Sync Pulse Usage in CASPER DSP Blocks

Henry Chen, Peter McMahon, Aaron Parsons

August 1, 2008

1 Overview

Most CASPER DSP blocks have a streaming architecture in which, after some initialization period, valid data is constantly being output. In addition, data tends to be defined in frames or windows, typically some spectrum or a signal of some known and repetitive vector length. To aid in managing the data stream, most major blocks in the library can use a sync pulse as a timing fiducial, as well as a reset.

2 Usage

2.1 Alignment

The sync pulse is used as a "vector warning signal," meaning that it preceeds a frame of data 1 clock cycle prior to the first data sample of the frame. This allows the sync pulse to be used as an in-situ reset signal; counters and other state elements can be reset to their initialization conditions in line with the arrival of new data.

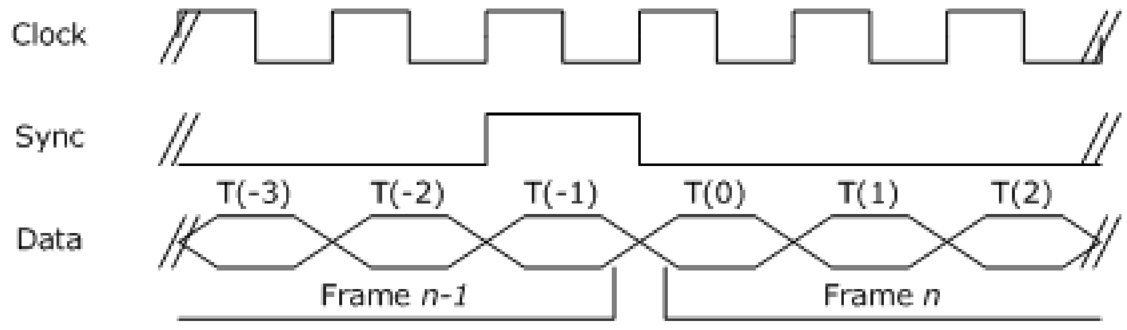

Figure 1: Sync pulse alignment to data

2.2 Propagation

Due to the reconfigurable, black-box nature of the CASPER DSP library, processing latency through a block varies depending on parameters, and can change with library revisions. Therefore, a latency-matched sync pulse is used to abstract data propagation delay in the datapath. Blocks—particularly those with variable latency—are responsible for internally delaying a sync pulse input to match the data latency. This way, if a sync pulse is input to a block along with a frame of data, the results from that frame of data are aligned with the sync pulse output from that block, regardless of the block's internal datapath latency.

3 Timing

Sync pulses should be a periodic signal with a pulse width of 1 clock cycle. If a sync comes out of alignment, it will act as a reset for the datapath. With the proper periodicity, the reset caused by the sync pulse should be coincident with the rollover to the initial conditions of a block's internal state. Period requirements vary with each particular design, depending on which processing elements are used. As channelization is the most common operation, its sync pulse timing is detailed in this memo.

Since sync pulses delineate spectrum frames, one would assume that an FFT block has a base minimum sync pulse period equal to the FFT length. However, the number of clock cycles it takes before a frame repeats is affected by two modifiers to the base spectrum length: the number of simultaneous inputs (if using a wideband FFT) and the orders of the reorder blocks in the FFT.

