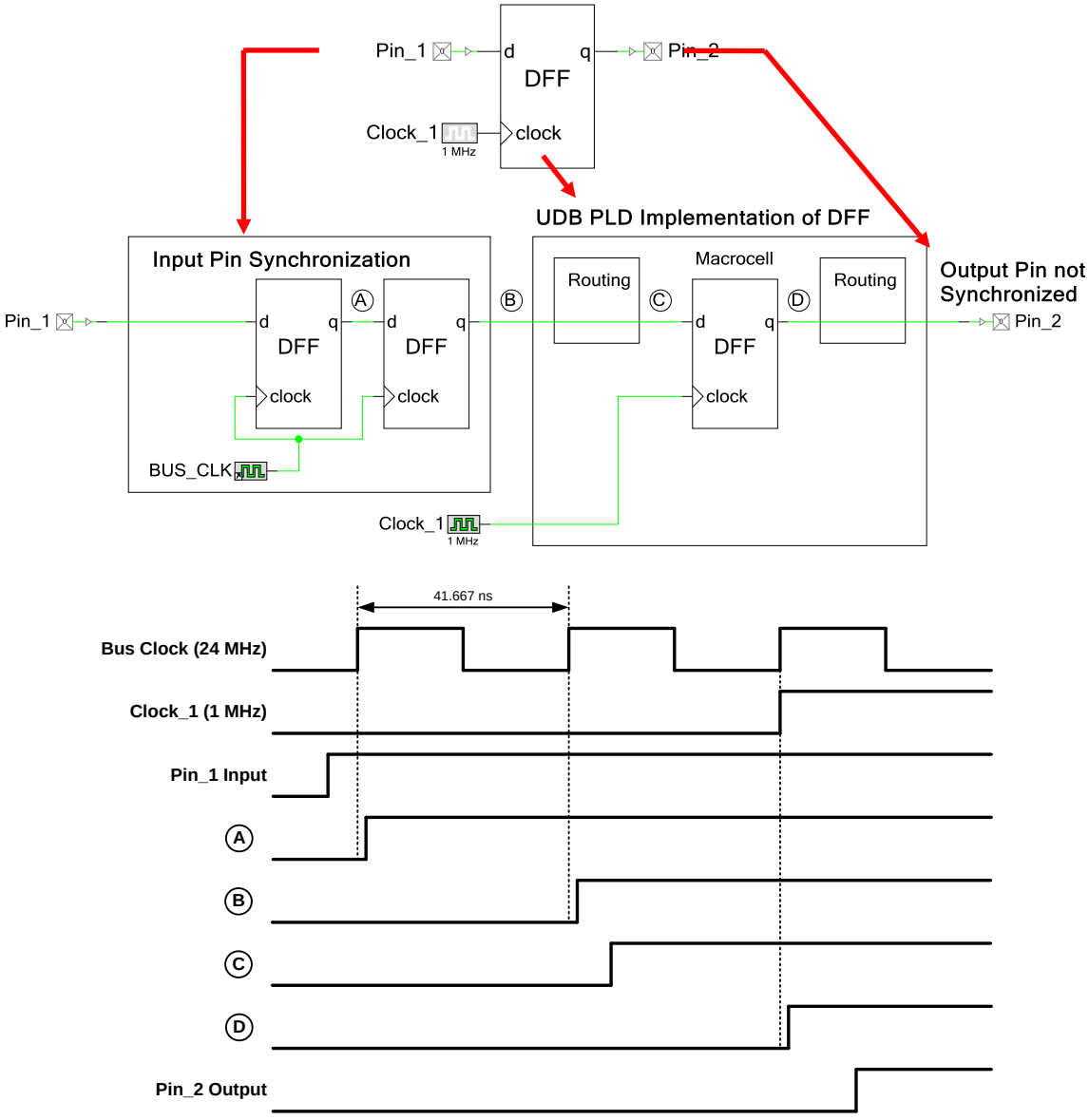
+ Clock Summary Section

Clock	Domain	Nominal Frequency	Required Frequency	Maximum Frequency	Violation
CyILO	CyILO	1.000 kHz	1.000 kHz	N/A	
CyIMO	CyIMO	3.000 MHz	3.000 MHz	N/A	
CyMASTER_CLK	CyMASTER_CLK	24.000 MHz	24.000 MHz	N/A	
Clock_1	CyMASTER_CLK	1.000 MHz	1.000 MHz	69.677 MHz	
CyBUS_CLK	CyMASTER_CLK	24.000 MHz	24.000 MHz	69.677 MHz	
CyPLL_OUT	CyPLL_OUT	24.000 MHz	24.000 MHz	N/A	

When the STA report is opened, it shows a summary of timing violations and a list of clocks with clock speed violations. In this example there are no violations. The report shows that bus clock and Clock_1 could go as fast as 69.677 MHz (which actually may be higher than the device frequency limit) without clock speed violations.

Before we look at the STA report timing details, it is important to see the relationship between the design on the schematic and what actually happens in the PSoC device. In Figure 38 on page 27, the Input Pin Synchronization block shows that, since input pin synchronization is on, the input pin signal goes through two registers which are clocked by bus clock. Then, the signal is routed from the pin through the DSI to a UDB macrocell, which implements the DFF in the schematic. Finally, the macrocell output is routed through the DSI to the output pin, which is not synchronized.

Figure 38. Example #1 Hardware Implementation



For details on the delays in the STA report, click on the "Expand All" (see Figure 37). Figure 39 on page 29 shows the expanded report, and the significance of the numbers in the detail subsections:

- **Setup Subsection**, the “Path Delay Requirement”: 41.6667 ns, corresponding to the 24 MHz bus clock
- In the table in the same subsection, “Delay” listing: 14.352 ns, the calculated delay from the Pin_1 output to the macrocell input, due to routing and gate delays
- In the same table, “Slack” listing: 27.315 ns, or 41.667 minus 14.352 – the calculated setup time for the macrocell
- **Clock to Output Section** “Delay” listing: 22.883 ns, the calculated delay from the macrocell output to Pin_2, due to routing delays.

Further expansion is possible by clicking the "Show All Paths", but is usually not necessary. In Figure 39 the **Hold Subsection** is collapsed – for details on hold violations see Hold, Recovery and Removal Violations on page 41.

