Front-end ASIC Design and Verification of Programmable and Precision Delay Timing Generator for CCD Detectors

Major Project Report

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology
in
Electronics & Communication Engineering
(VLSI Design)

By

Urvi Mehta
(13MECV11)

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481
May 2015

Front-end ASIC Design and Verification of Programmable and Precision Delay Timing Generator for CCD Detectors

Major Project Report

Submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Electronics & Communication Engineering (VLSI Design)

By

Urvi Mehta
(13MECV11)

Under the guidance of

External Project Guide:

Shri Mohammad Waris
Shri Rajiv kumaran
Sci/Eng.,SAC
ISRO,Ahmedabad.

Internal Project Guide:

Dr. Usha S. Mehta
Professor,EC
Institute of Technology,
Nirma University, Ahmedabad.

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481
May 2015

Declaration

This is to certify that

1. The thesis comprises my original work towards the degree of Master of Technology in VLSI Design at Nirma University and has not been submitted elsewhere for a degree.
2. Due acknowledgment has been made in the text to all other material used.

- Urvi Mehta
13MECV11

NIRMA UNIVERSITY
INSTITUTE OF TECHNOLOGY

Certificate

This is to certify that the Major Project entitled **"Front-end ASIC Design and Verification of Programmable and Precision Delay Timing Generator for CCD Detectors"** submitted by **Urvi K. Mehta (13MECV11)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design, Nirma University, Ahmedabad is the record of work carried out by her under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.The results embodied in this major project, to the best of our knowledge,haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Internal Guide

Dr. U. S. Mehta
(Professor,EC)

External guide

Shri. Mohammad Waris
(Sci/Engr -SD)

Shri. Rajiv Kumaran
(Sci/Engr -SF)

Shri. Sanjeev Mehta
(Head -SFED)

Program Co-ordinator

Dr. N. M. Devashrayee
(Professor,EC)

HOD

Dr. P.N. Tekwani
(Professor,EE)

Director

Dr.K Kotecha
(Director,IT-NU)

Acknowledgements

I would like to express my gratitude and sincere thanks to **Dr.P.N.Tekwani**, Head of Electrical Engineering Department, and **Dr. N. M. Devashrayee**, PG Coordinator of M.Tech VLSI Design program for allowing me to undertake this thesis work and for their guidelines during the review process.

I am deeply indebted to my thesis supervisor **Dr. Usha S. Mehta**, Professor, E.C.Dept., Nirma University for her constant guidance and motivation.

I would like to express my gratitude towards thesis supervisor **Shri. Mohammad Waris** (Scientist/Engineer-SD) **Shri. Rajiv Kumaran** (Scientist/Engineer-SF), and **Shri. Sanjeev Mehta** (Head, Sensor Front End Electronic Division), Sensor Electronic Group, Sensor Development Area (SEDA), at Space Application Centre (SAC), Ahmedabad , Indian Space Research Organization (ISRO). Without their experience and insights,it would have been very difficult to do quality work.I am grateful to **Shri. J. Ravisankar**, Head, HRDD, for giving me the opportunity to pursue this thesis in the premises of SAC-ISRO.

Last, but not the least,I would like to thank my family members for their constant support.

- Urvi Mehta
13MECV11

Abstract

Charge Coupled Devices (CCD) work by converting light into a pattern of electronic charge in a silicon chip.CCDs have features like high sensitivity, high linear dynamic range, electronic shuttering capability and low dark current.Various CCD architectures like linear array, full frame, interline, TDI, etc. are used for various application.One of the main unit in a CCD based satellite imaging system is Timing Generator producing various kinds of readout clocks and control signals for different CCD architectures .

This Project discusses about the front-end ASIC design of programmable and precision delay timing generator for various CCD architectures. Various CCD requires different types and number of clocks for its operation. The aim is to provide flexibility in terms of number of different types of clocks, effective image area, read out feature and a very precise delay control for each clock. Programmable clock generation with precise delay is implemented with an all digital Delay Locked Loop (DLL) because of its features like low power, smaller area, high noise immunity and synthesizable circuit. Precise delay resolution in sub nano seconds (approximately 620 ps) is obtained at clock frequency of 20 MHz. Programmable pulse generation logic is designed which can generate pulses with variable frequency and with variable pulse widths. The design is targeted with 180 nm technology. A single chip can cater to support various CCD architectures with full programmability.

Assertion based and coverage driven constrained random verification of precision delay timing generator is carried out using system verilog.

Contents

List of Figures

List of Tables

List of Abbreviations

ADC	Analog to digital convertor
ADDLL	All Digital Delay Lock Loop
ADLL	Analog Delay Lock Loop
AFE	Analog front end
ASIC	Application specific Integrated circuit
CCD	Charge coupled Device
CDS	Correlated double sampling
CLPOB	Clamping Optical black
DLL	Delay Locked Loop
FF	Full Frame
FIL	Frame Interline
FT	Frame Transfer
HBLK	Horizontal Blank
HD	Horizontal Sync pulse
IC	Integrated Circuits
IL	Interline
PBLK	Pre-Blank
PD	Phase Detector
PLL	Phase Locked Loop
SPI	Serial Peripheral Interface
STA	Static Timing Analysis
TDI	Time delay Integration
VBLK	Vertical Blank
VD	Vertical Sync pulse

Chapter 1

Introduction

Charge Coupled Devices (CCD) work by converting light into a pattern of electronic charge in a silicon chip.CCDs have features like high sensitivity, high linear dynamic range, electronic shuttering capability and low dark current.One of the main unit in a CCD based satellite imaging system is Timing Generator producing various kinds of readout clocks and control signals for various CCD architectures like linear, full frame, interline, TDI, etc. based on the application. In general, CCD camera comprises of high-speed CCD sensor, CCD output signal processing circuits, timing controller, timing generator, timing driver and external optics system. One of the critical requirements of these CCD are accurate clock pulse and sequences which are the main function of Timing generator and Control block.[1][2]

1.1 Objective and scope of the project

The aim of this project is to carry out front end design of programmable high precision timing generator ASIC which can generate the various driving time and processing time sequences for various CCD architectures. It should also provide the flexibility in terms of effective image area, number of clocks required for a particular CCD along with the precise and programmable delay control for each clock. An All Digital Delay Locked Loop (ADDLL) is used in order to achieve the precise delay control of the order of 620 ps.. A single chip can support various CCD architectures with programmable and precision delay controls for critical clocks.

CCD driving clocks includes horizontal (readout) clocks, vertical clocks (image transfer/transport clocks), storage clocks, stage selection clocks, reset clocks, exposure clocks etc. Processing clocks includes clocks for CDS (correlated double sampling), black pixel clamping, Preblanking clocks, ADC clock etc. to assure the high image quality of CCD camera.

In the Phase-I of this project, block level front-end ASIC design of timing generator is developed using VHDL. and the pre-synthesis simulation is done by developing the test-benches in verilog(directed test cases). Also the synthesis, Static timing analysis and post synthesis simulation of a Delay locked Loop (DLL),a critical block for precise timing generation is carried out.,Synopsys tools are used for design and analysis. The design is targeted with 180nm technology. In Phase-II of the project, the synthesis of remaining design blocks is carried out and analyzed for timing requirements (STA). Also SPI is developed for serial writing into memory. Assertion based and coverage driven constrained random verification of whole design is carried out using system verilog.

1.2 Organization of Thesis

The thesis is organized with Chapter 1 providing the introduction. Chapter 2 describes about CCD fundamentals and introduction to CCD based satellite imaging system which is part of literature survey. It provides basics of CCD operation, CCD structures, charge storage and transfer operation, clock requirements in processing blocks and some details about image data acquiring and processing system.

Chapter 3 discusses about different methodologies to obtain the precise delay control like PLL, delay lines and DLL along with the methods for generating pulses with different pulse width control.

Chapter 4 discusses design requirements and specifications including overall clock requirements which support all architectures, readout features for detector clocks,methods for generating programmble clocks, design specification and parameters with different clock categories.

Chapter 5 is about detailed design of timing generator, starts with proposed architecture for timing generator which contains proposed architecture with details description of identified blocks of design.This description includes block level information and functional description for each block in detail.

Chapter 6 discusses about simulation and synthesis results.The design blocks are synthesized and STA is done.The post synthesis simulation results are shown for various sub blocks of the design.

Chapter 7 is about verification of whole design.The verification environment is developed and various testcases are given randomly with constraints to achieve desirable coverage.

Chapter 8 is about the conclusion and future work of timing generator design.

Chapter 2

Introduction to CCD Fundamentals

2.1 Introduction to CCD

CCD (Charge Coupled Devices) detectors are widely used for imaging applications found in space and astronomy.CCDs detect a faint amount of light, convert light into pattern of electronic charge in a silicon chip and produce high resolution images.The CCD is a discrete-time device, i.e., analog signal such as light intensity is sampled at discrete times.The memory function comes by shifting these charges, simultaneously, down a row of cells, also in discrete time.[3] CCD can be used as analog memory. Applications include voice storage as in a telephone answering machine.

The more important and universally recognized applications of CCD are seen in image sensors. As an image sensor a CCD has an array of cells to capture a light image by the photo-electric effect. The packets of charge are not initially converted to an electrical signal, but rather moved from cell to cell by the coupling and decoupling of potential wells within the semiconductor that makes up the CCD. At the end of the line the charges, from all the different picture elements (pixels), can be converted to electrical signals. The idea here is to have a large number (may be millions) of sensing cells in order to achieve good resolution, but a small number (may be one) of readout cells for practicality.[4] [5]

2.1.1 Structure of CCD

The basic structure of CCD is shown in the figure 2.1. It consists of an image area (a square array of pixels) of the CCD.An image then builds up that consists of a pattern of electric charge. At the end of the exposure this pattern is then transferred, pixel at a time, by way of the serial register to the on-chip amplifier. Electrical connections are made to the outside world via a series of bond pads and thin gold wires positioned around the chip periphery.

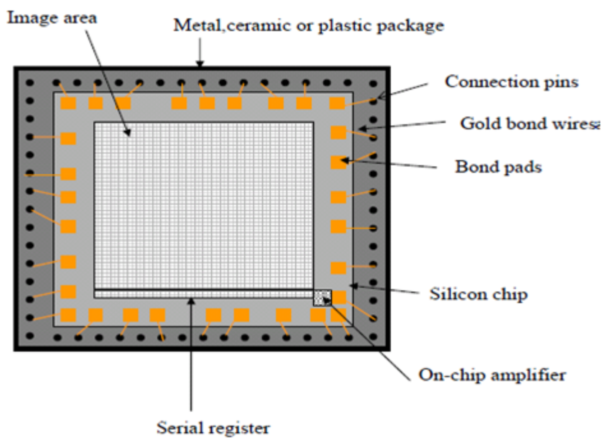


Figure 2.1: Basic structure of CCD[4]

2.1.2 Steps of CCD imaging Operation

CCD imaging is performed in a three step process:

(1) Charge collection in a CCD

Exposure converts light into an electronic charge at discrete sites called pixels.Photons entering the CCD create electron-hole pairs. The electrons are then attracted towards the most positive potential in the device where they create 'charge packets'. Each packet corresponds to a pixel.

(2) Charge transfer which moves the packets of charge within the silicon substrate

Once charge has been integrated and held locally by the bounds of the pixel architecture, there should be some means of getting that charge to the sense amplifier which is physically separated from the pixels. The common methods used involve differing charge transfer techniques like techniques for four phase (4ϕ) CCD, three phase (3ϕ) CCD, pseudo two phase ($P2\phi$) CCD, true two phase ($T2\phi$) CCD[6].

(3) Charge to voltage conversion and output amplification [4] [5]

The packets of charge are eventually shifted to the output sense node where the electrons (which represent a charge) are converted to a voltage that is easier to work with off chip. Conventional techniques usually employ a floating diffusion sense node followed by a charge to voltage amplifier such as a source follower.[6]

2.2 CCD based Satellite imaging System

High speed CCD camera systems are better choice for high-end real time visible imaging systems. All CCD based applications are completely dependent on the architecture of CCD detectors, their operation and processing of their output signals.The timing generator part of the CCD camera critically determines its performance.The working of CCD sensor depends on driving sequences which assure CCD with high transfer efficiency, high signal-to-noise ratio and credible CCD output signal while accurate CCD signal processing sequences ensures high quality of CCD image.[1][2]

Figure 2.4 shows Timing Generator along with other system interfaces for any CCD based satellite imaging system. Complete system is divided into five blocks: CCD detector, Timing Generator, clock drivers, CCD output signal processing blocks and image data acquiring system.

Primarily Timing Generator generates clocks and sequences for two blocks i.e. CCD detector and CCD output signal processing. These clocks are required for image transfer, readout, exposure, reset, shuttering etc. The CCD output is processed by signal processing blocks which contain digitizer, CDS circuitry, black level clamp etc. Clocks required for signal processing block are CDS clocks, blanking/clamping pulses for pre-blanking as well as clamping the optical black region and latching clocks. The digitized pixel information is sent to the digital image processor chip, which performs the post processing which forms part of image data acquiring system. For synchronization between various blocks, additional clocks are required which is generated in timing generator block. Each of these blocks are explained below in detail.

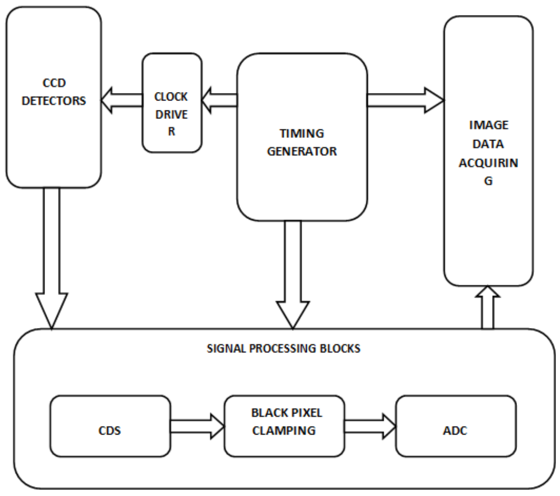


Figure 2.2: System level architecture of CCD based satellite imaging system

2.3 CCD Detector Array Architectures

CCD array architectures are driven by the application. These array architectures can be categorized into various types:[7]

Linear arrays, Time delay and integration (TDI), Full frame, Frame transfer and Interline transfer devices are used for various industrial applications, scientific applications, consumer camcorders and professional television systems.Table 2.1 summarizes about clock requirements for these detector architectures.

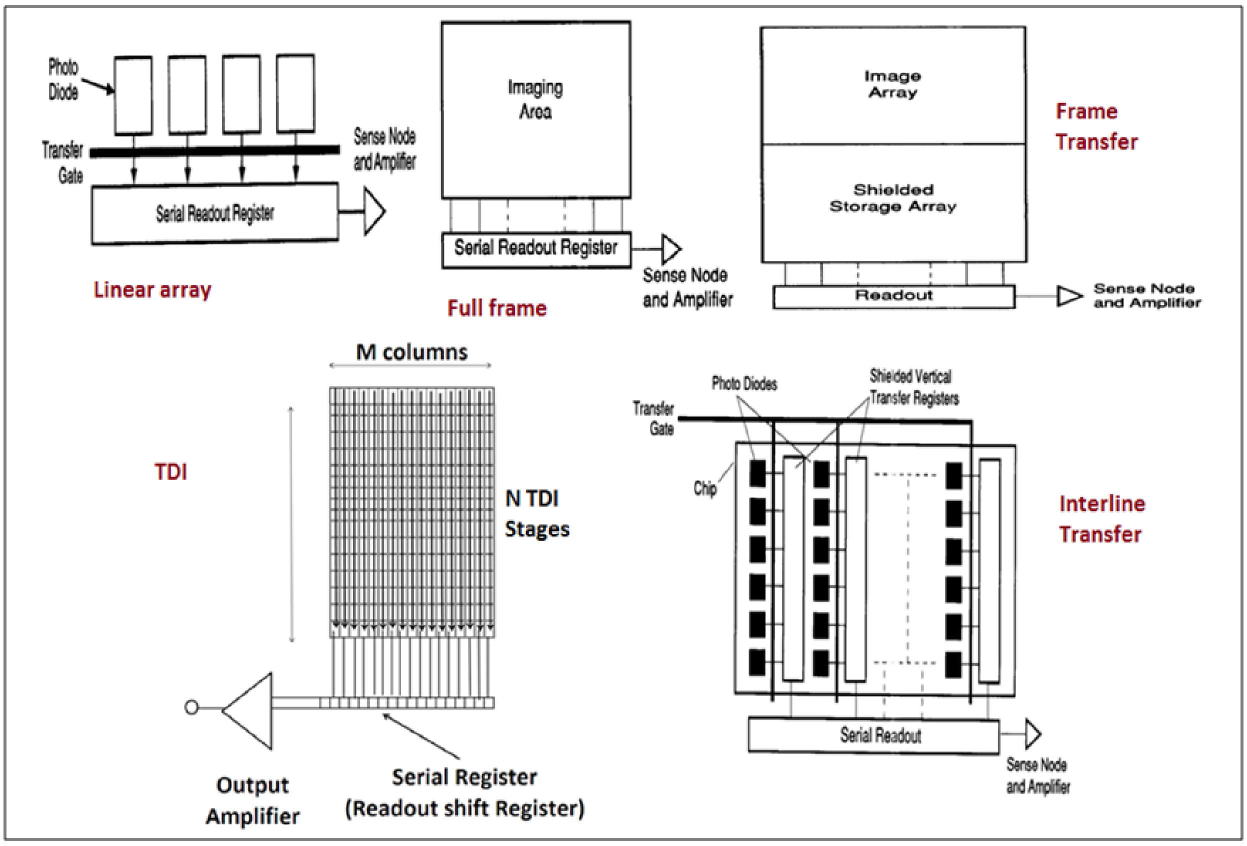


Figure 2.3: CCD Array Architectures[7]

