Examensarbete

A Wide Range Low Power Low Jitter All Digital DLL for Video Applications

Examensarbete utfört i Elektronik System
vid Tekniska högskolan i Linköping
av

Yasir Ali Shah,
Muhammad Touqir Pasha

LiTH-ISY-EX--10/4380--SE

Linköping 2010
A Wide Range Low Power Low Jitter All Digital DLL for Video Applications
En heldigital, bredbandig DLL med lågt jitter och låg effektförbrukning för videotillämpningar

A Wide Range Low Power Low Jitter All Digital DLL for Video Applications

Författare  Yasir Ali Shah,Muhammad Touqir Pasha
Author

Sammanfattning  Abstract

Technological advancements in video technology have placed stringent requirements on video analog front ends (AFEs) to deliver high resolutions crisp images while consuming low power to deliver optimal performance.

One of the vital parts of an AFE is a delay locked loop (DLL). The DLL is a first order system that aligns a delayed signal with respect to a reference signal while working in a feedback manner. DLLs find their applications in many electronic devices that deal with clocks in their operation. They are used to improve timing margins and clock delays in microprocessors, memory elements and other such applications. The vital function of a DLL is to delay the input clock (one period delay), by passing it through delay line and aligning the input clock and the delayed clock of the DLL through phase detector. Once this is done multiple phases can be derived from various stages of the delay line with each providing a stable clock signal that is a delayed version of the input clock. Due to the increasing clock speeds this task of deriving multiple phases has become quite cumbersome. The task may become complicated due to noise generated from switching activity in digital circuits thus resulting in jitter at DLL output. As the design of analog circuits becomes quite exigent especially below the 100 nm mark, the goal here is to design an all digital DLL to take advantage of the 65 nm process and a simplified design cycle.

The aim of this thesis is to implement an all digital delay locked loop with an input frequency range of 60 MHz to 300 MHz with a worst case jitter of 66 ps. The DLL provides 32 uniformly spaced phases between input and output clocks.

The DLL operation is divided into two stages. In the first step the first delay line quantizes input clock period with the help of a binary time to digital converter. Based on this quantization information second delay line introduces actual delay between input and output clocks with 32 intermediate phases in between. The entire process takes up to 9 clock cycles until a lock state is achieved. These 32 phases provide a greater phase resolution enhancing the sync processing characteristics of the video AFE thus improving the one screen display characteristics.
Abstract

Technological advancements in video technology have placed stringent requirements on video analog front ends (AFEs) to deliver high resolutions crisp images while consuming low power to deliver optimal performance.

One of the vital parts of an AFE is a delay locked loop (DLL). The DLL is a first order system that aligns a delayed signal with respect to a reference signal while working in a feedback manner. DLLs find their applications in many electronic devices that deal with clocks in their operation. They are used to improve timing margins and clock delays in microprocessors, memory elements and other such applications. The vital function of a DLL is to delay the input clock (one period delay), by passing it through delay line and aligning the input clock and the delayed clock of the DLL through phase detector. Once this is done multiple phases can be derived from various stages of the delay line with each providing a stable clock signal that is a delayed version of the input clock. Due to the increasing clock speeds this task of deriving multiple phases has become quite cumbersome. The task may become complicated due to noise generated from switching activity in digital circuits thus resulting in jitter at DLL output. As the design of analog circuits becomes quite exigent especially below the 100 nm mark, the goal here is to design an all digital DLL to take advantage of the 65 nm process and a simplified design cycle.

The aim of this thesis is to implement an all digital delay locked loop with an input frequency range of 60 MHz to 300 MHz with a worst case jitter of 66 ps. The DLL provides 32 uniformly spaced phases between input and output clocks.

The DLL operation is divided in to two stages. In the first step the first delay line quantizes input clock period with the help of a binary time to digital converter. Based on this quantization information second delay line introduces actual delay between input and output clocks with 32 intermediate phases in between. The entire process takes up to 9 clock cycles until a lock state is achieved. These 32 phases provide a greater phase resolution enhancing the sync processing characteristics of the video AFE thus improving the one screen display characteristics.
Acknowledgements

Praise be to Allah SWT the most gracious, the most merciful, for giving us the strength and conviction to undertake and complete this work.

We would like to show gratitude to our supervisor Dr. J Jacob Wikner for the opportunity to work under his kind supervision as part of the Video IC group. Jacob, thank you very much for providing us with much needed advice during the course of this thesis, being patient, answering all the silly questions and for the prompt replies to all the emails.

We would also like to thank the Swedish educational system, not only for the great educational experience and learning but also for allowing us to experience the Swedish lifestyle.

We would take this opportunity to thank fellow students at the ES division:

- Syed Ahmed Aamir, Syed Asad Alam for their support and motivation, especially during the documentation phase of this thesis.
- Khan ji, Abdul Raheem for providing great company and the much needed breaks during this work.
- Khalid, Abdul Mateen and Sohail for being great friends and making our stay in Linköping memorable.

Lastly we would like to thank our parents for unconditional love and support. Without your encouragement this work would not have been possible. Thanks for being there when we needed you.
Contents

Chapter 1
1.1 Background to Thesis ............................................................................................................. 1
1.2 Work Environment .................................................................................................................. 1
1.3 Chip Setup .............................................................................................................................. 1
  1.3.1 Digitizing Channel ........................................................................................................ 2
  1.3.2 Time Reference Channel .............................................................................................. 6
1.4 Project Execution ..................................................................................................................... 8
1.5 Thesis Outline .......................................................................................................................... 8

Chapter 2
2.1 DLLs Introduction ................................................................................................................... 11
2.2 DLL Principle ........................................................................................................................ 12
2.3 DLL Loop Dynamics .............................................................................................................. 12
2.4 DLL Jitter Analysis ................................................................................................................. 14
2.5 DLL Architectures .................................................................................................................. 16
  2.5.1 Analog DLLs ................................................................................................................ 16
  2.5.2 Digital DLLs ................................................................................................................. 21
2.6 Motivation for using Digital DLLs ........................................................................................ 26

Chapter 3
3.1 Video Basics .......................................................................................................................... 29
3.2 Sync Signals ........................................................................................................................... 29
3.3 Video Resolution .................................................................................................................... 31
3.4 Color Spaces ......................................................................................................................... 31
  3.4.1 RGB ............................................................................................................................ 31
  3.4.2 CMYK .......................................................................................................................... 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.3</td>
<td>YIQ</td>
<td>32</td>
</tr>
<tr>
<td>3.4.4</td>
<td>YPbPr</td>
<td>32</td>
</tr>
<tr>
<td>3.5</td>
<td>The Composite Video Signal</td>
<td>32</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Synchronization Signals</td>
<td>32</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Front Porch</td>
<td>32</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Sync Tip</td>
<td>32</td>
</tr>
<tr>
<td>3.5.4</td>
<td>Color Burst</td>
<td>32</td>
</tr>
<tr>
<td>3.5.5</td>
<td>Back Porch</td>
<td>33</td>
</tr>
<tr>
<td>3.5.6</td>
<td>Blanking Interval</td>
<td>33</td>
</tr>
<tr>
<td>3.6</td>
<td>Motivation of Video AFEs</td>
<td>33</td>
</tr>
<tr>
<td>3.7</td>
<td>Requirements on Video AFE</td>
<td>35</td>
</tr>
<tr>
<td>3.8</td>
<td>Why 32 Phases</td>
<td>37</td>
</tr>
</tbody>
</table>

**Chapter 4**

4.1 Working Principle | 38 |
4.2 Issues with Paper | 39 |

**Chapter 5**

5.1 Timing Control Unit (TCU) | 43 |
5.2 First Delay Line | 43 |
5.3 Second Delay Line | 45 |
5.4 Binary Time to Digital Conversion | 47 |
5.4.1 Control signal Generation | 48 |
5.4.2 Bits Generation | 50 |
5.5 Structure Functionality | 51 |
5.6 Counter | 56 |
5.7 Lock Detector | 56 |
5.8 Dynamic Frequency Detector (DFD) | 57 |
Chapter 6

6.1 Clock Pulse Generation ........................................................................................................................................ 59
6.2 Bits and Output Clock Generation ....................................................................................................................... 60
6.3 Output Clock alignment and Lock Signal Generation .......................................................................................... 61
6.4 Inter Phase Delay .............................................................................................................................................. 63
6.5 Jitter Analysis .................................................................................................................................................. 63

Conclusions and Future Work.................................................................................................................................... 65
Appendix A............................................................................................................................................................ 66
References .............................................................................................................................................................. 68
List of Figures

Figure 1.1 Video IC designed at the Division of Electronic System, Linköping University. .......................... 2
Figure 1.2 Effects of input multiplexer bandwidth on video picture .......................................................... 3
Figure 1.3 Effects of clamp non linear distortion on video picture ............................................................. 4
Figure 1.4 Effects of analog to digital converter offset errors on video picture ........................................ 6
Figure 1.5 Effects of synchronization errors on video picture .................................................................... 7
Figure 2.1 First type of DLL structure[4] .................................................................................................... 13
Figure 2.2 Second type of DLL structure[4] ................................................................................................. 13
Figure 2.3 Effects of jitter on video picture ................................................................................................. 15
Figure 2.4 XOR based phase detector [4] .................................................................................................. 17
Figure 2.5 Phase detector based on D flip flop ......................................................................................... 17
Figure 2.6 Dynamic phase detector [4] ....................................................................................................... 18
Figure 2.7 Basic charge pump structure [4] ............................................................................................... 18
Figure 2.8 Voltage control delay line structure [4] ............................................................................... 19
Figure 2.9 Single ended voltage controlled delay line cell [4] ................................................................. 20
Figure 2.10 Voltage controlled delay line cells with active loads [4] .................................................... 21
Figure 2.11 D flip flop based phase detector ............................................................................................ 22
Figure 2.12 Dual output phase detector [4] ............................................................................................... 22
Figure 2.13 Inverter based digital controlled delay line[15] .................................................................. 23
Figure 2.14 Nand based variable delay digital controlled delay line [15] ............................................... 24
Figure 2.15 Typical subgate digital controlled delay line [15] ............................................................... 25
Figure 2.16 Thermometric/logarithmic subgate digital controlled delay line [15] .............................. 25
Figure 2.17 Control mechanism in Digital DLLs [15] ............................................................................ 26
Figure 2.18 Finite state machine based control block in Digital DLLs [15] ............................................. 26
Figure 3.1 Interlaced display scanning process[26] .................................................................................. 30
Figure 3.2 Progressive display scanning process [26] ......................................................................... 30
Figure 3.3 General structure of a video signal [26] ............................................................................... 33
Figure 3.4 Video AFE block diagram as illustrated in Synopsys high definition video AFE: far beyond ADC, April, 2010 [22] ............................................................................................................. 34
Figure 3.5 Video AFE digitizing channel block diagram as illustrated in Synopsys high definition video AFE: far beyond ADC, April, 2010 [22] .................................................................................................... 35
Figure 3.6 Coasting caused due to inaccurate sync processing as illustrated in Synopsys high definition video AFE: far beyond ADC, April, 2010 [22] .................................................................................. 36
Figure 3.7 Effects on video picture due to poor jitter performance .......................................................... 36
Figure 4.1 Reference DLL block diagram [23] ....................................................................................... 38
Figure 4.2 Issues with phase extraction in reference architecture [23]. .................................................................40
Figure 4.3 Modified DLL architecture ....................................................................................................................41
Figure 5.1 Timing control unit structure for clock pulse generation [23].................................................................43
Figure 5.2 Timing diagram for first delay line [23]. ...............................................................................................44
Figure 5.3 First delay line schematic [23]. ............................................................................................................45
Figure 5.4 Second delay line schematic ................................................................................................................45
Figure 5.5 Second delay line single delay block structure [24]. ............................................................................46
Figure 5.6 Linear behavior of delay vs. digital code for second delay line .........................................................47
Figure 5.7 Binary Time to digital converter block diagram [23]. .................................................................47
Figure 5.8 A 5 bit time to digital converter [23] ......................................................................................................48
Figure 5.9 Timing diagram for clock pulse quantization in binary time to digital converter .........................49
Figure 5.10 Control signals (M’s) generation in time to digital converter [23] .....................................................50
Figure 5.11 Encoder circuitry for bits generation [23] .......................................................................................55
Figure 5.12 Counter schematic ..............................................................................................................................56
Figure 5.13 Lock detector schematic [23] ............................................................................................................57
Figure 5.14 Dynamic frequency detector schematic ..........................................................................................58
Figure 6.1 Clock Pulse generated for 300 MHz input ...........................................................................................59
Figure 6.2 Bits generated for 300 MHz input .........................................................................................................60
Figure 6.3 Initial output clock for 300 MHz input ...............................................................................................61
Figure 6.4 Output clock for 300 MHz input after fine tuning ...............................................................................62
Figure 6.5 Linear behavior of DLL delay line .......................................................................................................63
Figure 6.6 Jitter analysis at 300 MHz ..................................................................................................................64
List of Tables