Figure 39. Expanded STA Report #1, and Corresponding Timing

Static Timing Analysis

Project : MyProject
Build Time : 07/23/13 15:55:08
Device : CY8C3866AXI-040
Temperature : -40C - 85/125C
Vdda : 5.00
Vddd : 5.00
Vio0 : 5.00
Vio1 : 5.00
Vio2 : 5.00
Vio3 : 5.00
Voltage : 5.0
Vusb : 5.00

Expand All | Collapse All | Show All Paths | Hide All Paths

– Timing Violation Section

No Timing Violations

– Clock Summary Section

Clock	Domain	Nominal Frequency	Required Frequency	Maximum Frequency	Violation
CyILO	CyILO	1.000 kHz	1.000 kHz	N/A	
CyIMO	CyIMO	3.000 MHz	3.000 MHz	N/A	
CyMASTER_CLK	CyMASTER_CLK	24.000 MHz	24.000 MHz	N/A	
Clock_1	CyMASTER_CLK	1.000 MHz	1.000 MHz	69.677 MHz	
CyBUS_CLK	CyMASTER_CLK	24.000 MHz	24.000 MHz	69.677 MHz	
CyPLL_OUT	CyPLL_OUT	24.000 MHz	24.000 MHz	N/A	

– Register to Register Section

– Setup Subsection

– Source Clock : CyBUS_CLK : Positive edge(Required Frequency 24 MHz)

– Destination Clock : Clock_1 : Positive edge(Required Frequency 1 MHz)

Path Delay Requirement : 41.6667ns(24 MHz)
Affects clock : CyMASTER_CLK

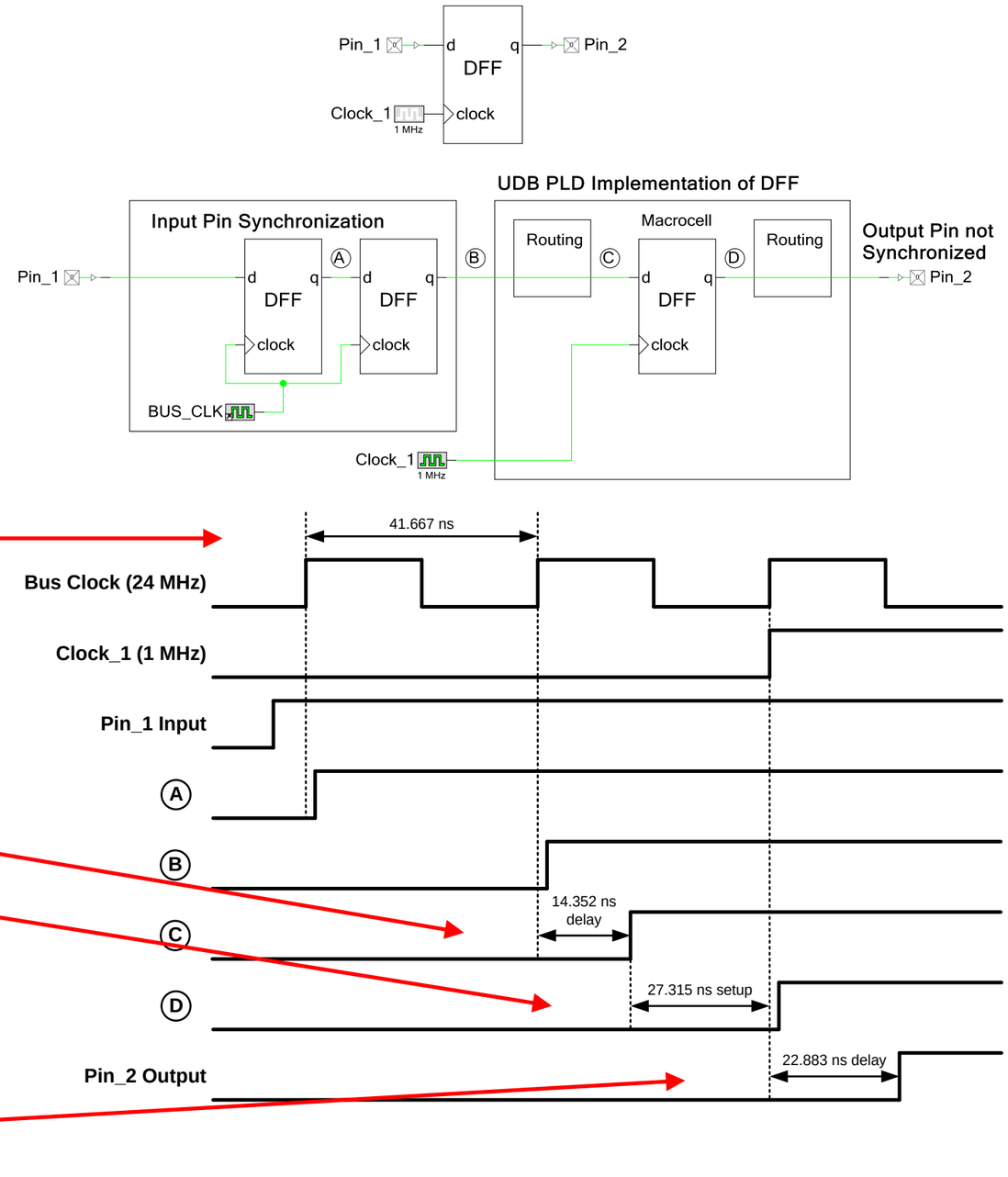
Source	Destination	FMax	Delay (ns)	Slack (ns)	Violation
Pin_1(0)/fb	cydff_1/main_0	69.677 MHz	14.352	27.315	

+ Hold Subsection

– Clock To Output Section

– Clock_1

Source	Destination	Delay (ns)
cydff_1/q	Pin_2(0)_PAD	22.883



If you click on any line in the report's tables, that line is expanded to show further detail. In Figure 40, the **Clock to Output Section** is expanded to show each part of the delay from Clock_1 through the DFF to Pin_2.

The expanded report shows that most of the delay, 16.2 ns, is through the pin itself. There is a small amount, 5.4 ns, from the macrocell output through the DSI routing to the pin, and a very small amount through the DFF itself.

Figure 40. Example #1 Expanded Clock to Output Section

- Clock_1

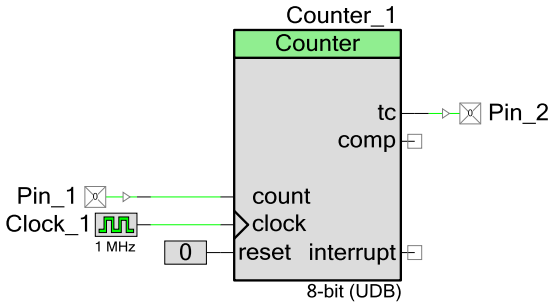
Source	Destination	Delay (ns)
cydff_1/q	Pin_2(0)_PAD	22.883

Type	Location	Fanout	Instance/Net	Source	Dest	Delay (ns)
macrocell1	U(3,4)	1	cydff_1	cydff_1/clock_0	cydff_1/q	1.250
Route		1	cydff_1	cydff_1/q	Pin_2(0)/pin_input	5.433
iocell	P0[1]	1	Pin_2(0)	Pin_2(0)/pin_input	Pin_2(0)/pad_out	16.200
Route		1	Pin_2(0)_PAD	Pin_2(0)/pad_out	Pin_2(0)_PAD	0.000
Clock					Clock path delay	0.000

