FPGA Based Prototyping of an ADPLL Network.

C. Dooley, E. Blokhina, B. Mulkeen
School of Electronic Engineering
University College Dublin
Belfield, Dublin 4, Ireland.
email address

Abstract—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.*

Index Terms—FPGA, ADPLL, Simulation, Prototype

I. MOTIVATION

The technical motivation underlying this paper is the creation of an Field Programmable Gate Array (FPGA) based prototyping platform for All Digital Phase Lock Loop (ADPLL) networks. As the creation of Application Specific Integrated Circuits (ASICs) is an expensive and time consuming process, any mistake has the potential to be rather costly. As such prior to the manufacture of an IC it is important to ensure that any errors made in the design have been weeded out prior to this stage. Simulations at either at theoretical or gate/transistor levels have global usage in minimising such errors due to the ubiquity of simulators and their ease of use.

However simulations are only as good as the model used to describe the dynamics of the system, and emulating real jitter and other behaviours of a complex system is a rather difficult task. An FPGA based prototype allows the system designers to validate the performance of both design and simulation, particularly the response to key noise sources such as power supply noise. An FPGA is ideal for this task as it leverages the existing skill set of a digital designer, permits the re-use of certain blocks, and most importantly enables cost effective and rapid reworking of the design.

An FPGA as a prototyping tool has seen use in the work of Dimitri Galayko and his research group in UPMC Sorbonne and in Elena Blokhina's team in University College Dublin, and has been used in the testing of established designs prior to their implementation in custom silicon [?], [?], the creation of realistic models for use in high level simulations [?] and the exploration of new designs for digital blocks before progressing on to true digital design on a gate level. In the case of Shan et. al [?] direct testing of control parameters was performed using an FPGA based design where all parameters were scaled down proportionally and thus the ideal gains for the loop filter found in simulation could be tested in a more realistic setting.

Similarly Koskin et. al [?] used the analysis of a number of ADPLLs implemented on an FPGA both independently and in a network to validate previously obtained theoretical results

while avoiding the costs associated with the development of a custom IC.

Prototyping on an FPGA does have its drawbacks, most notably the mixed-signal circuits central to the operation of an ADPLL are not implementable on an FPGA and while most fundamental digital circuit elements are present these elements are not true transistor level implementations of their functionality but rather implemented by lookup tables and other multi-role structures on the FPGA. These drawbacks present a challenge to a designer as they restrict the potential architectures of key blocks. This is of particular concern when trying to mimic the behaviour of an established design on an FPGA.

The particular system that this paper will discuss is an ADPLL network. Based on an idea first proposed in a 1995 paper by Pratt and Nguyen [?], in which they proposed an alternative clock distribution network for ICs using a Cartesian grid of clocking areas, each with their own Phase Lock Loop, which has become known as a PLL Network . This methodology was implemented by Gutnik et. al [?] who found it to be feasible. Subsequently Javiden et. al [?] proposed the implementation of such a system using All Digital PLLs which better suited the use cases of the technology and avoided some of the flaws pointed out in the original paper.

This paper will examine the process of creating such a system, highlight the differences between potential designs and address some of the challenges and pitfalls that may be encountered along the way. This paper will also demonstrate that FPGA based prototyping can play a central role on the pathway to the implementation of an ADPLL network on a custom chip, or indeed any number of similar applications. The paper is organised as follows: Section II describes the system that will be used on the FPGA and Section III the implementation thereof and the challenges encountered in the process. Finally Section IV will present example measurements made on the platform and discuss their merit.

II. SYSTEM ARCHITECTURE

An ADPLL network as the name would suggest is created from a number of ADPLLs that coupled using digital phase comparators which attempt to measure the phase error between two oscillators, with each non edge node being connected to four neighbours. Figure 1 illustrates such a system on a small scale, with a number of ADPLLs laid out in a Cartesian grid. The individual ADPLLs, often called distributed ADPLLs as

the phase detectors are shared between nodes, are made up of a digitally controllable oscillator, an error combination block and a digital loop filter with the aforementioned digital phase detectors lying between each pair of oscillators.
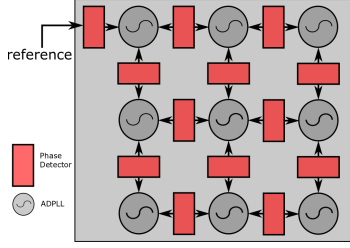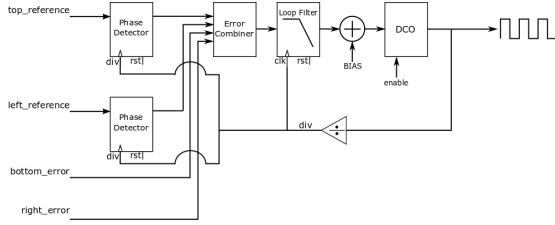

Figure 1. ADPLL Network Architecture.