Table 1.1 Multiplexer specifications .................................................................................................................. 3
Table 1.2 Programmable gain amplifier specifications .......................................................................................... 5
Table 1.3 Analog to digital converter specifications .............................................................................................. 5
Table 1.4 Phase locked loop specifications ......................................................................................................... 6
Table 1.5 Delay locked loop specifications ......................................................................................................... 7
Table 3.1 Summary of various video formats ....................................................................................................... 31
Table 5.1 Bit values and corresponding delay as generated through 2nd delay line .............................................. 46
Table 5.2 Delay between intermediate clock signals w.r.t. input clocks when all bits are zero ............................ 49
Table 5.3 Values of M signals of encoder for 80 MHz input frequency ................................................................. 50
Table 5.4 Values of bits generated for 80 MHz input frequency ............................................................................. 51
Table 5.5 Clock signal and bit values relation to internal delay of first delay line ............................................... 52
Table 5.6 Internal control signal & bit values for various frequency ranges ......................................................... 53
## List of Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFE</td>
<td>Analog front end</td>
</tr>
<tr>
<td>ADC</td>
<td>Analog to digital converter</td>
</tr>
<tr>
<td>BTDC</td>
<td>Binary time to digital conversion</td>
</tr>
<tr>
<td>CP</td>
<td>Charge pump</td>
</tr>
<tr>
<td>CRT</td>
<td>Cathode ray tube</td>
</tr>
<tr>
<td>DLL</td>
<td>Delay locked loop</td>
</tr>
<tr>
<td>DFF</td>
<td>D flip flop</td>
</tr>
<tr>
<td>DFD</td>
<td>Dynamic frequency detector</td>
</tr>
<tr>
<td>DCDL</td>
<td>Digital control delay line</td>
</tr>
<tr>
<td>FSM</td>
<td>Finite state machine</td>
</tr>
<tr>
<td>Hsync</td>
<td>Horizontal synchronization</td>
</tr>
<tr>
<td>IP</td>
<td>Intellectual property</td>
</tr>
<tr>
<td>LD</td>
<td>Lock detector</td>
</tr>
<tr>
<td>LTJ</td>
<td>Long term jitter</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase locked loop</td>
</tr>
<tr>
<td>PD</td>
<td>Phase detector</td>
</tr>
<tr>
<td>PGA</td>
<td>Programmable gain amplifier</td>
</tr>
<tr>
<td>PVT</td>
<td>Process, voltage and temperature</td>
</tr>
<tr>
<td>SAR</td>
<td>Successive approximation register</td>
</tr>
<tr>
<td>TCU</td>
<td>Timing control unit</td>
</tr>
<tr>
<td>Acronym</td>
<td>Term</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>TDC</td>
<td>Time to digital conversion</td>
</tr>
<tr>
<td>Vsync</td>
<td>Vertical synchronization</td>
</tr>
<tr>
<td>VCDL</td>
<td>Voltage control delay line</td>
</tr>
<tr>
<td>VCO</td>
<td>Voltage controlled oscillator</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background to Thesis

This thesis is a part of the “video project” being carried out at the Division of Electronic systems (ES), Department of Electrical Engineering, Linköping University Sweden, where over recent period the aim has been to couple the master’s thesis projects with the ongoing research undertaken by the faculty members. The aim of this project is to implement a digitizing video device for handheld devices like laptop computers and micro projectors. The aim is to achieve a low power device that satisfies all the stringent timing and performance metrics. A single platform is available for work by both students and faculty members each contributing to the entire project. The video project involves a variety of components that need to be implemented and most of them are discussed in the next section. The completion of a single component will not result in a taped out chip rather the aim here is to get the work done by this batch of students refined by the next batch until the work is mature enough for taping out.

1.2 Work Environment

Within the project we used tools from vendor like Cadence and MATLAB. All the students met for two hours at least once a week starting with a 15 minute “around the table” based on SCRUM model with primarily two questions

- current progress
- plans for next week

This is followed by a 45min lecture by the supervisor about various design practices and discussions regarding various issues related to implementation. During the second hour one of the AFE building blocks was focused upon usually by the students working on the block.

1.3 Chip Setup

The chip setup of the video AFE is shown in Figure 1.1. All the components shown here are being implemented as part of the master thesis by various students under the supervision of Dr. J. Jacob Wikner at the ES Division. In this section, we briefly discuss each of these components and their required specification along with their affects on the picture in case the design requirements are not fulfilled.
1.3.1 Digitizing Channel

The video digitizing channel is composed of a multiplexer for selecting various interfaces followed by a clamp circuit to restore the signal’s DC point. This analog signal is then passed through a low pass filter to cater of anti aliasing since filter also band limits the input signal. The signal is then fed to a programmable gain amplifier (PGA) for amplifying input analog signals or scaling the signal’s amplitude and finally a digitizing analog to digital converter (ADC) for converting the stream for digital post processing along with some assisting components like bias generator that is used for generating bias voltages for ADC and PGA. The above mentioned components are likely to be found in any video AFE that is being employed in modern day TVs and display units.

In the following sections we briefly discuss each of the above mentioned components.

Figure 1.1 Video IC designed at the Division of Electronic System, Linköping University.
1.3.1.1 Input Multiplexer, Clamping and DC Restoration

The input multiplexer selects between the various AFE inputs for example, computer, DVD player, set top box etc. After selection in order to make digital conversion possible this component restores the DC component of the analog video signal. This is necessary because the variation of the video content varies the AC coupled signal bias level thus eliminating the brightness information. This is done during the “back porch” or “sync-tip” area of the video signal. The horizontal timing needs to be set prior to clamping so that it can be performed during the specified interval mentioned above.

In this chip a single implementation tackles both the input multiplexer and clamping circuits having the specs given in Table 1.1.

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of inputs</td>
<td>8</td>
</tr>
<tr>
<td>Number of outputs</td>
<td>3</td>
</tr>
<tr>
<td>Input voltage swing</td>
<td>200-2000 mVpp</td>
</tr>
<tr>
<td>DC settling time</td>
<td>1 Frame</td>
</tr>
<tr>
<td>Clamp time</td>
<td>6 Pixels</td>
</tr>
<tr>
<td>Isolation</td>
<td>70 dB</td>
</tr>
</tbody>
</table>

*Table 1.1 Multiplexer specifications*

It is imperative that this block performs the required task accurately as any design lapse would directly be visible on the picture being displayed on screen.

*Figure 1.2 Effects of input multiplexer bandwidth on video picture.*

Figure 1.2a shows a generally used Philips PM5544 pattern for video. Figure 1.2b shows the effects of bandwidth limitation on the test video picture; note that due to bandwidth limitation
the vertical boundaries between various colors gets blurred. If the clamp circuit has linearity issues, the picture would be distorted as the color and brightness information is lost, and the resulting picture would like the one as shown in Figure 1.3b. These linearity issues affect the brightness of all the colors in the test picture. If observed carefully it can be seen that the overall image brightness changes due to this non linear distortion.

![Standard frame for video display testing](image)

![Effect of Distortion](image)

**Figure 1.3** Effects of clamp non linear distortion on video picture.

1.3.1.2 Anti Aliasing Filter

In order to band limit the ADC input signal and reduce noise bandwidth, it is preferred to use an analog filter. Although some ADCs might use oversampling along with a digital filter to prevent folding in baseband but in many designs an analog filter is still employed.

1.3.1.2.1 Filter Type

While designing such an analog filter, a suitable filter type must be selected that satisfies the requirements placed on the analog filter. These requirements include group delay, pass band amplitude, and pass band edge. Among the design parameters group delay is an important parameter, as it should prevent overshooting due to abrupt changes in video data. If this constraint is not tightly met, the image sharpness is compromised. Possible alternatives maybe Chebychev, Elliptic, Bessel or Butterworth type filters. The first two filter types although might have good attenuation properties, may not cater to the ringing issue due to their non consistent group delay. The Bessel type filters exhibit good group delay characteristics but might have stability issues. Butterworth filter on the other hand reasonably satisfies most of the above mentioned requirements placed on the analog filter as discussed in [1]. Although a Butterworth filter may satisfy the requirements but the resulting filter order might be
unreasonably high thus complicating filter design. A workaround this problem is to divide the filter in active and passive parts, with active part design being incorporated with the PGA.

### 1.3.1.3 Programmable Gain Amplifier

The programmable gain amplifier (PGA) is required to scale the analog signal’s amplitude to the required levels of the ADC. It provides single to differential ended transformation which helps in noise/ harmonics suppression and increased output swing. In general, a low voltage fully differential architecture is used with a common mode feedback circuit to achieve desired results.

The PGA architecture selected for implementation is fully differential transconductance amplifier (OTA) with a common mode feed forward (CMFF) and inherent common mode feedback (CMFB) detector. The specifications for PGA are shown in Table 1.2.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMRR</td>
<td>50 dB</td>
</tr>
<tr>
<td>3-dB bandwidth</td>
<td>500 MHz</td>
</tr>
<tr>
<td>Gain settings</td>
<td>0.5,1,2</td>
</tr>
<tr>
<td>Linearity at 30 MHz</td>
<td>60 dB</td>
</tr>
<tr>
<td>Supply tolerance</td>
<td>+/- 10%</td>
</tr>
<tr>
<td>Power consumption</td>
<td>50 mW</td>
</tr>
</tbody>
</table>

*Table 1.2 Programmable gain amplifier specifications*

### 1.3.1.4 Analog to Digital Conversion

The differential output of the PGA is fed to a high resolution, high data rate time-interleaved ADC. The required ADC can be implemented to have any of say sigma delta, pipelined or successive approximation register (SAR) architecture, but two former architectures have issues like low bandwidth and power consumption respectively.

For this project the ADC is based on SAR architecture which has been selected mainly due to the process environment and due to the fact that the aim of this research work is to implement as many “digital” components as possible. The specifications for ADC are shown in Table 1.3.

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of input bits</td>
<td>12 bits</td>
</tr>
<tr>
<td>Sampling frequency</td>
<td>300 MHz</td>
</tr>
<tr>
<td>Input bandwidth</td>
<td>500 MHz</td>
</tr>
<tr>
<td>SFDR</td>
<td>60 dBC</td>
</tr>
<tr>
<td>Offset error</td>
<td>1%</td>
</tr>
<tr>
<td>Gain error</td>
<td>1%</td>
</tr>
</tbody>
</table>

*Table 1.3 Analog to digital converter specifications*
In the case that the ADC has issues like offset errors the picture quality deteriorates and then on screen picture would be as shown in Figure 1.4b. Notice due to offset errors in the ADC the test image intensity varies at fixed intervals as can be seen by the stripes that appear in the test image.