5.3.2 Example #2

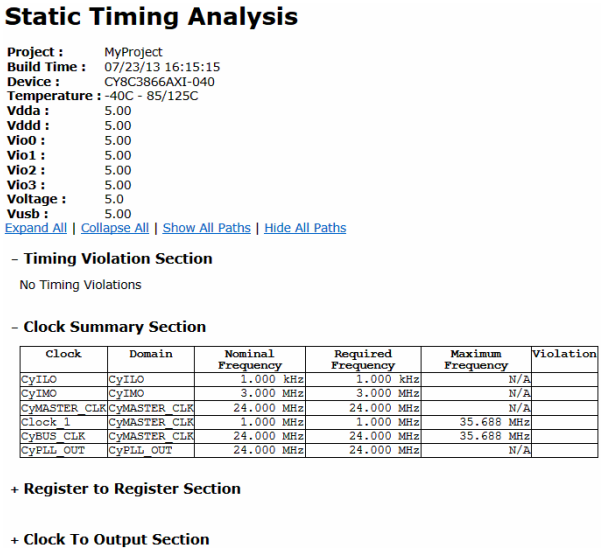
Let us now change Example 1, by replacing the DFF with a UDB-based 8-bit counter, as Figure 41 shows:

Figure 41. STA Report Example #2



A portion of the resultant STA report file is shown in Figure 42. Although there are still no timing violations, the **Clock Summary Section** shows that there is now much less margin for bus clock; the maximum frequency has been reduced to ~35 MHz.

Figure 42. STA Report for Example #2



Static Timing Analysis

Project : MyProject
Build Time : 07/23/13 16:15:15
Device : CY8C3866AXI-040
Temperature : -40C - 85/125C
Vdda : 5.00
Vddd : 5.00
Vio0 : 5.00
Vio1 : 5.00
Vio2 : 5.00
Vio3 : 5.00
Voltage : 5.0
Vusb : 5.00

Expand All | Collapse All | Show All Paths | Hide All Paths

– Timing Violation Section

No Timing Violations

– Clock Summary Section

Clock	Domain	Nominal Frequency	Required Frequency	Maximum Frequency	Violation
CyILO	CyILO	1.000 kHz	1.000 kHz	N/A	
CyIMO	CyIMO	3.000 MHz	3.000 MHz	N/A	
CyMASTER_CLK	CyMASTER_CLK	24.000 MHz	24.000 MHz	N/A	
Clock_1	CyMASTER_CLK	1.000 MHz	1.000 MHz	35.688 MHz	
CyBUS_CLK	CyMASTER_CLK	24.000 MHz	24.000 MHz	35.688 MHz	
CyPLL_OUT	CyPLL_OUT	24.000 MHz	24.000 MHz	N/A	

+ Register to Register Section

+ Clock To Output Section

To understand why, look at the detailed report in the **Register to Register Section, Setup Subsection**, in Figure 43 on page 31:

Figure 43. Example #2 Expanded Setup Subsection

– Register to Register Section

– Setup Subsection

– Source Clock : Clock_1 : Positive edge(Required Frequency 1 MHz)

– Destination Clock : Clock_1 : Positive edge(Required Frequency 1 MHz)

Path Delay Requirement : 1000ns(1 MHz)

Source	Destination	FMax	Delay (ns)	Slack (ns)	Violation
\Counter_1:CounterUDB:sCTRLReg:SyncCtl:ctrlreg\/control_7	\Counter_1:CounterUDB:sC8:counterdp:u0\/cs_addr_1	45.304 MHz	22.073	977.927	
\Counter_1:CounterUDB:count_stored_i\/q	\Counter_1:CounterUDB:sC8:counterdp:u0\/cs_addr_1	48.256 MHz	20.723	979.277	
\Counter_1:CounterUDB:sC8:counterdp:u0\/z0_comb	\Counter_1:CounterUDB:sC8:counterdp:u0\/cs_addr_0	51.454 MHz	19.435	980.565	
\Counter_1:CounterUDB:sC8:counterdp:u0\/z0_comb	\Counter_1:CounterUDB:sSTSReg:rstSts:stsreg\/status_3	61.501 MHz	16.260	983.740	
Net_22/q	\Counter_1:CounterUDB:sSTSReg:rstSts:stsreg\/status_3	70.507 MHz	14.183	985.817	
\Counter_1:CounterUDB:prevCompare\/q	\Counter_1:CounterUDB:sSTSReg:rstSts:stsreg\/status_0	92.902 MHz	10.764	989.236	
\Counter_1:CounterUDB:sC8:counterdp:u0\/z0_comb	Net_22/main_0	94.073 MHz	10.630	989.370	
\Counter_1:CounterUDB:sC8:counterdp:u0\/z0_comb	\Counter_1:CounterUDB:sSTSReg:rstSts:stsreg\/status_1	99.661 MHz	10.034	989.966	

– Source Clock : CyBUS_CLK : Positive edge(Required Frequency 24 MHz)

– Destination Clock : Clock_1 : Positive edge(Required Frequency 1 MHz)

Path Delay Requirement : 41.6667ns(24 MHz)
Affects clock : CyMASTER_CLK

Source	Destination	FMax	Delay (ns)	Slack (ns)	Violation
Pin_1(0)/fb	\Counter_1:CounterUDB:sC8:counterdp:u0\/cs_addr_1	35.688 MHz	28.021	13.646	
Pin_1(0)/fb	\Counter_1:CounterUDB:count_stored_i\/main_0	69.677 MHz	14.352	27.315	

The Counter Component is constructed from a combination of UDB datapaths and PLDs. In Figure 43 the **Source Clock: Clock_1** table shows that the UDB elements internal to the Counter are clocked by the Counter's clock input, in this case Clock_1. The table also shows that since Clock_1 is only 1 MHz there is almost 1 µs of slack – a very comfortable margin indeed.

More interesting numbers exist in the **Source Clock: CyBUS_CLK** table. Note that the Path Delay Requirement for this table is the period of the 24-MHz bus clock. The reason for this is explained in Multiple Clocks on page 15.

The table shows that the input pin, synchronized by bus clock, is routed to two UDB elements within the Counter. One of them has a delay of over 27 ns which reduces available slack to less than 14 ns. This is the reason why bus clock is limited to 35 MHz.

Finally, we can change the Counter Component from 8-bit to 32-bit and, for demonstration purposes, raise the bus clock frequency to 28 MHz. After rebuilding we now get a setup time violation, as Figure 44 shows.

Figure 44. Timing Violation with 32-bit Counter

– Timing Violation Section

Note: If your design will only ever run at typical room temperatures, selecting the narrower temperatur range in the system DWR for your application helps the tool to find timing-compliant routing solutions.

Violation	Source Clock	Destination Clock	Slack(ns)
Setup			
	CyBUS_CLK	Clock_1	-2.705

– Clock Summary Section

Clock	Domain	Nominal Frequency	Required Frequency	Maximum Frequency	Violation
CyILO	CyILO	1.000 kHz	1.000 kHz	N/A	
CyIMO	CyIMO	3.000 MHz	3.000 MHz	N/A	
CyMASTER_CLK	CyMASTER_CLK	28.000 MHz	28.000 MHz	N/A	
Clock_1	CyMASTER_CLK	1.000 MHz	1.000 MHz	26.029 MHz	
CyBUS_CLK	CyMASTER_CLK	28.000 MHz	28.000 MHz	26.029 MHz	Frequency
CyPLL_OUT	CyPLL_OUT	28.000 MHz	28.000 MHz	N/A	

The reason for the violation is more apparent in the **Clock Summary Section**, which shows that the bus clock frequency of 28 MHz is too high for the Component.