Table 2.1 Summary of Clock requirements for CCD array architectures[7]

Type of CCD	Description	Clocks Required
Linear array	• As shown in the figure 2.3 located next to each sensor is the transfer gate followed by CCD shift register. • A variant of this is bilinear array. For a fixed pixel size,the bilinear readout increases the effective charge transfer effciency as the number of transfers is reduced by a factor of 2. • Linear CCDs are used for fast operation and high resolution.	• Transfer gate clock pulse to move the charge from photodiode into CCD transfer register. • Clock pulse for serial readout operation. • Reset gating clock when the charge is transferred from output register to amplifier.[7]
Full Frame Arrays	• Here, after the integration, the image pixels are read out line by line through a serial register which then clocks its contents onto the output sense node. • All charge must be clocked out of the serial register before the next line can be transferred. An electronic shutter can be used to shield the array during read out to avoid smear. • The effective readout rate can be increased by factor of N, if N sub arrays are read out simultaneously. There is a tradeoff between the frame rate and image size. • The full frame design is used for CCD imagers with higher resolution and higher density.	• Exposure clocks which pulses the CCD substrate to clear out accumulated charge for electronic shuttering. • Horizontal clocks for pixel read out operation
Frame Transfer	• A frame transfer imager consists of two identical arrays as shown in figure 2.3, one devoted to image pixels and one for the storage. • After the integration cycle,charge is quickly transferred from light sensitive pixels to storage cells and readout off chip is carried out simultaneously when storage array is integrating the next frame. • Continuous or shutter less operation is achieved resulting in faster frame rates than full frame device. • Lower resolution and higher cost than full frame architecture.	• Exposure clocks for electronic shuttering • Horizontal clocks for pixel read out operation • Vertical clocks for shifting each line of pixels into serial readout register • Reset clocks for reset gating when charge is transferred from output register to amplifier • Storage clocks for transferring each line of pixel from image area to storage area.[7]

Interline Transfer	• The interline array consists of photodiodes separated by vertical transfer registers (which are covered by an opaque metal shield).The signal charge produced is transferred to vertical shift register through transfer gate. • The main advantage is that transfer from active sensors to shielded storage area is quick as charge transfer is performed simultaneously for all pixels. • To solve the problem of smear, the signal charge is transferred to storage section at high speed.	• Exposure clocks for electronic shuttering • Horizontal clocks for pixel read out operation • Vertical clocks for shifting each line of pixels into serial readout register • Reset clocks for reset gating when charge is transferred from output register to amplifier • Storage clocks for transferring each line of pixel from image area to storage area.[7] • Vertical sensor gate pulses to transfer pixel charges from light sensitive area to vertical registers
Time Delay Integration (TDI)	• TDI, shown in figure 2.3, is a specialized detector readout mode used for observing a high-speed moving object under low light conditions normally undetectable by classic CCD imaging. • TDI is designed to allow continuous movement of the object past the sensor to produce a continuous video image of a moving, two-dimensional object: a definite advantage over stop-then-start capture. • The number of lines on the sensor corresponds to the increase in sensitivity than a single line scan camera. • It provides exposure control by controlling the number of stages. This is generally achieved by turning off selection clocks.	• Horizontal clocks for pixel read out operation • Vertical clocks for shifting each line of pixels into serial readout register • Reset clocks for reset gating when charge is transferred from output register to amplifier • Vertical transfer clock pulses are used to transfer pixel charges from light sensitive area to vertical registers • TDI stage selection clock for selecting the number of stages for exposure and the last stage of each register has a separate clock to speed up the signal transfer and reduce the video settling time.[7]

2.4 CCD Signal Processing Blocks

CCD Analog Front-end (AFE)clocks or Processing blocks include double correlation samplings (CDS) circuitry, black pixels clamp and ADC to assure the high image quality of CCD camera, which is sent to image acquiring system after digitization.

These clocks and pulses include CDS (Correlated Double Sampling) Clocks, Blanking/Clamping clocks and clocks for optical black region (CLPOB) which can reduce noise and sample true image data, Pre-blanking pulses (PBLK) for pre-scan pixel removal and ADC clocks which samples the true video signal and further sent to image data acquiring system.. All these clocks and sequences are high frequency clocks.

The timing diagram for CLPOB and PBLK clock pulses are shown in figure 2.4 and figure 2.5 respectively. Here HD and VD are horizontal and vertical synchronizing pulse respectively.

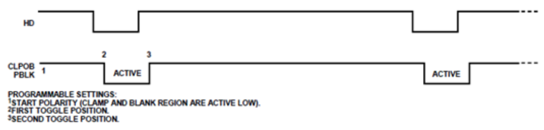


Figure 2.4: CLPOB and PBLK placement[9]

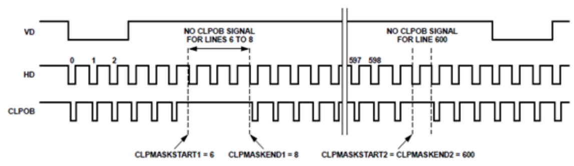

VD
NO CLPOB SIGNAL FOR LINES 6 TO 8
NO CLPOB SIGNAL FOR LINE 600
0
1
2
597
598
HD
CLPOB
CLPMASKSTART1 = 6
CLPMASKEND1 = 8
CLPMASKSTART2 = CLPMASKEND2 = 600

Figure 2.5: CLPOB masking[9]

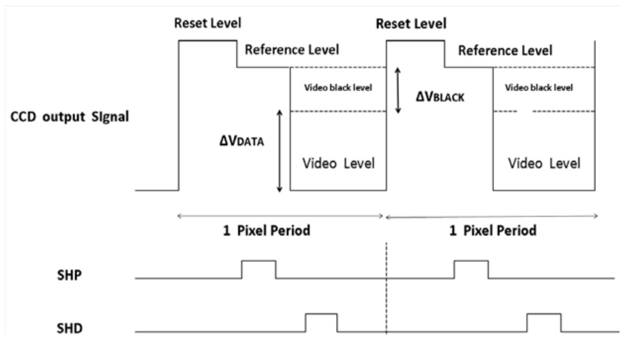


Figure 2.6: CCD signal Double correlation sampling

This CDS technology is not only the important mean to restraint CCD camera noise but also an important part of the video signal processing. The changes based on the reset noise and the reference level can be restrained by characteristics of correlation. The reset noise from the reset gate pulse will be neglected when CDS technology is used. CDS can also restraint the 1/f noise of the source follower.

The CDS circuit samples each CCD pixel twice to extract the video information and reject low frequency noise. Figure 2.6 illustrates how two CDS clocks, SHP and SHD, are used to sample the reference level (floating gate level) and data level (video level) of the CCD signal respectively. The reference level corresponds to the zero photo electrons. But the video level is not the actual signal level we required; it includes the video black level. If we only use these two levels we would not get the true signal or image data which represent the incident photons. The CCD has the black reference pixel or lines which show the video black level. The Black clamp clocks inform the processing circuit the positions of CCD black pixels or lines. So the CDS circuit gets the video black level and floating gate level. Then the black pixel signal, VBLACK and the true image data, VDATA are attained in succession. The value of VDATA is digitized by the analog-to-digital circuit. The A/D clocks should be synchronous with CCD video signal. For digital image processing and data acquiring, three synchronous clocks (pixel, line and frame) are necessary. [2][8]

2.5 Timing Generator

Timing generator is the critical block of a satellite imaging system. It provides various clocks to all other blocks. Most importantly it generates various timing sequences like exposure clocks, vertical (image transfer) clocks, Horizontal (readout) clocks, reset clocks, storage clocks etc for driving the given CCD architecture and processing timing sequences like CDS sampling clocks, latching clocks for digitizer and clamping clocks for clamping circuitry.Apart from these, it also generates some synchronization pulses (line and frame synchronization) for image acquiring system for post-processing and compression.

This Thesis discusses about design of precision timing generator which can support various CCD detector architectures. The aim is to provide different types of clocks with different readout features like rising/falling edge programmability with the precise delay control, polarity control, enabling control, masking, toggle positions, patterns, sequence and repeats pattern groups, etc. This Design also supports flexible image area for CCDs. Following Chapters show details about its design architecture and various features.

2.6 Image data acquiring and processing systems

CCD detector output signal processed and digitized by signal processing block. All processed image information is sent to image data acquiring and processing systems. This system can have different sub-blocks based on requirements. In general, this block can have digital image processing ASIC, image compression circuits, encoders and other image acquiring systems. This section performs different operations like image compression, final output image, image data acquisition and other processing.

2.7 Summary

This chapter gives basic idea about CCD architectures ,its working and also about system level blocks for CCD based satellite imaging system.Following section provides detail of Timing Generator block.

Chapter 3

Precision Timing Generation

Different types of CCD architectures and signal processing blocks require various types of clocks as described in the previous chapter. One of the important feature required in these horizontal and vertical clocks is the precise delay control. Typically delay control is achieved by using different phases of clock.In order to obtain this precise delay control following circuits can be used:

1. Phase Locked Loop (PLL)
2. Delay line
3. Delay locked loop (DLL)

3.1 Phase Locked Loop (PLL)

PLLs are used for clock synthesis and hence can be used to generate different phases from a clock.

A Phase-Locked Loop (PLL) is a feedback system that causes an output signal to track a reference signal in phase and frequency. When both the frequency and phase of these signals are synchronized, the PLL is said to be in the locked condition. The phase difference between the output and reference is a known value when the loop is locked. PLLs are found in a wide range of applications, including clock recovery, noise and jitter suppression in communications, clock synchronization in memory interface and high-performance microprocessors, and frequency synthesis for instrumentation and RF transceivers.

A basic PLL is a negative feedback system that consists of a phase detector, a low-pass loop filter and a voltage-controlled oscillator (VCO), as shown in Figure 3.1 A divide-by-N counter can be added in the feedback path to provide a frequency synthesis function. Figure 3.2 shows the waveform of a reference signal and PLL output when the loop is in lock (using a divide-by-2 in the feedback path). The frequency of the PLL output signal is twice the reference with zero-phase difference (assuming N=2).

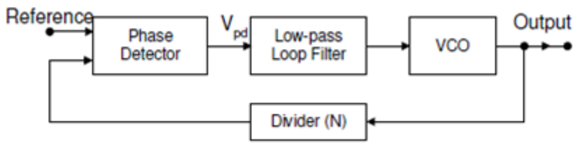


Figure 3.1: Block diagram of PLL

The basic operations of a PLL can be divided into three steps: First, the phase detector (PD) matches the phase difference between two inputs and generates an error signal V_{Pd} whose average value is linearly proportional to the phase difference.A Loop filter (LF)(which is a first order system) is then used to suppress the high-frequency components of the PD output, allowing the average value (DC or low-frequency) to control the VCO frequency.Finally, an oscillator generates an output signal whose frequency is a linear function of the control signal out of the LF. The generated signal is fed back to the input of the PD and another phase comparison is started until the phase difference achieves a fixed relationship.

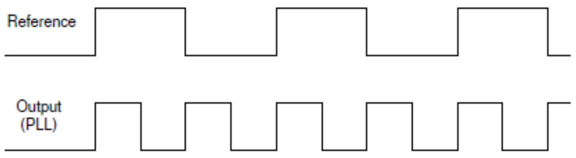


Figure 3.2: PLL in lock (N=2 in this case)

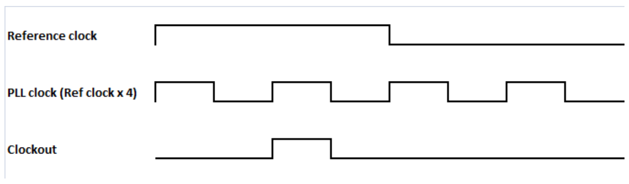


Figure 3.3: precision delay control with PLL clock

Figure 3.3 shows how PLL can be used to generate clocks with multiple and programmable phase shift. Here ,for example to obtain 4 edge placement within the reference clock period,a divider N=4 is required such that PLL output clock frequency is (4 x Reference clock) .So if frequency of reference clock is 20 MHz, frequency of PLL output is (4 x 20) 80 MHz and the phase resolution changes from 50 ns to 12.5 ns and a precise delay control can be achieved.

3.2 Delay line

In PLL based approach frequency multiplication is carried to generate different clock phases out. Another way to obtain precise delays is to use a Delay Line.

Digital delay lines are classified into two types:[10]

1. Coarse delay line also referred to as Gate delay DCDL
2. Fine delay line also referred to as Sub gate-delay DCDL.

There are various number of delay line structures that are employed or that have been proposed for different applications. The following section briefly discusses few of the commonly used DCDL structures.

3.2.1 Coarse Delay Line

The coarse delay lines are composed of CMOS logic gates and are cascaded to form a delay line. The simplest of the delay lines for digital DLL is a chain of cascaded inverters with each stage consisting of a pair of inverters and the required output tap is selected by a multiplexer. The minimum delay of each stage is $2T_D$ where $T_D$is the average CMOS gate delay and requires $\log_2 N$ storage elements where N is the number of stages. A coarse delay line design is shown in the Figure 3.4. [10]

This implementation has four delay stages between the input and output. If the required number of delay stages is more then it may load the input clock. To overcome the loading, separate buffers must be introduced thus increasing the delay of each cell to more than $2T_D$. This increase in delay would depend on the amount of added buffering.

A work-around to this problem is shown in Figure 3.5.[10] 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 thus maintaining the delay of each stage to $2T_D$. A shift register controls the delay of each cell. For example, when Q [0] =1 and Q [3:1] =0 the output clock signal would have a two nand gate delay i.e. the delay of A and B nand gates in cell 0. When 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.[10]
Figure 3.6 shows another implementation of coarse delay line. Here, the delay line consists of multiple delay cells connected in cascade. Each delay cell consists of a multiplexer (MUX) and a buffer. The input1 of the MUX of first delay cell is connected to supply and for all other delay cells the input1 is connected to the output of the previous stage cell. The input2

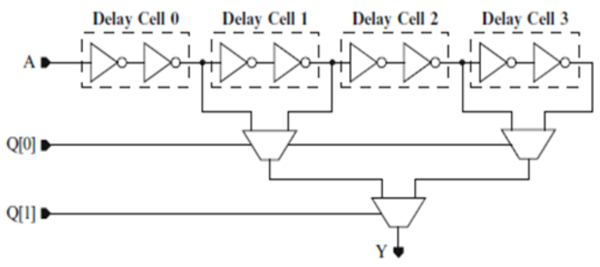


Figure 3.4: Inverter based coarse delay line[10]

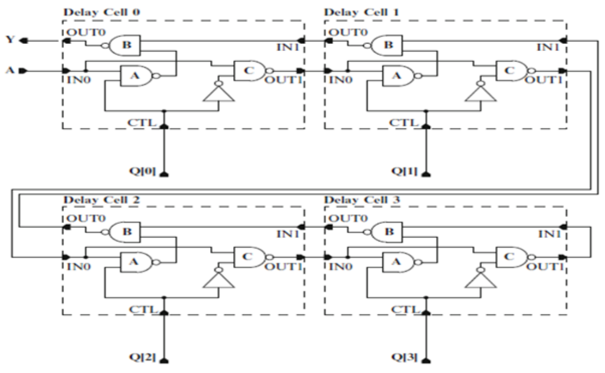


Figure 3.5: NAND based coarse delay line[10]

of all the MUX are connected to clock. The delay line is controlled through the select lines which are the output of the control logic.[10]

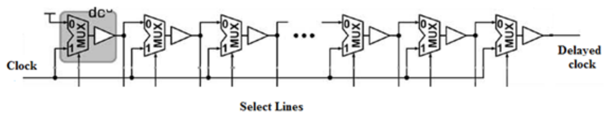


Figure 3.6: MUX based Delay line[10]

3.2.2 Fine Delay Line

Two designs of the fine delay line, which generate delay by using the RC delay characteristics are shown in Figure 3.7 and Figure 3.8.

The structure in Figure 3.7 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.[10]
In the second design of the fine delay shown in Figure 3.8, the RC delay increases with the number of ON devices. The digital control bits can be encoded using both thermometric

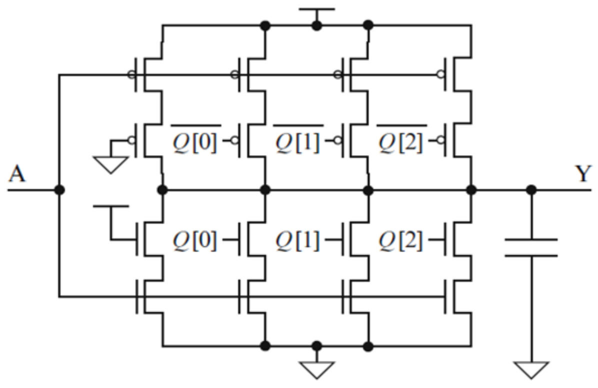


Figure 3.7: Typical digital controlled fine delay line[10]

and logarithmic codes due to linear characteristics of the circuit. As in former structure the dynamic range increases by cascading several stages

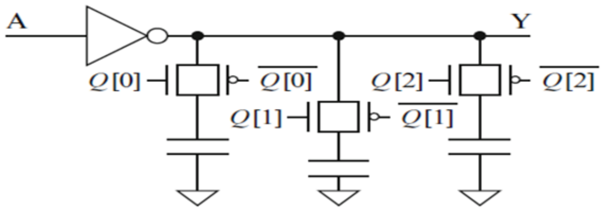


Figure 3.8: Thermometric/logarithmic digital controlled fine delay line [10]

In case of coarse delay line the delay resolution obtained is in several hundreds of ps whereas in case of fine delay line the delay resolution is in tens of ps.

