XAPP652 (v1.0) November 15, 2002

Word Alignment and SONET/SDH Deframing

Author: Nick Sawyer

Summary

This application note describes the logic to perform basic word alignment and deframing specifically for SONET/SDH systems, where data is being processed at 16-bits or 64-bits per clock cycle.

Introduction

A word aligner and deframer is a block of logic commonly needed to interface processing logic to serial data streams. Typically, an incoming serial data stream is processed in 16-bit or 32-bit words. As an example, with SONET OC48, a ~2.5 Mb/s serial stream is usually converted to 16-bit words for processing at ~155 MHz inside the FPGA. The designer needs to determine what relation these 16-bit words have to the 1-bit original data stream. In SONET systems this is done by detecting the A1A2 transition (present in the frame header at 8 KHz). Once the transition is found, the data words can be shifted in time to correspond to the byte boundaries of the incoming data.

In a sixteen-bit data path, the actual data to be processed could, for example, start at bit [5] of a particular word. This then requires the movement of bits [15:5] of the input word into bit positions [10:0] of the output word. Bits [15:11] of the output will come from bits [4:0] of the previous input word, as SONET data is transmitted MSB first. In this case, the selection of how many bits to shift will be made with a four-bit input code. The functionality for wider data paths is identical, but the numbers get larger.

In addition, the deframing function needs to generate signals allowing the designer to determine the current position in the frame. This is normally done by generating two binary values indicating the current line number and word number.

The basic SONET frame (STS-1/OC1) is shown in Figure 1.

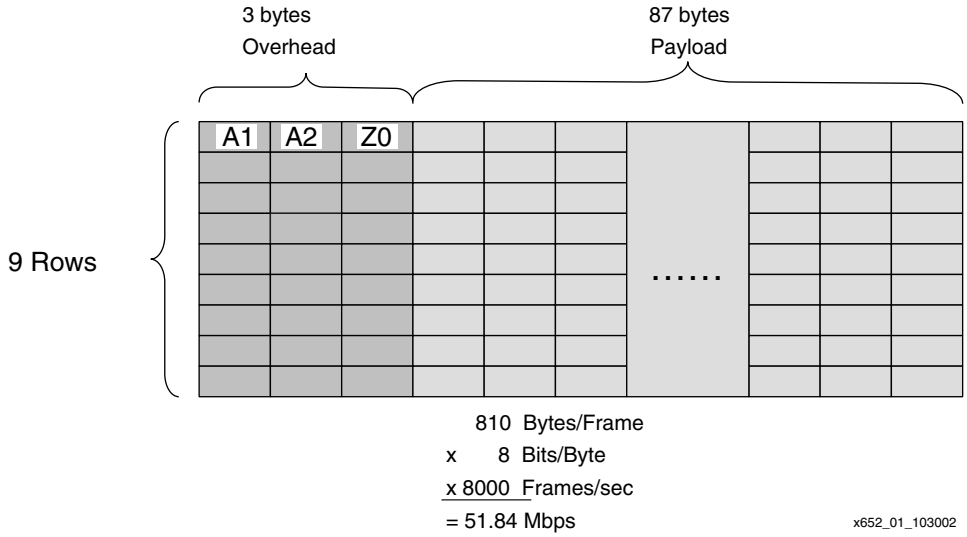

Figure 1: **STS-1/OC1 Frame Format**

Higher line rates are achieved by byte multiplexing the STS-1 frames. For instance, OC48 is 48 byte-multiplexed OC1 frames. This is shown in Figure 2.