The details of the violation are shown in Figure 45 on page 32. A 32-bit counter requires four UDB datapaths, chained together. Consequently the input pin, which is synchronized to bus clock, is routed to many different destinations within the UDB elements. For one of those destinations the routing and other delays is 38.419 ns, which is greater than the 35.7143 ns period of bus clock, and causes the setup time violation. Changing the Clock_1 frequency does not help - to correct this problem you must either reduce the bus clock frequency or synchronize pin 1 to Clock_1.

Figure 45. Details of Timing Violation with 32-bit Counter

– Source Clock : CyBUS_CLK : Positive edge(Required Frequency 28 MHz)

– Destination Clock : Clock_1 : Positive edge(Required Frequency 1 MHz)

Path Delay Requirement : 35.7143ns(28 MHz)

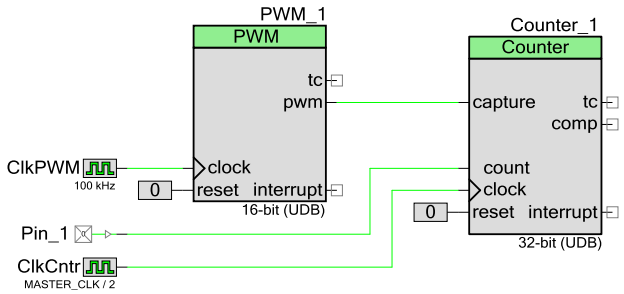
Affects clock : CyMASTER_CLK

Source	Destination	FMax	Delay (ns)	Slack (ns)	Violation
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	26.029 MHz	38.419	-2.705	SETUP
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u2\/ci	28.483 MHz	35.109	0.605	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	28.484 MHz	35.108	0.606	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	30.431 MHz	32.861	2.853	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u1\/ci	31.448 MHz	31.799	3.915	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u2\/ci	31.449 MHz	31.798	3.916	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u2\/cs_addr_1	33.805 MHz	29.581	6.133	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u3\/cs_addr_1	33.807 MHz	29.580	6.134	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u0\/cs_addr_1	35.064 MHz	28.519	7.195	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u1\/cs_addr_1	35.066 MHz	28.518	7.196	

5.4 Using the STA Report to Remove Warnings

As a final exercise, let us now use the techniques from the previous sections to understand and resolve a series of STA warnings in a moderately complex example. First, let us look at the design shown in Figure 20, which is repeated in Figure 46 with one change – ClkCntr is now master clock divided by 2:

Figure 46. Multi-Clock Example from Figure 20



Remember that in this design both clocks are synchronized to the master clock; the counter clock ClkCntr is specifically set to be master clock divided by 2. When the master clock frequency is high enough, in this case 60 MHz, two STA setup time violations are detected:

```
Warning-1366: Setup time violation found in a path from clock (ClkCntr) to
clock(ClkCntr)
```

```
Warning-1366: Setup time violation found in a path from clock (CyBUS_CLK) to clock
(ClkCntr)
```

Figure 47 shows these same warnings in the Timing Violation Section of the STA report

Figure 47. STA Report for Figure 46.

Static Timing Analysis

Project : Fig19
Build Time : 07/23/13 09:27:41
Device : CY8C3866AXI-040
Temperature : -40C - 85/125C
Vdda : 5.00
Vddd : 5.00
Vio0 : 5.00
Vio1 : 5.00
Vio2 : 5.00
Vio3 : 5.00
Voltage : 5.0
Vusb : 5.00

Expand All | Collapse All | Show All Paths | Hide All Paths

- Timing Violation Section

Note: If your design will only ever run at typical room temperatures, selecting the narrower temperature range in the system DWR for your application helps the tool to find timing-compliant routing solutions.

Violation	Source Clock	Destination Clock	Slack(ns)
Setup			
	ClkCntr	ClkCntr	-2.533
	CyBUS_CLK	ClkCntr	-14.795

- Clock Summary Section

Clock	Domain	Nominal Frequency	Required Frequency	Maximum Frequency	Violation
CyILO	CyILO	1.000 kHz	1.000 kHz	N/A	
CyIMO	CyIMO	3.000 MHz	3.000 MHz	N/A	
CyMASTER_CLK	CyMASTER_CLK	60.000 MHz	60.000 MHz	70.952 MHz	
ClkCntr	CyMASTER_CLK	30.000 MHz	30.000 MHz	27.882 MHz	Frequency
Clk_PWM	CyMASTER_CLK	100.000 kHz	100.000 kHz	42.411 MHz	
CyBUS_CLK	CyMASTER_CLK	60.000 MHz	60.000 MHz	31.784 MHz	Frequency
CyPLL_OUT	CyPLL_OUT	60.000 MHz	60.000 MHz	N/A	

To remove the warnings, first examine the setup violation for the counter clock, that is, where the source and destination clocks are both ClkCntr. To do this, click on the ClkCntr to ClkCntr row in the **Timing Violation Section**. This opens up the details further down in the report, as Figure 48 shows:

Figure 48. STA Setup Violations for ClkCntr to ClkCntr

- Register to Register Section

- Setup Subsection

- Source Clock : ClkCntr : Positive edge(Required Frequency 30 MHz)

- Destination Clock : ClkCntr : Positive edge(Required Frequency 30 MHz)

Path Delay Requirement : 33.3333ns(30 MHz)

Source	Destination	FMax	Delay (ns)	Slack (ns)	Violation
\Counter_1:CounterUDB:sC32:counterdp:u0\/z0	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	27.882 MHz	35.866	-2.533	SETUP
\Counter_1:CounterUDB:sC32:counterdp:u1\/z0	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	29.039 MHz	34.436	-1.103	SETUP
\Counter_1:CounterUDB:sC32:counterdp:u2\/z0	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	30.298 MHz	33.006	0.327	
\Counter_1:CounterUDB:sCTRLReg:SyncCtl:ctrlreg\/control_7	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	30.714 MHz	32.558	0.775	
\Counter_1:CounterUDB:sC32:counterdp:u0\/z0	\Counter_1:CounterUDB:sC32:counterdp:u2\/ci	30.716 MHz	32.556	0.777	
\Counter_1:CounterUDB:sC32:counterdp:u3\/z0_comb	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	31.670 MHz	31.576	1.757	
\Counter_1:CounterUDB:sC32:counterdp:u0\/z0	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	31.742 MHz	31.504	1.829	
\Counter_1:CounterUDB:count_stored_i\/q	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	32.050 MHz	31.201	2.132	
\Counter_1:CounterUDB:sC32:counterdp:u1\/z0	\Counter_1:CounterUDB:sC32:counterdp:u2\/ci	32.127 MHz	31.126	2.207	
\Counter_1:CounterUDB:sC32:counterdp:u1\/z0	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	33.251 MHz	30.074	3.259	

Because the Counter is 32-bit, four UDB datapaths are required to implement it. Figure 48 shows that delays through some of the UDBs are too long for the 30 MHz counter clock (ClkCntr), and we must reduce ClkCntr to below 27.882 MHz. Because ClkCntr is the master clock divided by 2, we can reduce the master clock to 55 MHz, which makes ClkCntr be 27.5 MHz.