Challenges in designing delay line:
1. Every delay element should have equal delay.
2. The resolution of delay line is obtained by the amount of delay that can be achieved with a single delay element.
3. Another important thing is to get a 50 percent duty cycle of the output clock.Here due to the rise and fall time propagation delay differences of different gates, the duty cycle variations may occur.

3.3 Delay Locked Loop (DLL)

DLL can be used for a variety of applications which require precise time intervals. The function of the 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.

DLL is a feedback system that aligns the feedback clock to the reference clock. This is done by delaying the input feedback clock after passing it through a delay line and controlling the delay using the control mechanism. Once the input feedback 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.

A DLL is able to provide multiple clock signals which are separated from each other by a well-controlled phase shift (delay). This application of DLL is utilized in obtaining high precision clocks required for timing generators.

Generally a phase detector, delay line and a charge pump or counter make up a DLL.DLL can be either primarily analog or digital depending on the design components.

3.3.1 Analog DLL

Analog DLLs (ADLL) were first used in clock distribution applications. The components of a conventional Analog DLL are, phase detector, charge pump, low pass filter and voltage controlled delay line as shown in Figure 3.9.[11]

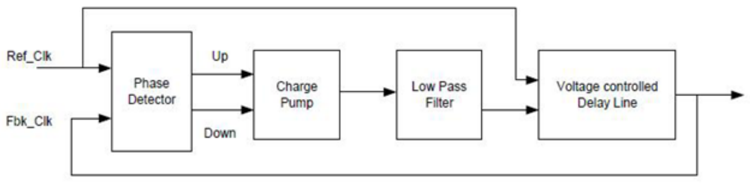


Figure 3.9: Analog DLL

A phase detector compares two input signals and generates "UP" and "DN" output pulses that represent the direction and magnitude of the input phase error.Considering simple type of phase detectors such as the phase-frequency detectors are those which are only rising and falling edge sensitive. They produce a single pulse at the UP or DN output for each pair of input reference and feedback edges depending on which edge arrives first.

Some of the simple PD architectures are shown below. An XOR based PD is shown in figure 3.10 . 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:

1. Any variation in the input duty cycle may cause this PD to generate wrong phase information.
2. Since this PD has a single output it complicates the design of subsequent blocks namely charge pump.

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. If the setup and hold times are not symmetric this may introduce a regular phase error as lagging and leading decisions based on the hold and setup times, respectively. This type of PD has been proposed in [12] and is shown in Figure 3.11.

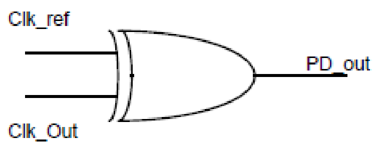


Figure 3.10: XOR based PD[12]

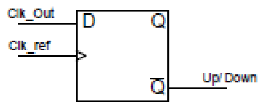


Figure 3.11: D-Flipflop based PD[12]

A charge pump, connected to the phase detector, sources or sinks current for the duration of the UP and DN pulses from the phase detector. The net output charge is proportional to the difference between the pulse widths of the UP and DN outputs. The charge pump drives the loop filter which integrates and filters the charge current to produce the control voltage.

Figure 3.12 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.[12]

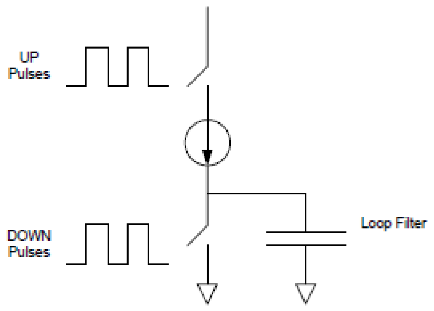


Figure 3.12: Basic charge pump structure[12]

A charge pump based on this working principle has certain limitations: Due to mismatches

between the charging and discharging currents may occur in jitter particularly once the DLL is locked.The non ideal behavior of switches may also introduce jitter in the signal.
The most critical component in the performance of a DLL is a VCDL which directly influence DLL jitter performance and stability. The control Voltage drives a voltage-controlled delay line (VCDL) in a DLL which generates a delay proportional to the control voltage.

Ideally, the delay generated by each stage is equal to T_{ref}/n where T_{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 is shown in Figure 3.13. Current changes with Vcontrol, thus changing the driving capability of the delay stage.

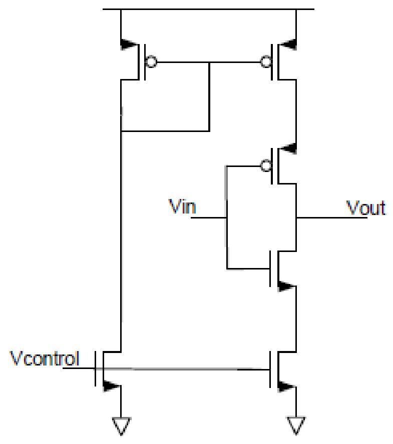


Figure 3.13: Single ended voltage controlled delay line cell

Figure 3.14 shows a differential implementation of a VCDL delay line cell as discussed in [12].The resistance of the diode connected structure would be:

$$r_{out} = \frac{1}{g_{out}} + \frac{1}{r_n} + \frac{1}{r_p}$$

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

$$g_m = \sqrt{\mu_p}\, C_{ox}\, \frac{W}{L}\, I_{ss}$$

where I_{ss} is the tail source current.Substituting this value of gm to the above equation rout can be written as

$$\mathrm{r}_{out} = \frac{1}{g_m}$$

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

$$\mathrm{t}_d = \sqrt{2}\mathrm{r}_{out}\, C_{out}$$

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}.

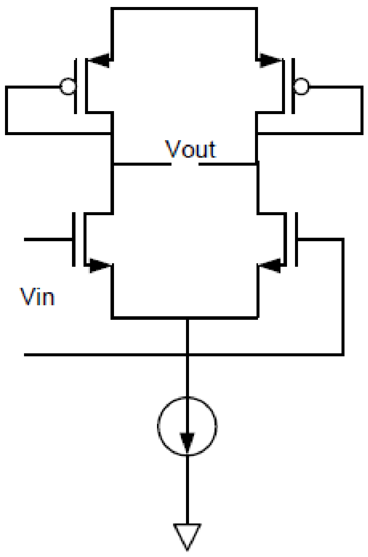


Figure 3.14: differential implementation of a VCDL delay line cell[12]

Important points:

1. An Analog DLL is a complex analog circuit requiring process-specific implementation.
2. An Analog DLL generally provides better jitter performance at the expense of greater complexity.
3. Scaling of analog integrated circuits directly affects output resistance and intrinsic gain of the circuit. This results in higher power consumption and enhanced design complexity.

3.3.2 Digital DLL

Digital DLLs (ADDLL) are characterized by their use of digitally controlled delay line. They are typically made of digital circuit elements. Figure 3.15 shows a conventional Digital DLL.Here there is a Phase Detector which compares the phase of reference and feedback clocks and generates a single up/down signal depending on the phase error. 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.

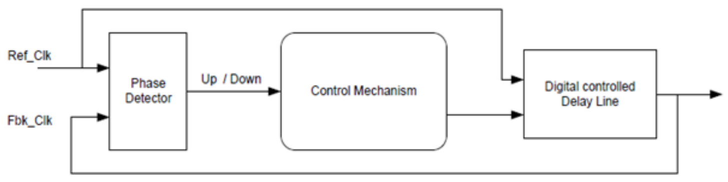


Figure 3.15: Digital DLL[12]

Control mechanism (which can be of different types) is used to generate signals in order to control the delay of digitally controlled delay line depending on up/down signal.

The delay line is the most important component of the DLL and has a profound effect on the overall DLL performance. The various types of the Digitally controlled delay lines have already been discussed in previous section.

Important Points:
1. If the clock frequency is f and its period is T,length of a delay line is N delay of a unit delay element is t ,then for delay line to get locked over its entire length T= N * t resolution = t.
2. Digital DLL has more noise immunity and synthesizable circuit makes it very attractive for clock alignment applications.

A comparision of PLL, Delay line, ADLL and ADDLL is summarized in table 3.1.

Table 3.1 Summary of PLL, Delay line and DLL [12][17][18]

PLL	**Delay line**	**ADLL**	**ADDLL**
At least a second order system	Zero order system	First order system (LPF)	Zero order system
To generate a fractional delay clock frequency multiplication is required. Thus power consuming.	No clock frequency multiplication needed. Less power consuming	No clock frequency multiplication needed. Less power consuming	No clock frequency multiplication needed. Least power consuming
Susceptible to PVT variations due to non linear locking behavior	Susceptible to PVT variations	Less PVT invariant as compared to PLL.	Less PVT invariant as compared to PLL.
VCO is supply and	Noise immunity is	Noise immunity better	Substrate and supply

substrate noise sensitive, phase error accumulates.	poor	than PLL but poorer than digital DLL.	noise immune.
Occupies more area than DLL	Occupies less area than PLL and DLL	Occupies more area than digital DLL.	Occupies less area than analog DLL.

3.4 Approaches For Digital DLL Designs:

The Digital DLLs are characterized by the controller that is used to control the delay line.

Digital DLLs control can be divided into three categories.

1. Register controlled DLL
2. Counter controlled DLL
3. Successive Approximation Register controlled DLL (SAR -DLL)

3.4.1 Register controlled DLL (RDLL)

Fig. 3.16 shows the block diagram of the RDLL. The replica input buffer dummy delay in the feedback path is used to match the delay of the input clock buffer. The phase detector (PD) is used to compare the relative timing of the edges of the input clock signal and the feedback clock signal, which comes through the delay line, controlled by the shift register. The outputs of the PD, shift-right and shift-left, are used to control the shift register.

In the simplest case, one bit of the shift register is high. This single bit is used to select a point of entry for CLKIn in the symmetrical delay line .When the rising edge of the input clock is within the rising edges of the output clock and one unit delay of the output clock, both outputs of the PD, shift-right and shift-left, go to logic LOW and the loop is locked.[13]

As shown in Figure 3.17, the input clock is a common input to every delay stage. The shift register is used to select a different tap of the delay line. The complementary outputs of each register cell are used to select the different tap: Q is connected directly to the input A of a delay element, and Q* is connected to the previous stage of input B. From right to left, the first LOW-to-HIGH transition in the shift register sets the point of entry into the delay line. The input clock will pass through the tap with a high logic state in the corresponding position of the shift register. Since the Q* of this tap is equal to a LOW, it will disable the previous stages; therefore, it does not matter what the previous states of the shift register are.

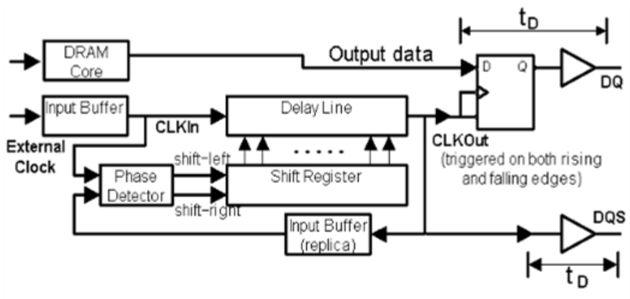


Figure 3.16: Block diagram of RDLL[13]

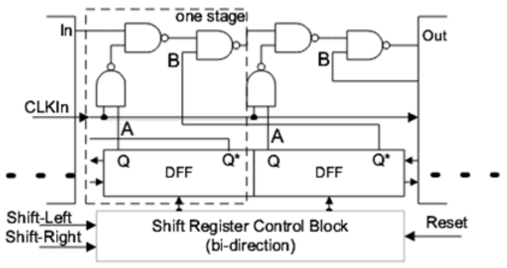


Figure 3.17: Delay element in RDLL[13]

Important points:
1. RDLL is stable against temperature, process, and power-supply variations.
2.The locked time and the number of delay cells increase exponentially as the number of control bits increases.
3.A symmetrical delay line is used in this approach . The delay line used in this DLL has the same delay whether a high-to-low or a low-to -high logic signal is propagating along the line.

3.4.2 Counter controlled DLL (CDLL)

Figure 3.18 shows the block diagram of the CDLL. It is similar to the RDLL, except that an up/down counter substitutes for the shift register to control the delay line. The n-bit control word from the up/down counter determines whether the input clock goes through the delay stage or just passes it. The principle of operation is similar to that case in a RDLL except that the n-bit up/down counter counts up or down to control the delay line.[14]
Important points:
1. The counter-based controller replaces the register-based to achieve better performance as higher level abstraction is used for optimization.
2. The locked time and the required delay cells of the counter-controlled DLL are similar to those of the register-controlled DLL.

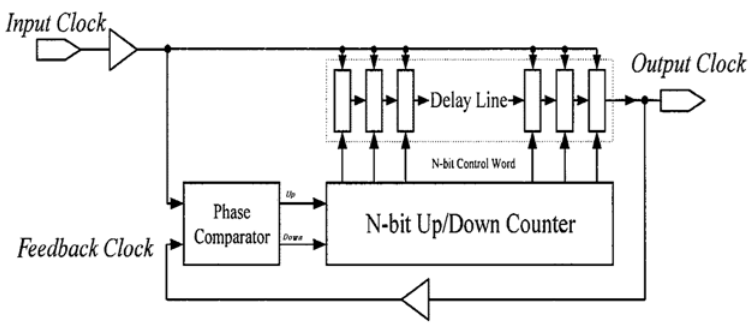


Figure 3.18: Block Diagram of CDLL[14]

3.4.3 Successive Approximation Register Controlled DLL (SAR-DLL)

Besides the cost of chip area, another important parameter to evaluate the performance of DLL is the lock time. In the case of the RDLL, if the point of entry of the delay line is chosen in the middle at first, the longest lock time will be 32 input clock periods for the 64-stage delay line. In the case of the CDLL, if the initial value of the counter is set as 32, the longest lock time will also be 32 input clock periods when a 6-bit up/down counter is used. Both of the digital DLL mentioned above exhibit the same lock time.[14]

The block diagram of the SARDLL is shown in Figure 3.19. It consists of a Phase Comparator, a digital-controlled delay line as in the conventional digital DLL's, and an additional frequency divider, an initial circuit (INCKT), and a 6-bit SAR to provide a control word for the delay line.

The binary search algorithm is the main operation principle of the proposed SARDLL. Theoretically, when a 6-bit binary-weighted delay line is used, a SARDLL can achieve lock time of six clock periods, which is faster than a RDLL or a CDLL.

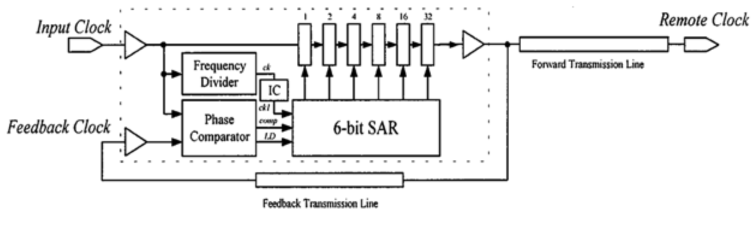


Figure 3.19: Block Diagram of SAR DLL[14]

Figure 3.20 shows the entire binary-weighted digital-controlled delay line. The number on the top of each stage denotes the number of unit delay it provides. The delay line is divided into two sections, coarse section and fine section. Instead of combining the stages controlled by b4 or b5 into two stages with longer delay time, the 16-unit delay stage (controlled by b4) is composed of two 8-unit delay stages and the 32 delay stage (controlled by b5) is

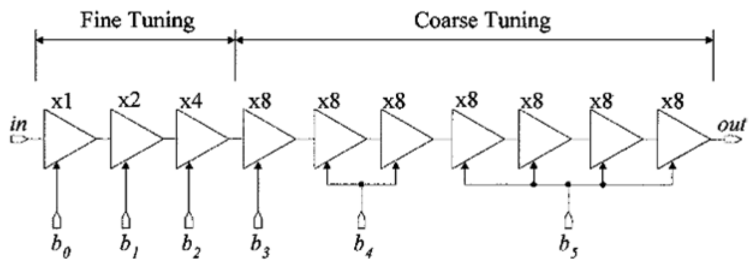


Figure 3.20: Binary weighted delay line in SAR- DLL[14]

composed of four 8-unit delay stages. This is due to fact that the longer the delay time per stage is, the more difficult it is for the rise time and fall time of the delay stage to match with each other.[14]
Important points:
1. A SAR-DLL can reduce the locked time by using the binary search algorithm but it increases the chip area.
2. Also as the binary weighted delay line is used , there would be considerable mismatch between rise time and fall time of each delay stage.

It is planned to implement all digital logic (DLL) with counter control approach considering overall advantage in terms of power,complexity and size.

3.5 Edge Programmable timing signal generation (clock pulse generation)

Timing circuits are typically required to generate signals having programmable phases and pulse widths.