![Standard frame for video display testing](image1) ![Raster due to Offset errors (16 ADC slices)](image2)

(a) Normal picture  (b) Picture after offset errors

*Figure 1.4 Effects of analog to digital converter offset errors on video picture*

1.3.2 Time Reference channel

The time reference channel, as shown in bottom of Figure 1.1, comprises of slicer, multiplexer, phase locked loop (PLL) and delay locked loop (DLL). Each of these individual components and their desired specifications for this video chip are described below.

1.3.2.1 Phase Locked Loop

The signal will go to a phase-locked loop (PLL) which will align its output clock (at a higher frequency) to the input reference. For this project a time to digital converter (TDC) based all digital PLL architecture has been selected with a multiplication factor up to 2600 thus generating output frequency in the range of 10-300 MHz. Table 1.4 below summarizes some of the PLL specifications.

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input frequency</td>
<td>10-150 KHz</td>
</tr>
<tr>
<td>Output frequency</td>
<td>10-300 MHz</td>
</tr>
<tr>
<td>Jitter value</td>
<td>2%</td>
</tr>
<tr>
<td>Latency</td>
<td>2 Clock Cycles</td>
</tr>
</tbody>
</table>

*Table 1.4 Phase locked loop specifications*
Figure 1.5 Effects of synchronization errors on video picture

1.3.2.2 Slicer

A slicer is employed to determine the timing information from an analog signal and thus may be used to provide a reference exact location of the sync signal.

1.3.2.3 Delay Locked Loop

In order to provide the exact clock edge to the ADC for sampling purposes an all digital DLL is employed. The input to the DLL is a 10-300 MHz signal provided by the PLL and the DLL in turn produces an output clock of the same frequency as the input but with 32 phases of the input equally distributed in between. Table 1.5 below summarizes some of the DLL specifications.

In case the DLL has issues like high cycle to cycle jitter value this would result in varying pixel sampling time due to the skew introduced in sync-tip. On the test video picture shown in Figure 1.5b this could be seen as variation at the vertical starting location of each new line.

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input frequency</td>
<td>10-300 MHz</td>
</tr>
<tr>
<td>Number of phases</td>
<td>32</td>
</tr>
<tr>
<td>Supply voltage</td>
<td>0.9 - 1.1 V</td>
</tr>
<tr>
<td>Power consumption</td>
<td>&lt;10 mW</td>
</tr>
<tr>
<td>Jitter value</td>
<td>2%</td>
</tr>
<tr>
<td>Latency</td>
<td>2 Clock Cycles</td>
</tr>
</tbody>
</table>

Table 1.5 Delay locked loop specifications.
1.4 Project Execution

For an efficient project execution the task was divided in to several phases and design goals were set so that each design requirement could be met and the progress could be checked at every stage.

- **Literature study**
  It was suggested that we start with a literature search on the internet/library to get familiar with the concept: *digitizers for video applications*, with emphasis on most commonly employed delay-locked loop (DLL) architectures.

- **Model description**
  The next phase of the project was to start implementing the DLL in matlab and Verilog or Verilog AMS high level description languages. With the implementation of the model taking in to account, all required is to demonstrate that the DLL can be implemented in silicon.

- **Schematic-level description**
  The third phase of the project is the implementation of the DLL in schematic level representations.

For the project execution the following design rules were set:

- Design and verify the DLL blocks while checking that requirements are met for each block.

- The design of each block should be as digital as possible to take advantage from technology scaling.

- Ensure that all the performance parameters are met as well as block functionality is checked in all four corners.

1.5 Thesis Outline

This report is organized in to the following sections

- Chapter 2 discusses the motivation for using DLLs their working principle, performance metrics and DLL types i.e., analog and digital DLLs.

- Chapter 3 briefly discusses video signal basics, the motivation for using AFEs.

- Chapter 4 discusses the behavioral design and working of the selected architecture and the problems encountered during the implementation.
• Chapter 5 discusses each component block in detail, its working and implementation.

• Chapter 6 lists the simulation results.

• Conclusions and future work.

• Appendix A

• References
Chapter 2

Delay Locked Loops

Both phase locked loops (PLLs) and delay locked loops (DLLs) are based on the same underlying principle that they try to align the input clock and its delayed version (one period delayed from the input signal) as described in [2]. For this purpose the PLL employs a voltage control oscillator (VCO) block to generate the output by integrating frequencies and comparing the phases of input and output clocks using a phase detector (PD). The PLLs, due to this integrating nature, are used for frequency multiplication applications. The output of the phase detector drives the loop filter which controls the voltage control oscillator thus controlling the alignment. Due to this integrating nature of VCO and filter structures, PLLs are generally higher order systems compared to DLLs. Therefore PLLs have several disadvantages mainly location of zeros inserted to cater for the two poles might be affected due to PVT variations and thus affecting the loop stability.

The DLLs, on the other hand, align the input and delayed clocks. This is done by delaying the input clock after passing it through a delay line and controlling the delay using some mechanism. Once the input clock is delayed a phase detector (PD) compares the phases of the two inputs. Based on PD output value, the delay is adjusted (increased or decreased) until the two phases are aligned.

An important consideration while selecting between PLLs and DLLs is the behavior of both circuits in digital IC environments. As mentioned in [3], PLLs are more sensitive to power supply and substrate noises as opposed to DLLs.

PLLs are prone to noise or signal level variations for example; any change in the supply or substrate voltage of VCO could result in change in operating frequency which eventually results in increasing phase error which accumulates until feedback loop is engaged to cater for it.

DLLs on the other hand are not as adversely affected as PLLs due to power supply and substrate noises as a voltage variation on delay line only changes the delay of the delay line. The phase error does not accumulate due to non circulation of output clock in this case and eventually decreases until the locked state is achieved.

2.1 DLLs—an Introduction [4]

Delay locked loops find their applications in many electronic devices that deal with clocks in their operation. They are used to improve timing margins and clock delays in microprocessors, memory elements and other such applications. The vital function of a DLL is to
align the input clock and delayed clock through phase detector (PD). Once this is done multiple phases can be derived from various stages of the delay line with each providing a stable clock signal that is a phase shifted version of the input clock. Due to the increasing clock speeds this task of deriving multiple phases has become quite cumbersome as explained in [5]. The task may become complicated due to noise generated from switching activity in digital circuits thus resulting in jitter at DLL output. The main aim of this thesis is to design a wide range, fast locking, low jitter DLL that can provide a phase resolution of 32 phases between the input and output clocks.

2.2 DLL Principle [4]

Although DLLs are classified as nonlinear negative feedback systems, a study of their linear model provides a sound estimate about various characteristics of DLLs although the results obtained may not be very accurate.

Figure 2.1 shows a typical DLL block diagram: the input signal in a DLL propagates through the delay line being time delayed or phase shifted by each stage in the delay line. A phase detector is then employed to compare the input and delayed clock phases. In case of analog DLLs the phase detector generates a voltage (or current) at its output that is fed through to the charge pump which uses it to set the delay of the delay line using loop filter. In case digital DLLs the phase detectors generates an up/down signal that is utilized by the counter in order to set the delay of the delay line. Due to this feedback mechanism the phase difference between the input and output clocks is eventually reduced until it becomes zero indicating the fact that the delay produced by the delay line is equal to the period of the input clock and a lock signal is generated to indicate this state.

2.3 DLL Loop Dynamics [4]

The loop dynamics of DLLs are of interest in many applications and extensive studies about these characteristics can be found in e.g. [6, 7]. The DLL architectures can be divided into two types.

- The first type of DLL compares the input clock, Clk_ref, with a delayed version of itself by using a phase detector. This type of DLLs finds their usage in frequency synthesis and synchronization applications.
In the second type of DLLs the input clock Clk_ref is compared with a delayed version of an uncorrelated clock signal as described in [7]. This type of DLLs finds their usage in clock recovery circuits. The Figure 2.2 shows this type of architecture.

As we require the DLL for synchronization purposes we will emphasize on first type of DLLs.
2.4 DLL Jitter Analysis [4]

One of the vital performance metrics judging the DLL performance is jitter. Jitter is the random variation in the period of the output clock. Let us consider an example where the time period of the mth clock $T_m$ is

$$T_m = t_{m+1} - t_m$$

where $t_m$ refers to the mth zero crossing of the clock.

Ideally the clock period should be $T$ and the difference $\Delta T = T_m - T$ indicates the jitter [8]. Three different kinds of jitter have been defined in [8].

- **Absolute jitter** or long term jitter (LTJ) is the accumulated jitter over N clock cycles and is given by

$$\Delta T_{abs} = \sum_{n=1}^{N} \Delta T$$

- **Cycle to cycle jitter** is the average effect of long term fluctuations

$$\Delta T_C = \lim_{N \to \infty} \sqrt{\frac{1}{N} \sum_{n=1}^{N} \Delta T^2}$$

- **Cycle to cycle jitter** deals with the variation in rms value of two successive clock cycles and is given as

$$\Delta T_C = \lim_{N \to \infty} \sqrt{\frac{1}{N} \sum_{n=1}^{N} (T_n - T_{n-1})}$$

where $T_n$ and $T_{n-1}$ is the time period of the nth and n-1th clock cycles respectively.

All of these jitter types are used to measure the performance of timing components. Usually jitter is both random and deterministic in nature. Random jitter is caused due to the inherent noise sources in circuits like thermal noise, substrate noise etc. The deterministic jitter is generally caused by device mismatches and other related effects.

The main sources of noise contribution in a DLL circuit are input clock noise and noise originating from the delay line stages. The noise of the delay line refers to the noise of the transistors used to realize the delay line. Since the jitter is very critical to the performance of the DLL, any such factor would directly show its result on the screen as shown in Figure 2.3. This would result in varying pixel sampling time due to the skew introduced in sync-tip and would introduce a raster in the video picture.
The noise sources include both thermal and flicker noise and noise caused by gate leakage current as in [9]. For long channel MOSFET devices, while operating in saturation region, a noise source connected between gate and source can be used to simulate the effects of thermal noise. Let us consider the source has a spectral density from [10]

\[ \overline{i_{n,t}^2} = 4kT\gamma g_m \]

In the above equation k is the Boltzmann constant and \( g_m \) being the device transconductance and \( \gamma \) is the process dependent parameter.

In mixed signal environments the noise contribution by power supply and substrate noise becomes significant. The power supply noise is characterized in [9] as

- Static power supply noise
- Dynamic power supply noise

Special techniques like supply filtering, etc., are proposed in [9] to cater for this kind of noise. Substrate noise refers to the changes in the substrate voltage due to switching activity of on chip circuits. These variations do not affect the pmos devices as much as nmos devices since they share the same substrate in an n-well process. These variations alter the threshold voltage thus changing the operating characteristics of nmos devices.
This new threshold voltage is given by

\[ V_{th,N} = V_{\text{th}0} + \gamma \left( \sqrt{2\phi_p} + V_{SB} \right) - \sqrt{2\phi_p} \]

where \( \gamma \) is the coefficient of body effect and \( \phi_p \) is process dependent. A general rule to reduce the substrate noise is to apply the supply voltage to the pmos devices as described in [6].

2.5 DLL Architectures [4]

Generally a phase detector, delay line and a charge pump or counter make up a DLL. The DLLs can be classified as

- Analog DLLs
- Digital DLLs

Usually, a digital DLL exhibits low supply voltage requirements, superior process migration, simpler design cycle and short lock time but at the same time they exhibit poor jitter performance as compared to their analog counter parts.

Analog DLLs usually exhibit good noise immunity, better jitter and skew suppression characteristics but on the other hand are process sensitive, require larger areas and are quite complex to design as compared to their digital counterparts.

Here we discuss a few analog and digital DLL architectures that are commonly employed for different applications.

2.5.1 Analog DLLs

As mentioned above an analog DLL is composed of the following basic components

- Phase detector (PD)
- Charge pump (CP)
- Voltage controlled delay line (VCDL)

This section first discusses the architecture and performance of each of these components and then some common analog DLL architectures.