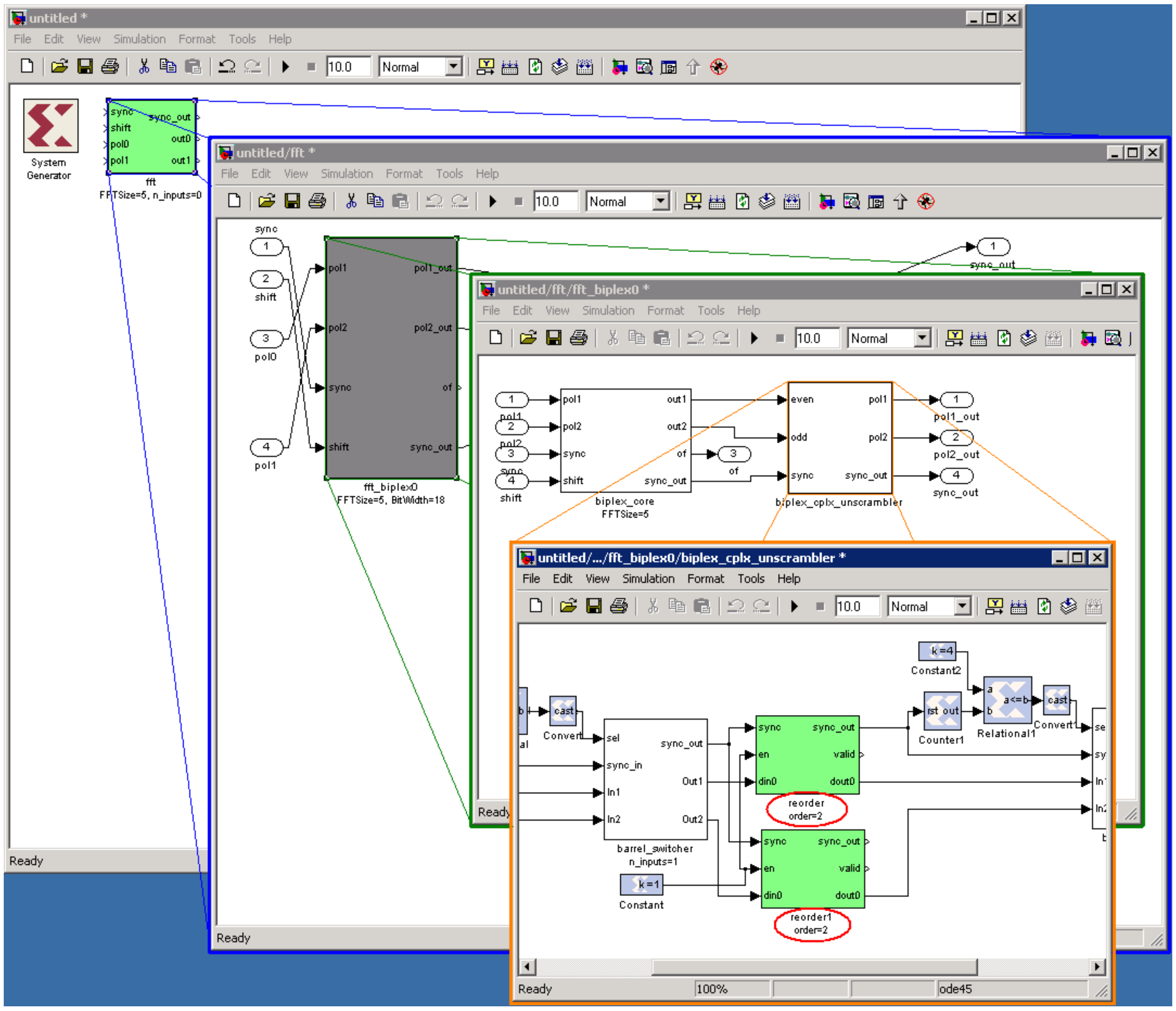

Figure 2: Reorder blocks in Biplex FFT

Reorder blocks have a certain order[1] r. The effect of the set order is that the reorder block must work

[1]Parsons, A., *The Symmetric Group in Data Permutation, with Applications to a High-Bandwidth Streaming FFT Archi-*

through r reorders before cycling back to its initial state, and so should not be reset beforehand. Each reorder block thus contributes a factor r to the minimum sync period.

$$\text{Minimum Sync period} = LCM(\text{reorder orders}) \cdot \frac{\text{FFT Size}}{\text{Simultaneous inputs}} \tag{1}$$