An edge programmable timing generator is shown in the figure 3.21. It is a circuit consisting of three DLL and some combinational gates.[16]

A method of generating an output timing signal having variable pulse width, comprises of the following steps

1. Generating an input timing signal having a predetermined logic high pulse width

2. Delaying the input timing signal by a predetermined fixed length of time and generating a delayed input timing signal (say signal 1)

3. Delaying input timing signal by a predetermined length of time corresponding to ris-

ing edge value and generating a rising edge signal (say signal 2)

4. Again delaying signal 1 by a predetermined length of time corresponding to falling edge value and generating a falling edge signal (say signal 3)

5. Performing a logic OR or AND operation on said rising and falling edge signals (signal 2 and signal 3)and generating an output signal pulse.Figure 3.22 shows the entire process of

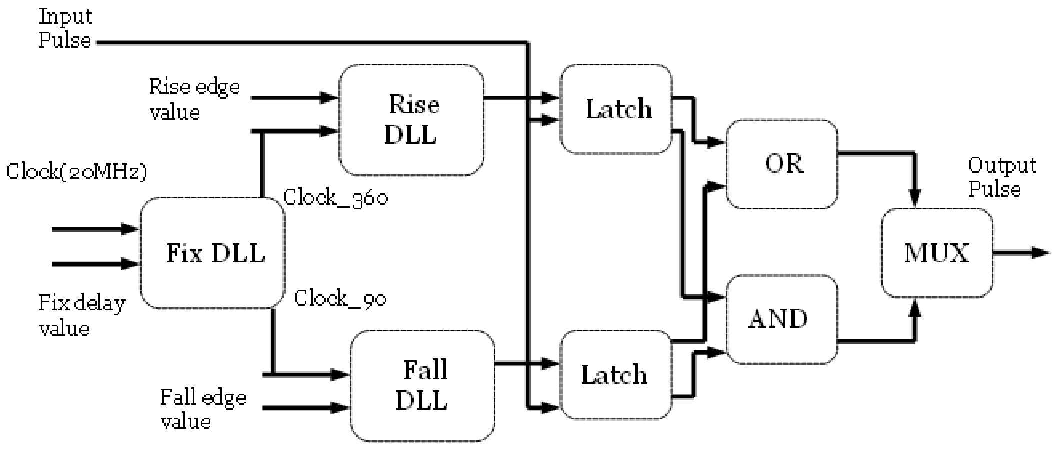


Figure 3.21: An edge programmable timing signal generator

edge programmable output pulse generation from the input signal.

In operation, referring to Figure 3.22 A and B, a coarse “timing signal is shown in Figure 3.22 A having a predetermined logic high level pulse width, for example 1.5 nanoseconds. The signal output from fix DLL is shown in Figure 3.22 B offset by a predetermined length of time (eg. 0.75 nanoseconds) from the input coarse timing signal of Figure 3.22 A. With no delay programmed into either of the DLLs (rise and fall) , the signal output from combinational gate logic is shown in Figure 3.22 C. Thus signals having logic high level pulse widths greater than or equal to the pulse width of the coarse timing input signal can be generated.

Now, in the event it is desired to generate a signal having a pulse width less than that shown in Figure 3.22 C yet greater than the pulse width of the input signal, a delay factor is loaded into rising edge DLL such that the output signal from DLL is delayed by a predetermined amount (less than the delay of delay register), as designated by the letter “D” and the dashed lines in Figure 3.22 A. In response, the signal output from combinational gate logic Figure 3.22 D.

Further, in the event it is desired to generate a signal having a pulse width greater than that shown in Figure 3.22 C, a delay factor is loaded into falling DLL , as shown in dashed lines and designated by the letter "E" in Figure 3.22 B. In response, combinational gate logic generates the signal illustrated in Figure 3.22 E.

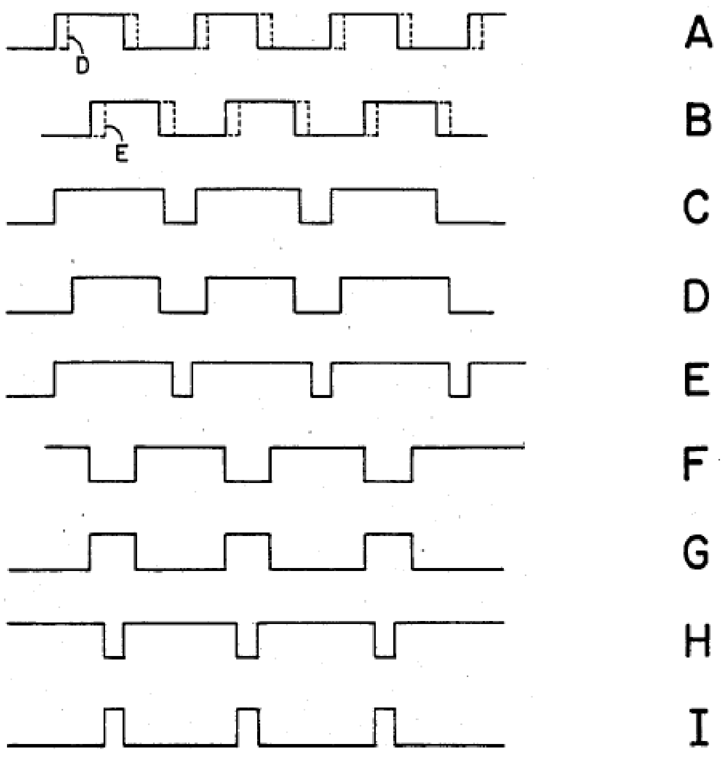


Figure 3.22: Programmable Pulse generator waveforms[16]

It is possible to generate a low going pulse, having predetermined programmable rising and falling edges which define a pulse width which is less than the minimum pulse width generated by the coarse timing circuit , as described above. However, in order to generate short high going pulses, a short low going pulse is generated first by the above method, and the pulse is subsequently inverted.

In order to generate a signal having pulse width less than the pulse width of the input signal, first the pulse shown in figure 3.22 F is generated and is the inverted to obtain final output pulse Figure 3.22 G. Further to generate a signal having yet a shorter pulse width, refer to the figure 3.22 H and figure 3.22 I.

3.6 Summary

This chapter discusses about the various methods to achieve the precise delay control for various horizontal and vertical clocks. An ADDLL is used due to its features like high noise imuunity, low power, synthesizable circuit, etc. Also Delay lines can be used but they are

PVT invariant. The method for programmable pulse generation using DLL with pulse width control is discussed.

Chapter 4

Design Requirements And Specifications

4.1 Clock Requirements for Timing Generator

As discussed, different CCD detectors requires different types of clocks. Based on literature survey and study of different available CCD architectures [7], clocks are classified in two categories i.e. CCD driving clocks and CCD signal processing clocks.

4.1.1 CCD Driving Clocks

• Exposure clocks
The CCD image exposure time is controlled by the exposure clock which is also known as substrate clock signal, which pulses the CCD substrate to clear out accumulated charge. This is nothing but electronic shuttering.

• Vertical clocks (image zone vertical / image transfer/transport clock):
These are low frequency clocks for shifting each line of pixels into horizontal outputs registers of the CCD.

• Horizontal clocks (readout register/ serial clocks):
These are very high frequency clocks for performing horizontal (serial) pixel readout operation.

• Reset clocks (reset gating clock):
This reset clock is required for reset gating when charge is transferred from horizontal output register to output amplifier.

• Stage selection clocks:

This clock is used for selecting the stages for exposure (especially for TDI CCDs). Its frequency is same as image zone clocks.

• Storage area clocks :
This clock is required for the storage to transfer each line of pixels from image area to storage area. These clocks are specifically used for frame type architectures.

• Vertical sensor gate pulses:
The vertical sensor gate (VSG) pulses are used to transfer the pixel charges from the light-sensitive image area (i.e. photo diodes) into light- shielded vertical registers. These clocks are specifically used for interline based architectures.

4.2 CCD signal Processing Clocks

• Blanking and clamping pulses:
These are internal clamping blanking pulses which are required for clamping and blanking optical black region and also for horizontal vertical Masking.

• CDS clocks (Sampling clocks):
The reset noise from reset gate pulse will be neglected when CDS methodology is used. Two sampling clocks are required for sampling which are SHP and SHD to sample reference level and video (data) level respectively.

• Latching clocks:
These are also known as ADC clocks which are used to latch the output signal for post-processing and other operations.

4.3 Programmability features of Detector Clocks

Different CCD requires different types of clocks with different features. These features provide various types of control on CCD operation, image readouts, exposure, processing etc.. Based on survey, different features are discussed below.

• Rising/falling edge programmability
This feature provides rising /falling edge programmability control for any high frequency as well as low frequency clock within a pixel period as well as for multiples of pixel period.

• Polarity control for clocks
This feature provides the polarity control to any clock from high/low to low/high and vice-

versa. This feature is used for high as well as low frequency clock generation.

• Mode selection for grouping of clocks
This feature provides grouping of clocks by programming in various mode. We can group low and high frequency clocks in given mode for CCD readout.

• Blanking/ Masking/Clamping of pulses
This is very important feature in timing generator which is used in order to mask/blank/clamp the clock with in a line. These features are useful for both driving (for internal masking) as well as signal processing category of clocks.

• Toggle position control
Within a line, the different toggle positions of high as well as low frequency clocks can be programmed using this feature. The toggle control is also provided within masking region for high frequency clocks.

• Pattern groups
Within a line, pattern groups for low frequency clocks are generated by giving different start position with respect to reference pixel location.

• Repeat pulse/ sequences
Low frequency clocks and sequences can be repeated with in line by adding information of pattern group, pattern length of pixels with start position.

• Clock enabling control
Provides control on particular number of clock for given CCD detector.

• Variable frequency selection control
This feature provides the variable frequency selection control for high frequency clocks, for a pixel clock with particular frequency.

4.4 Targeted Specifications for Timing Generator

The following section shows the specifications for Timing Generator

1. Maximum Pixel rate: 20 MHz
2. Maximum CCD size supported up to 64K x 64K pixels.
3. 20 high frequency and 30 low frequency clocks.
4. Required memory configuration is 1024 x 16 bit.

5. Variable frequency selection for high frequency clocks.
6. 32 edge placement within pixel clock for coarse timing control of high frequency clocks.
7. Minimum programmable step size for high as well as low frequency clock edge is in sub nano seconds at 20MHz.
8. For vertical clocks with variable start position (with reference pixel location), pattern length and number of repeats within a line length selection, 4 pattern groups are required each having 32 patterns for start, pattern length and repeat.

4.5 Summary

In this chapter the design requirements and specifications are worked out. Also the details about different features are provided.

Chapter 5

Detailed Design

5.1 Top level architecture of Precision timing generator for CCD detectors

The overall system architecture is shown in the figure 5.1. The precision timing generator generates 20 high frequency and 30 low frequency clocks i.e. total 50 CCD output clocks required for various CCD driving and signal processing blocks.

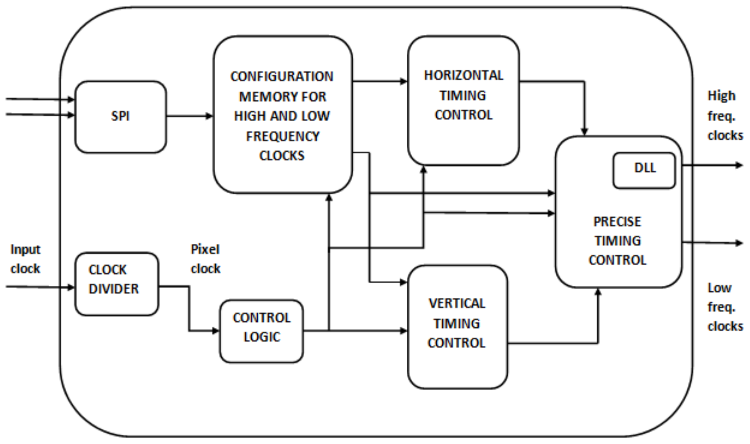


Figure 5.1: Top level architecture of Precision timing generator

The overall system design can be divided into various blocks as shown in the figure 5.1. The SPI is used to write various attributes of CCD clocks related information to the memory. The configuration memory consists of the information like rising edges, falling edges,enabling and polarity control, masking and toggle related information,etc. related to various CCD clocks. The clock divider is responsible for obtaining the pixel clock from input clock using appropriate divide logic (both even and odd). The control logic block decodes the clock related information from memory for other blocks. The horizontal and vertical timing control contains coarse timing generators which are responsible for coarser delay control of clocks ,where the delay control is in multiples of pixel clock period .Whereas a common precise timing control has a fine clock generator that is used to have a fine delay control (in fractions of pixel clock period) on high frequency and low frequency clocks. For the fine delay control a Delay Locked Loop is used to obtain precise delays in sub-nano seconds. These high and low frequency clocks also have features like programmable masking and toggle within masking regions. The low frequency clocks also has a feature of pattern group selection to generate vertical timing sequences with various start, length and repeat control.

The various blocks of precision timing generator are explained in detail in the following section.

5.2 Serial Peripheral Interface (SPI)

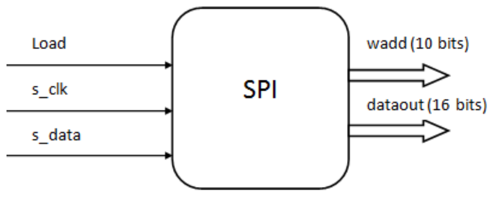


Figure 5.2: Block diagram of SPI

To configure the low frequency and high frequency clocks SPI is required for high speed serial writing operation of clocks information in configuration memory.

As shown in the block diagram of SPI (fig 5.2), whenever load signal goes low and at every rising edge of serial clock s_clk, data is written in memory serially. Here 1 bit overhead is used to differentiate between address and data. In this SPI, first a serial to parallel conversion is carried out which stores a 17 bit of data in a register. If the MSB is 0, the value stored in the parallel register is address (wadd) and if MSB is 1 the value stored is data (dataout). The timing diagram for SPI is shown in the figure 5.3.

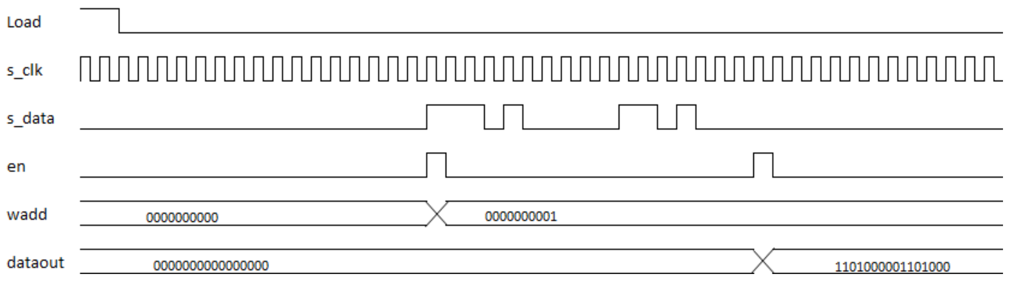


Figure 5.3: Timing diagram of SPI

5.3 Configuration Memory for high frequency and low frequency clocks.

Figure 5.4 shows the block diagram of configuration memory. It has a RAM and an initialization sequencer. There are 5 inputs to this memory block: Register data in i.e. a data bus which is 16 bits wide, address bus which is 10 bits wide, reset,clock and load. The size of this memory is 1K x 16 bits and hence can have 1024 address locations.

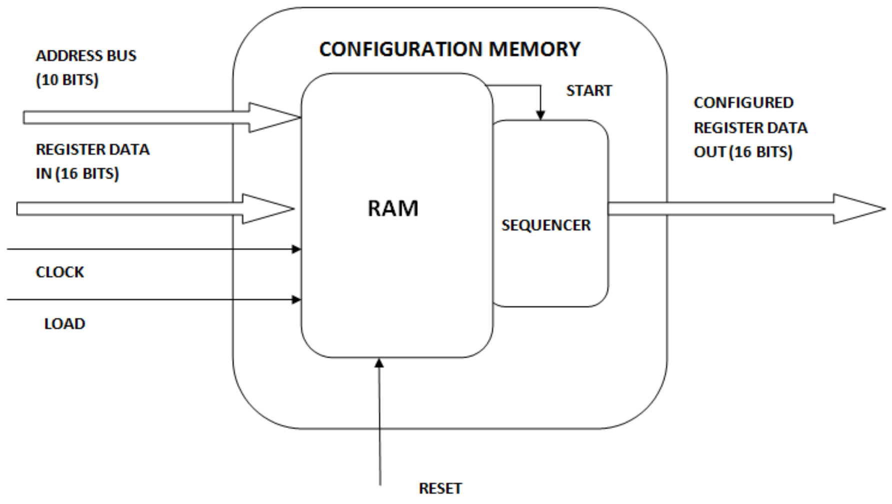


Figure 5.4: Block diagram of configuration memory

Whenever the reset goes high, first the memory write operation takes place. After all the information has been written in the RAM, lastly a fixed address location is written which initiates the sequencer as the start signal is asserted and memory read operation takes place in to the various configured data out registers one by one.

As per proposed architecture, for generating different types of clocks the information of these clocks are required to be written in this memory. Details regarding register like register type, purpose, description of that register, number of bits (memory requirements, data bits position in location, memory location requirements, memory word estimation etc are described here. All details are required to be written in memory at given location and bit position.

5.4 Clock divider

The clock divider is designed to provide flexibility in the design as the detector can be read out with different frequency.

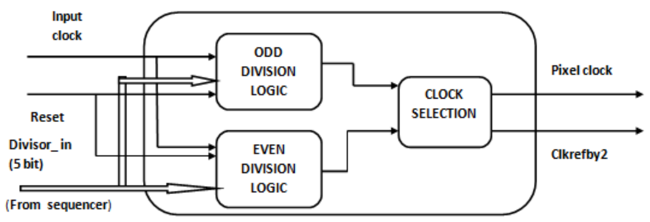


Figure 5.5: Block diagram of clock divider

As shown in the figure 5.5, whenever active high reset is asserted the input clock is divided by the appropriate value to get the pixel clock with 50 % duty cycle and a divide-by-2 reference clock of pixel clock (clkrefby2). Divisor in obtained from memory output which is 5 bit wide and support maximum 20 division.The divider logic is split in two parts : odd division logic and even division logic. The input clock is divided using any one of the two described above based on the divisor value. The pixel clock is used as master clock for the rest of the blocks. Figure 5.6 shows the waveform for clock divider.