2.5.1.1 Phase Detector (PD)

The phase detector is a vital component of a DLL. The purpose of the PD is to detect the difference between its two inputs. One of the inputs to a phase detector is the input clock and the other is a delayed version of this input clock from the delay line or a delayed version of an uncorrelated clock depending on the architecture of the DLL. The simplest of the phase detectors is an XOR gate shown in Figure 2.4. Once the difference is detected this information is
passed on to the charge pump for further action where the magnitude of PD output is proportional to the phase difference between inputs. This kind of PD, however, suffers from several limitations:

- Any variation in the input duty cycle may cause this PD to generate wrong phase information as explained in [11].
- Since this PD has a single output it complicates the design of subsequent blocks namely charge pump.

![Figure 2.4 XOR based phase detector [4].](image)

A much superior PD is based on flip flops as it can detect edges irrelative of the input signal’s duty cycle. This type of PD can be used for frequency detection along with phase but the setup time may become a limiting factor in achieving good performance results. This type of PD has been proposed in [12] and is shown in Figure 2.5.

![Figure 2.5 Phase detector based on D flip flop.](image)

A much more widely used PD found in high speed applications is proposed in [13] and is referred to as a “dynamic PD”. This type of PD generates an up or a down signal depending on the input values. The “dynamic PD” provides fast locking characteristics but this kind of PD must be thoroughly analyzed as there might be “dead zones” for this PD’s operation. Figure 2.6 below shows the schematic of dynamic PD.
2.5.1.2 Charge Pump (CP)

Once the phase difference is detected by the PD, the difference is sent to the CP which in turn makes changes to the loop filter thus setting the VCDL delay in order to minimize the phase error between the inputs of the PD. Figure 2.7 shows a general structure of a charge pump: it consists of a couple of controlled switches being controlled by the up/down signals from the PD and a couple of current sources where one is a current source and the other is a sink. Depending on the position of the switch, loop filter is either charged or discharged. This process continues until the DLL is locked, after that charge over loop filter remains unchanged. In some cases, as in [6], it is desired to slightly charge and discharge the filter (strictly by equal amounts) thereby reducing jitter.

![Figure 2.6 Dynamic phase detector [4].](image)

![Figure 2.7 Basic charge pump structure [4].](image)
A charge pump based on this working principle has certain shortcomings:

- Due to mismatches between the charging and discharging currents may occur in jitter particularly once the DLL is locked. In [14] the drift caused by this mismatch is approximated as

\[
\phi_{\text{Offset}} = 2\pi \frac{\Delta t_{\text{on}}}{T_{\text{ref}}} \frac{\Delta i}{I_{\text{CP}}}
\]

where \( \Delta t_{\text{on}} \) is the duty cycle of the up/down signal, \( \Delta i \) is the difference between charge and discharge currents and \( T_{\text{ref}} \) is the time period of the input clock. This can be catered by reducing \( \Delta t_{\text{on}} \).

- Secondly, the non ideal behavior of switches may also introduced jitter in the signal.

Typically both single and double ended structures of charge pump are implemented with each having its own pros and cons. More details are discussed in [14, 15].

### 2.5.1.3 Voltage Controlled Delay Line (VCDL)

The most critical component in the performance of a DLL is a VCDL which directly influence DLL jitter performance and stability. The basic purpose of a VCDL is to delay the input signal by one clock period or equivalently 360°. Figure 2.8 shows the basic structure of a VCDL.

Ideally, the delay generated by each stage is equal to \( T_{\text{ref}}/n \) where \( T_{\text{ref}} \) is the time period of the input clock and \( n \) is the number of delay stages in a VCDL. Theoretically there is no limit to the number of delay stages in a VCDL but there are some practical constraints. It is to be noted here that the delay may not necessarily be an inverter. Both single ended and differential structures have been implemented where the differential structures having better common mode noise rejection characteristics.

An example of single ended delay cells based on current starved inverters used in [16] is shown in Figure 2.9. Current changes with Vcontrol, thus changing the driving capability of the delay stage.
Figure 2.9 Single ended voltage controlled delay line cell [4].

Figure 2.10 shows a differential implementation of a delay line cell as discussed in [9]. The resistance of the diode connected structure in Figure 2.10 would be

\[ r_{out} = \frac{1}{g_m} + \frac{1}{r_p} + \frac{1}{r_n} \]

where \( g_m \) is pmos load’s transconductance, while \( r_p \) and \( r_n \) are the output resistances of pmos and nmos transistors. The transconductance can be expressed as

\[ g_m = \sqrt{\mu_p C_{ox} \left(\frac{W}{L}\right)_p I_{ss}} \]

where \( I_{ss} \) is the tail source current. Substituting this value of \( g_m \) to the above equation \( r_{out} \) can be written as

\[ r_{out} = \frac{1}{\sqrt{\mu_p C_{ox} \frac{W}{L_p} I_{ss}}} \]

The delay of each stage is proportional to the RC constant and can be expressed as

\[ t_d = \sqrt{2} r_{out} C_{out} = \sqrt{\frac{2}{\mu_p C_{ox} \frac{W}{L_p} I_{ss}}} C_{out} \]

where \( C_{out} \) is the output capacitance at each node. It can be seen from this expression that the time delay \( t_d \) can be directly controlled by controlling \( I_{ss} \).
2.5.2 Digital DLLs

Like its analog counterparts the digital DLL is composed of the following main component blocks:

- Phase detector (PD)
- Control mechanism (CM)
- Digital controlled delay line (DCDL)

Below we outline each of these blocks.

2.5.2.1 Phase Detector (PD)

For the case of digital DLL the output of the phase detector is a single signal (up/down) that drives the subsequent counter or FSM depending upon the application of DLL. Generally the PD is made up of a DFF whose output is a single digital bit representing leading or lagging between the inputs of the PD. Conventionally, this kind of PDs are called “bang bang phase detectors”. Generally a PD must satisfy the following design requirements:

- To avoid false detection the PD should spend minimum time in metastable state.
- The setup and hold times must be reasonably small.
- The setup and hold times should be comparable so as the resulting phase information is not biased.
- The clock to Q delay should be reasonably small but is usually not a stringent requirement.
The simplest of the phase detectors can be a single D flip flop (DFF) as shown in Figure 2.11. The DFF is clocked by the input reference clock and detects the difference between the clocks by selecting the $\bar{Q}$ output of the DFF. If the $\bar{Q}$ output is low it means that the output clock (clock signal form the end of digitally controlled delay line) is leading the input reference clock and thus the delay of the delay line must be increased so as to compensate for the difference in phase until the lock condition is achieved.

![Figure 2.11 D flip flop based phase detector](image)

Although this design is simple and effective in most cases but it has several shortcomings. If the setup and hold times are not symmetric this may introduce a regular phase error as lagging and leading decisions base on hold and setup times, respectively.

A very efficient design that does not suffer from above flaws has been is discussed in [4]. Figure 2.12 shows this kind of PD. It is composed of two identical blocks generating “UP” or “DOWN” signal. Within each block there are two stages with a pre-charge pmos in each stage. Output of the first stage controls the second stage pre-charging as can be from Figure 2.12.

![Figure 2.12 Dual output phase detector [4]](image)
2.5.2.2 Digital Controlled Delay Line (DCDL)
The delay line is the most important component of the DLL and has a profound effect on the overall DLL performance. Commonly, digital DLLs are composed of two delay lines: a coarse delay line and a fine delay line.

In [4] the coarse delay line is also referred to as gate delay DCDLs while the fine delay line is also sometimes referred to as subgate-delay DCDL. The following section briefly discusses the various aspects of this type of DCDL.

2.5.2.2.1 Gate Delay DCDLs

Gate delay DCDLs are composed of CMOS logic gates and are cascaded to form a delay line. The simplest of the delay lines for the digital case is a chain of cascaded inverters with each stage composed of a pair of inverters and the required output is selected by a multiplexer. The minimum delay of each stage is $2T_D$ and requires $\log_2 N$ storage elements. An accepted coarse delay line discussed in [15] is shown in Figure 2.13.

![Figure 2.13 Inverter based digital controlled delay line][15]

This structure has four delay stages between the input and output while the delay per stage is $2T_D$ where $T_D$ is the delay of a single CMOS inverter.

However, if the number of delay stages is increased, this may load the input clock and separate buffers must be introduced thus increasing the delay of each cell to more than $2T_D$ and this increase in delay would depend on the amount of buffering introduced.

A work-around to this problem is shown in Figure 2.14. In this structure, the loading on the input clock is prevented by varying the delay line in a “telescopic” fashion. Adapting this structure eliminates the need to introduce internal buffering of the clock signal and thus maintains the delay of each stage to $2T_D$. A shift register controls the delay of each cell. When for example $Q[0] = 1$ and $Q[3:1] = 0$ the output clock signal would have a two nand gate delay i.e. the A and B nand gates in cell 0. When for example $Q[1] = 1$ and the rest of the values are low the output path would be nand gate C in cell 0, nand gate A in cell 1, nand gate B in cell 1,
nand gate B in cell 0. The “wrap around” in the delay cell 3 toggles the delay line between various modes.

Here we discuss only a few of the commonly used DCDL structures and it is worth mentioning that these are not the only structures that are employed or have been proposed. It can be safely stated that there is no limit on the number of delay line structures and it is up to the designer to select the type and structure of delay line that is required for a particular application.

![Diagram of nand gate-based delay line](image)

*Figure 2.14 Nand based variable delay digital controlled delay line [15]*

### 2.5.2.2.2 Subgate-Delay DCDL

Subgate-Delay DCDLs generate delay by using the RC delay characteristics. Figure 2.15 shows an example of a fine delay line. This structure relies on varying cell resistance. The branches are controlled by using digital bits Q [2:0] which switch in a fixed specified pattern. The bits can only be thermometrically encoded for this circuit due to poor linearity behavior of the circuit. For increased dynamic range several stages can be cascaded. The Greater the number of devices that are ON at a particular time smaller will be the RC delay.
A second structure that can be used as a fine delay line is shown in Figure 2.16. For this structure the RC delay will increase with the number of ON devices. The digital control bits can be encoded using both thermometric and logarithmic codes due to linear characteristics of the circuit. As in former structure the dynamic range increases by cascading several stages. This type of delay lines are discussed with more detail in [15].

![Diagram](image1.png)

Figure 2.15 Typical subgate digital controlled delay line [15]

![Diagram](image2.png)

Figure 2.16 Thermometric/logarithmic subgate digital controlled delay line [15]

2.5.2.3 Control [15]

The controlling mechanism varies based on the application of DLL. As discussed earlier there can be two kinds of DLLs. First one in which the same reference signal is fed to both the PD inputs and the second in which one of the PD inputs is a reference clock and the other being uncorrelated clock of the same frequency. Due to mismatches in actual circuit implementations, the clock frequencies may differ and in cases where data recovery is to be performed this may intentionally be the case. However, for our case the former structure is of interest.
Figure 2.17 Control mechanism in Digital DLLs [15]

Figure 2.17 shows a straightforward control mechanism. In this arrangement the PD controls an up/down counter that is used to set the delay of the DCDL so as to compensate for the phase difference between the PD inputs.

When the output of the PD is high, the counter is incremented, increasing the delay of DCDL and when PD output is low, the counter is decremented, to decrease the delay of DCDL, thus attaining the lock condition.

Figure 2.18 shows an advanced control mechanism in which a finite state machine (FSM) is placed between the PD and the up/down counter. In this case FSM sends a high or low signal to the up/down not only based on the PD output but also on the internal state of the FSM, which is set based on the initial conditions of the system. The design of such an FSM is very complex and may directly affect the dynamic range and stability of DLL.

