The fPLL as a Replacement for External VCXOs

Chris Maryan
IPD Department
Altera Toronto
150 Bloor Street West Suite 400
Toronto, Ontario, Canada
Tel: +1.416.926.8392
cmaryan@altera.com

Boon Hong Oh
IPD Department
Altera Penang
Plot 6, Bayan Lepas Technoplex
11900 Bayan Lepas, Penang
Tel: +6.04.636.8366
bhoh@altera.com

Aiman Zakwan Jidin
IPD Department
Altera Penang
Plot 6, Bayan Lepas Technoplex
11900 Bayan Lepas, Penang
Tel: +6.04.636.6821
jazakwan@altera.com

ABSTRACT

Broadcast video products require every independently clocked video path to have at least one external voltage controlled crystal oscillator (VCXO) and corresponding PLL circuitry. This causes VCXOs to be a significant percentage of the bill of materials (BOM) cost for video customers, especially as FPGAs sizes have increased to allow handling of many video paths within a single device.

In this paper we present the development of a system to use the fractional PLLs in Altera's 28nm products as a replacement for costly external VCXOs and space consuming external PLL circuitry. Our implementation covers the design of a PLL in soft logic, using DSP resources, which is used to control an fPLL via dynamic reconfiguration. We describe some of challenges in generating the various required video clock rates from different types of reference sources. Our work also included experiments on configuring the engineering features of the fPLL to minimize transfer of fractional spurs and meet SDI jitter specifications.

The resulting IP allows fPLLs to completely replace external VCXOs and any external circuitry. Multiple instances of this IP can be implemented to drive independent video paths. For broadcast customers this produces very significant cost savings, potentially exceeding the cost of the FPGA. This work has implications for future FPGA PLL and clocking architecture decisions. It may also serve as the foundation for VCXO replacement in areas such as Optical Transport Network (OTN).

Keywords

SDI, VCXO, fPLL, PFD, BOM, Mfrac, DSM, HSYNC, PID, Kp, Ki

1. INTRODUCTION

Digitization of video signals has been common practice in broadcast video for many years. Early digital video was commonly encoded on a 10-bit parallel bus, but as higher processing speeds became practical, a serial form of the digitized video signal called the Serial Digital Interface (SDI) was developed and standardized. Serialization of the digital video stream greatly facilitates its distribution within a professional broadcast studio.

SDI is based on a 27MHz, 74.25MHz, 148.5MHz, 74.25/1.001 or 148.5/1.001 reference pixel clock depending on the exact standard – at the highest data rate this results in an approximately 3Gb/s serial signal for 20 bit/pixel High Definition (1080p) video.

As with any source clocked streaming interface, there are small differences in local oscillator frequencies, causing different devices to produce video streams with slight offsets in their frequencies. The local oscillator frequency on a given device also varies over time due to voltage and temperature fluctuations – this shows up as jitter. The SDI specification allows for a maximum offset of +/-10ppm from the nominal refence clock frequencies for static offsets. There is also a specification on the maximum jitter allowed over a particular bandwidth.

This variation in the clock frequency requires a device that retransmits the signal to lock to the incoming clock recovered from the data, and regenerate the clock to use as a reference for transmit.[1] This is known as reclocking.

The process of generating the transmit clock by using the recovered receive clock as a reference is accomplished using a phase locked loop (PLL). Because of the small frequency range involved, a VCXO is usually used as the oscillator.[2]

Most SDI devices perform some type of reclocking, causing PLLs and correspondingly VCXOs to be commonplace in video equipment. Every independently clocked video path requires at least one VCXO device. As a result, VCXOs tend to be a significant percentage of the bill of materials. As FPGAs have grown to handle more video paths in a single device, the number of VCXOs on the board has also grown. To further complicate matters, becaue PLLs and VCXOs are analog circuits, board layout is made difficult by the need to make consessions to complex analog layout requirements.

For Altera's 28nm products, a delta-sigma fractional PLL architecture was introduced on top of the existing integer PLL. A fractional PLL (fPLL) allows output frequency to be multiplied by an integer and a high granularity fraction. In an fPLL the feedback divider is modulated continuously such that the averge divide ratio is a fractional value. A third-order delta-sigma modulator is

[1] The common exception to the rule that a device must regenerate the clock are non-reclocking video routers. These are often effectively digital switches based on multiplexers, where no clock recovery happens. However, non-reclocking devices generally add jitter. Note that not all routers are non-reclocking.