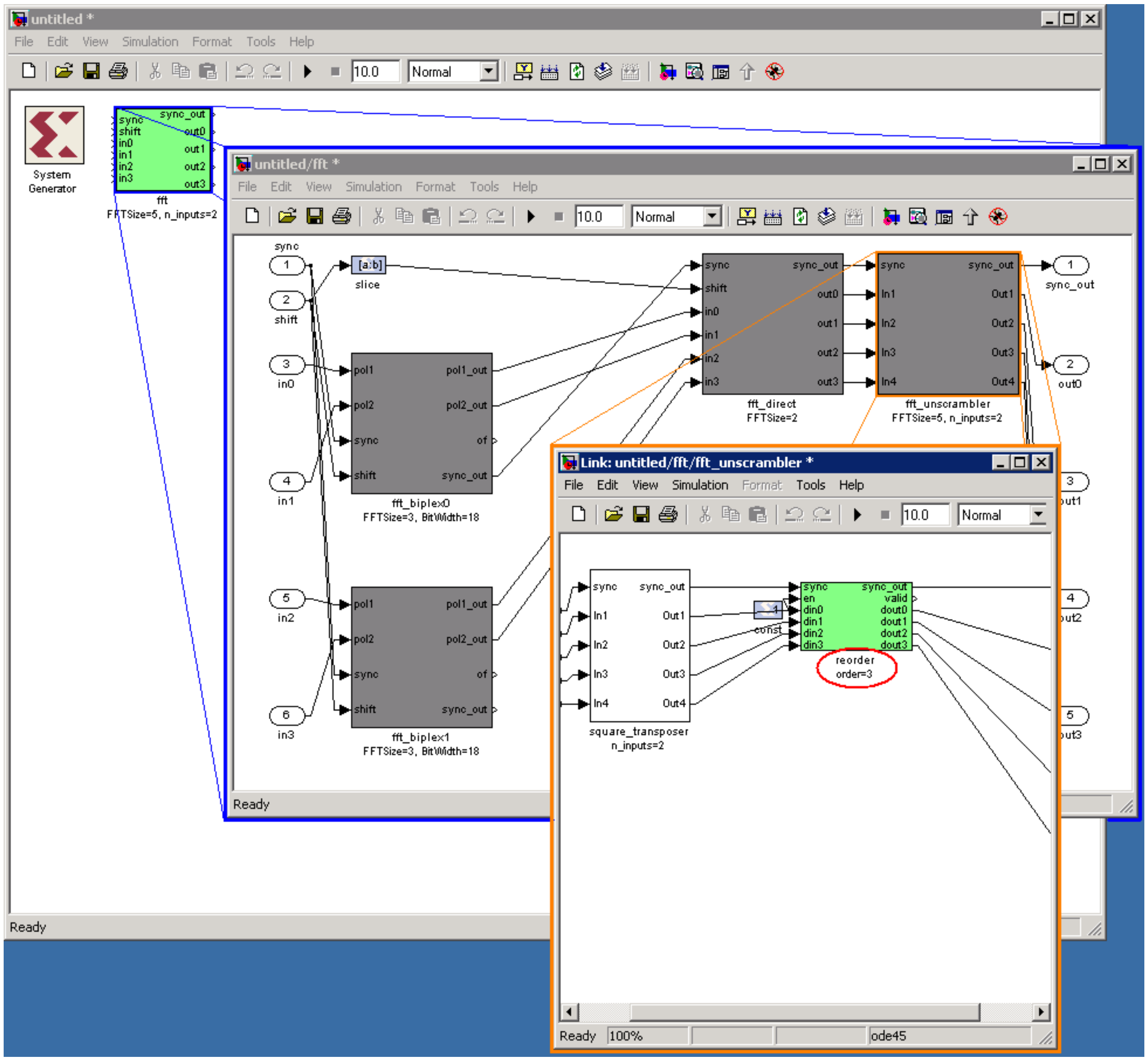

Figure 3: Reorder blocks in Wideband FFT

In FFTs, reorder blocks are used to perform the bit-reversals to unscramble a spectrum into bin order. A standard biplex FFT has two reorder blocks in its biplex unscrambler, as shown in Figure 2 with their orders highlighted. If a wideband FFT is used, there is an additional reorder block used in the FFT unscrambler following the direct FFT, as shown in Figure 3. This is in addition to the biplex unscramblers that are still implemented in each of the biplex FFTs used to assemble the full wideband FFT.