Figure 2.18 Finite state machine based control block in Digital DLLs [15]

2.6 Motivation for using Digital DLLs [20]

Scaling of integrated circuits results in high performance and low system cost. This scaling is driven by the fact that most of the components that make up a system are digital in nature. This directly affects the system performance by improving circuit switching speeds and
area of chip. Whereas for analog circuits, actual transistor sizes are of considerable importance and changing them may alter the circuit’s functionality. Scaling directly affects output resistance and intrinsic gain of the circuit. As we cross below the 100nm mark, the design of these analog circuits becomes quite exigent, especially for low supply voltages around 1 V. This results in higher power consumption and enhanced design complexity. Clearly both these affects are conflicting with the aims of scaling. None the less, there are functions that only analog circuits can perform so as to achieve respectable results.

While the analog circuits may be adversely affected by scaling, digital circuits become increasingly efficient by operating at higher speeds, consuming less power and occupying smaller area. For example, due to fast switching, the affects of flicker noise is considerably reduced. Although scaling has many advantages but there are a few disadvantages as well, like degraded power supply rejection ratio (PSRR) and susceptibility to supply noise.

Even though there might be some disadvantages but gains outweigh these cons and thus motivate the digital implementation of circuits. If the functionality of an analog circuit can be improved or performed by digital components, then it is worth the effort because it is only a one time design effort due to better process migration characteristics of digital circuits. Though this may result in a larger design that consumes higher power, but the effort will pay off due to further scaling.
Chapter 3

Video Basics & Video AFEs

3.1 Video Basics [21]

In the beginning video signals was composed of only black and white colors. When color broadcasts were introduced, analog red, green and blue colors also known as RGB were employed for the purpose. Although successful in revolutionizing the video transmission, the RGB color transmission required approximately 3 times the bandwidth required for black and white transmission. Thus other methods like Y, R-Y and G-Y methods of color representations were employed and instead of transmitting three independent signals, a single signal for the transmission of this Y, R-Y and G-Y was used, utilizing the same bandwidth as required for the transmission of black and white signal. Formats like PAL, SECAM and NTSC are based on this merged signal. Although there are many other representations of signals but all of them are somehow related to the original RGB technique. A variation of Y, R-Y and G-Y that has recently gained popularity is YPbPr and is primarily used to interface various devices due to high definition video transfer characteristics.

3.2 Sync Signals [21]

Although a video signal appears as a continuous signal but in reality it is a sequence of still images changing at a specified rate fast enough for the human eye to detect. This image changes 60-70 times a second for TVs and consumer displays and 70-90 times for computer displays. A signal called “vsync” represents the starting of a new image. Every one of these still images is composed of “scan lines” occurring sequentially after each other. Another signal called “hsync” is generated at the end of each line representing the start of next line. The two synchronization signals are communicated in one of the following ways

- Separate hsync and vsync signal transmission.
- A composite sync signal separate from video signal.
- A composite sync signal implanted in the video signal.

As the images are painted on screen after one another and while the actual image is being painted, different techniques have been employed for this actual image painting

- Interlaced Displays
- Progressive Displays
Interlaced displays were primarily employed to reduce the quantity of information sent per image. Interlaced displays painted the image in two steps, firstly transferring the odd number lines of the image and then the even numbered lines. Figure 3.1 shows interlaced display scanning.

![Figure 3.1 Interlaced display scanning process](image)

Progressive Displays are commonly employed in CRT screens. They paint the image on the screen by starting at the top left corner of the screen, scanning to the right corner of the same line. This process is repeated until the bottom right corner of the screen is reached. Figure 3.2 shows the progressive display scanning.

![Figure 3.2 Progressive display scanning process](image)

In interlaced scans, each line is refreshed half as often as it is in progressive scans to reduce the line flicker, as the number of line to line changes is restricted, the image is low pass filtered. On the other hand progressive scans have unlimited line to line changing characteristics resulting in a flicker free sharper image.
3.3 Video Resolution

We commonly see displays with resolutions like 1280X800 or 800X600 but these are merely numbers and do not in any way represent information about the video signal. For example, a certain analog video signal can be sampled at 15 MHz to generate 800 samples/line or the same signal when sampled at 30 MHz would yield 1600 samples/line. This represents a change in only the number of samples and not the actual resolution of the video content. Video quality is therefore measured in terms of “lines of resolution” i.e., how many distinctive black and white lines could be seen on the screen. This number is then normalized with the required aspect ratio. Commonly employed aspect ratios are 4:3 or 16:9 depending upon applications.

Table 3.1 summarizes the various aspects of some commonly used video formats.

<table>
<thead>
<tr>
<th>Format Name</th>
<th>Scan Type</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Aspect Ratio</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4:3</td>
</tr>
<tr>
<td>Standard Definition</td>
<td>Interlaced</td>
<td>720X480i</td>
</tr>
<tr>
<td></td>
<td></td>
<td>720X576i</td>
</tr>
<tr>
<td>Enhanced Definition</td>
<td>Progressive</td>
<td>720X480p</td>
</tr>
<tr>
<td></td>
<td></td>
<td>720X576p</td>
</tr>
<tr>
<td>High Definition</td>
<td>Progressive</td>
<td>1280X720p</td>
</tr>
<tr>
<td></td>
<td>Interlaced</td>
<td>1920X1080i</td>
</tr>
</tbody>
</table>

*Table 3.1 Summary of various video formats*

Some commonly used video formats are listed here with decreasing quality as we go down the list

- HDMI
- DVI
- Analog YPbPr
- Analog RGB
- Analog S-Video
- Analog composite

3.4 Color Spaces

In this section we briefly discuss some commonly used color spaces.

3.4.1 RGB

The mathematical representation of colors via the standard RGB (Red, Green and Blue) is termed as color space. These three primary colors, when combined together can produce almost any color. Often it is the case that instead of true RGB colors, “Gamma Corrected” colors referred to R’G’B’ are used in the same manner as true RGB colors.
3.4.2 CMYK

CMYK stands for Cyan, Magenta, Yellow and Key. The CMYK model masks colors on white backgrounds thus reducing the reflected light. This format works by storing color values of cyan, magenta, yellow and black. This model is sometimes also referred to as subtractive because of brightness of white is subtracted from white.

3.4.3 YIQ

YIQ stands for luma, in phase and quadrature components. Luma refers to the brightness of the image and I, Q deal with the chrominance, dealing with the color information. YIQ is utilized in NTSC format, being used in North and Central America. In YIQ, luminance value is stored in two “chrominance values” according to the extent of blue and red colors. It is a shifted version of the YUV employed in PAL, used mostly in Europe.

3.4.4 YPbPr

This color space is a modified version of Y, R-Y and G-Y which was introduced for the transmission of colored video at lower bandwidth than RGB. YPbPr is not directly used but has a numerical equivalent that is most commonly employed and is called YCbCr.

3.5 The Composite Video Signal

In a composite video, the synchronization information is embedded in the analog video signal to ensure synchronization between display device and video signal.

3.5.1 Synchronization Signals

As mentioned earlier there are two sync signals namely, the “Horizontal Sync” also called the “hsync”, which indicates the beginning of a new horizontal line. The other signal is “Vertical Sync” also called “vsync”, which indicates the end of one image or field. Figure 3.3 shows various portions of a general video signal.

3.5.2 Front Porch

Front porch is the part of the active video signal in the blanking interval before the sync tip arrives.

3.5.3 Sync Tip

Sync tip is also sometimes called sync pulse. The duration of this sync tip is of vital importance for the accurate functioning of video AFEs.

3.5.4 Color Burst

Color burst provides color information by providing its phase and amplitude information. Usually has 10 cycles with amplitude of ± 20 IREs of reference frequency. It is used
to synchronize the color information carrier (sometimes referred to as chrominance sub carrier) to the screen.

### 3.5.5 Back Porch

The part of the video signal between color burst and active video signal is called back porch. It is primarily used to restore black reference level in analog video.

![General structure of a video signal](image)

*Figure 3.3 General structure of a video signal [26]*

### 3.5.6 Blanking Interval

The blanking interval is composed of front and back porches and all of the intermediate signals in between. The blanking interval may be of two types:

- **Horizontal Blanking**: Once a line is completed, the time required to move from right most corner of the completed to the left most edge of the next line.
- **Vertical Blanking**: Once an image is completed, the time required to move from right most corner of the last line on the screen to move back to the top right corner of the screen is called vertical blanking interval.

### 3.6 Motivation of Video AFEs [22]

Video AFE’s are a vital part of today’s entertainment systems. Although the current trend in technology is scaling and migration towards all digital components, analog interfaces are still a vital part of many audio/video interfaces due to their superior performance and
efficient power consumption. While digital interfaces like HDMI, Ethernet etc provide digital signals but traditional electronic items like DVD players, video cameras and set top boxes still provide an analog video signal. So in order to keep devices backward compatible, such interfaces are imperative for entertainment system in near future so that both xVGA and analog video formats, along with the recently popular 1080p formats could be supported. Figure 3.4 shows complete video AFE block diagram.

Figure 3.4 Video AFE block diagram as illustrated in Synopsys high definition video AFE: far beyond ADC, April, 2010 [22]
The simplest of the video AFEs employs one or more ADCs, preceded by some conditioning components like clamping circuits, PGA and filters. The video AFEs also incorporate sync processing elements that extract timing and sync information for HDTV and xVGA graphics formats. The Figure 3.5 shows the typical block diagram of a digitizing channel in a video AFE.

The operating characteristics of several components vary with varying formats, like for example; the SDTV format may require 10 bit resolution at a sampling frequency of 27 MS/s or 12 bit resolution with sampling frequency of 54 Ms/s. The HDTV 1080p format requires 10bit resolution at around 148 Ms/s whereas most computer applications only require either an 8 or 10 bit resolution with much higher sampling frequencies reaching up to 205 MS/s. The variations in the ADC requirements are considerably minute as compared to other components in the analog signal chain.

Hence design of each component in an AFE is as important for the performance as any other component. It is possible for system designers to pick up individual IP blocks from several vendors and incorporate them to make up the required AFE but interfacing such IP blocks sometimes becomes quite hectic and the resulting complexity may be of the same level as that faced while designing a new system from scratch. Therefore home entertainment equipment manufacturers tend to buy an entire video AFE rather than individual IP blocks.

![Figure 3.5 Video AFE digitizing channel block diagram as illustrated in Synopsys high definition video AFE: far beyond ADC, April, 2010 [22]](image)

**3.7 Requirements on Video AFE [22]**

Perhaps the weakest point in the entire Video AFE signal is the sync recovery mechanism. Any imperfection in the design of this component directly impacts the picture display on screen. One of the affects that is observed due to malfunctioning of sync recovery circuits is referred to as “Coasting”. Figure 3.6 below shows the effects of a poorly designed implementation as a result of which the picture is bent at the top due to “Poor Coasting” properties of the sync recovery circuit.
Another effect of poor DLL design is the variation in hsync starting point due to jitter and may result in a distorted image, as each line of the image may start from a different point. Therefore, it is imperative that the jitter does not exceed the set minimum limit. Figure 3.7 shows the effect of poor jitter on the video picture.

Figure 3.6 Coasting caused due to inaccurate sync processing as illustrated in Synopsys high definition video AFE: far beyond ADC, April, 2010 [22]

Figure 3.7 Effects on video picture due to poor jitter performance
3.8 Why 32 Phases

The main aim of employing a DLL in the time/ref channel is to provide a reference clocking signals to the ADC for sampling the video signal at the appropriate instant. According to Figure 1.1, three DLLs are employed, which get the input signal after being processed through various components like slicer, multiplexer and PLL, each with its own processing delay. The basic aim of using 32 phases is to increase the likelihood of the clocking signal overlap at each DLL output, so that the exact sampling instant can be provided to the ADC.
Chapter 4

Behavioral Level Design

4.1 Working Principle [23]

Figure 4.1 shows the block diagram of the proposed all digital DLL system. The operation of the proposed system can be divided into three stages. In the first step, a clock pulse CLK_p with half the period of the input clock is generated by the timing control unit (TCU). The width of CLK_p is then quantized using the signals from the 1st delay line which is composed of delay cells that are digitally controlled. The first delay line generates 32 signals which are used for the quantization process. Firstly the bits at the output of the binary time to digital converter (BTDC) (B1-B6) and counters output (C1-C6) are set to zero. The quantization result for the pulse CLK_p is stored in the six bits B1-B6 at the output of BTDC which controls the delay through the first delay line. For the reference design presented in [23] this step requires a maximum of one clock cycle for the entire range of input frequencies.