[2] In recent years VCXOs have been partially replaced by clock generating integrated circuits that perform similar functions. However the price and performance remains comparable.

used to shift the fractional noise to high frequencies to be filtered out by the PLL. The high granularity and jitter reduction properties of Altera's fPLL make it good system to use as an arbitrary frequency clock source.

In this paper, we present the development of a system to use the fPLLs in Altera's 28nm products as a replacement for costly external VCXOs and space consuming external PLL circuitry. This paper is organized as follows. The need of synchronization within video network components by using VCXO solution is described in Section 2. It is followed by introducing the VCXO-less implementation that covers the development of an innovative clock control scheme and the creation of an extremely useful soft IP block for broadcast video applications in Section 3. The work covers the challenges of tuning the system to have sufficiently low jitter to meet the SDI specifications, while not relying on low noise external VCXOs. This extends to the challenges that faced and resolutions taken to counter the challenges that described in Section 4. The performance of the implementation is proven by several experiments. The results are presented in Section 5. Finally, the conclusion is drawn in Section 6.

2. VCXO SOLUTION

Many components are involved in the production of the final broadcast video signal. Distribution amplifiers send video signals around the studio and perform the job as a repeater to help clean up the signals they distribute. Frame synchronizers ensure that all video sources are synchronized making the process of switching, keying, and editing possible. Video routers provide a single point of switching and routing video signals to and from other studios. All of these video components must perform a receive and then subsequently re-transmit. An SDI reclocker is required in all of these components to help remove the jitter from the SDI data stream or to reclock the data onto a new clock domain.

Reclocking can be used to recreate the receive clock to use for transmit, potentially with lower jitter. When an SDI device reclocks a video stream to a recoverd clock it can include a short FIFO buffer (phase compensation FIFO) between RX and TX. This will compensate for small temporary differences between the RX clock and the generated TX clock that are caused by jitter. Thus, if the PLL has a low bandwidth, reclocking can be used to reduce jitter on an SDI signal.

Reclocking can also be used to transfer the video to a completely new clock domain. In video applications it is also desireable to switch between two video streams. Doing this without creating broken video output requires the change from one video stream to another to happen between the active portion of the video frame. The active portion is effectively the portion that contains the image, the remainder of the frame contains either ancilliary data or is blank. When two video streams come from different sources, with different offsets in their clock rates, they must be synchronized onto a common clock domain to allow convenient switching between them. A device known as a master clock provides a reference signal that defines the new common clock domain. Each of the incoming asynchronous video streams is passed through a device that generates a transmit clock based on the master clock input. The device uses an elastic frame buffer to drop or repeat video frames whenever the accumulated offset between RX and TX reaches a point that would cause the buffer to overflow or underflow. The result is that video can be reclocked onto a completely new clock domain, and multiple video sources can be reclocked onto a common clock domain. The device in this case is known as a frame synchronizer.

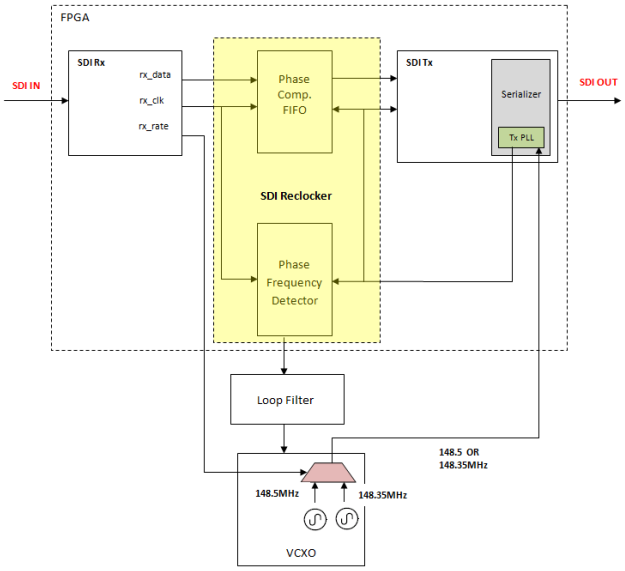

Figure 1 SDI reclocker

Regardless of the application, the PLL circuitry is similar. A reference is provided, either from the incoming video or a master clock. This feeds a phase-frequency detector (PFD) that measures the error between the reference and the generated clock. The PFD error, in the form of up/down pulses, is then averaged in a loop filter. Typically in VCXO based solutions the filter is implemented external to the FPGA as a charge pump and R-C lowpass filter made from discrete components. The output of the loop filter drives the control voltage of the VCXO. The VCXO output is used as the TX reference clock, and it is also used to close the feedback loop to the PFD. See references 1 and 2 for a thorough discussion of PLLs.