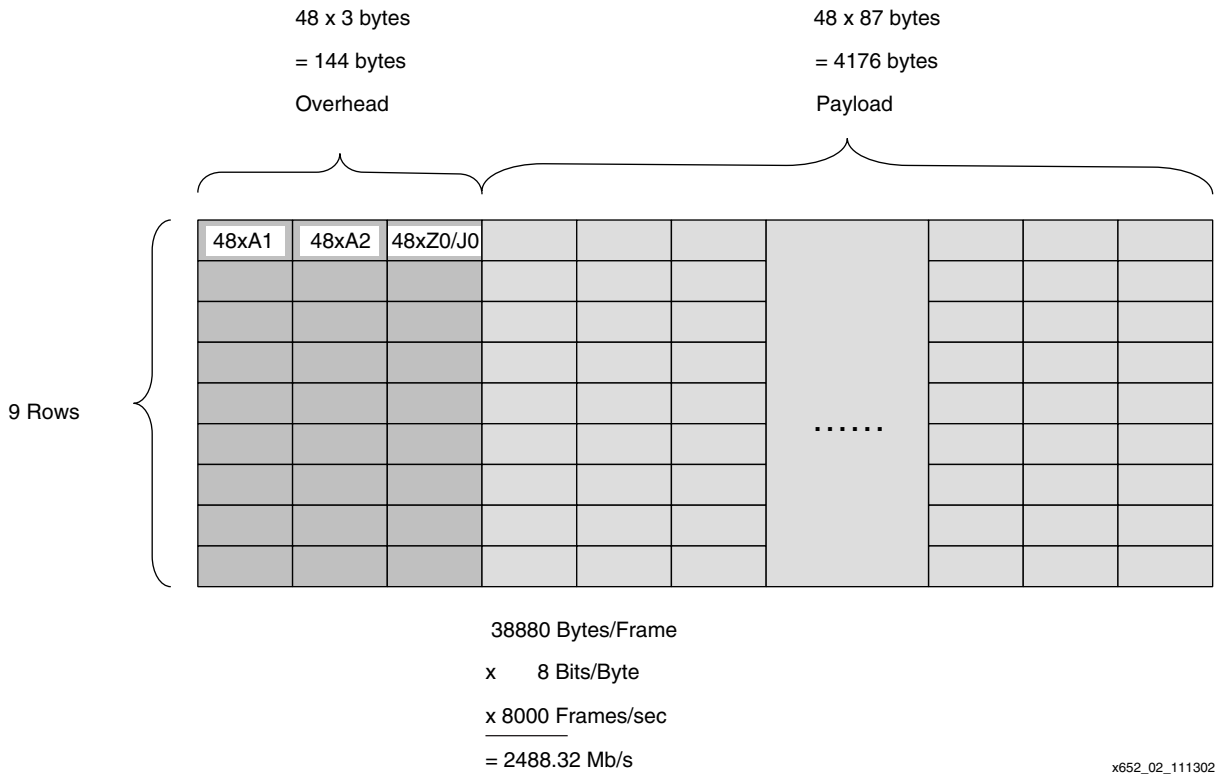

Figure 2: **STS-48/OC48 Frame Format**

Circuit Description

The circuitry for a word aligner is very simple, an example for a 16-bit data path is shown in Figure 3. Assuming the incoming data is initially unaligned, operation is initiated by the search input being asserted for one or more cycles. Once the search input transitions Low, searching for the frame header data can start.