![Block diagram of the proposed all digital DLL system](image)

*Figure 4.1 Reference DLL block diagram [23]*

In the second step, the output bits of the BTDC are copied to C1-C6: the outputs of the counter which in turns controls the delay through the second delay line. Ideally, the delay of the second delay line should be half the clock period but in reality it is always slightly less as the delay of the first delay line is always slightly greater than half the clock period. Due to this, a small error exists between the input and output clocks. After the generation of CLK_out, the phase detector (PD) tracks the error that exists between the input and output clocks and depending upon its value the lock detector (LD) generates the lock signal that enables or disables the six bit binary
counter controlling the delay of the second delay line. If the phase error is large, then the error is reduced by making the counter to count up or down depending on the signal it receives from the PD. The phase difference is checked again and this process continues until the lock signal is generated by the LD. At the end of the chain an SR-Latch is employed is to ensure that the duty cycle of the output clock is around 50%. This step requires a maximum of 8 to 9 clock cycles to complete. In the third step, once the lock is achieved the DFD is activated which checks for any changes in the input clock frequency. When the input clock frequency changes, the DFD resets the entire DLL and the entire process is repeated again. Here, the reset signal is kept low for at least 4 clock cycles so that all the internal signals within the DLL are also cleared before the lock process is started again.

4.2 Issues with Paper

When deciding the architecture of the DLL that satisfies the requirements placed by the video IC on the DLL, we focused on finding digital DLL architectures that had both wide input range and also provided intermediate phases at the output. For this purpose, we had initially selected [25] and [23] but we selected the later due to listed performance metrics achieved in the selected architecture [23]. Once the architecture was finalized, we concentrated on understanding the selected architecture’s operation and detailed functionality of each component. After a better understanding of the underlying architecture was developed, we started implementing the individual components. While implementing the various components we found the following issues

- Firstly the paper has quite a few confusing texts and some misguiding figures that are quite confusing. For example, Figure 9 in the selected architecture [23] shows the block diagram of the timing control unit, in the lower path the second DFF is clocked by the inverted output of the first DFF thus always resulting in a low value at output but actually the second DFF should be driven by first DFF’s output rather than its inverted inversion in order to reset the RS latch and generating the half clock pulse signal Clk_p.

- Secondly, during the initial study of the paper, we assumed that the signals d1-d26 and Clk_1-CLK_6 that appear at the output of both the delay lines are delayed versions of the input clock and thus we would achieve 32 clock phases from each of the two delay lines employed and a total of 64 clock phases. But as we gained more insight in to the working of the DLL structure proposed in [23], it was observed that the these are merely internal signals of the delay line structure used by the BTDC and could not be used as clock. For example, as shown in figure 4.2, when a certain bit say bit B3 is high the delay between Clk_2 and Clk_3 will be that of only two nand gates and the signal d1 that is located half way between Clk_2 and Clk_3 will not be generated.
The second issue meant that while the selected architecture satisfied metrics like jitter and power consumption, it is impossible to extract the required 32 phases mentioned in section 3.8. In order to cater for this problem, some changes had to be made to the basic DLL architecture proposed in [23]. For this purpose instead of using the second delay line in the reference architecture we replaced it with delay line similar to the one proposed in [24] and is discussed in more detail in section 5.3. The modified DLL architecture is shown in Figure 4.3.

The modified architecture enables us to extract the required the 32 phases equally spaced between input and output clocks from the second delay line. This also modifies the system functionality and might also effect some performance metrics. In this new architecture, the first delay line is only used in conjunction with BTDC to quantize the input clock pulse while the output clock and 32 phases are extracted from the second delay line.
Figure 4.3 Modified DLL architecture
Chapter 5

Schematic Level design

5.1 Timing Control Unit (TCU)[23]

The first step in the DLL operation is the clock pulse Clk_p generation. The width of the clock pulse is the half the period of the input clock frequency.

The circuit for clock pulse generation is shown in Figure 5.1. At the rising edge of the input frequency, the upper half of the circuit is activated, the Clk_p signal makes a transition from low to high and remains high until the falling edge of the input clock arrives and the falling edge activates the lower half of the circuit and at this time the Clk_p signal makes a transition from high to low. For a single input frequency the clock pulse Clk_p is generated only once and Clk_p remains low for the rest of the time until the DLL remains in locked state. When the input signal frequency changes the reset signal goes low and remains low for 4 clock cycles resetting the DFFs, once the reset signal goes high the entire process is repeated for the clock pulse generation.

Figure 5.1 Timing control unit structure for clock pulse generation [23].

5.2 First Delay Line [23]

As shown in Figure 4.3, our design employs two digitally controlled delay lines. First delay line along with the binary time to digital circuit (BTDC) is used to quantize the pulse width into bits B1-B6. The bits B1-B6 generated in the previous step are fine tuned by the counter and are used to control the delay of input clock through the second delay line.

The first delay line is divided into six delay groups with each group generating a delay, depending upon the input bit controlling a particular group. Each delay element is made up of nand gates as shown in Figure 5.3. For example, the delay of the first group is controlled by the input bit B1 of the BTDC and subsequent bits controlling that of following groups.
When B1 is high, the delay between the input and output clocks, Clk_in and Clk_1, will be equal to the delay of two nand gates and when B1 is low, the delay between Clk_in and Clk_1 will be equal to four nand gate delays. It is worth mentioning here that the delay of a single nand gate is quantified as $td/2$ where $td$ is 180 ps. So when B1 is high, the delay between Clk_in and Clk_1 is 180ps and when B1 is low, the delay between Clk_in and Clk_1 will be 360 ps. In general, it can be said that when a certain bit $Bn$ is low, the delay is increased by $2^{n-1}+1 \times td$ between Clk_n-1 and Clk_n where $n$ varies from two to six. Please refer to Table 5.5.

The signals d1-d26 shown in Figure 5.3, are uniformly distributed between clocks Clk_2 and Clk_6. For example d1 is halfway between Clk_2 and Clk_3, d2-d4 between Clk_3 and Clk_4, d5-d11 between Clk_4 and Clk_5 and d12-d26 between Clk_5 and Clk_6. All these signals are used by the BTDC to quantize Clk_p i.e., the clock pulse will be asserted only once for each input clock with half the period of input clock.

Figure 5.2 Timing diagram for first delay line [23].
5.3 Second Delay Line [24]

In order to achieve the required 32 phases of the input clock as per DLL specifications, we employ a second delay line similar to the one proposed in [24]. This delay line is used to achieve fine delay between various phases. We employed 32 delay blocks as shown in Figure 5.4. Each delay block is composed of an inverter and six switch controlled capacitances as shown in Figure 5.5. The load of each inverter is composed of six binary weighted capacitors. The delay of each block is controlled by the six bits from the counter, so that all the 32 blocks contribute equal delay. Hence equally delayed phases will be measured.
As can be seen from Figure 5.5, the switch is enabled when controlling bits are zero. When all the bits are “one”, the switch is “off” and thus, the only significant delay would be provided by the inverter. For example, for a bit configuration of B1-B6=000111, each stage would generate a delay of 77 ps+69 ps where 69 ps is the delay contributed by the inverter. The total delay contributed by all 32 stages of the delay line would be 4.67 ns. Table 5.1 summarizes the delay produced by the individual delay stages.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Delay Contributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>0</td>
<td>11 +69 [ps*]</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>69 [ps]</td>
</tr>
<tr>
<td>B2</td>
<td>0</td>
<td>22 +69 [ps]</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>69[ps]</td>
</tr>
<tr>
<td>B3</td>
<td>0</td>
<td>44 +69[ps]</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>69[ps]</td>
</tr>
<tr>
<td>B4</td>
<td>0</td>
<td>88 +69 [ps]</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>69 [ps]</td>
</tr>
<tr>
<td>B5</td>
<td>0</td>
<td>176+69 [ps]</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>69[ps]</td>
</tr>
<tr>
<td>B6</td>
<td>0</td>
<td>352+69[ps]</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>69[ps]</td>
</tr>
</tbody>
</table>

*69ps is the delay of each inverter

Table 5.1 Bit values and corresponding delay as generated through 2nd delay line.

Figure 5.6 shows the relationship between input code and its corresponding time delay that is generated by the delay line.
The foremost design metric taken into consideration during the design of this delay line, ensure linear performance according to the input controlling signal. As can be observed from Figure 5.6 the delay line exhibits a nearly linear behavior.

5.4 Binary Time to Digital Conversion [23]

The binary time to digital converter is composed of a 5-bit time digital circuit and an encoder. As mentioned in previous sections, the task of TDC is to digitize the pulse, CLK_p. In order to achieve this, TDC employs 32-DFFs which are clocked by the 32 clocks from the digital delay line.

Figure 5.7 Binary Time to digital converter block diagram [23].
The output of the DFFs is a set of digital codes that are in turn used for the signals that control the bits governing the initial delay of both delay lines.

In order to understand the operation of this TDC, an example is used to illustrate the working of this block. The operation of the TDC can be divided in to two main following steps,

- Control signal Generation
- Bits Generation

The above mentioned steps now will be explained consecutively.

### 5.4.1 Control signal Generation [23]

The control signal generation is shown in Figure 5.8. In order to generate these signals (dc1-dc6 and dd1-dd26), 32 DFFs are employed which generate two sets of codes, one of them produced when the input clock of DFF is one of the six clock signals (Clk_1-Clk_6) from the first delay line with their outputs named dc1-dc6 which are in turn used to generate more internal signals for the encoder. The second set is produced, when the input clocks of the DFFs are intermediate signals (d1-d26) generated from the first delay line. These signals are named as dd1-dd26.

For example let us consider the input signal of 80 MHz at the DLL input. Since this is virtually the first step in the locking process, the clock pulse generated by the TCU is quantized by using the clock signals at the output of the first delay line. Since all the input bits are set to zero at the beginning, the clocks will be generated with the some initial delays. These delay values are shown in Table 5.2. A more detailed timing information table can be found in appendix A.

Now, all the clocks signals (Clk_1-Clk_6) and the intermediate clocks (d1-d26) from the first delay line that are high within the duration of Clk_p will yield a high value at the DFF output.
The Figure 5.9, shows the timing diagram for this case, as it can be seen from Figure 5.9 all clocks that have a delay of less than 34.7td (34.7*180 ps = 6.25 ns i.e., pulse width of 80 MHz) will produce an active high signal at the DFF output. In this case (dc1-dc4) and (dd1-dd11) would be high while the remaining signals (dc5-dc6) and (dd12-dd26) would be low.

<table>
<thead>
<tr>
<th>Clock Name</th>
<th>Delay*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clk_1</td>
<td>2td</td>
</tr>
<tr>
<td>Clk_2</td>
<td>5td</td>
</tr>
<tr>
<td>Clk_3</td>
<td>10td</td>
</tr>
<tr>
<td>Clk_4</td>
<td>19td</td>
</tr>
<tr>
<td>Clk_5</td>
<td>36td</td>
</tr>
<tr>
<td>Clk_6</td>
<td>69td</td>
</tr>
</tbody>
</table>

*Delay w.r.t. input clocks when all bits are zero

\` 1 td = 180ps

*Delay w.r.t. Input clock

Table 5.2 Delay between intermediate clock signals w.r.t. input clocks when all bits are zero

Figure 5.9 Timing diagram for clock pulse quantization in binary time to digital converter for 80 MHz.
5.4.2 Bits Generation [23]

Bits generation can be divided into two steps:

- Control signal or M’s generation
- Actual bits generation

M’s are actually control signals that are generated as shown in Figure 5.10. The signals generated in this step (named as M1-M5) are used to in the next step to set any one of the bits to zero. It is to be noted that for a particular input frequency only one of these signals M1-M5 is zero.