If any of the video components has more than one independent channel, more than one PLL circuit and VCXO is required is required. For example, a 4x4 video router acts like a switchboard for video signals. It routes video signals from one place to another. Video signals are switched through a cross-point matrix, reclocked, and sent to its destination. The video signal is processed as a serial stream from input to output. A total of 4 VCXOs are required in this 4x4 video router. This is illustrated in Figure 2

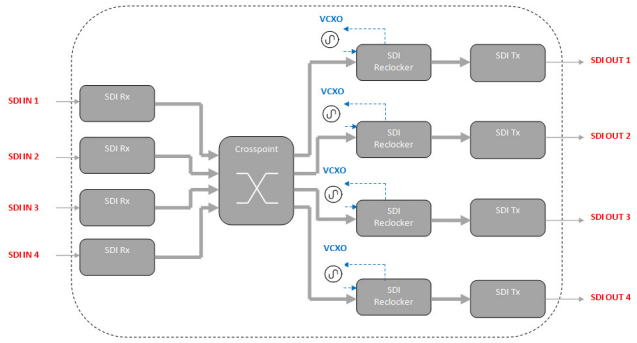

Figure 2 SDI 4x4 video router

Considering another example, an SDI dual distribution amplifier has 2 SDI inputs to receive multiple video signals and help extend the distance that the video signal can reach or not to distribute the video signal to multiple sources. A total of 2 VCXOs are required in this dual distribution amplifier. This is illustrated in Figure 3.

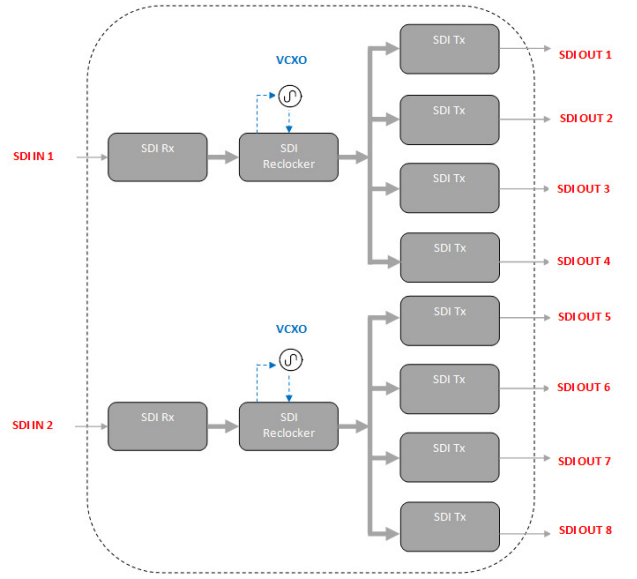

Figure 3 SDI dual distribution amplifier

Using traditional VCXO design methods, a production switcher or broadcast router can require dozens of VCXOs to allow FPGAs to pass through (SD/HD/3G-SDI) video channels. Priced at US$15-$25 each, the VCXOs add a significant and growing cost to the BOM because video networking equipment is scaled up to carry large numbers of channels.

Altera FAEs are currently working with customers looking to place more than 16 independent video paths in a single FPGA. At this point the BOM cost of VCXOs can exceed the cost of the FPGA, and board design becomes extremely challenging.

3. VCXO-LESS SOLUTION

The VCXO-less solution replaces the conventional VCXO and discrete PLL with an fPLL and soft logic contained entirely within the FPGA. A dynamically reconfigurable fPLL can conceptually serve the same function as a VCXO, provided that it has sufficient granularity, range and frequency update speed.

Fractional PLLs use divide counters to perform frequency synthesis. The counter settings can be reconfigured to adjust the fPLL output clock in real time without reconfiguring the entire 28nm FPGA devices. In the VCXO-less solution, the only fPLL component that is needed to be reconfigured in real time using the dynamic reconfiguration IP is the fractional division (Mfrac) for delta-sigma modulator (DSM).