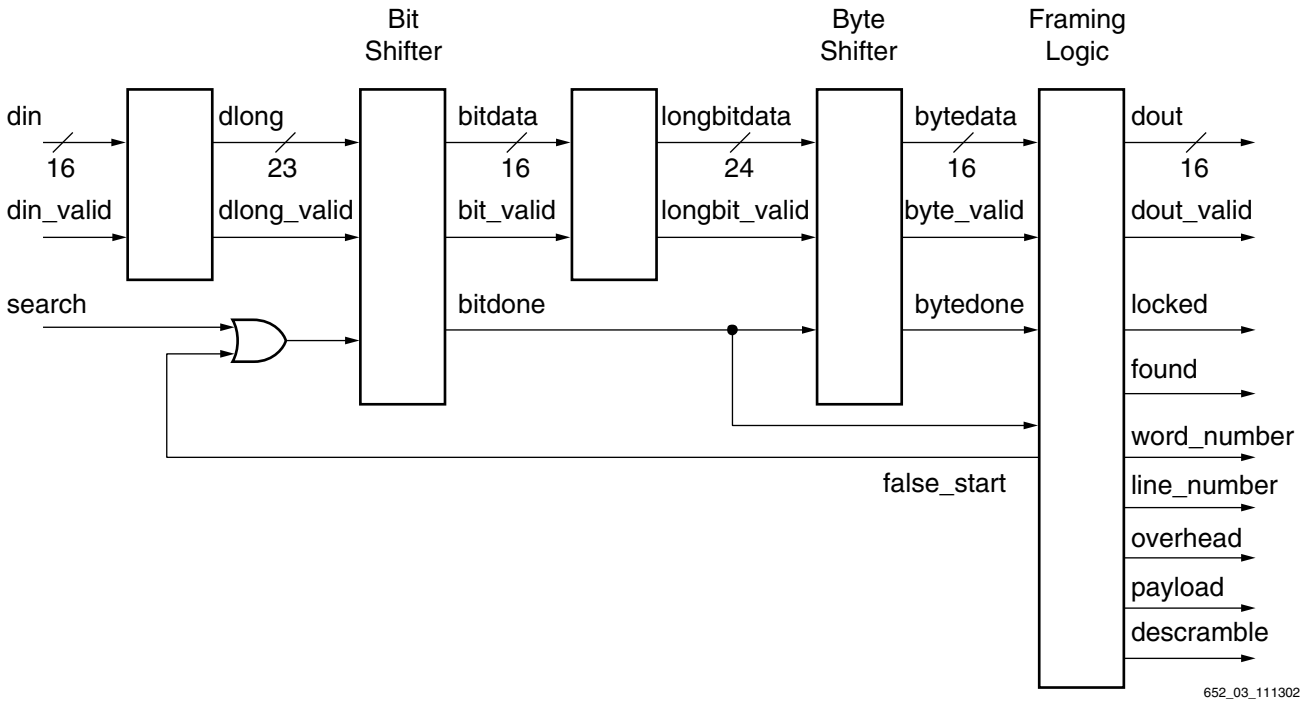

Figure 3: **16-bit Data Path Block Diagram**

The incoming 16-bit data word is combined with the lower seven bits of the data word from the previous cycle, and then fed into a bit-shifter module (shown in Figure 4). The bit shifter contains 16-bit comparison logic to test for the presence of the A1A1 bit pattern (`0XF6F6h`) in any of the eight possible positions of the 23-bit input word. If the A1A1 is found in the same position on two consecutive cycles, assumes the first part of the frame header period of the frame is found. There are 48 consecutive A1 bytes in an OC48 frame header. The encoder generates a 3-bit signal to shift the incoming data word from 0 to 7 bit positions. The incoming data is now assumed to be bit aligned, and the *bit_valid* flag is raised.

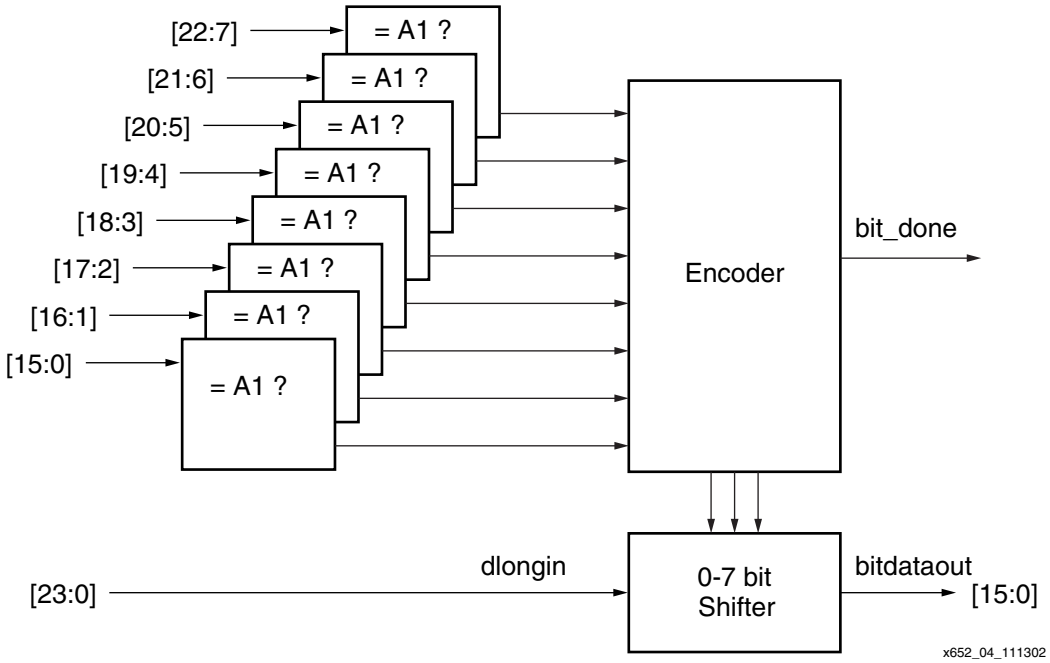

Figure 4: **16-bit Data Path Bit-Shifter**

A similar process is then performed for byte alignment. A 16-bit data word is combined with 8 more bits and then fed to the byte-shifter module shown in Figure 5. The byte shifter contains 8-bit comparison logic that searches for both the A1 octet (`0XF6h`), and the A2 octet (`0X28h`) once the *bit_done* input has gone high and the input data can be assumed to be bit-aligned. If the search logic finds the pattern A1A2 in either of the two possible positions (bits [23:8] or [15:0]), it is assumed that this is the A1A2 transition in the frame header, and the *byte_done* flag is asserted. The encoder will generate a one-bit signal that shifts the incoming data either 0 or 8-bit positions, so that the output data is precisely aligned on the A1A2 boundary. The output will therefore be A1A1 followed by A2A2.