Figure 2. Distributed ADPLL Design.

A. Digitally Controlled Oscillators

While regular Voltage Controlled Oscillator (VCO) is tuned by the input voltage over a continuous range of values in a digital system there are a very limited number of voltages representable, most commonly just two levels. As such using the voltage itself to control an oscillator is not a viable strategy and instead Digitally, or as they are sometimes known as Numerically, Controlled Oscillators (DCO) accept an *n* bit wide control signal that alters the period of the output waveform. While tuning range, centre frequency and linearity apply as in a VCO, a DCO also has a minimum frequency step as a result of the non continuous control values. In combination with the bit width of the control signal the minimum frequency step limits the range over which the oscillator can be tuned. Frequently the output of the DCO is divided down to allow the control blocks in the feedback path that are clocked on this signal to run at a lower and thus easier frequency.

B. Loop Filter

The loop filter is a key aspect of a phase lock loop as without it only the current output of the phase detector could be used to compute the control voltage of the VCO. In an ADPLL it performs an identical role, however the digital environment results the traditional loop filter being replaced by a discrete time Proportional Integral (PI) controller, with the integral path of the controller implemented by an accumulator.

C. Error Combiner

The Error Combiner, as the name suggests, performs a summation of the error signals from the phase detectors lying

between any neighbouring oscillator and its neighbours. It is often convenient for this to be performed as a weighted sum, thus avoiding different gain requirements for the loop filter of oscillators in the corners of the grid that only have a pair of neighbours. The ability to change the weights during operation enables "uni-directional start-up" which Javiden et. al [?] found prevented the system from entering unwanted stable equilibriums warned of in Pratt' and Nguyen's seminal paper.

D. Phase Detector

As in a conventional PLL the phase detector outputs a value proportional to the difference in phase between a pair of signals, which is then output as a continuous voltage. A digital phase detector then attempts to do the same by measuring the difference between fixed reference points in the waveform, which as a digital system deals with square waves this is typically the rising edge of each input to the block. Similarly to the oscillator quantisation also plays a role in the behaviour of this block, with the time difference between each rising edge having a minimum possible measurement step. In distributed ADPLLs the phase detector has dual outputs, each the inverse of the other, in order to feed the correct value to both of the connected error combiners.

III. FPGA PROTOTYPING

A. FPGA Restrictions

The use of an FPGA comes with many restrictions as to how a complex system such as an ADPLL can be designed with the mixed signal blocks, the DCO and phase detectors, in particular being curtailed by the lack of transistor level control of the design and layout.

Two potential solutions exist which alleviate this problem, and are suited to different frequency ranges and/or goals of the designer. Firstly the clock signal generated by the FPGA's clock distribution block can be used to drive accumulator based oscillators and phase detectors a technique suitable in two main use cases: The emulation of the performance of an already designed system operating at a proportionally cut down frequency. At frequencies that are orders of magnitude lower than the maximum obtainable clock frequency of the FPGA's clock manager. The downside of this approach is that the minimum steps are dictated by the clock frequency of the FPGA, which at frequencies that are not orders of magnitude lower than the FPGA clock will result in coarse resolutions for detector or oscillator, and thus greater jitter seen at the oscillator output.

The second solution to the problem is more akin to the real world system, and uses the delay through primitive elements such as inverters, which can be deduced from measurements on the FPGA of choice but will vary depending on the layout selected. In the case of the FPGA used for this paper, a Xilinx Artix-7 (XC7A100T-1CSG324C) this delay is in the region of 300 picoseconds but varies depending on the routing between elements. This design is better suited to scenarios where the desired operating frequency is a significant fraction

of the maximum usable clock of the FPGA or it is particularly desirable to have asynchronous behaviour in the system as in an ASIC. It is not suited to lower frequency designs as the number of inverters required will run into spacial constraints.

B. Digitally Controlled Oscillators on an FPGA

There are three key designs that be used to implement a DCO on an FPGA, two of which rely on the FPGA clock and one using inverter primitives. Each of these designs fulfil a different role in mimicking the behaviour of an ASIC based mixed signal DCO.

Design 1: The first of these is a linear in period design using an accumulator architecture, with a variable reload value on overflow. Here the control code modifies the reload value around a bias point and thus adjusts the time taken for overflow to be reached with the oscillator's period given as $T_{osc} = (2^{\text{width}} - (\text{bias} + \text{code})) \times \text{clock period}$. Period linearity is a desirable characteristic as it mimics the behaviour of many mixed-signal implementations, and is the key use case of this method. The choice of the accumulator's Most Significant Bit (MSB) as the output signal however highlights the drawbacks of this method as less than half of the control codes are feasible. As the minimum period is set by the number of unused bits and the maximum period fixed at twice the minimum the frequency and tuning range combinations are very limited. In addition the wave output by this oscillator is only square at the maximum period and control code changes vary the duty cycle of the signal.