The Altera fPLL supports a 32-bit fractional divide word – this results in an approximate granularity of Fout*$1/2^{32}$. When generating 148.5MHz, this translates to $3.5x10^{-2}$Hz or 0.002 ppm per LSB step. This is comparable or better than the noise floor that can be achieved with a typical 100ppm pull range VCXO.

Characterization results on dynamic reconfiguration of the fPLL show that the fractional value can be updated more than the equivalent of 1000ppm in a single step without the PLL losing lock. Additionally, the settling time for a small frequency update has been measured to be 10us. Both of these figures indicate that the fPLL has the reconfiguration characteristics to emulate a VCXO.

In the VCXO-less implementation, soft IP with DSP resources is created to work in conjuction with the fPLL as a functional replacement of a discrete PLL. This is illustrated in Figure 4. The soft IP is complete PLL implementation, including a phase-frequency detector (PFD) and loop filter. The reference signal is the HSYNC signal that is extracted from the incoming video stream, or an HSYNC pulse from a master clock, while the feedback signal is a divided version of the fPLL output clock. The HSYNC signal is a horizontal sync pulse indicating the start of a line of video. It is commonly used as timing reference since it occurs as a repetitive and accurate rate. In this application, the HSYNC signal is used in place of using the RX recovered clock directly to make the system easily switchable to use a master clock output, which only provides HSYNC signals.

The PFD is a traditional flip-flop based PFD, that compares both reference and feedback signals to generate the up and down pulse width. The pulse length of the up and down pulses is measured relative to a fast clock, and subtracted to generate a phase error signal. This is then sent to the digital loop filter. The digital loop filter implements the Proportional-Integral (PI) control algorithm. The phase error drives the proportional and integral elements of the PI controller. The resulting signals are added and used to drive the fPLL.

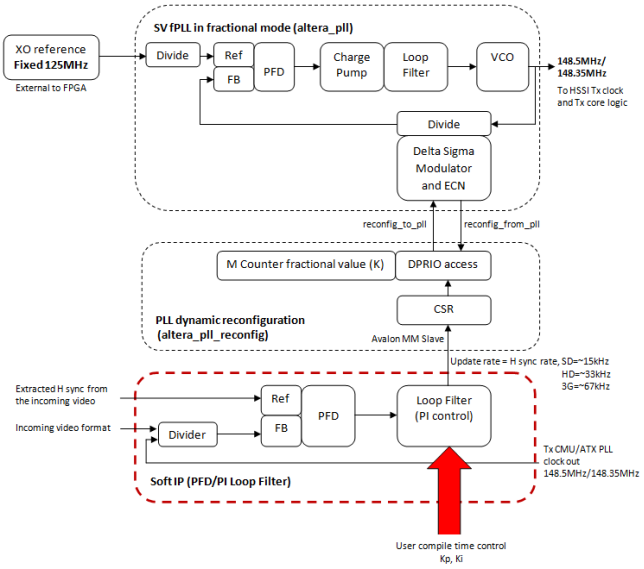

Figure 4 VCXO-less implementation

In a typical PI controller, each of the proportional and intergral elements performs a different tasks and has a different effect on the functioning of a system. These elements are driven by a combination of the system command and the feedback signal from the object that is being controlled. Their outputs are added together to form the system output. The PI control algorithm that has been implemented does not require heavy mathematics and without requiring user to learn any control theory.

Proportional control is the easiest feedback control to implement, and simple proportional control is probably the most common kind of control loop. A proportional controller is just the error signal multiplied by a proportional gain constant *Kp* and fed out to the drive. The proportional term gets calculated with the following formula:

$$Pout = Kp * e(t)$$

A high proportional gain results in a large change in the output for a given change in a error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. Figure 5 shows the pure proportional gain response.