This resolves the first warning but we still have another setup violation:

```
Warning-1366: Setup time violation found in a path from clock (CyBUS_CLK) to clock
(ClkCntr)
```

To remove this one we can examine the CyBUS_CLK to ClkCntr section of the STA report - click on the relevant row in the **Timing Violation Section**. The details are shown in Figure 49:

Figure 49. STA Setup Violations for Master Clock at 55 MHz

– Register to Register Section

– Setup Subsection

+ Source Clock : ClkCntr : Positive edge(Required Frequency 27.5 MHz)

+ Source Clock : Clk_PWM : Positive edge(Required Frequency 100 kHz)

– Source Clock : CyBUS_CLK : Positive edge(Required Frequency 55 MHz)

– Destination Clock : ClkCntr : Positive edge(Required Frequency 27.5 MHz)

Path Delay Requirement : 18.1818ns(55 MHz)
Affects clock : CyMASTER_CLK

Source	Destination	FMax	Delay (ns)	Slack (ns)	Violation
Pin_1(0)_SYNC/out	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	31.758 MHz	31.488	-13.306	SETUP
Pin_1(0)_SYNC/out	\Counter_1:CounterUDB:sC32:counterdp:u2\/ci	35.489 MHz	28.178	-9.996	SETUP
Pin_1(0)_SYNC/out	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	35.528 MHz	28.147	-9.965	SETUP
Pin_1(0)_SYNC/out	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	39.005 MHz	25.638	-7.456	SETUP
Pin_1(0)_SYNC/out	\Counter_1:CounterUDB:sC32:counterdp:u1\/ci	40.212 MHz	24.868	-6.686	SETUP
Pin_1(0)_SYNC/out	\Counter_1:CounterUDB:sC32:counterdp:u2\/ci	40.263 MHz	24.837	-6.655	SETUP
Pin_1(0)_SYNC/out	\Counter_1:CounterUDB:sC32:counterdp:u3\/cs_addr_1	44.431 MHz	22.507	-4.325	SETUP
Pin_1(0)_SYNC/out	\Counter_1:CounterUDB:sC32:counterdp:u2\/cs_addr_1	44.727 MHz	22.358	-4.176	SETUP
Pin_1(0)_SYNC/out	\Counter_1:CounterUDB:sC32:counterdp:u0\/cs_addr_1	46.322 MHz	21.588	-3.406	SETUP
Pin_1(0)_SYNC/out	\Counter_1:CounterUDB:sC32:counterdp:u1\/cs_addr_1	46.389 MHz	21.557	-3.375	SETUP

First, note that the Path Delay Requirement is the period of the 55 MHz master clock, and not ClkCntr. The reason for this is explained in Multiple Clocks on page 15. The delays in the routes to the counter UDBs are too long for Pin_1. Two solutions are available:

1. Reduce the master clock frequency. If your overall system timing does not require a high-speed master clock, this method is preferred because it may also reduce noise and current consumption. Figure 49 shows that the master clock must be brought below the smallest FMax value, i.e., 31.758 MHz. However bringing bus clock below 33 MHz causes a different synchronization method to be applied to Pin_1 – see Pins on page 12 – which in turn causes different routing and as a consequence another timing violation. Figure 50 shows that the master clock frequency must actually be reduced even more, to below 26.036 MHz.

Figure 50. Setup Violations for Master Clock at 31 MHz

– Register to Register Section

– Setup Subsection

+ Source Clock : ClkCntr : Positive edge(Required Frequency 15.5 MHz)

+ Source Clock : Clk_PWM : Positive edge(Required Frequency 100 kHz)

– Source Clock : CyBUS_CLK : Positive edge(Required Frequency 31 MHz)

– Destination Clock : ClkCntr : Positive edge(Required Frequency 15.5 MHz)

Path Delay Requirement : 32.2581ns(31 MHz)
Affects clock : CyMASTER_CLK

Source	Destination	FMax	Delay (ns)	Slack (ns)	Violation
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	26.036 MHz	38.409	-6.151	SETUP
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u2\/ci	28.491 MHz	35.099	-2.841	SETUP
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	28.492 MHz	35.098	-2.840	SETUP
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u3\/ci	30.440 MHz	32.851	-0.593	SETUP
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u1\/ci	31.457 MHz	31.789	0.469	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u2\/ci	31.458 MHz	31.788	0.470	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u2\/cs_addr_1	33.817 MHz	29.571	2.687	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u3\/cs_addr_1	33.818 MHz	29.570	2.688	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u0\/cs_addr_1	35.077 MHz	28.509	3.749	
Pin_1(0)/fb	\Counter_1:CounterUDB:sC32:counterdp:u1\/cs_addr_1	35.078 MHz	28.508	3.750	

After this is done (by setting the PLL to 52 MHz, and dividing it by 2 to get a 26 MHz master clock), STA produces no warnings, as Figure 51 shows:

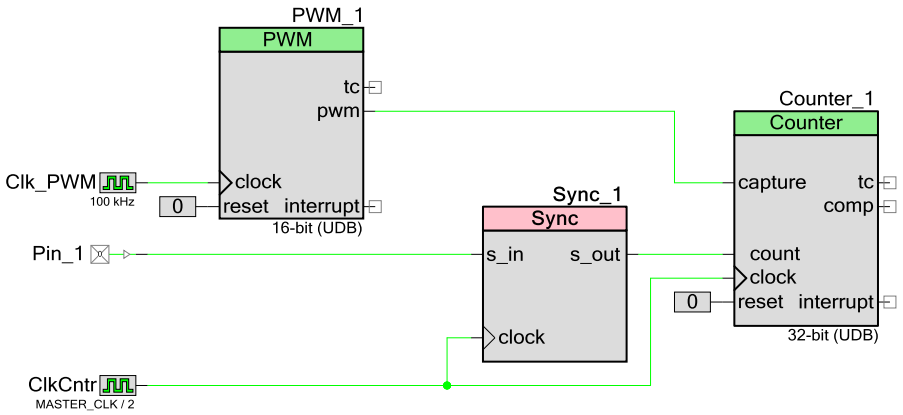
Figure 51. Clock Summary for Master Clock at 23 MHz

– Clock Summary Section

Clock	Domain	Nominal Frequency	Required Frequency	Maximum Frequency	Violation
CyILO	CyILO	1.000 kHz	1.000 kHz	N/A	
CyIMO	CyIMO	3.000 MHz	3.000 MHz	N/A	
CyMASTER_CLK	CyMASTER_CLK	26.000 MHz	26.000 MHz	76.225 MHz	
ClkCntr	CyMASTER_CLK	13.000 MHz	13.000 MHz	26.036 MHz	
Clk_PWM	CyMASTER_CLK	100.000 kHz	100.386 kHz	40.125 MHz	
CyBUS_CLK	CyMASTER_CLK	26.000 MHz	26.000 MHz	26.036 MHz	
CyPLL_OUT	CyPLL_OUT	52.000 MHz	52.000 MHz	N/A	

2. You can always reduce the master clock frequency to a low enough value to remove all STA warnings. However, if you need a high-speed master or bus clock, a better solution is to add a PSoC Creator Sync Component, in this case to synchronize Pin_1 to ClkCntr. See Figure 52.