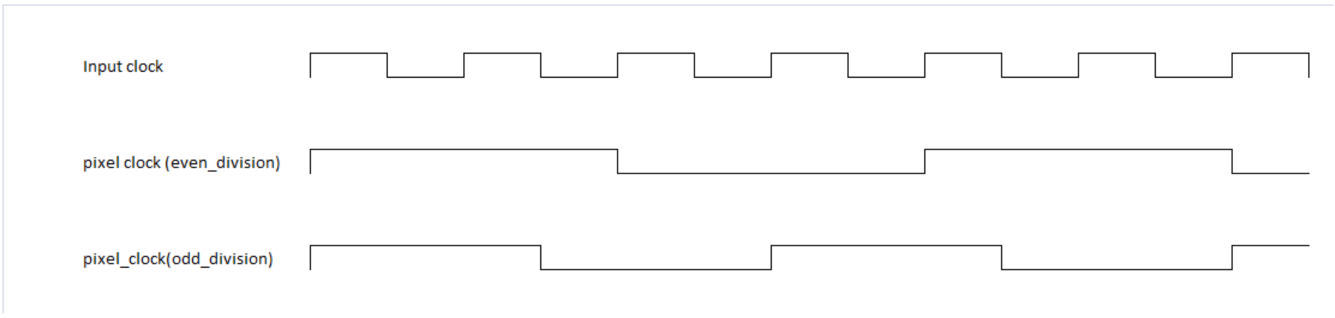


Figure 5.6: Timing diagram for clock divider

5.5 Precision timing control

Precision timing control is one of the most important block to obtain very precise delays and to improve the resolution of clocks generated coarsely. DLL with counter controlled technique is used to generate precise delay control.

Referring to the block diagram of DLL as shown in figure 5.7,active high power on reset (which is the primary input of design) is used. Synchronous reset logic is used to avoid meta stability issue. When Reset signal goes high,Phase Detector compares the phase of input clock and feedback clock,which is the output of Mux based delay line, and generates up signal. The counter based controller operates at clkrefby2 (which is input clock divide by 2) and generates the count value based on the phase difference between input clock and feedback clock. The control value obtained from the controller is used to select the tap needed to align the input clock and feedback clock at the same phase and the loop is locked.

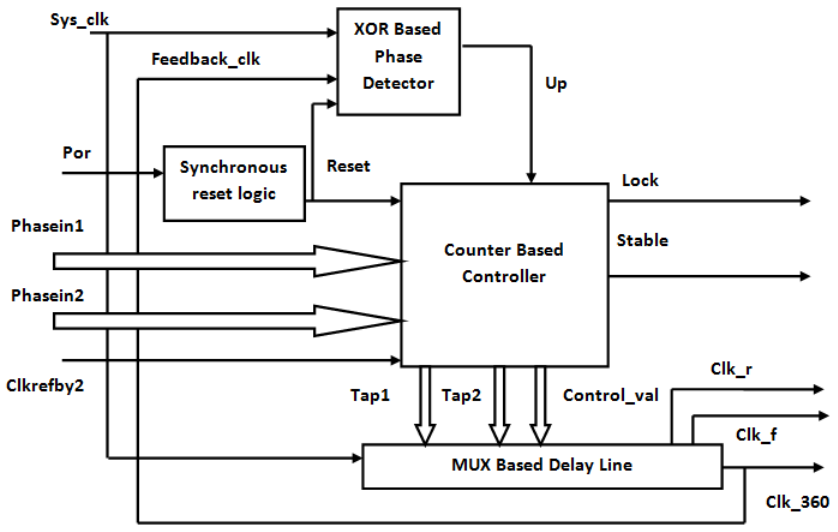


Figure 5.7: Block Diagram of DLL

After the loop is locked,based on the phases given (Phasein1 and Phasein2) the corresponding tap values are obtained(tap1 and tap2) from controller such that the output clock will have required phase shift compared to the input clock.

Therefore the controller first shifts the reference clock by 360° .Now for achieving the desired phase shift of PS° the controller selects the tap by given equation:

$$\frac{PS^\circ \times N}{360^\circ}$$

Where N is the number of taps after lock is achieved resulting in 360° phase shift between reference clock and clock output from the delay line.

The mux based delay line has a mux followed by a delay buffer as its unit delay element.This lock signal of DLL is used as a delayed start signal for rest of the blocks of timing generator. Hence all other blocks (except memory and clock divider) functions only after the DLL has been locked. The three main blocks of a DLL are 1. Phase Detector 2. Counter based Controller 3. Mux based delay line.

5.5.1 Phase detector

Referring to the block diagram of Phase Detector (PD) in figure 5.8, here an XOR based PD is used. The input reference clock and feedback clock (which is the output of the delay line) are XORed in order to generate a control signal ,up xor which is used to start the counter in the controller block.

Finally the control signal ,up, is obtained at every rising edge of the reference clock so as to avoid the metastability issues (setup and hold time violations).

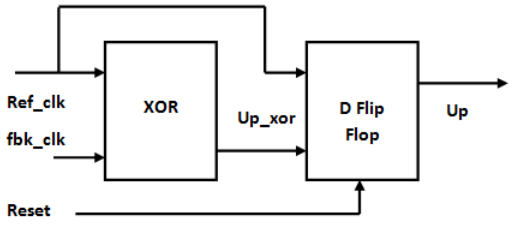


Figure 5.8: Block diagram of Phase detector

As shown in the figure 5.9 XOR gate compares the phases of two clocks and outputs the up xor signal. At every rising edge of the reference clock , up signal is obtained. This signal remains low up to the phase difference between these clocks is less than 180. Whenever the feedback clock goes 180°out of phase to reference clock, up signal goes high and again becomes low when the two clock phases are aligned.

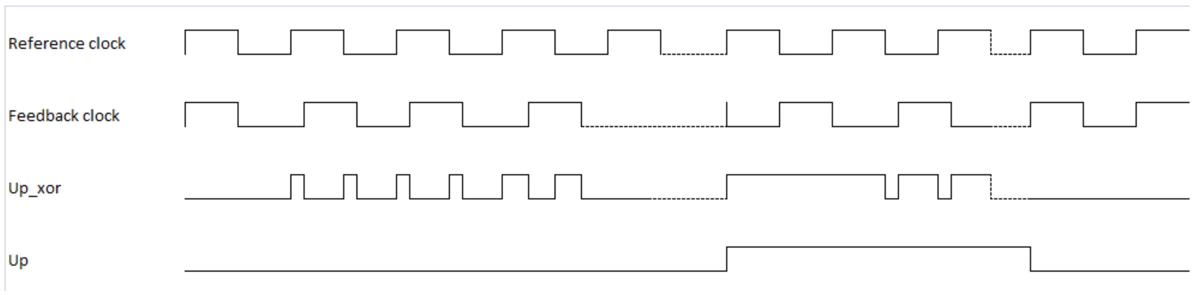


Figure 5.9: Timing diagran of Phase detector

5.5.2 Counter based Controller

The block diagram of the counter based controller is shown in the figure 5.10. This controller operates at clkrefby2 (divide by 2 clock of pixel clock) to slow down the counter operation. As a 96 tap delay line is used, a 7 bit (96 modulo) counter is needed to calculate the control value. Whenever an active high reset is asserted the counter increments ,as per the up control signal, until the phases of reference and feedback clock aligns and then becomes stable at a particular value. Here the counter is initialized to a fix value (rather than starting it form 0) to avoid false locking. Here a decoder logic is used to obtain one hot encoding corresponding to control val , which is used as a selection control for delay line.

After the DLL is locked, the required number of taps for the desired phase shift (phase value) is calculated using a multiplier and a subtraction logic. In order to assure that multiplier starts only after the DLL is locked, a multiplier pipeline delay is added using a chain of Flip flops. The stable signal goes high after all the required tap values have been calculated and finally the tap value and the control value obtained are fed to the delay line.

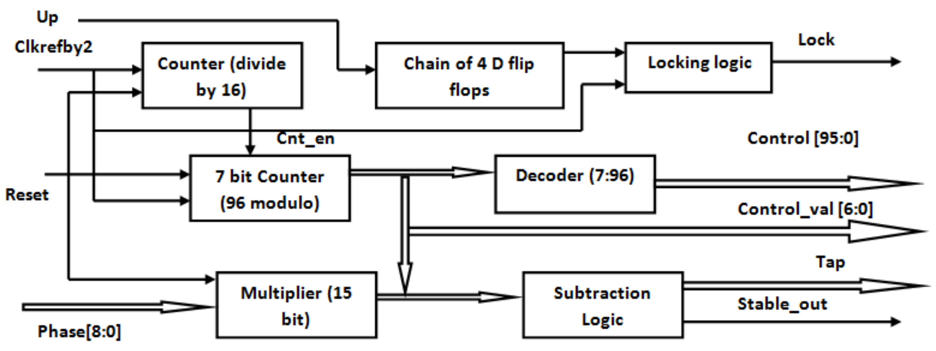


Figure 5.10: Block diagram of Counter based controller

5.5.3 MUX based delay line

As shown in the figure 5.11 a unit delay element comprises of a Mux followed by a pair of NAND gates. The design of this delay line is the most critical part in the entire DLL design. To achieve delay resolution of approximately 620 ps, a 96 tap delay line is used for the clock frequency of 20 MHz. The number of Mux selected depends on the control value. Whenever the select signal is high the input clock passes as it is without any delay and whenever it goes low, the input clock is delayed by a unit delay element value. Clock tap can be taken out from anywhere in the delay line. (clk1 to clk96)

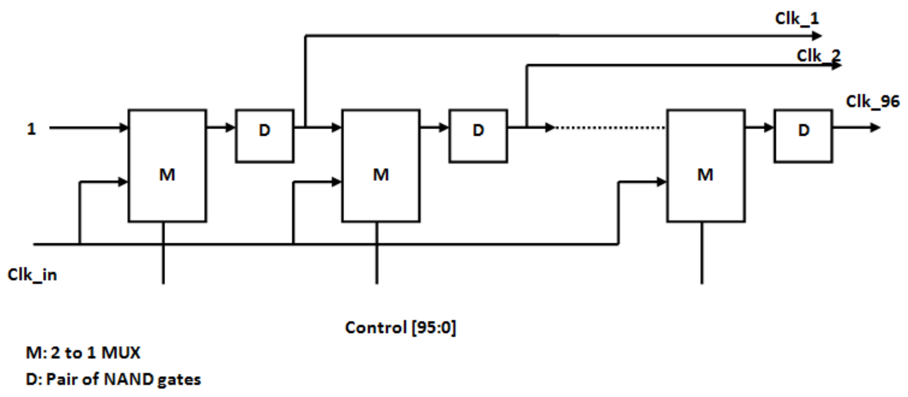


Figure 5.11: Block diagram of MUX based delay line

5.5.4 Fine clock pulse generator

By using DLL, precise delays of different clocks are obtained. Pulse generator is needed to obtain the variable pulse widths of the input pulses obtained from the horizontal and vertical clock generators. The block diagram of the fine clock pulse generator is shown in the figure 5.12. The core block of this logic is DLL. The values corresponding to fine rise and fall edge

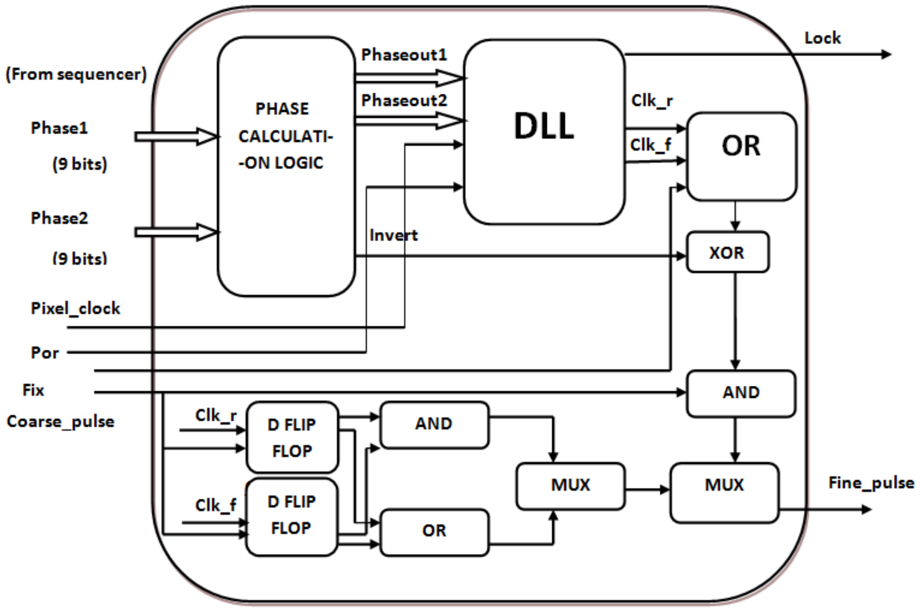


Figure 5.12: Block diagram of fine clock pulse generator

programmability are fed to this block from memory. After the DLL gets locked and delayed start signal goes high the phase calculation logic consisting of multiplier and subtraction logic calculates the phase values corresponding to these edges. DLL gives two output clocks Clkr and Clkf for rising and falling edges values. Using these clocks the input coarse pulse is latched, followed by some combinational logic, final output fine pulse is obtained.

For a 96 tap delay line the fine rise and fall edge values can be between (0, 95). The coarse pulse latched with Clkr is clockr and the one with Clkf is clockf. The phase inputs given to DLL are phaseout1 and phaseout2 respectively.

If pulse width of input coarse pulse is greater than a pixel clock period then fix signal remains at logic low otherwise at logic high..

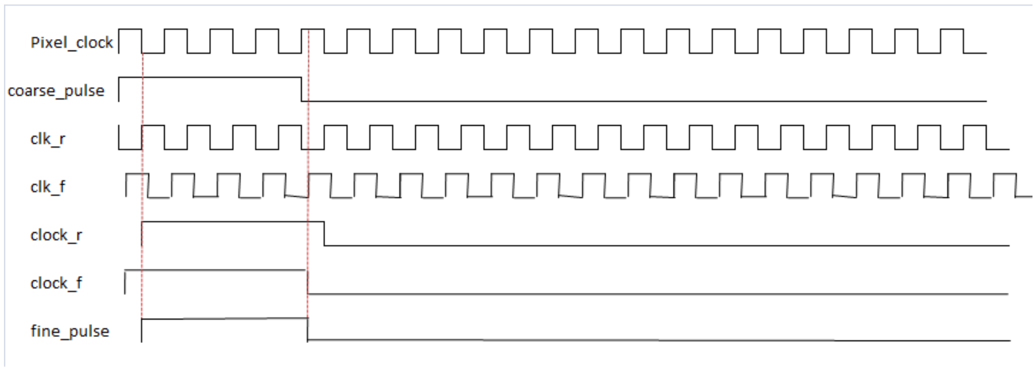


Figure 5.13: Timing diagram for case 1

The concept used here is that, in order to obtain shorter pulse widths, first wider pulse

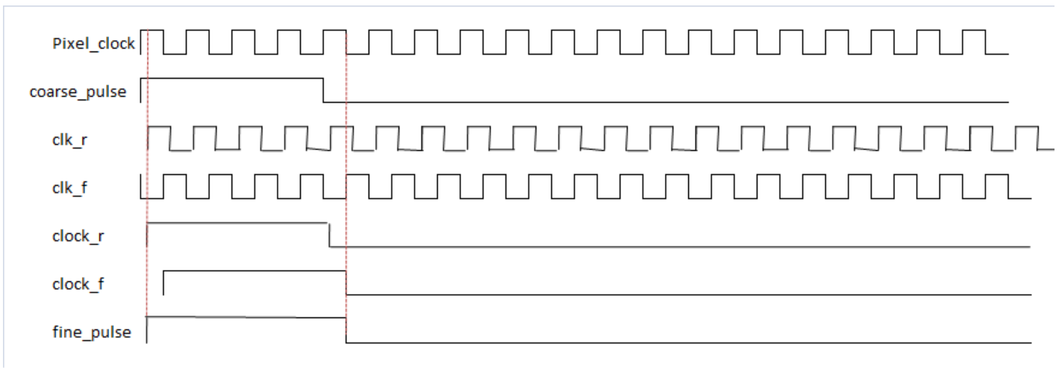


Figure 5.14: Timing diagram for case 2

widths are obtained and then these are inverted to get the final output pulse.The timing diagram for obtaining the pulse width greater than and and less than original pulse width are shown in the figure 5.11 to figure 5.14.