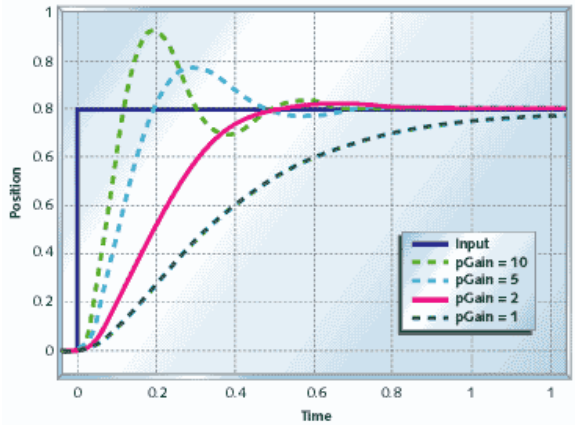

Figure 5 Pure proportional gain response

Integral control is used to add long-term precision to a control loop. It is almost always used in conjunction with proportional control. The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PI controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain *Ki* and added to the output. The integral term is given by:

$$Iout = Ki * \sum e(t)$$

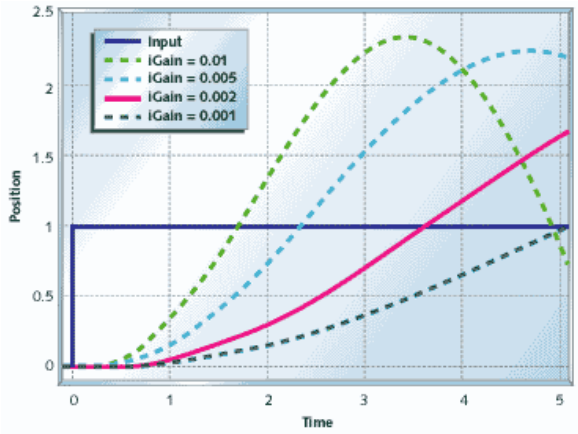

Figure 6 Pure integral gain response

The integral term accelerates the movement of the process towards setpoint and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the setpoint value. Integral control by itself usually decreases stability, or destroys it altogether. Figure 6 shows the system with pure integral control and the system does not settle or it may takes extremely longer time to settle. If the system is equipped with proportional and integral control as shown in Figure 7, the position takes longer to settle out than the system with pure proportional control (Figure 5), but it will not settle to the wrong spot.

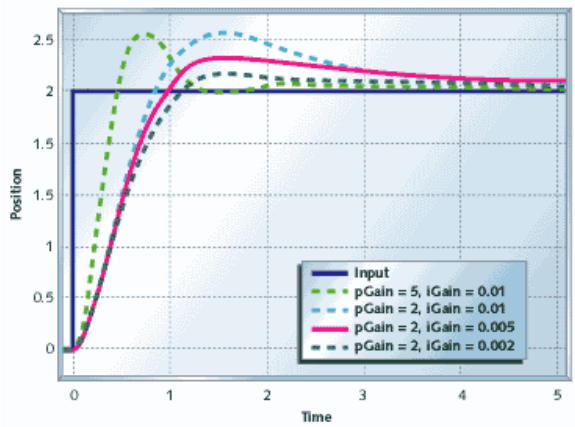

Figure 7 Combined proportional and integral gains response

The sum of the integral and proportional terms, the control signal, is the fractional division (Mfrac) value provided to the the fPLL. This value will is updated in the fPLL via the PLL dynamic reconfiguration IP. The update rate varies between different video rates, but generally matches the HSYNC rate. The worst case is approximately 15kHz for standard definition video (SD-SDI - 270Mbps) HSYNC rate. The update is continuously performed as long as the PFD detects phase error between the reference and feedback signals. The fPLL output clock is cascaded to the transmitter PLL in the high-speed serial interface (HSSI) channel. Each fPLL together with this soft IP in each SDI channel can be independently, dynamically, and continuously changed to track an independent reference and hence act as a VCXO replacement.

4. CHALLENGES

Tuning a control loop is the adjustment of its control parameters such as proportional (*Kp*) and integral (*Ki*) gains to the optimum values for the desired control response. Stability or bounded oscillation is a basic requirement, but beyond that, different systems have different behavior, different application have different requirements, and requirements may conflict with one another. Designing and tuning a PI controller appears to be conceptually intuitive, but can be hard in practice, if multiple and often conflicting objectives such as short lock acquisition time and high stability or low accumulated jitter are to be achieved.