Figure 52. Multi-Clock Example from Figure 20, with Sync Component



Doing this gives an STA report that indicates that ClkCntr can now go as high as 28 MHz, as Figure 53 shows:

Figure 53. Clock Summary for Master Clock at 26 MHz and Sync Components Added

– Clock Summary Section

Clock	Domain	Nominal Frequency	Required Frequency	Maximum Frequency	Violation
CyILO	CyILO	1.000 kHz	1.000 kHz	N/A	
CyIMO	CyIMO	3.000 MHz	3.000 MHz	N/A	
CyMASTER_CLK	CyMASTER_CLK	26.000 MHz	26.000 MHz	77.149 MHz	
ClkCntr	CyMASTER_CLK	13.000 MHz	13.000 MHz	28.988 MHz	
Clk_PWM	CyMASTER_CLK	100.000 kHz	100.386 kHz	42.411 MHz	
CyBUS_CLK	CyMASTER_CLK	26.000 MHz	26.000 MHz	N/A	
CyPLL_OUT	CyPLL_OUT	52.000 MHz	52.000 MHz	N/A	

Since ClkCntr can go to 28 MHz, the master clock can go as high as 56 MHz, as Figure 54 shows.

Figure 54. Clock Summary for Master Clock at 56 MHz and Sync Component Added

– Clock Summary Section

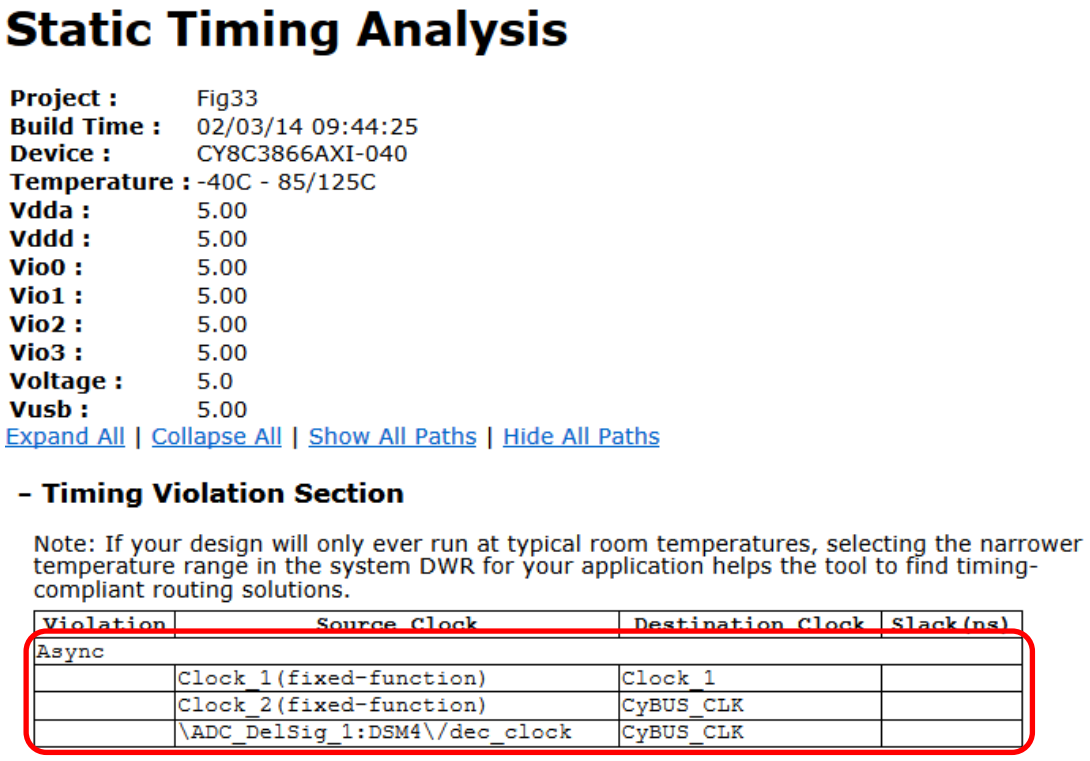
Clock	Domain	Nominal Frequency	Required Frequency	Maximum Frequency	Violation
CyILO	CyILO	1.000 kHz	1.000 kHz	N/A	
CyIMO	CyIMO	3.000 MHz	3.000 MHz	N/A	
CyMASTER_CLK	CyMASTER_CLK	56.000 MHz	56.000 MHz	75.228 MHz	
ClkCntr	CyMASTER_CLK	28.000 MHz	28.000 MHz	28.988 MHz	
Clk_PWM	CyMASTER_CLK	100.000 kHz	100.000 kHz	42.411 MHz	
CyBUS_CLK	CyMASTER_CLK	56.000 MHz	56.000 MHz	N/A	
CyPLL_OUT	CyPLL_OUT	56.000 MHz	56.000 MHz	N/A	

In addition, if you need a maximum-speed master and bus clock, that is, as high as 66 MHz, you can always set ClkCntr to be master clock divided by 3 instead of 2, for a frequency of 22 MHz, which is less than the maximum frequency requirement.

5.4.1 Asynchronous Clock Crossings

Another type of violation that the STA report may produce is “Async”, as Figure 55 shows. These are usually due to routing fixed-block outputs to the inputs of UDB-based Components. The easiest way to handle these is to add Sync Components to your design, as Topic #7 describes.

Figure 55. Asynchronous Timing Violations



Static Timing Analysis

Project : Fig33
Build Time : 02/03/14 09:44:25
Device : CY8C3866AXI-040
Temperature : -40C - 85/125C
Vdda : 5.00
Vdd : 5.00
Vio0 : 5.00
Vio1 : 5.00
Vio2 : 5.00
Vio3 : 5.00
Voltage : 5.0
Vusb : 5.00