Design 2: If frequency linearity is a desired characteristic of the oscillator an accumulator based, FPGA clock driven design exists for this use case also. Here the control code k is added to the value stored in the accumulator at each rising edge of the FPGA clock until overflow is reached at $2^n - 1$ where n is the bit width of the counter thus valuing each control code increment at $\frac{\text{FPGA clock}}{2^{\text{width}}}$ Hz. Once again the MSB of the accumulator is used as the output signal, however with this design the MSB will spend an equivalent duration at both logic high and low. Again this design is better suited to lower frequencies, where the output of the DCO is orders of magnitude lower than the clock signal driving it.

Design 3: The third and final potential DCO design is an inverter ring/chain which makes use of the delay through inverter primitives, thus bearing significant resemblance to DCOs used on ASICs. On an FPGA the lack of precise control over the layout of blocks means the exact value of the delay is uncertain and may vary noticeably from one implementation to another, however this misalignment and the asynchronous behaviour stemming from the lack of an FPGA driven clock increase the similarity between this design and an ASIC. The period is set by the number of inverters at $2 \times (\text{num inverters}) \times (\text{delay})$, and is varied by adding or removing a pair of inverters for every control code increment. This design works best when the desired period is within a couple orders of magnitude of the delay through each pair of inverters, beyond this point the number of look-up tables and

routing resources required to implement the design becomes unsustainable.

C. Phase Detectors on an FPGA

The two options for replicating the behaviour of a phase detector also employ the same two design principles, with the option better suited to lower frequency signals once again being driven by the clock generated by the FPGA and an inverter based design better suiting higher frequency systems where the greater resolution offered is beneficial.

A Sign-Number (SigNum) detector is a common method of performing phase-frequency detection in an ADPLL and is the method that will be discussed in this paper. As the name suggests this design constitutes two sections, with one determining the sign of the phase difference and the other its magnitude. The first issue with implementing a phase detector on an FPGA is the issue of metastability, where the output of a circuit element is at an undetermined level, which can occur if both signs were to have edges within a short time of another or the set-up/hold times of a register are violated. This issue is easily addressed in an ASIC with an arbiter, however an arbiter is a mixed signal circuit that is not synthesisable on an FPGA.

Design A: The first potential design uses a Finite State Machine (FSM) to determine which edge which edge has occurred first and thus the sign, and a counter that computes the time difference between each rising edge. This counter is clocked by the FPGA clock, thus giving the system a time resolution of $\frac{1}{F_{\text{FPGA}}}$. As before this results in a design that is only suitable when the frequency of operation is well below that of the FPGA clock due to resolution constraints. In order to avoid the problem of metastability the incoming waveforms can be synchronised to the clock of the FPGA for no penalty beyond the extra clock cycle taken as this synchronisation performs the same quantisation that would otherwise have occurred in the counter measuring the time difference. This design is better suited for use with the accumulator based oscillators due to the overlapping frequency ranges, however it can also be used in conjunction with an inverter ring. Important to note is that in the case of the Artix-7 used in this paper this results in a four times worse phase difference resolution than period resolution which made low jitter operation difficult, but this combination could be made possible by removing eight rather than two inverters for each control code step.

Design B: Better suited for use with the inverter based oscillator is a system that mimics the behaviour of a tapped delay line working asynchronously to the FPGA clock. As the FPGA clock is not used, and that of the ADPLL cannot be used to measure itself an unclocked sign detection circuit is required, an example of which can be seen in Figure 3. This is nearly identical to sign detection methods used on an ASIC with the arbitration circuit replaced by a second SR Latch which alleviates the issue of metastability.

A tapped delay line can be emulated on an FPGA through the use of inverter pairs due to their short and approximately known delay placed between each tap. The resolution then is

the propagation time through the pair, although this will once again depend on the layout specifics, which was approximately 600 picoseconds on an Artix-7. This is significantly lower than the 4 nanoseconds achieved using the previous design and thus more suitable for use with an inverter ring oscillator as this resolution is better than the time resolution of the oscillator. Figure 4 illustrates the configuration of the circuit used to emulate the function of a delay line that will be used in this paper.
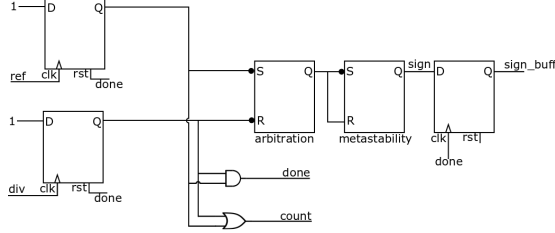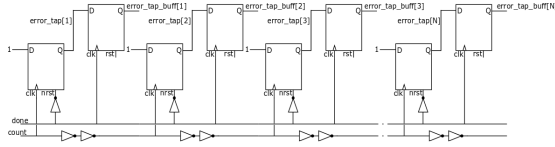
Figure 3. Design B Sign Detection RTL Diagram.