There are several methods for tuning a PI loop. One of most effective methods for clock synthesis application is to first set *Ki* value to zero and increase the *Kp* until the output of the loop oscillates. For example, consider that the incoming serial data rate is 2.96993396Gbps, for which the recovered clock rate is 148496698Hz, and the while the local reference clock rate which is the fPLL output clock is idealy 148500000Hz. By just tuning the *Kp*, keeping *Ki*=0, the system can achieve frequency lock between the incoming rate and the local reference rate. The fPLL output clock will be dynamically reconfigured by the control loop from 148500000Hz to 148496698Hz. Figure 8 shows the signaltap waveform after *Kp* tuning. The first data bus shows the number of recovered clock cycles measured in one second. The second data bus shows the number for fPLL output clock cycles measured in one second. The third signal is the HSYNC reference extracted from the incoming video stream based on recovered clock domain. The fourth signal is the divided version of the fPLL output clock that is frequency matched and locked to the incoming HSYNC rate. The resf of the signals are PLL and Rx lock signals, respectively. From the figure, the system is able to achieve frequency lock, not phase lock with just proportional control.

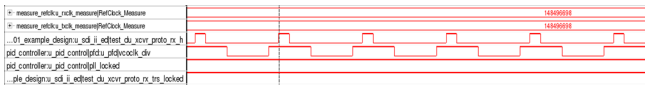

Figure 8 Frequency lock with *Kp* tuning

Then, *Kp* is set and *Ki* is increased until the system achieves both frequency and phase lock. This is shown in Figure 9.

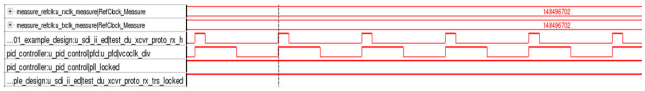

Figure 9 Phase lock with *Ki* tuning

As described in section 3, a high *Kp* results in a large change in the output for a given change in a error, meaning the system tends to achieve steady-state with short lock acquisition time. However, the system also tends to become unstable and oscillates. This results in bad accumulated jitter during steady-state. In contrast, a low *Kp* improves steady-state behavior in terms of oscillation and accumulated jitter, but the system will take longer time to settle. The two main objectives are conflicting with each other.

To fulfill both objectives which are short lock acquisition time and good accumulated jitter, the system is enhanced to initially set a high *Kp* before it reaches steady-state and then switch to take low *Kp* once it reaches steady-state. The steady-state is considered reached if the very small phase error has been found consecutively between the reference and feedback signals.

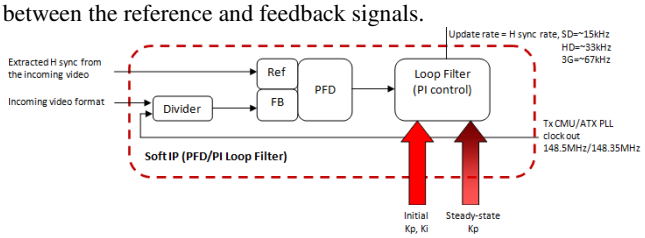

Figure 10 Initial *Kp* and steady-state *Kp*

Apart from loop tuning, the system has to accommodate for various incoming reference rates to generate the desired output clock frequency. It could be as high as 67kHz for HD 1080p video (3G-SDI - 2.97Gbps or 2.967Gbps), around 33kHz for 1080i or 720p video (HD-SDI - 1.485Gbps or 1.4835Gbps) and at worst of around 15kHz from SD-SDI (270Mbps). Further criteria that must be met are an ability to slew between +10ppm and -10ppm as required by the SMPTE SDI specification. In practice to accommodate for the variation in the reference clock feeding the fPLL the target lew range should be at least +/- 100ppm. A solution good enough for video test equipment would cover up to 1000ppm.

5. EXPERIMENTS

The VCXO-less solution is first implemented in Altera's high-end 28nm device – Stratix V. The design and experiment setup is illustrated in Figure 11.

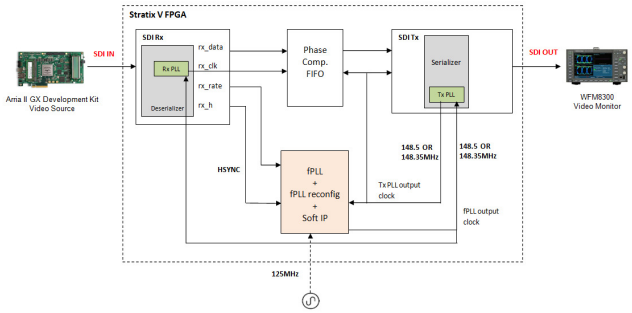

Figure 11 Experiment setup

An Arria II GX development kit is configured as a video source to transmit a SD/HD/3G-SDI video test stream with a colorbar test signal. A Stratix V is the device under test, loaded with test IP composed of an SDI receiver, transmitter, phase compensation FIFO and the fPLL together with the soft IP. A Tektronix WFM8300 video monitor is used to measure the jitter performance of the video stream that is clocked by the fPLL.