Expand All | Collapse All | Show All Paths | Hide All Paths

– Timing Violation Section

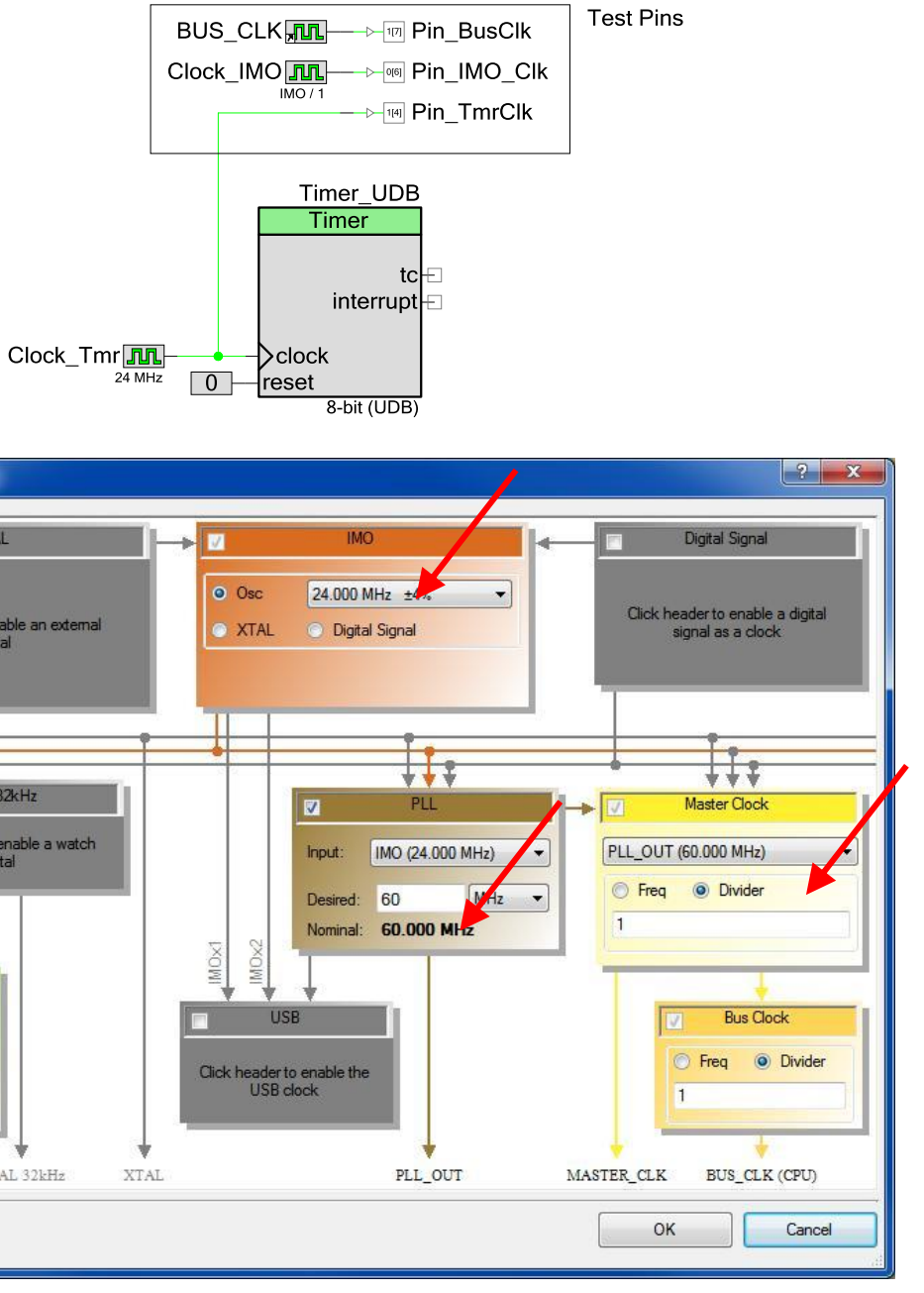
Note: If your design will only ever run at typical room temperatures, selecting the narrower temperature range in the system DWR for your application helps the tool to find timing-compliant routing solutions.

Violation	Source Clock	Destination Clock	Slack(ns)
Async			
	Clock_1(fixed-function)	Clock_1	
	Clock_2(fixed-function)	CyBUS_CLK	
	\ADC_DelSig_1:DSM4\/dec_clock	CyBUS_CLK	

5.5 Clock Nominal and Required Frequencies

Examine the previous figures and note that in the STA Clock Summary Section all clock frequencies are the same across the columns Nominal Frequency and Required Frequency. This is the typical case; however it is possible for the frequencies to be different. This can happen when the master clock is not an integral multiple of another clock, as Figure 56 shows. Clock_Tmr is sourced from the IMO, and the master clock is 60 MHz; the master clock is running at 2.5 times the timer clock.

Figure 56. Design Causing Different Nominal and Required Clock Frequencies



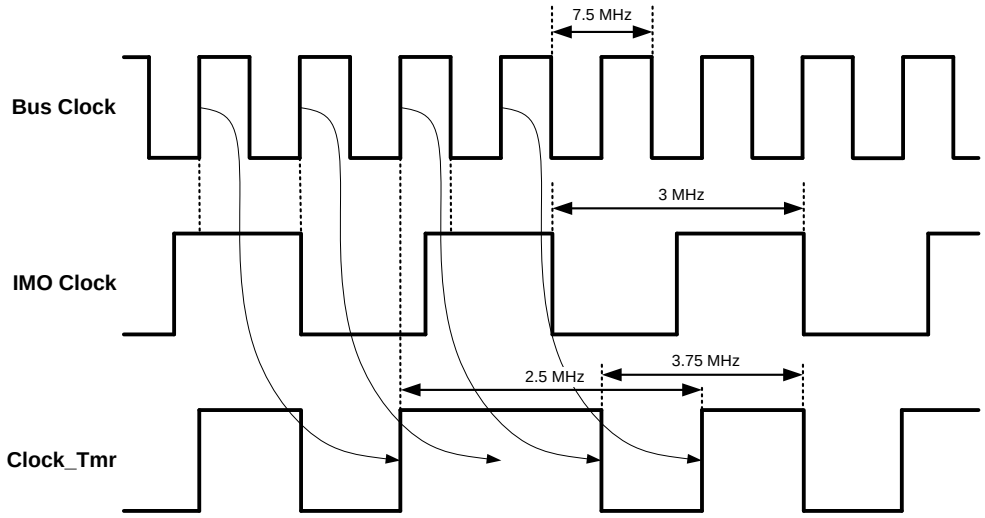
You can get the same effect at lower frequencies, which may be easier to see with an oscilloscope on the test pins shown in Figure 56 on page 39. Figure 57 shows a similar example with Clock_Tmr at 3 MHz and the master clock at 7.5 MHz (60 MHz divided by 8).