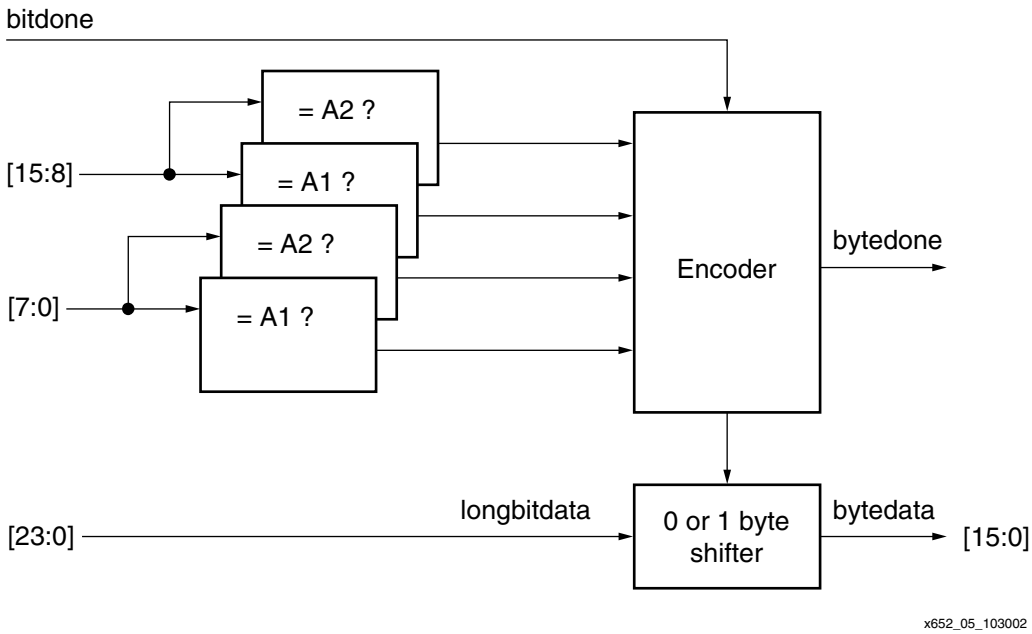

Figure 5: **16-bit Data Path Byte Shifter**

The byte-aligned data is now fed to the framing logic. This again performs a comparison looking for the A1A2 transition. If this transition is found in the correct place, three times running, then the signal "locked" will be asserted. In addition, each time the A1A2 transition is found, the *found* signal will be asserted for one clock cycle. If the transition is not found in the correct place, then the *false_start* signal is used to reset all the logic, and the search begins again. If the frame pattern is found in the correct place *n* times, then the locked signal is asserted. The value *n* is determined by a constant in the Verilog/VHDL code, and is set to a default value of "3".

Found implies that the circuit finds an A1A1 word immediately followed by an A2A2 word.

The "correct place" is determined by the type of data that is being received, and the basic 8 KHz rate of telecom systems. In the case of OC48, the A1A2 transition should occur every 38,880 octets (19,440 clocks assuming 16-bit wide data.)

Once alignment is achieved the output signals available are:

word_number	This is the word number of the current data (0 to 2159 for OC48)
line_number	This is the line the word is in (0 to 8)
overhead	This indicates the receipt of the overhead portion of the frame
payload	This indicates that the word is part of the payload
descramble	This is used to enable and disable the scrambling logic described in **XAPP651**.

Once locked, the circuit continues in operation until either a search command is issued to it, or an A1A2 is not found in the correct place for *m* consecutive cycles. The value *m* is determined by a constant in the Verilog/VHDL code, and is set to a default value of "3". In either case the *locked* output signal is de-asserted.

Finally, a clock enable *din_valid* signal and output data valid *dout_valid* signal are included for cases where the incoming data is coming from a FIFO.