Figure 5.10 Control signals (M’s) generation in time to digital converter [23].

The bits generation circuitry is shown in Figure 5.11. Initially all the bits are set to zero by setting the reset signal to low. Once all the required signals are generated the signals M1-M5 and dd1-dd26 are encoded to generate the bits that are used to set the delay of the second delay line to the required value. It is worth mentioning here that the bit B1 is always set to zero in order to minimize hardware. Also the bit relating to the low M signal generated above (only one of the five M signals is low) will be low.

For the case of 80 MHz discussed above, this step would yield the values of M’s shown in Table 5.3.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Logic value</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>1</td>
</tr>
<tr>
<td>M2</td>
<td>1</td>
</tr>
<tr>
<td>M3</td>
<td>1</td>
</tr>
<tr>
<td>M4</td>
<td>0</td>
</tr>
<tr>
<td>M5</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.3 Values of M signals of encoder for 80 MHz input frequency
The bits generated for this case are shown in Table 5.4:

<table>
<thead>
<tr>
<th>Bit name</th>
<th>Logic value</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>0</td>
</tr>
<tr>
<td>B2</td>
<td>0</td>
</tr>
<tr>
<td>B3</td>
<td>0</td>
</tr>
<tr>
<td>B4</td>
<td>0</td>
</tr>
<tr>
<td>B5</td>
<td>0</td>
</tr>
<tr>
<td>B6</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.4 Values of bits generated for 80 MHz input frequency.

Bits generation circuit is shown in Figure 5.11.

- The value of B1 for our case is always set to zero due to reasons that will become obvious below.
- The value of B2 will be zero as no path exists between Vdd and ground.
- The value of B3 will be zero as no path exists between Vdd and ground.
- The value of B4 will be zero as no path exists between Vdd and ground.
- The value of B5 will be zero, the pmos with gate input of M4 conducts resulting in B5 to be zero.
- The value of B6 is one as the pull down network is active and thus generates an active high value.

In order to quantize the digital delay line, a 7-bit TDC would have been required meaning a total of 128 DFFs but in our case by using only six bits for the quantization we save about 64 DFFs in comparison to a conventional TDC. Also by setting B1 to zero the number of DFFs in the BTDC are further halved.

In general, the requirements for the number of DFFs are about a quarter of that required for the conventional TDC. Since the delay lines generate a delay that is slightly greater than half the clock period, the residual phase error is corrected for by using a PD along with a counter to tune the delay of the second delay line thus minimizing the phase error.

### 5.5 Structure Functionality

The basic underlying concept in designing the encoder is that for a given input frequency, the delay of the first delay line should be approximately equal to the half the input clock period. In order to get such a delay, the bits at the input of the delay line need to be set accordingly. For example, as mentioned earlier when bit B1 is high the delay generated by group G1 of delay line is 1td but when B1 is low the delay is 2tds. Table 5.5 summarizes the delay through each group G1-G6.
For each input clock frequency we need to generate bits accordingly, so that the required delay i.e. half the input clock period or equal to length of the clock pulse Clk_p is achieved.

To set a bit value to zero there are two ways:

- Either the value of M related to the bit value is set to zero
- Pull down network should not conduct. In other words the value of the bit should remain to its pre set value of 0.

To set the bit value to one, the pull down network should provide a path to the ground. As can be seen from the encoder structure shown in Figure 5.11, there are multiple branches in the pull down network for a particular bit.

<table>
<thead>
<tr>
<th>Clock Signal</th>
<th>Bits</th>
<th>Logic Value</th>
<th>Delay^</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clk_1</td>
<td>Bit1</td>
<td>High</td>
<td>1td*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>2td</td>
</tr>
<tr>
<td>Clk_2</td>
<td>Bit2</td>
<td>High</td>
<td>1td</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>3td</td>
</tr>
<tr>
<td>Clk_3</td>
<td>Bit3</td>
<td>High</td>
<td>1td</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>5td</td>
</tr>
<tr>
<td>Clk_4</td>
<td>Bit4</td>
<td>High</td>
<td>1td</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>9td</td>
</tr>
<tr>
<td>Clk_5</td>
<td>Bit5</td>
<td>High</td>
<td>1td</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>17td</td>
</tr>
<tr>
<td>Clk_6</td>
<td>Bit6</td>
<td>High</td>
<td>1td</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>33td</td>
</tr>
</tbody>
</table>

Table 5.5 Clock signal and bit values relation to internal delay of first delay line

^ delay represents the delay between two adjacent clocks *1td=180ps

Our desired frequency range is 62-300 MHz, the clock period and pulse width range from 16-3.33 ns and 8 -1.66 ns respectively. As mentioned above, initially all the bits are set to zero, Clk_1-Clk_6 are generated with delays mentioned in Table 5.2. Since Clk_2 comes after 5td and Clk_3 comes after 10td all the frequencies which have pulse width between 5td and 10td (for frequencies between 280-300 MHz) have dc1-dc6 = 110000. As Clk_4 comes after 19td all the frequencies having pulse width between 10td and 19td (for frequencies between 146-280 MHz) have dc1-dc6=111000. Similarly as CLK_5 that comes after 36td all the frequencies having pulse width between 19td and 36td (for frequencies between 78-146 MHz) have dc1-dc6=111100. For other frequency ranges refer to Table 5.6. This information of dc1-dc6 is used to generate M signals. Table 5.6 shows the frequency ranges and corresponding values of M signals.

When any of the M signals is zero it gives information about a particular frequency range. The respective M signal that is low for a particular frequency range, changes a specific bit related to that M signal to zero(regardless of its previous value). This bit is used to set the delay close to half the period of input clock.
For fine tuning, the delay of the first delay line to be equal to half of the input clock period (as the delay in the previous step is not exact) the signals dd1-dd26 are used. For example, for the input frequency range 78-146 MHz, the pulse width for this frequency range is 6.49 -3.42 ns, in terms of td this delay will be around 19td-36td. From Table 5.6 it can be seen that for this case dc1-dc6=111100 and M4=0 which sets B5=0 (generating a 17td delay). For each input frequency in a given frequency range dd1-dd26 are always unique that are used to set the remaining bits to achieve the exact delay.

<table>
<thead>
<tr>
<th>Input Frequency Range (MHz)</th>
<th>Period (ns)</th>
<th>Pulse Width (ns)</th>
<th>M1-M5</th>
<th>B1-B6</th>
<th>Delay Produced (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>62-65</td>
<td>16-15.38</td>
<td>8-7.69</td>
<td>11110</td>
<td>000110</td>
<td>45td</td>
</tr>
<tr>
<td>65-68</td>
<td>15.38-14.7</td>
<td>7.69-7.35</td>
<td>11110</td>
<td>010110</td>
<td>43td</td>
</tr>
<tr>
<td>68-72</td>
<td>14.7-13.88</td>
<td>7.35-6.94</td>
<td>11110</td>
<td>001110</td>
<td>41td</td>
</tr>
<tr>
<td>72-78</td>
<td>13.88-12.98</td>
<td>6.94-6.49</td>
<td>11110</td>
<td>011110</td>
<td>39td</td>
</tr>
<tr>
<td>78-83</td>
<td>12.98-12.03</td>
<td>6.49-6.01</td>
<td>11101</td>
<td>000001</td>
<td>37td</td>
</tr>
<tr>
<td>83-88</td>
<td>12.03-11.36</td>
<td>6.01-5.68</td>
<td>11101</td>
<td>010001</td>
<td>35td</td>
</tr>
<tr>
<td>88-94</td>
<td>11.36-10.37</td>
<td>5.68-5.18</td>
<td>11101</td>
<td>001001</td>
<td>33td</td>
</tr>
<tr>
<td>94-101</td>
<td>10.37-9.9</td>
<td>5.18-4.95</td>
<td>11101</td>
<td>011001</td>
<td>31td</td>
</tr>
<tr>
<td>101-109</td>
<td>9.9-9.17</td>
<td>4.95-4.58</td>
<td>11101</td>
<td>000101</td>
<td>29td</td>
</tr>
<tr>
<td>118-129</td>
<td>8.47-7.75</td>
<td>4.24-3.88</td>
<td>11101</td>
<td>001101</td>
<td>25td</td>
</tr>
<tr>
<td>129-146</td>
<td>7.75-6.84</td>
<td>3.88-3.42</td>
<td>11101</td>
<td>011101</td>
<td>23td</td>
</tr>
<tr>
<td>146-168</td>
<td>6.84-5.95</td>
<td>3.42-2.98</td>
<td>11011</td>
<td>000111</td>
<td>21td</td>
</tr>
<tr>
<td>168-193</td>
<td>5.95-5.18</td>
<td>2.98-2.59</td>
<td>11011</td>
<td>010011</td>
<td>19td</td>
</tr>
<tr>
<td>193-224</td>
<td>5.18-4.46</td>
<td>2.59-2.23</td>
<td>11011</td>
<td>001011</td>
<td>17td</td>
</tr>
<tr>
<td>224-280</td>
<td>4.46-3.57</td>
<td>2.23-1.78</td>
<td>11011</td>
<td>010111</td>
<td>15td</td>
</tr>
<tr>
<td>280-300</td>
<td>3.57-3.33</td>
<td>1.78-1.66</td>
<td>10111</td>
<td>010111</td>
<td>11td</td>
</tr>
</tbody>
</table>

Table 5.6 Internal control signal & bit values for various frequency ranges

\[1td=180ps\]

For example, when input frequency is 120 MHz, pulse width is 4.167 ns, approximately a delay of 23.2td (23.3*180 ps=4.17 ns) is required, setting B5=0, gives a delay of 17td. Signal d5 arrives after a delay of 21.5td and d6 after 23.5td as shown in Table A1, appendix A. For a 23.3td pulse width, the corresponding signals have value dd1-dd26=111110...00. These signals (dd1-dd26) along with M1-M5 are used to generate following bit values. Bits generation circuit is sown in Figure 5.11.

- Bit1 is always 0 which gives a delay to 2td.
- Bit2 is maintained at 0 as no path connects output to Vdd or ground, this gives a delay of 3td (total delay=5td)
Bit3 is set to 1 by activating the one of the pull down networks (the 3\textsuperscript{rd} branch since $M_4 = 1, \overline{dd6} = 1, \overline{dd8} = 1$) generating a delay of 1td. (total delay=6td)

Bit4 is set to one as the pull down network is active (the branch since $M_3=1$ and $dd8 = 1$) generating a delay of 1td. (total delay=7td)

Bit5 is zero for this frequency range resulting in a delay of 17td (refer to the above paragraph) (total delay=24td)

Bit6 is one for this frequency range as $M_5$ is 1 generating a delay of 1td. (total delay=25td)

This delay is slightly larger than the required delay i.e., 23.2td of half clock period. Once the bits are generated, they are copied to the second delay line and the counter. Second delay line should generate a delay of one clock period i.e., 46.4td between input and output clocks with equally spaced 32 phases in between.

For the above example, the bits loaded to the second delay line are B1-B6=001101. According to Table 5.1, the delay generated from each stage of second delay line is $11\text{ ps} + 22\text{ ps} + 0\text{ ps} + 0\text{ ps} + 176\text{ ps} + 0\text{ ps} + 69\text{ ps} = 278\text{ ps}$. The total delay generated by the second delay line is $32*278\text{ ps}=8.89\text{ ns}$ which is equal to 49.4td. As this delay is 3td more than the required delay of 46.4td, this delay is catered for by the counter by counting down so that the required delay is achieved.