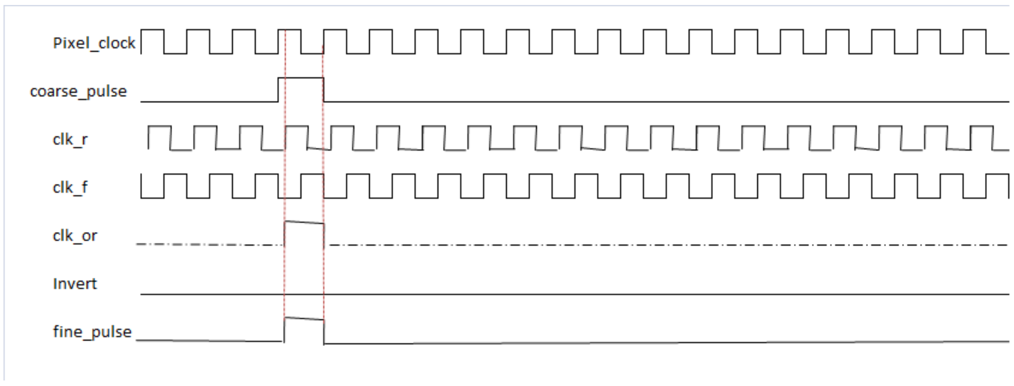


Figure 5.15: Timing diagram for case 3

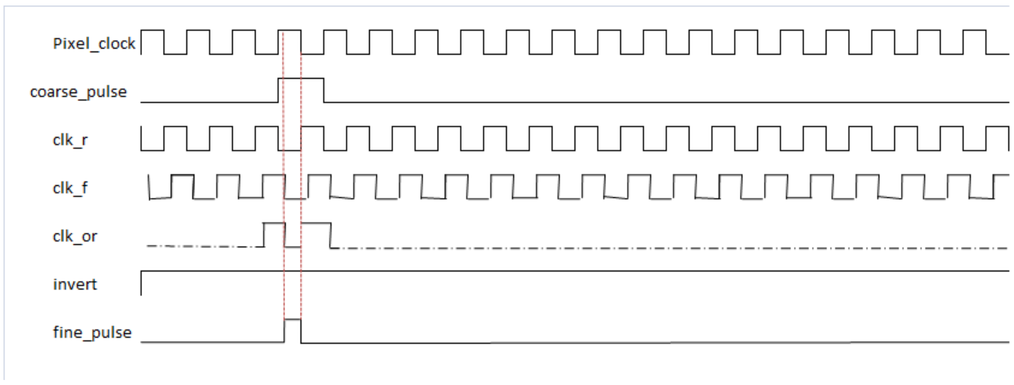


Figure 5.16: Timing diagram for case 4

5.6 Horizontal timing control

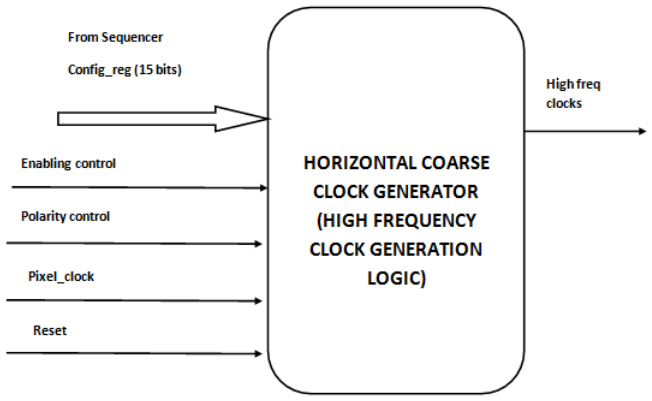


Figure 5.17: Block diagram of horizontal coarse clock generator

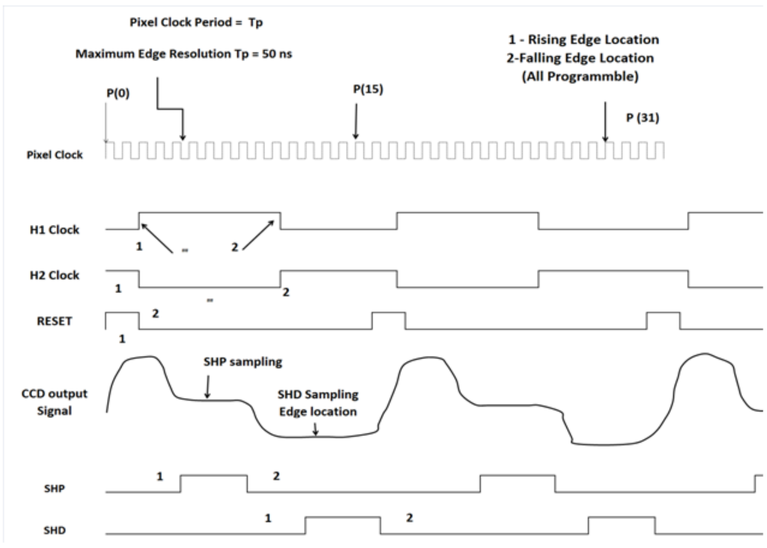


Figure 5.18: Timing diagram for high frequency clock

Horizontal timing control block generates 20 horizontal high frequency clocks which include read out clocks, sampling clocks like SHP and SHD, latching clocks like ADC, reset clocks, etc.

The block diagram is shown in figure 5.17. This block is responsible for generating coarser delays (where delay value is in multiple of pixel clock period). The config reg from memory contains the information regarding coarse rising edge and falling edge value along with the period information. These coarse rise and fall edge values can be between (0,31) (32 edges) which provides the resolution of 50 ns for pixel clock of 20 MHz. The period value is to vary the frequency of these horizontal clocks which division of pixel clock by integer value , period value can be between 0 to 31 (5 bits). Here enabling and polarity control is also provided for each clock. The timing diagram for high frequency clocks is shown in figure 5.18.

Information stored in configuration register is decoded by FSM which provides edge delay control for configured bits which contains rising and falling edge information with polarity control . The delayed start signal used here is the lock signal of DLL, thus reset is not of delayed start. Here a 5 bit counter is used to count up to the maximum value of 31. Hclock is the high frequency output clock.

5.7 Vertical Timing Control

Vertical timing control block generates 30 low frequency clocks which include exposure clocks, blanking and clamping clock pulses, storage and stage selection clocks. Apart from these line and frame synchronization pulses are also included in this category.

The block diagram is shown in figure 5.19. This block is responsible for generating coarser delays for vertical clocks (where delay value is in multiple of pixel clock period). The configreg (16 bits) from memory contains the information regarding coarse rising edge and falling edge value for vertical clocks. These coarse rise and fall edge values can be between (0, 65536) which provides the resolution of 50 ns for pixel clock of 20 MHz. Here enabling and polarity control is also provided for each clock.

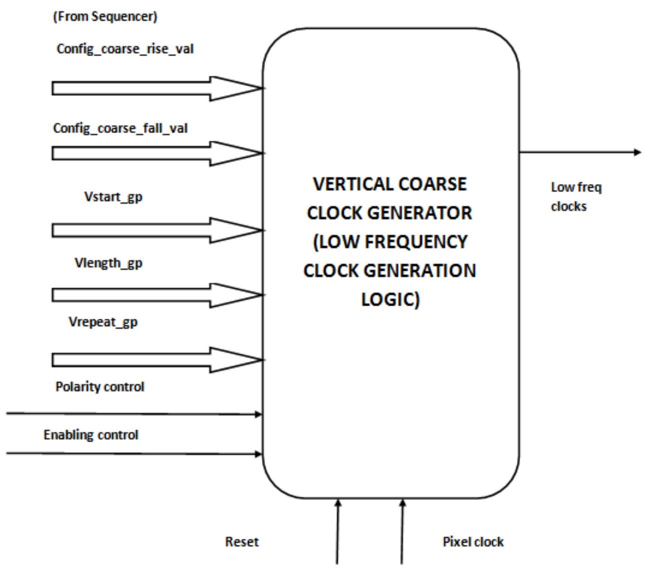


Figure 5.19: Block diagram of Vertical coarse clock generator

Apart from these, one more feature is provided in vertical clocks which is the pattern group and sequence selection for vertical pattern start, length and repeat information. Pattern group consists of coarsely generated vertical clock with start position within a line whereas pattern sequence consists of pattern group with number of repeats within a line. Start value gives information regarding the offset value for different vertical clocks. Length value gives

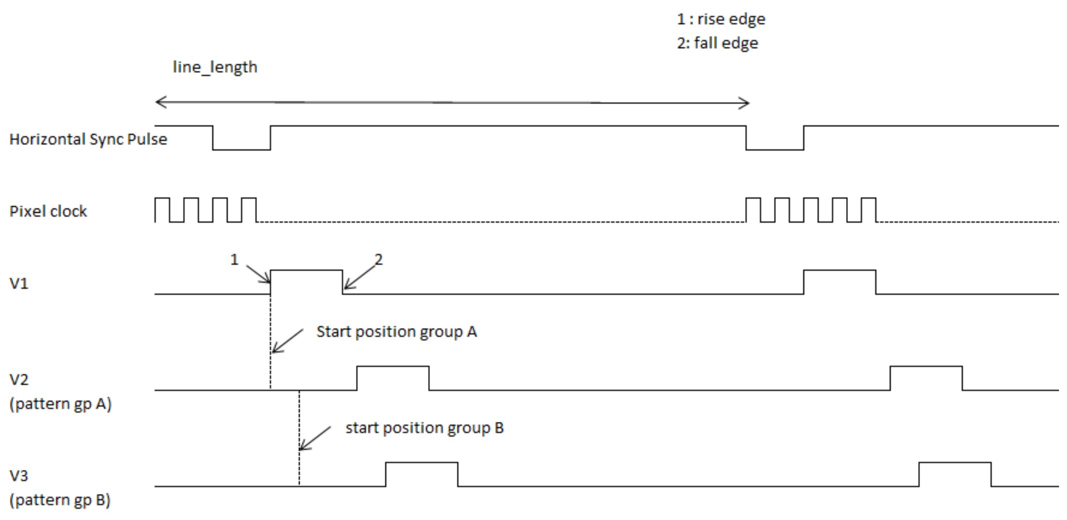


Figure 5.20: timing diagram for vertical pattern group

information regarding the vertical pattern length and repeat value determines the number of times vertical sequence to be repeated within a line. By default the number of repeats is 1 and so the pattern length is the line length. All these information about pattern groups is obtained from memory. The registers corresponding to these information are 16 bits wide.

There are 4 pattern groups provided in this design : Group A ,B,C and D.Each of these groups have 32 patterns for start, 32 patterns for length and 32 patterns for repeat Thus each vertical clock can be configured with any of these 32 patterns for start, length and repeats.The timing diagram for vertical pattern group and vertical pattern sequence generation is shown in figure 5.20 and figure 5.21

Information stored in configuration register is decoded by FSM which provides edge delay control for configured bits which contains rising and falling edge information with polarity control. The delayed start signal used here is the lock signal of DLL, thus reset is not of delayed start. Here a 16 bit counter is used to count up to the maximum value line length. Also there are counter used for start position and number of repeats. Vclock is the low frequency output clock.

5.8 Line and Frame synchronizing pulse generation block

Image area registers in configuration memory in Timing Generator gives the information regarding line size and frame size for a given CCD detector. Figure 5.23 shows the timing diagram for line and frame size i.e. horizontal and vertical counter respectively. These pulses are used for indicating the information regarding line and pixel numbers for a given

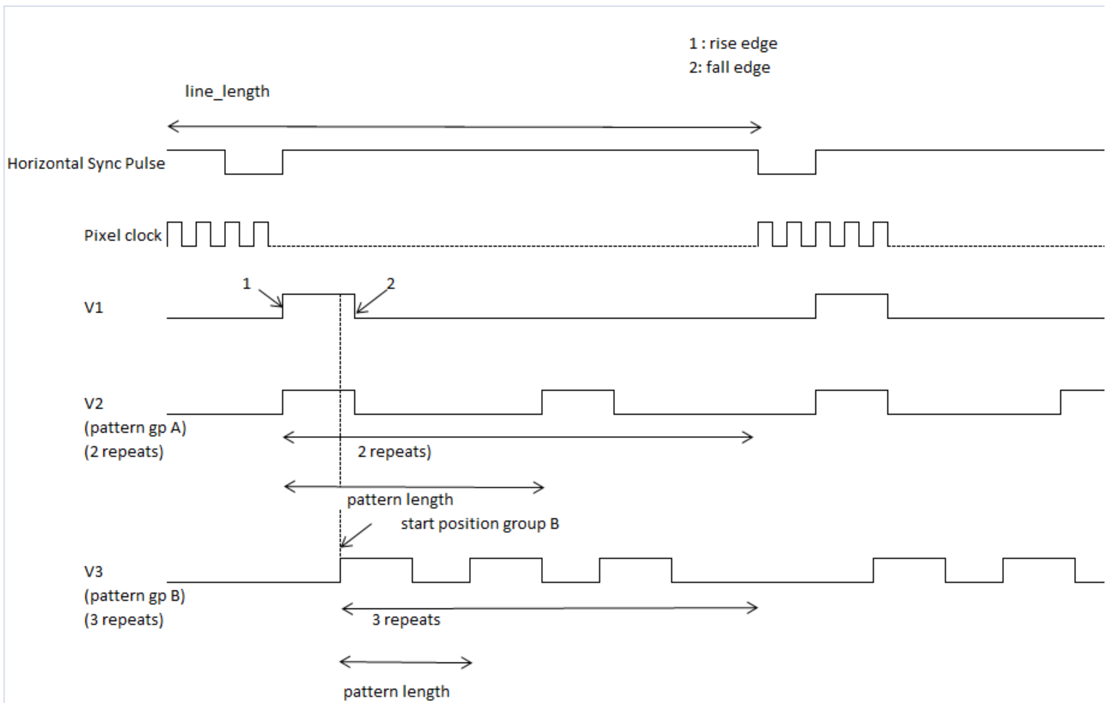


Figure 5.21: timing diagram for vertical pattern group

CCD.

Block diagram for line and frame synchronizing pulses is shown in Figure 5.22, in which image area registers of configuration memory having information regarding line size, frame size, rising edge pulse polarity, enabling control etc. CCD size is supported upto 64K x 64 K. By default, HD and VD both are active low pulse with default falling at pixel location 0.

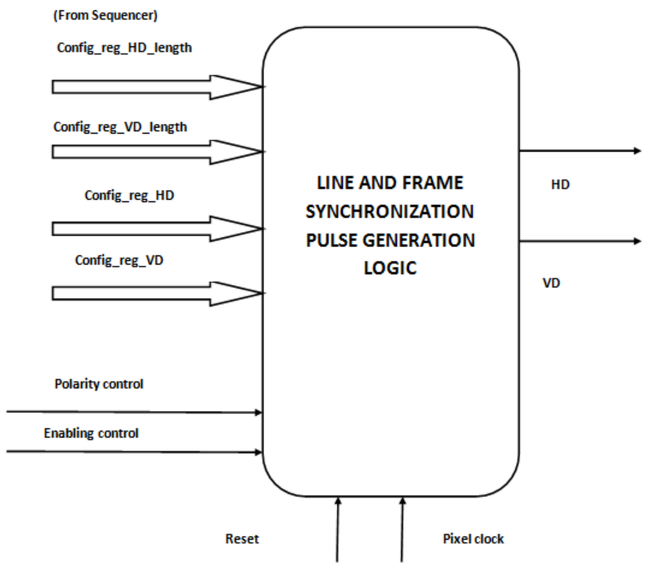


Figure 5.22: Block diagram of line frame Synchronization pulse

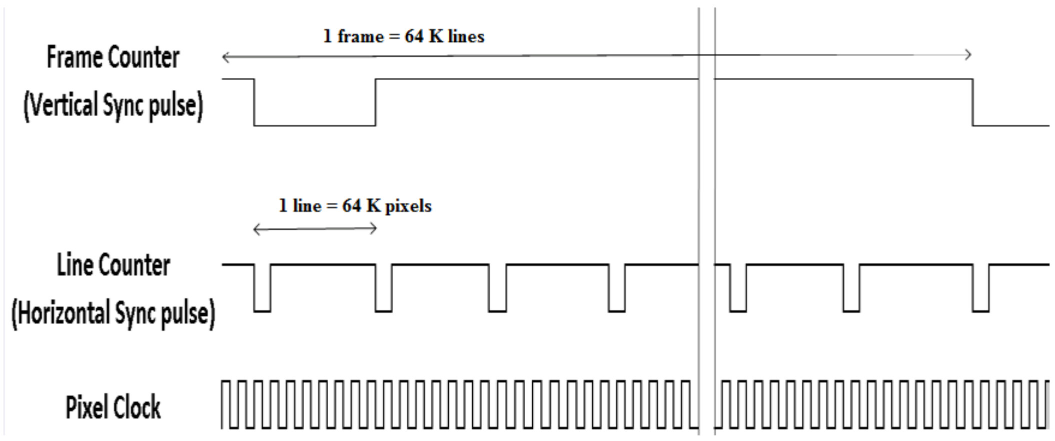

1 frame = 64 K lines
Frame Counter (Vertical Sync pulse)
1 line = 64 K pixels
Line Counter (Horizontal Sync pulse)
Pixel Clock

Figure 5.23: Timing diagram of line frame Synchronization pulse

5.9 Exposure clock generation block

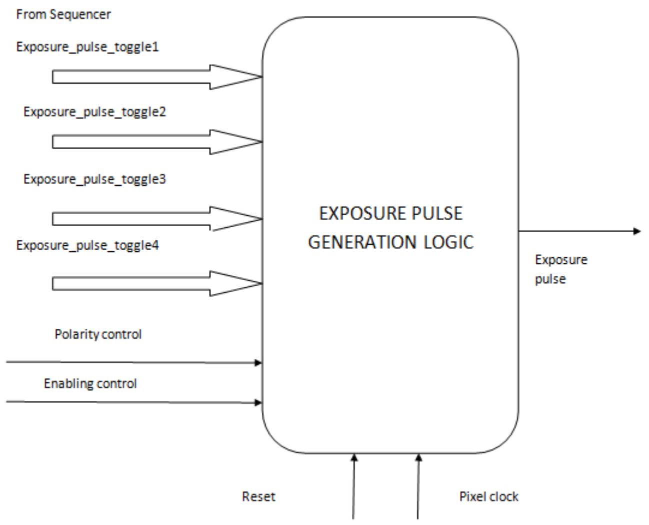


Figure 5.24: Block diagram of exposure clock generation

Timing generator generates low frequency clock pulses for shuttering operation. Exposure clocks provide shuttering time in which light incidents, i.e. exposure of photon on CCD's effective image area. When exposure operation gets over, line transfer operation will be initialized.Figure 5.24 shows the block diagram of exposure clock generation logic.
Figure 5.25 shows the timing diagram for exposure clocks. Fast shuttering can be provided by High precision exposure clock while low / normal shuttering can be provided by Low speed/ Normal exposure clocks.Exposure clock can support upto 4 toggle positions. User can programme the toggle position in configuration memory based on shuttering operation requirements.