Two types of jitter are defined by Society of Motion Picture and Television Engineers (SMPTE) in the SDI spedcification: timing jitter and alignment jitter. Timing jitter covers the entire jitter frequency spectrum starting at 10Hz – it is aimed at ensuring that low frequency variation is bounded to allow buffers to be reasonably sized. Alignment jitter focuses on the jitter frequency spectrum that clock and data recovery unit (CDR) circuits cannot track. Table 1 shows the official jitters specification for different video standards. Jitter in the timing band is a function of the control loop, while jitter in the alignment band is a function of the clock source and transmitter. Thus tuning the control loop coefficients affects the resultant timing jitter, while ensuring that the fPLL and transmitter are optimally configured affects the alignment jitter.

Video Standard	**Timing Jitter (10Hz)**	**Alignment Jitter (1kHz:SD, 100kHz:HD/3G)**
SD-SDI (270Mbps)	0.2 UI	0.2 UI
HD-SDI (1.485 or 1.4835Gbps)	1.0 UI	0.2 UI
3G-SDI (2.97 or 2.967Gbps)	2.0 UI	0.3 UI

Table 1 Offical jitter specification

Other than getting the optimum gain values for the PI controller as described in Section 4, there are some other engineering features of the fPLL that could impact the jitter performance. Throughout the experiment, some hypotheses have been made that the following criteria must be met for minimal jitter:

1. High frequency of the fPLL input reference clock
2. High frequency of the fPLL VCO
3. Mfrac in 32-bit that is closer to 0.5
4. High bandwidth with optimum charge pump current
5. Delta-sigma modulator set to 3rd order

Based on the fPLL formulas,

$$Fin * (M + Mfrac / 2^{32}) / (N * C) = Fout$$

which *Fin* is the fPLL input reference clock; *Fout* is the desired fPLL output clock; *M, N, C* are the counters value; *Mfrac* is the fractional division that is dynamically changed.

$$Fvco = Fin / N * (M + Mfrac / 2^{32}) * pll\_vco\_div$$

which *Fvco* is the VCO frequency; *pll_vco_div* is the post-VCO divider.

To fulfill all of the criteria, the optimum fPLL settings were determined to be as listed in Table 2.

fPLL Params	**Value**
Fin	125MHz
Fout	148.5MHz
M	10
N	1
C	9
DSM mode	3rd order
DSM bit	32-bit
Initial Mfrac	2972117369
pll_vco_div	1
pll_cp_current	40
pll_bwctrl	2000
VCO frequency	1336.5MHz

Table 2 fPLL optimum settings

The VCXO-less solution with the optimum fPLL settings yields a low jitter stream and compliance with SMPTE SDI specifications. The jitter performance is competitive with a VCXO solution. Table 3 shows the results obtained from the experiment.

Video Standard	**Timing Jitter (10Hz)**	**Alignment Jitter (1kHz:SD, 100kHz:HD/3G)**
SD-SDI (270Mbps)	0.05 UI	0.04 UI
HD-SDI (1.485 or 1.4835Gbps)	0.12 UI	0.06 UI
3G-SDI (2.97 or 2.967Gbps)	0.26 UI	0.13 UI

Table 3 Jitter performance with VCXO-less solution

Figure 12-17 show the timing jitter eye diagram and alignment jitter number details for each video standard. These results are captured by WFM8300.