64-bit Data Path

Extending the data paths to 64-bits wide allows word alignment of OC192 data at 10 Gb/s. The circuit design is very similar, but the data paths are wider as shown in Figure 6. The only real change is in the complexity of the byte shift. It now has to shift the incoming data from zero to seven bytes. The framer still checks for only 32-bits of the header pattern, i.e., if A1A1A2A2 is correctly aligned. This is a generally accepted technique.

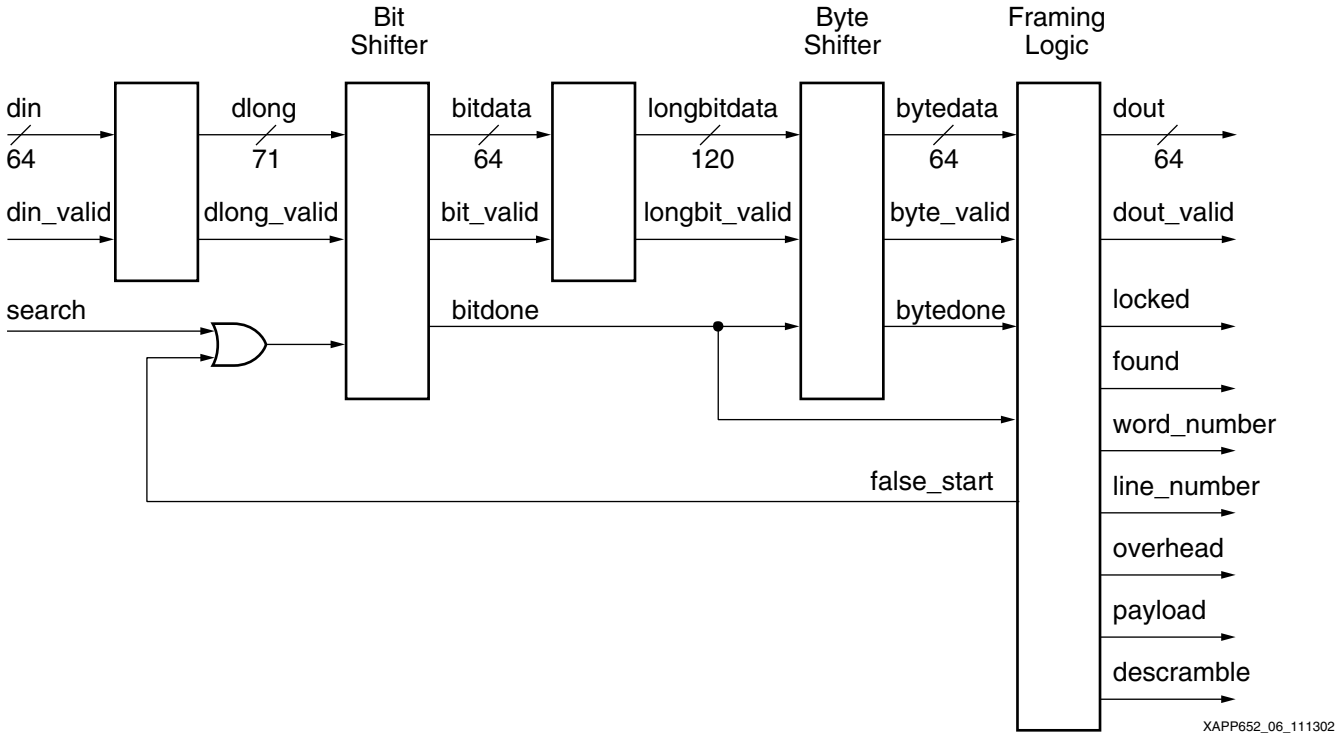

Figure 6: **64-bit Data Path Block Diagram**

Reference Design

Using a 16-bit or 64-bit data path, the reference design will run at around 180 MHz in a Virtex-II -4 speed grade device. The 16-bit design uses about 150 Virtex-II slices, and the 64-bit design uses around 500 slices.

The reference design files for the various cases, written for the Virtex-II series are available in both Verilog and VHDL on the Xilinx web site (**xapp652.zip**). The .zip file includes an up-to-date readme.txt file.

Conclusion

The reference design files show the flexibility of the Virtex-II Series Platform FPGAs when implementing word aligners. The files in this application note can be combined with those for scrambling (**XAPP651**) and rate conversion (**XAPP649**) to allow the use of the Virtex-II Pro RocketIO transceiver for SONET OC48 use across system backplanes.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/15/02	1.0	Initial Xilinx release.