Exposure clock generation logic has counter based FSM which works on pixel clock .Counter counts upto the line length.

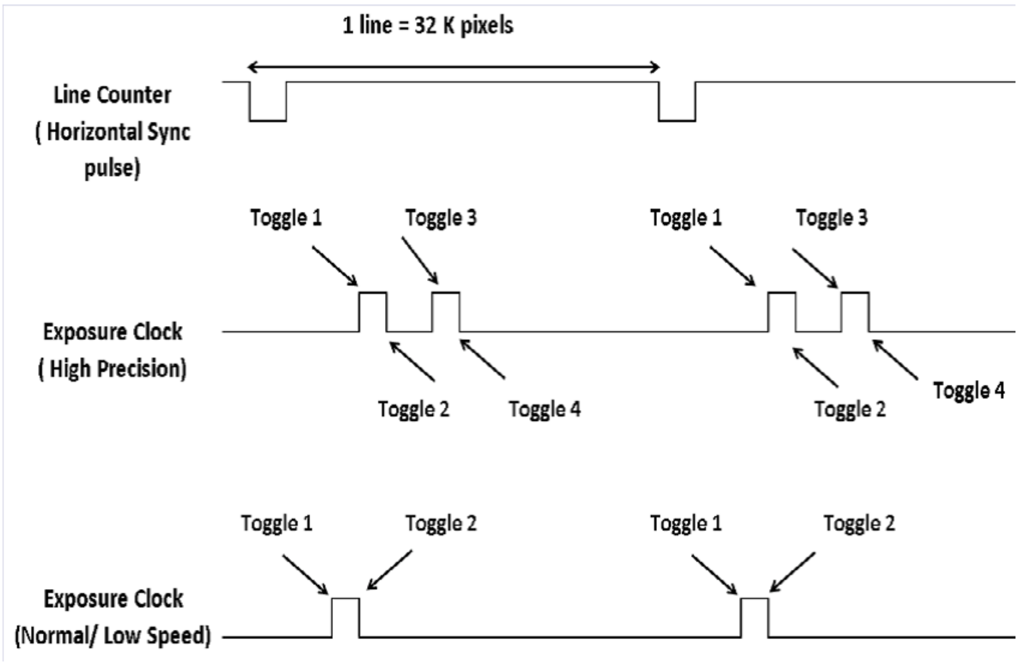


Figure 5.25: Timing diagram for exposure clock generation

5.10 Horizontal and Vertical Masking block

During CCD readout, there is need to mask some pixels and lines. Apart from that some driving clocks also requires masking in different readout modes. This block contains design for 3 different pulses i.e. horizontal/vertical masking, pre-blanking pulse and clamping optical black pulse. All these three pulse are required in order to mask horizontal/vertical clocks, pre-blanking region and optical black region of CCD detector respectively. The timing diagram shown in Figure 5.26 indicates all about horizontal blanking and clamping optical black region.

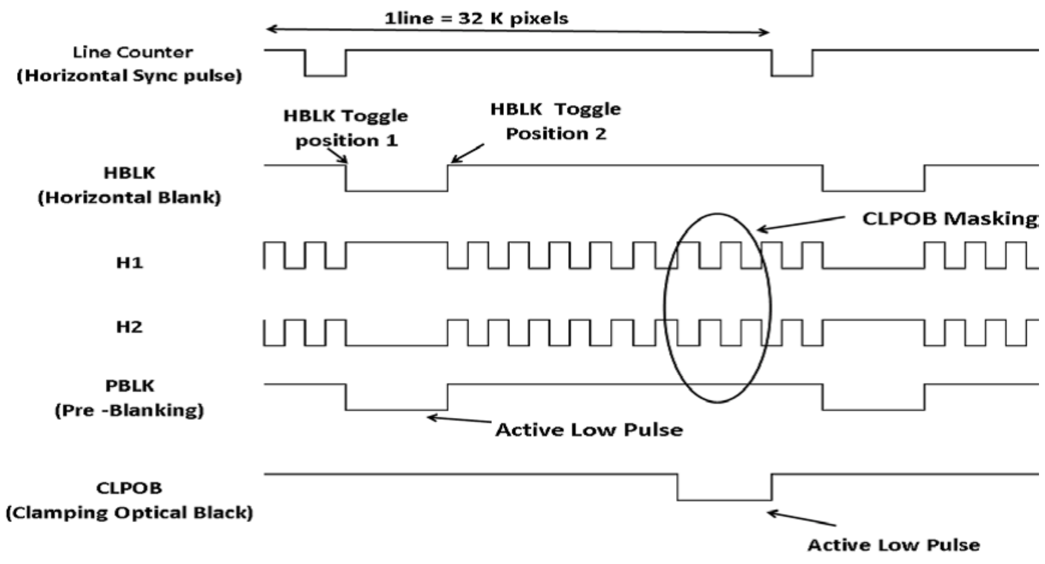


Figure 5.26: Timing diagram for Masking

The block diagram for masking signal generation is shown in figure 5.27. It supports up to 6 toggles. It is a counter based FSM working on pixel clock.

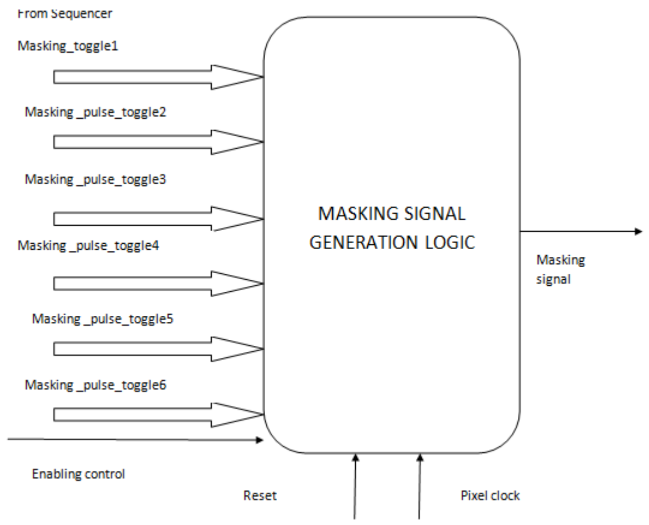


Figure 5.27: Block diagram of Masking signal generation

5.10.1 Masking Toggle generation logic

During Masking operation, when masking region is detected sometimes it requires the state change from 0 to 1 or vice-versa. This is known as masking time toggle generation. This feature is required special readout mode for specific detector like TDI architecture. In this design this feature is implemented for high frequency which requires state change during masking region. This logic works on pixel clocks which supports single toggle position within a line length.

The block diagram and timing diagram for masking toggle generation is shown in the figure 5.28 and 5.29

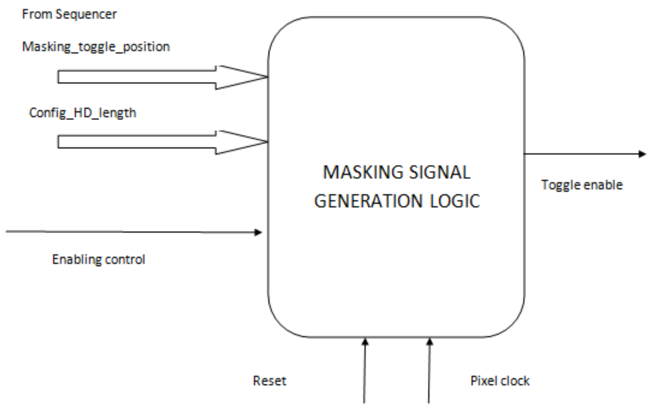


Figure 5.28: Block diagram of Masking toggle generation

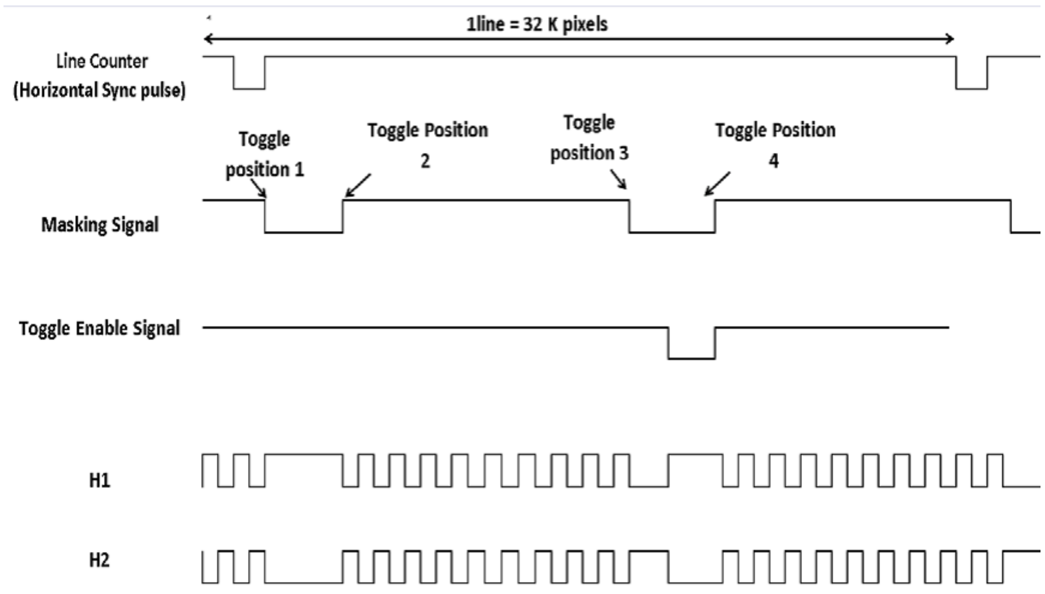


Figure 5.29: Timing diagram of masking toggle enable

5.11 Summary

This chapter summarizes about the detailed block level design of the timing generator. Each block has been described in detail along with its timing diagrams.

Chapter 6

Simulation And Synthesis Results

6.1 Block level STA results

The RTL coding of the design is done using VHDL language and synthesis, static timing analysis and simulation is carried out using Synopsys tool. The design is synthesized in 180 nm technology. The synthesis and STA results at 20 MHz for various design blocks are tabulated below:

Table 6.1 Operating conditions

Operating Condition	**Process**	**Voltage (V)**	**Temperature**
Min	0.8	1.98	-40
Typ	1	1.8	25
Max	1.2	1.62	125

6.1.1 Horizontal coarse clock generator

Table 6.2 STA results of Horizontal coarse clock generator

Worst Slack(ns)	**Min**	**Typ**	**Max**
Setup	13.41	12.52	11.41
Hold	0.34	0.56	0.78

Area: 438.54 units (1 unit = 4 nand gates)

6.1.2 Horizontal and vertical fine clock generator

Table 6.3 STA results of Horizontal and vertical fine clock generator

Worst Slack(ns)	**Min**	**Typ**	**Max**
Setup	4.37	3.01	1.77
Hold	0.24	0.27	0.29

Area: 6946.58 units (1 unit = 4 nand gates)

6.1.3 Horizontal and Vertical blanking

Table 6.4 STA results of Horizontal and Vertical blanking

Worst Slack(ns)	Min	Typ	Max
Setup	13.99	12.58	10.98
Hold	0.5	0.78	1.12

Area: 814.18 units (1 unit = 4 nand gates)

6.1.4 Vertical coarse clock generator

Table 6.5 STA results of Vertical coarse clock generator

Worst Slack(ns)	Min	Typ	Max
Setup	6.56	5.87	5.25
Hold	0.49	0.61	0.7

Area: 1344.50 units (1 unit = 4 nand gates)

6.1.5 DLL

Static timing Analysis was carried out to ascertain the operating frequency range of DLL. Table 6.6 shows that maximum frequency is 1.06 GHz for Best PVT corner and 431MHz for worst PVT corner; considering minimum 4 phases are obtained from DLL. Minimum frequency is 41.61 MHz for Best PVT corner and 16.96 MHz for worst PVT corner

Table 6.6 STA results of DLL

Delay Type	Min Frequency	Max Frequency (with 4 phases)
Max	16.96MHz	431MHz
Typ	23.20MHz	588.23MHz
Min	41.61MHz	1.06GHz

Average delay of a delay cell is approximately 620 ps. Figure 6.1 shows the linearity of delay line.

6.1.6 Line and frame synchronization pulse generator

Table 6.7 STA results of Line and frame synchronization pulse generator

Worst Slack(ns)	Min	Typ	Max
Setup	12.57	11.68	10.68
Hold	0.39	0.43	0.54

Area:754.61 units (1 unit = 4 nand gates)

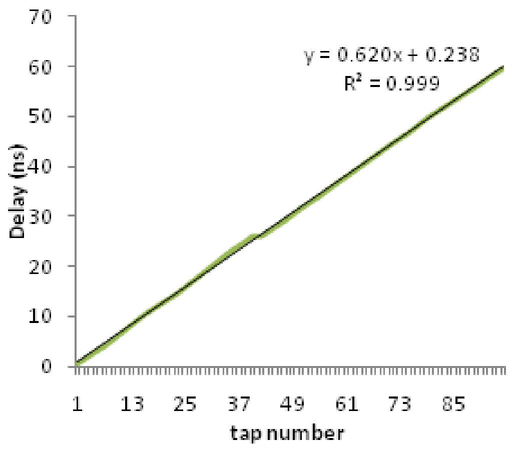


Figure 6.1: linearity of delay line

6.1.7 Exposure clock generator

Table 6.8 STA results of Exposure clock generator

Worst Slack(ns)	**Min**	**Typ**	**Max**
Setup	16.29	14.65	12.08
Hold	0.66	0.87	1.04

Area:622.55 units (1 unit = 4 nand gates)

6.2 Simulation Results

The post synthesis simulation results for different CCD clocks are shown in the following figures:

6.2.1 DLL

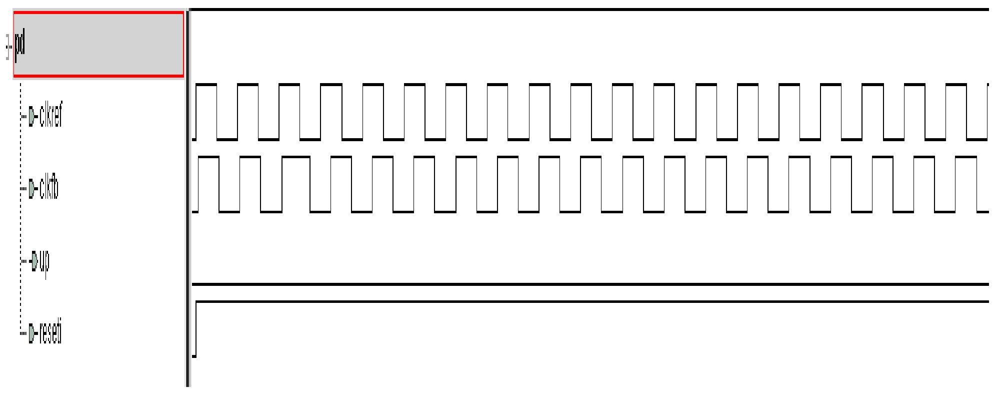


Figure 6.2: Phase detector when phase difference less than 180°

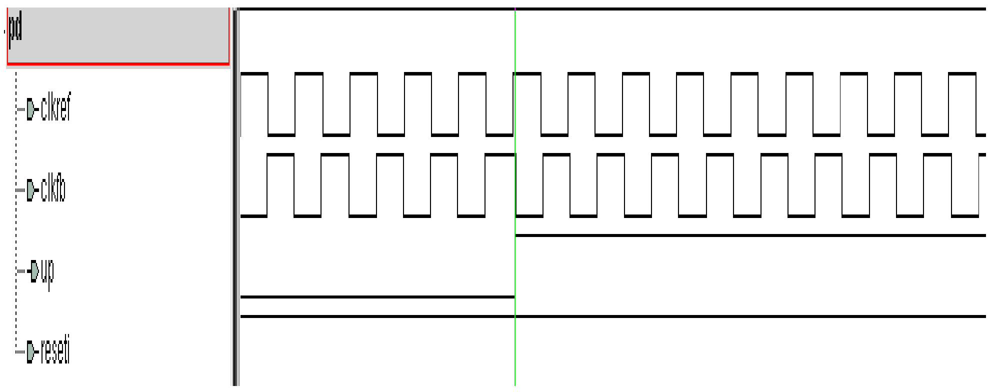


Figure 6.3: Phase detector when phase difference equal to 180°

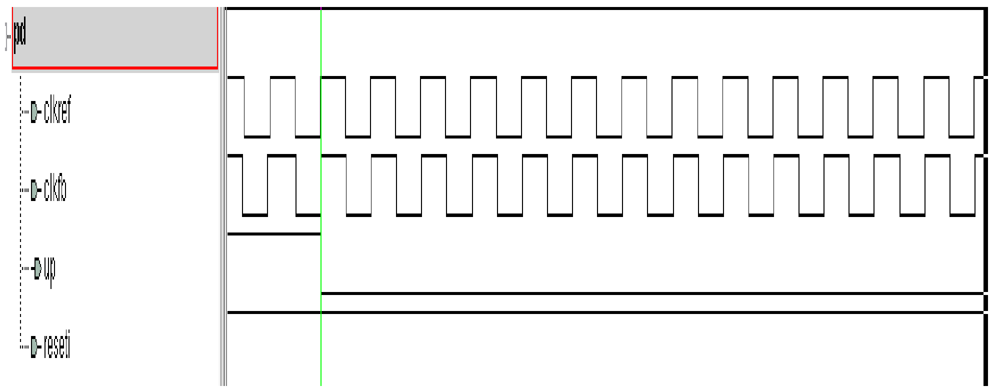


Figure 6.4: Phase detector when phase difference equal to 360°

The figure 6.5 shows the clock output obtained ,as per the input phase,after DLL gets locked.