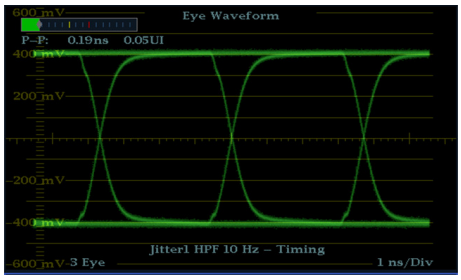

Figure 12 Timing jitter (10Hz) for SD-SDI

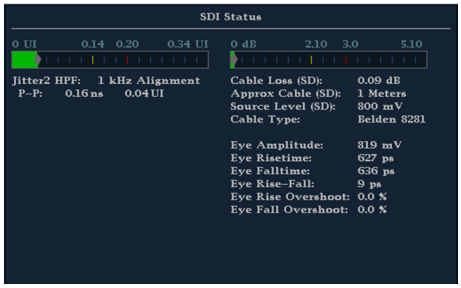

Figure 13 Alignment jitter (1kHz) for SD-SDI

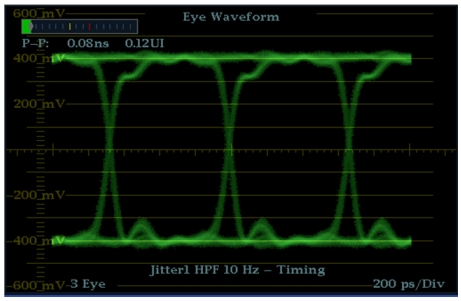

Figure 14 Timing jitter (10Hz) for HD-SDI

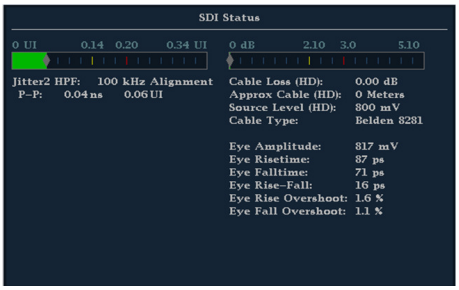

Figure 15 Alignment jitter (100kHz) for HD-SDI

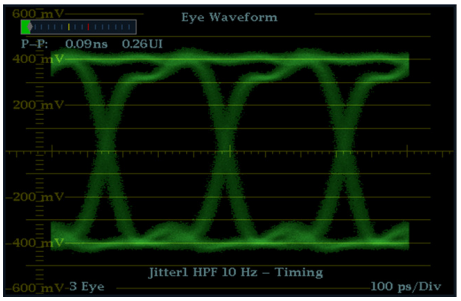

Figure 16 Timing jitter (10Hz) for 3G-SDI

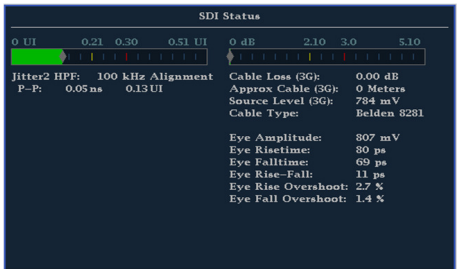

Figure 17 Alignment jitter (100kHz) for 3G-SDI

6. CONCLUSIONS

The VCXO-less method provides considerable cost savings for customers, potentially in excess of the cost of the FPGA. This is an IP solution that is in very high demand from broadcast customers. This lays the foundation for the removal of external VCXOs, which may prove to be the biggest step in FPGA feature integration since the addition of onboard transceivers. Beyond broadcast, this has potential implications in the much larger market of wireline communications.

7. REFERENCES

[1] Phase Locked Loops: Design, Simulation, and Applications, Roland Best, McGraw-Hill, 2007.

[2] Phaselock Techniques, Floyd M. Gardner, Wiley-Interscience, 2005.

[3] PID Control – PID without a P-hD, Tim Wescott, EE Times-India, October 2000.

[4] Timing and Synchronization in Broadcast Video, Silicon Laboratories, Rev.0.1 8/09, 2009.

[5] HD Video – Reclocking for restoring low-jitter HD digital video, Mark Sauerwald, EE Times-India.

[6] VCXO Tuning Slope, Stability, and Absolute Pull Range, Silicon Laboratories, rev.0.2 10/10, 2010.

[7] Implementing Fractional PLL Reconfiguration with ALTERA_PLL and ALTERA_PLL_RECONFIG Megafunctions, AN661, May 2012

[8] Stratix V PLL SWIP Functional Description, Christian Refvik & Kevin Mai, October 2009.

[9] PLL Dynamic Reconfiguration SWIP Functional Description, Lyndon Calvalho, June 2011.

[10] SDI MegaCore Function User Guide, Altera, November 2011

[11] SDI II MegaCore IPD Functional Description, Boon Hong Oh, August 2012.