tecture, IEEE Letters on Signal Processing

For example, for a 4096-channel, 4-input wideband FFT, there are 2 biplex FFTs, each with 2 order-2 reorder blocks in their biplex unscramblers. Its FFT unscrambler uses an order-5 reorder block. Therefore, the minimum sync pulse period for this FFT is

$$\begin{aligned}
\text{Minimum Sync period} &= LCM(\text{reorder orders}) \cdot \frac{\text{FFT Size}}{\text{Simultaneous inputs}} \\
&= LCM(2,2,2,2,5) \cdot \frac{4096}{4} \\
&= 10 \cdot 1024 \\
&= 10240 \text{ clock cycles}
\end{aligned}$$

In general, any processing block that affects what is considered to be a single "window" of data will act as a modifier to the minimum sync pulse period. If a PFB FIR filter is used in conjuction with an FFT to form a PFB, then the minimum sync period is multiplied by the number of taps in the filter. When vector accumulators are used to sum up multiple spectra, then the number of spectra accumulated is also a factor in the window length (Eq. 2).

$$\text{Sync period} = N_{accumulations} \cdot \text{PFB FIR taps} \cdot LCM(\text{reorder orders}) \cdot \frac{\text{FFT Size}}{\text{Simultaneous inputs}} \qquad (2)$$

In regular operation, the sync pulse should always have a period that is an integer multiple of the minimum period. Even though the Xilinx FPGAs issue a global reset after configuration, it is good practice to send at least one sync pulse to ensure a reset of the datapath. This is particularly true when using 2 iADC boards, in which case ADC synchronization—and thus clock and data validity—may not occur until after the FPGA global reset. Provided that the sync pulse meets the minimum period requirement and any special reset requirements, blocks should not depend on a particular periodicity of the sync pulse. No sync pulse, one-and-only-one pulse, and sync pulses repeating at any integer multiple of the minimum period should all be valid sync inputs to a block, with any special requirements being handled internally.

A recommended configuration is to compare the output of a counter to 0, and set the counter to count to $(SyncPeriod - 1)$. If the sync period happens to be a power of 2, then a free-running counter can be used; otherwise a count-limited counter is needed. By enabling the reset port of the counter, a sync pulse can be manually generated via a software register or any other user- or system-controllable signal (Figure 4).

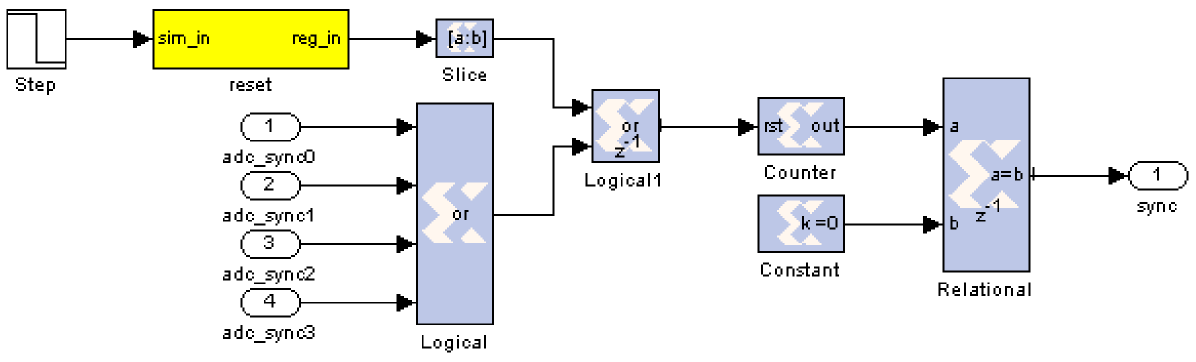

Figure 4: Example sync pulse generator configuration