For another input frequency of say 90 MHz in same input frequency range, pulse width is 5.556 ns, approximately a delay of 30.9td ($30.9*180\text{ ps}=5.556\text{ ns}$) is required, setting $B_5=0$, gives a delay of 17td. Signal $d_9$ comes after a delay of 29.5td and $d_{10}$ after 31.5td refer to TableA1 is appendix A. For a 30.9td pulse width the corresponding signals have value $dd1$-$dd26=1111111110\ldots00$. These signals ($dd1$-$dd26$) along with $M_1$-$M_5$ are used to generate following bit values. Bits generation circuit is shown in Figure 5.11.

- Bit1 is always 0 which gives a delay to 2td.
- Bit2 is maintained at 0 as no path connects output to Vdd and ground this gives a delay of 3td (total delay=5td)
- Bit3 is set to 1 by activating the one of the pull down networks (the 2\textsuperscript{nd} branch since $M_4 = 1, \overline{dd10} = 1, dd8 = 1$) generating a delay of 1td (total delay=6td).
- Bit4 is maintained at 0 as no path connects output to Vdd and ground this gives a delay of 9td (Total Delay=15td)
- Bit5 is zero for this frequency range resulting in a delay of 17td (refer to Table 5.5) (total delay=32td).
- Bit6 is one for this frequency range as $M_5$ is 1 generating a delay of 1td. (total delay=33td)

This delay is slightly larger than the required delay i.e., 30.9td of half clock period. Once the bits are generated, they are copied to the second delay line and the counter. Now the second delay line should delay the input clock by one clock period i.e., 61.8td.
Figure 5.11 Encoder circuitry for bits generation [23].
For this example, the bits loaded to the second delay line are B1-B6=001001. According to Table 5.1, the delay generated from each stage of second delay line is 11 ps + 22 ps + 0 ps + 88 ps + 176 ps + 0 ps + 69 ps = 366 ps. The total delay generated by the second delay line is 32*366 ps = 11.71 ns which is equal to 65.1td. As this delay is approximately 3td more than the required delay this delay is catered for by the counter by counting down so that the required delay is achieved.

5.6 Counter

The counter used to control the delay is a loadable six bit up/down counter with asynchronous reset. Figure 5.12 shows the counter schematic. The counter is activated or deactivated by the “enable” signal generated by LD, based on the fact that input and output clocks are locked. The counter also checks for “load signal” so that the counter could be loaded with bits generated by the BTDC, the load signal is a delayed version of the Clk_p signal generated by the TCU. Another of the counters’ input is the “up/down” signal which is read from the PD and consequently counts up or down based on the signal’s value.

![Figure 5.12 Counter schematic](image)

5.7 Lock Detector [23]

The lock detector (LD) is composed of DFFs, nand gates and delay components. The primary task of the LD is to detect the difference between the input and output clocks and generates corresponding signals to increase or decrease the delay of the delay line such that the time difference between the two signals reaches in the 180ps window. When the time difference between the two inputs signals is greater than 180ps the enable signal is high, keeping the counter in active state and when the time difference between the two input signals is in the window of 180ps the enable signal at LD’s output goes low, thus disabling the counter and indicating that the DLL is in locked state.
Figure 5.13 shows the employed lock detector. As seen from the figure at the first DFF a 1td delayed output clock signal is sampled by the input clock signal while at the second DFF it is sampled by the same signal which is 2td delayed. The first DFF always exerts a high signal at the nand gate. If the rising edge of the Clk_Out does not occur in the 180 ps window, the contribution of the second DFF to the nand input would be low, resulting in a high enable signal indicating that DLL is not in lock state. If however the rising edge of the Clk_Out occurs in the 180 ps time window, the second DFF contributes a high value to the nand gate resulting in a low enable signal indicating the DLL has achieved the lock state.

5.8 Dynamic Frequency Detector (DFD)

Dynamic frequency detector (DFD) is used to reset the DLL in case of change in input frequency. When the time difference between input and output clocks becomes greater than 1td (180ps), the enable signal from LD is de-asserted. By detecting this change, the DFD resets the entire DLL for 4 cycles. Figure 5.14 shows DFD structure. Both DFFs have “enable” signal as their input, one of the DFF being clocked by input clock (Clk_in) and the other by a delayed version of the input clock. When the input frequency changes and the enable signal go low, the output of the upper DFF is low while that of the lower is high. When these two signals i.e. the output of the upper DFF after being inverted and the output of the second DFF are applied to the nand gate, the output of the nand changes from high to low, activating the in-line negative edge triggered DFF. This results in a high to low transition of the global reset signal. The counter starts counting, which counts for 4 clock cycles. After 4 cycles, internal reset signal becomes low, which resets the DFF connected to counter. The global reset signal changes from low to high and counter is stopped. At this point one might ask why this entire process needs to be based on the clock signal, why not simply use the lock signal with a DLL to set the counter as shown in Figure 5.14. The reason for this is that the lock signal has glitches in it and is not a clean digital signal. This might result in false locking and unpredictable system behavior.
Figure 5.14 Dynamic frequency detector schematic
Chapter 6

Simulation Results

In this chapter, the simulation results achieved from Cadence environment are presented. Simulations were run for input frequency of 300 MHz. As mentioned in chapter 3, the operation is divided into the following steps.

6.1 Clock Pulse Generation

Figure 6.1 shows the generation of clock pulse for 300 MHz input frequency signal. The signal in blue is the generated clock pulse signal which is approximately equals to half the period of the input clock presented in yellow color.

![Clock Pulse generated for 300 MHz input](image)

Figure 6.1 Clock Pulse generated for 300 MHz input
6.2 Bits and Output Clock Generation

The BTDC quantize the clock pulse and generate bits B1-B6. For 300 MHz, the bit values are B1-B6=010111. Figure 6.2, shows the bit values generated for 300 MHz frequency.

Figure 6.2 Bits generated for 300 MHz input

As mentioned in chapter 5, each block in the second delay line is binary weighted, the delay generated by each block as a result of these bit values is 55 ps.

- B1=0  ->  11 ps
- B2=1  ->  0 ps
- B3=0  ->  44 ps
- B4=1  ->  0 ps
- B5=1  ->  0 ps
- B6=1  ->  0 ps
The delay through each stage of the second delay line is 55 ps+69 ps=124 ps where 69 ps is the delay through inverter of each stage. After passing through 32 stages (as we require 32 phases) this value comes to 3.96 ns while for this input frequency we require a delay of 3.3 ns (i.e., period of input frequency). As can be seen from red arrows in Figure 6.3, the output clock is not exactly aligned to the input clock.

**6.3 Output Clock alignment and Lock Signal Generation**

Once the output is generated, if it is located in the ± 1td range the lock signal is generated, otherwise if the delay is more than ± td (180 ps) the counter would have to decrease the delay (by counting up) so that lock state is achieved.
Figure 6.4 Output clock for 300 MHz input after fine tuning

It can be seen from Figure 6.4, when the difference between input and output clocks becomes smaller i.e., comes in the time span of ± 1td, the enable signal become low indicating the DLL is in locked state. The DLL remains in locked state until the arrival of a new input frequency. When a change in input frequency is detected, the DLL is reset and the entire process starts again.
6.4 Inter Phase Delay

An imminent feature of a DLL is that the output phases should be linearly distributed between input and output clocks. This behavior can be observed in Figure 6.5, where the distance of phases from the input clock linearly increases.

Figure 6.5 Linear behavior of DLL delay line

6.5 Jitter Analysis

Another important parameter used for characterizing DLL is Jitter. For a good DLL, the jitter should be Low. Figure 6.6 shows the long term peak to peak jitter for Proposed DLL. The long term jitter measured for 2000 clock cycles while simulating at the maximum frequency comes out to be 13 ps which is much smaller than the required value 90 ps.
Figure 6.6 Jitter analysis at 300 MHz
Conclusions and Future Work

In this thesis we presented a wide range all digital DLL for video applications by using standard digital blocks and components like D flip flops and logic gates. The proposed design was implemented using a 65 nm CMOS process with a 1V supply voltage.

During this thesis we have exploited technology scaling to achieve good switching characteristics and low power consumption and a small chip area.

The following conclusions were made:

- The proposed design consumes only 3.7 mW of power.
- The proposed DLL has lock time 9 clock cycles in the worst case.
- A worst case peak to peak jitter of 13 ps has been achieved for the proposed DLL.
- The proposed DLL operates in the frequency of 62 - 300 MHz.
- The proposed DLL achieves a phase resolution of 32 phases between input and output clocks.

A possible extension to the proposed design could be

- The input frequency range can be enhanced by lowering it to 10 MHz so as to cater for low frequency video formats currently in order to process these formats we assume the input video signal has to be upsampled six times in order to achieve desired 32 phases.
- Another enhancement could be the introduction of a phase selection mechanism that selects the desired phase and passes it on to the ADC for further processing.
- Layout of the designed DLL could be generated.
Table A1 shows the delay between intermediate clock signals of first delay line and the input clock when all the bits are zero. These intermediate clock signals are used by BTDC for clock pulse (Clk\_p) quantization to digital bits.

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clk_1</td>
<td>2td*</td>
</tr>
<tr>
<td>Clk_2</td>
<td>5td</td>
</tr>
<tr>
<td>d1</td>
<td>7.5td</td>
</tr>
<tr>
<td>Clk_3</td>
<td>10td</td>
</tr>
<tr>
<td>d2</td>
<td>12.5td</td>
</tr>
<tr>
<td>d3</td>
<td>14.5td</td>
</tr>
<tr>
<td>d4</td>
<td>16.5td</td>
</tr>
<tr>
<td>Clk_4</td>
<td>19td</td>
</tr>
<tr>
<td>d5</td>
<td>21.5td</td>
</tr>
<tr>
<td>d6</td>
<td>23.5td</td>
</tr>
<tr>
<td>d7</td>
<td>25.5td</td>
</tr>
<tr>
<td>d8</td>
<td>27.5td</td>
</tr>
<tr>
<td>d9</td>
<td>29.5td</td>
</tr>
<tr>
<td>d10</td>
<td>31.5td</td>
</tr>
<tr>
<td>d11</td>
<td>33.5td</td>
</tr>
<tr>
<td>Clk_5</td>
<td>36td</td>
</tr>
<tr>
<td>d12</td>
<td>38.5td</td>
</tr>
<tr>
<td>d13</td>
<td>40.5td</td>
</tr>
<tr>
<td>d14</td>
<td>42.5td</td>
</tr>
<tr>
<td>d15</td>
<td>44.5td</td>
</tr>
<tr>
<td>d16</td>
<td>46.5td</td>
</tr>
<tr>
<td>d17</td>
<td>48.5td</td>
</tr>
<tr>
<td>d18</td>
<td>50.5td</td>
</tr>
<tr>
<td>d19</td>
<td>52.5td</td>
</tr>
<tr>
<td>d20</td>
<td>54.5td</td>
</tr>
<tr>
<td>d21</td>
<td>56.5td</td>
</tr>
<tr>
<td>d22</td>
<td>58.5td</td>
</tr>
<tr>
<td>d23</td>
<td>60.5td</td>
</tr>
<tr>
<td>d24</td>
<td>62.5td</td>
</tr>
<tr>
<td>d25</td>
<td>64.5td</td>
</tr>
<tr>
<td>d26</td>
<td>66.5td</td>
</tr>
<tr>
<td>Clk_6</td>
<td>69td</td>
</tr>
</tbody>
</table>

Table A1. Delay between intermediate clock signals w.r.t. input clock when all bits are zero

*1td equals 180 ps
References


[22] Synopsys, High Definition Video AFE: Far beyond the ADC, April 2010


[25] Sebastian Hoyos et al., A 15MHz~600MHz, 20 mW, 0.38mm^2, Fast Coarse Locking Digital DLL in 0.13um CMOS, 34th European Solid-State Circuits Conference, pp. 90-93, Edinburgh, SCOTLAND, 2008