Figure 4. Tapped Delay Line Emulation RTL Diagram.

IV. COMPARISON OF DESIGNS

This paper will examine the performance of a number of combinations of these blocks and the performance thereof, specifically oscillator 2 with phase detector A and oscillator 3 with detectors A & B. For each test there were four ADPLLs on the FPGA arranged in a Cartesian Grid and as such each ADPLL will be referred to using matrix indexing corresponding to their grid location. For each configuration three different modes of operation were examined: Each ADPLL locking to the same 5 MHz external reference, each ADPLL connected to the left and above neighbours in a "uni-directional" network and finally a true ADPLL network where all neighbours are connected. For comparing each configuration gains which had previously been determined to provide the best jitter performance were chosen. The data was captured using an Agilent MSO7054 A at 4 GSa/s. Each point of data is the mean of five 200 microsecond long captures which were then analysed in Matlab, and Table I displays the results of this analysis.

The first columns of the table relate to the cycle-to-cycle jitter of the system, the difference between sequential distances, and the second portion gives the Time Interval Error, or difference between the actual timing of the edge and the "ideal" time as set by the reference. This combination of measurements will show how well each design performs the task of staying consistent from cycle to cycle both in relation to itself and the system at large.

From the table it is apparent that the combination of the inverter chain DCO and the FPGA clock driven phase detector

	Jitter Standard Deviation (ns)			Time Interval E	
	PLL 11	PLL 12	PLL 22	PLL 11	PLL 1
Osc 2, PDET A	-	-	-	-	-
Free PLLs	1.5988	1.6178	1.6220		
Uni-dir.	1.9804	2.0316	2.1317		
Bi-dir.	2.0124	2.0681	2.0732		
Osc 3, PDET A	-	-	-	-	-
Free PLLs	0.62499 μs	0.75919 μs	0.79049 μs		
Uni-dir.	0.76822 μs	0.9009 μs	1.1407 μs		
Bi-dir.	0.70393 μs	0.70201 μs	1.241 μs		
Osc 3, PDET B	-	-	-	-	-
Free PLLs	4.1680	3.9803	5.0741		
Uni-dir.	4.3820	5.7315	6.2336		
Bi-dir.	5.6137	7.1831	7.1612		

Table I

is an ill advised choice, as the jitter present in the system is several orders of magnitude worse than either other situation. In this system the FPGA clock is at 258 MHz, the maximum possible, thus giving a detection resolution of 4 nanoseconds. This is 4 times greater than the period resolution of approximately 1.2 nano seconds of the oscillator, and requires the gains of the PI filter to be increased fourfold in order to respond appropriately to the error detected. The problems of relatively inaccurate detection and increased gains combine to produce a system that is only able to achieve poor locking performance. Both other combinations work well however, delivering a level of jitter that is both low yet allows an indication how impactful variations in architecture or control parameters to be seen visually on an oscilloscope.

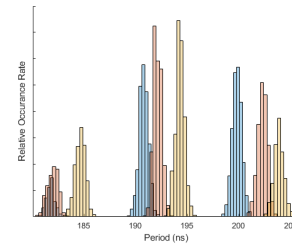Figures 5 & 6 paint a picture of the unrealistic nature of

Figure 5. Design B Sign Detection RTL Diagram.

the FPGA clocked designs as in comparison to the ring based design each oscillator will have identical intrinsic period steps thus eliminating any jitter that maybe be seen in the system due to misalignment. This effects the suitability of the FPGA clocked designs for the improvement of simulation models or the examination of how a novel block maybe impact the behaviour of the network.
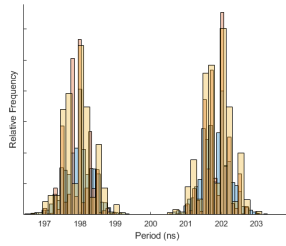
Figure 6.   Tapped Delay Line Emulation RTL Diagram.

CONCLUSION

An FPGA is a suitable environment for the prototyping of ADPLL networks and can fill a number of roles depending on the designer's requirements. However the designer should bear in mind the platform's limitations especially when attempting to use a frequency that is not orders of magnitude lower than the maximum clock available from the FPGA's clock manager. Inverter based and FPGA clocked designs both have a niche in which they are the preferred option, with the clocked design better suiting replication of an existing design at scaled down frequencies and the inverter based designs better in the modelling and analysis of ADPLL network behaviour.

REFERENCES