Figure 57. Different Nominal and Required Frequencies, Example #2

Type	Name	Domain	Desired Frequency	Nominal Frequency	Accuracy (%)	Tolerance (%)	Divider	Start on Reset	
System	USB_CLK	DIGITAL	48.000 MHz	? MHz	±0	-	1	☐	IMOx2
System	Digital Signal	DIGITAL	? MHz	? MHz	±0	-	0	☐	
System	XTAL 32kHz	DIGITAL	32.768 kHz	? MHz	±0	-	0	☐	
System	XTAL	DIGITAL	25.000 MHz	? MHz	±0	-	0	☐	
System	ILO	DIGITAL	? MHz	1.000 kHz	-50, +100	-	0	☑	
System	IMO	DIGITAL	3.000 MHz	3.000 MHz	±1	-	0	☑	
System	BUS_CLK (CPU)	DIGITAL	? MHz	7.500 MHz	±1	-	1	☑	MASTER_CLK
System	MASTER_CLK	DIGITAL	? MHz	7.500 MHz	±1	-	8	☑	PLL_OUT
System	PLL_OUT	DIGITAL	60.000 MHz	60.000 MHz	±1	-	0	☑	IMO
Local	Clock_IMO	DIGITAL	? MHz	3.000 MHz	±1	-	1	☑	IMO
Local	Clock_Tmr	DIGITAL	3.000 MHz	3.000 MHz	±1	-	1	☑	Auto: IMO
Local	Clock_UDB_1	DIGITAL	? MHz	7.500 MHz	±1	-	0	☑	BUS_CLK

In both cases, the timer clock is sourced from the IMO, but generally in order for it to be used with UDBs it must be synchronized to the bus clock. Figure 58 shows the timing diagram at the test pins; note that Clock_Tmr is double-synchronized to bus clock so is shifted rightward by two bus clock cycles.

Figure 58. Timing Diagram, Different Nominal and Required Frequencies



Because Clock_Tmr, sourced from the IMO, is synchronized by a bus clock whose frequency is not an integral multiple of the IMO, the Clock_Tmr output has two different frequencies – 2.5 MHz and 3.75 MHz – neither one of which equals that of the IMO. This effect is known as **synchronization jitter**.

This is indicated in the STA Clock Summary Section, as Figure 59 shows. It is important to distinguish between the terms “Desired Frequency” and “Nominal Frequency” in the PSoC Creator Clocks tab (Figure 57 on page 40), and “Nominal Frequency” and “Required Frequency” in the STA report (Figure 59). Nominal Frequency is the same between the two displays. Required Frequency is the frequency at which paths using this clock must be able to meet timing. It is the highest frequency in the clock – in this case, 3.75 MHz. It is possible for the Nominal Frequency to be less than the Maximum Frequency and the Required Frequency to be greater than the Maximum Frequency, which would give a timing violation with no obvious solution.

Figure 59. STA Showing Different Nominal and Required Clock Frequencies

+ Clock Summary Section

Clock	Domain	Nominal Frequency	Required Frequency	Maximum Frequency	Violation
Clock_IMO	Clock_IMO	3.000 MHz	3.000 MHz	N/A	
Clock_IMO (routed)	Clock_IMO (routed)	3.000 MHz	3.000 MHz	N/A	
Clock_Tmr (routed)	Clock_Tmr (routed)	3.000 MHz	3.000 MHz	N/A	
CyBUS_CLK (routed)	CyBUS_CLK (routed)	7.500 MHz	7.500 MHz	N/A	
CyILO	CyILO	1.000 kHz	1.000 kHz	N/A	
CyIMO	CyIMO	3.000 MHz	3.000 MHz	N/A	
CyMASTER_CLK	CyMASTER_CLK	7.500 MHz	7.500 MHz	N/A	
Clock_Tmr	CyMASTER_CLK	3.000 MHz	3.750 MHz	55.723 MHz	
CyBUS_CLK	CyMASTER_CLK	7.500 MHz	7.500 MHz	N/A	
CyPLL_OUT	CyPLL_OUT	60.000 MHz	60.000 MHz	N/A	

It is generally not a good idea to drive a timer, counter, PWM or other timing-dependent circuit with a clock with synchronization jitter. After building a design you should check the STA clock summary and make sure that this situation does not exist. If it does, there are two ways to fix it:

- Adjust the master clock and bus clock frequency to be an integral multiple of the other clock source's frequency. The best practice is that it should be at least 4 times the other clock's frequency – see discussion on page 17. In the above case, master clock could be changed to 12 MHz, which is 4x the frequency of Clock_Tmr.
- Select a source for the clock that is either master clock or synchronized to master clock. In this case, source Clock_Tmr from master clock instead of from IMO. Note that this may in turn cause your clock's frequency to change resulting in the need to change other portions of your design.

5.6 Hold, Recovery and Removal Violations

The previous examples have focused on setup violations, because these violations are by far the most common. The other violation types, as described in Metastability, and Register Timing on page 7, can be found under the following conditions:

- Hold violations are usually associated with clocks that are routed from the DSI, and thus delayed, or skewed relative to each other; see Figure 23. This can delay the clock and cause a hold time violation; see Figure 7.
- Recovery and removal violations are associated with asynchronous resets or presets; see Figure 30. Changing these to synchronous will usually remove the warning.

6 Summary

This application note has shown some general principles in digital design and how they are implemented in PSoC 3, PSoC 4, and PSoC 5LP. A set of best practice digital design topics were then presented.

Finally, detailed explanations of STA warnings were provided, as well as detailed examples of how to remove them.

The information presented in this application note should help you to implement robust and high-performance digital designs in PSoC. Detailed instructions for Verilog and datapath programming, and Component development, are beyond the scope of this application note. For more information on these topics, see the Component Development Kit tools and documentation included with PSoC Creator. See also PSoC Creator Design Tutorials.

7 Related Application Notes

- AN82250 - Implementing Programmable Logic Designs – An Introduction
- AN82156 - PSoC 3 and PSoC 5LP Designing PSoC Creator Components with UDB Datapaths
- AN54181 - Getting Started with PSoC 3
- AN79953 – Getting Started with PSoC 4
- AN77759 - Getting Started with PSoC 5LP
- AN60631 - PSoC 3 and PSoC 5LP Clocking Resources
- AN72382 - Using PSoC 3 and PSoC 5LP GPIO Pins
- AN62510 – Implementing State Machines with PSoC 3 and PSoC 5LP

About the Authors

Name:	Todd Dust
Title:	Applications Engineer Sr
Background:	BSEE, Seattle Pacific University
Name:	Mark Ainsworth
Title:	Applications Engineer Principal
Background:	BS in Computer Engineering, Syracuse University; MSEE, University of Washington.

Document History

Document Title: AN81623 - PSoC® 3, PSoC 4, and PSoC 5LP Digital Design Best Practices

Document Number: 001-81623

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3701524	MKEA	08/06/2012	New application note
*A	3772570	MKEA/ANTO	11/08/2012	1. Added links to AN82156 and AN82250 2. Minor edits throughout the document
*B	3818225	MKEA	11/21/2012	Updated for PSoC 5LP and PSoC Creator 2.1 SP1. Minor edits throughout the document.
*C	4078871	MKEA	07/25/2013	Updated throughout for PSoC 4 and PSoC Creator 2.2 SP1. Deleted Figure 27 and related text; renumbered subsequent figures. Updated figures 19, 31, and 37 for better clarity. Added SRFF Component to Figure 28. Added section Clock Nominal and Required Frequencies. Updated to *K template.
*D	4268107	MKEA	02/05/2014	Added Figure 17 and related text. Changed title and content for Topic #7 section, and added section Asynchronous Clock Crossings with new Figure 55. Updated to *L template. Miscellaneous minor edits.
*E	5347349	MKEA	07/12/2016	Updated template

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com