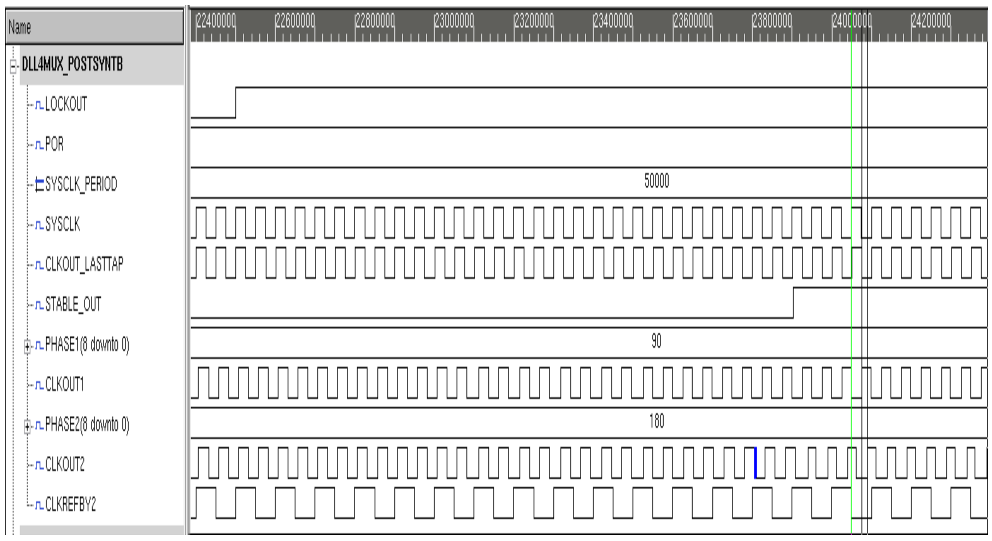


Figure 6.5: DLL

6.2.2 Fine clock generator

The simulation results for fine pulse generation with variable pulse widths are shown below:

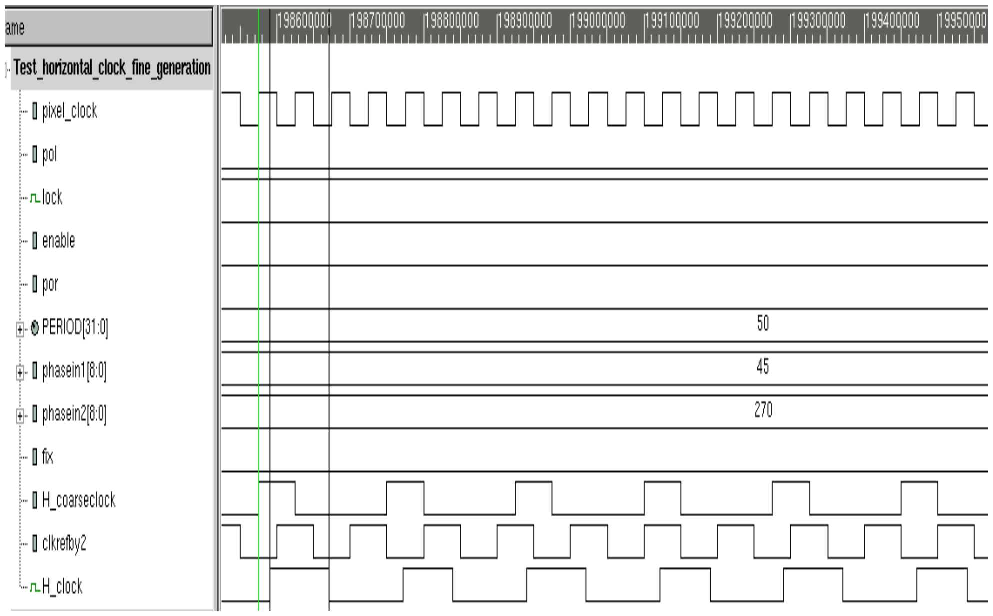


Figure 6.6: Clock pulse with pulse width greater than pixel clock period and phase2 - phase1 $> 180°$

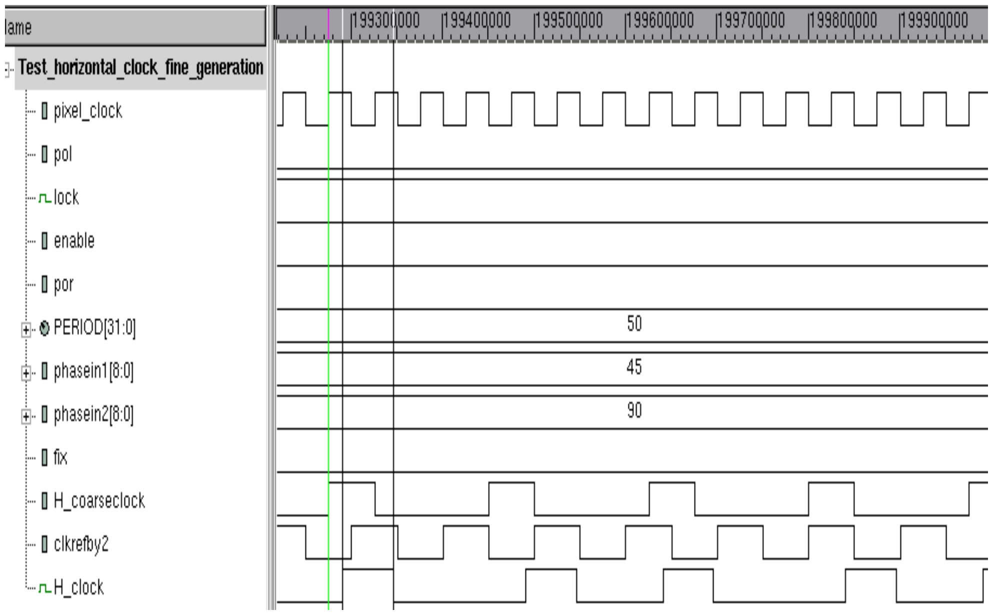


Figure 6.7: Clock pulse with pulse width greater than pixel clock period and phase2 - phase1 $< 180°$

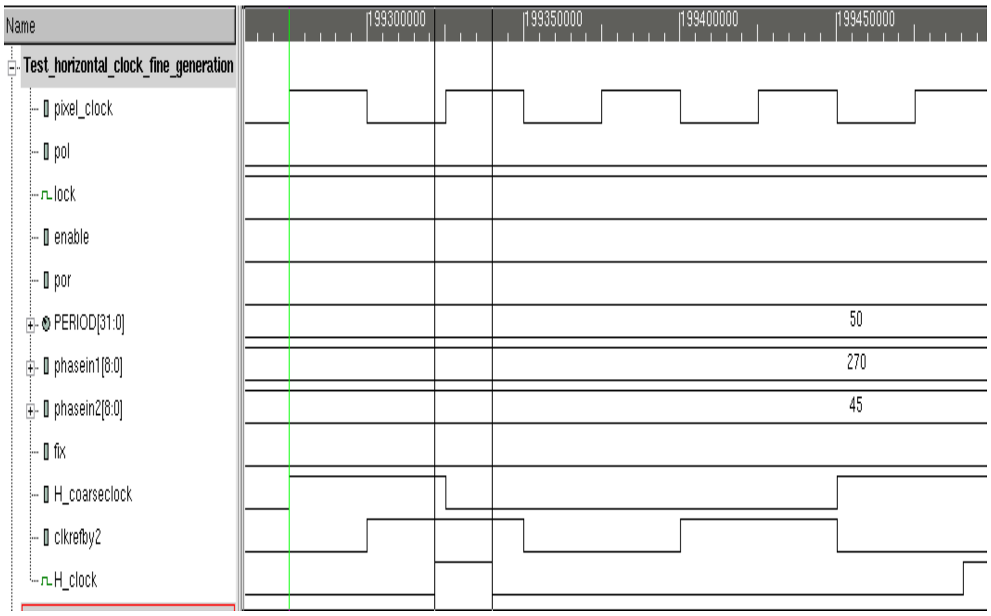


Figure 6.8: Clock pulse with pulse width greater than pixel clock period and phase1 - phase2 $> 180°$

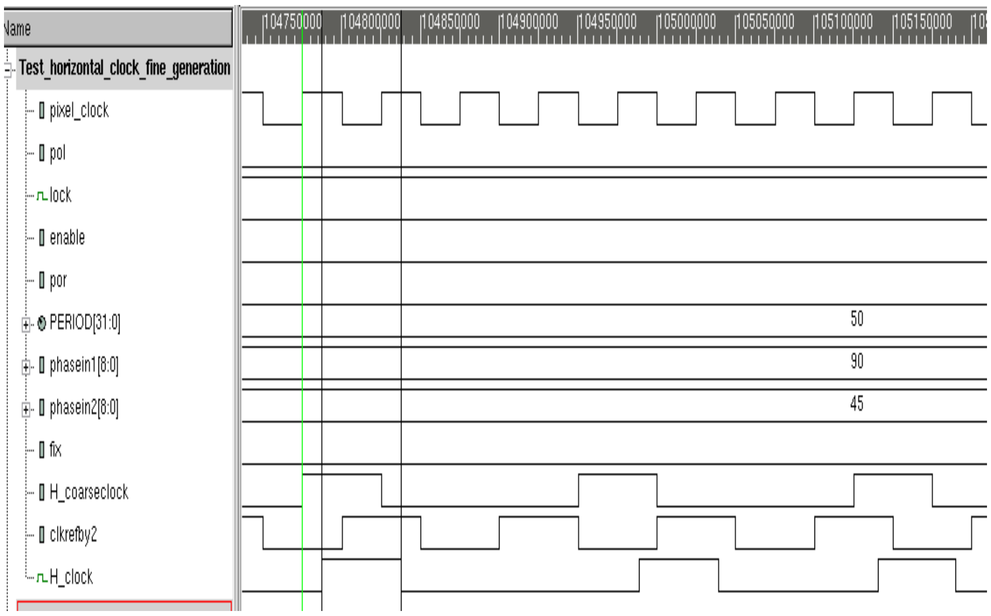


Figure 6.9: Clock pulse with pulse width greater than pixel clock period and phase1 - phase2 $< 180°$

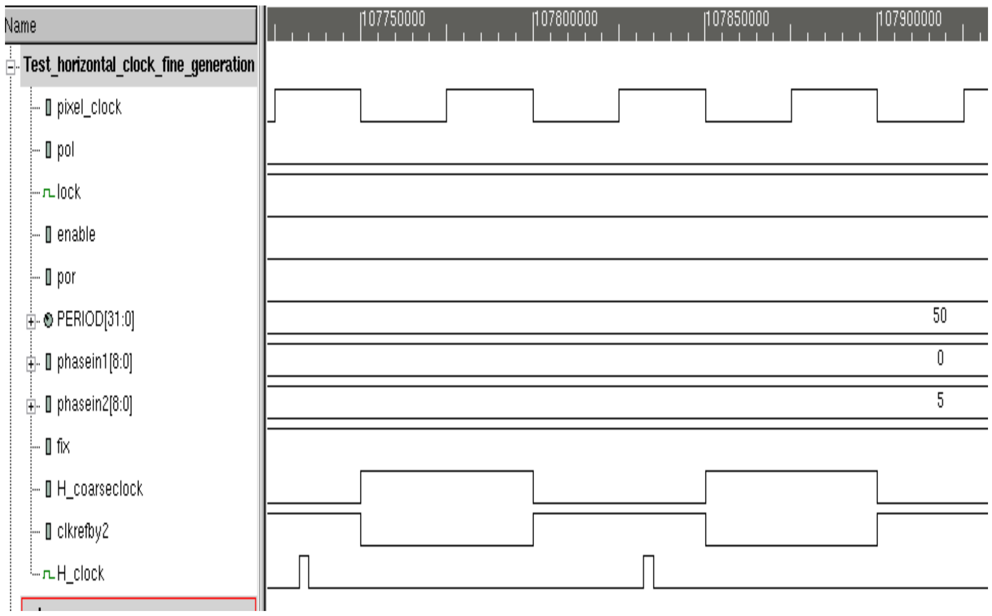


Figure 6.10: Clock pulse with pulse width less than pixel clock period and phase difference less than 180°

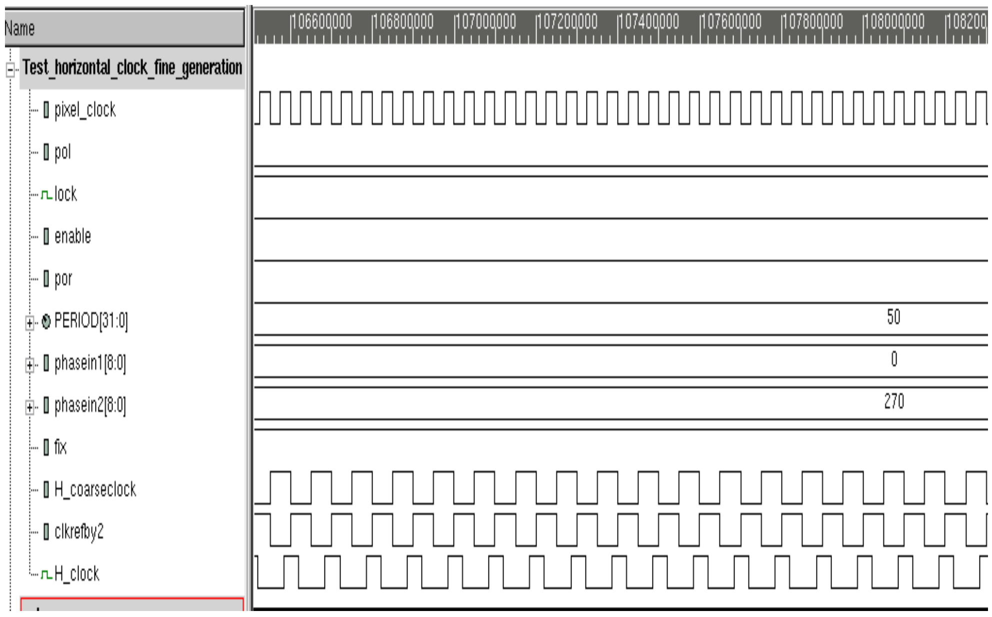


Figure 6.11: Clock pulse with pulse width less than pixel clock period and phase difference greater than 180°

6.2.3 Horizontal clock

The simulation result for a horizontal clock with all features is shown in figure 6.12 The coarse rise edge and fall edge are configured at 0 and 10 respectively. The rising edge of coarse clock is further delayed by 90 and falling edge by 180. Also,2 toggle masking and toggle within masking is shown.

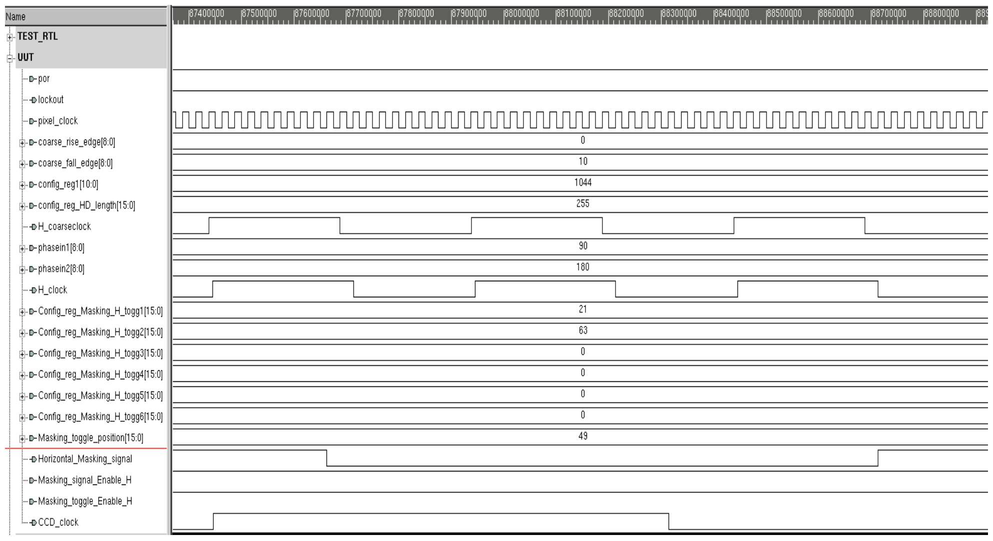


Figure 6.12: Horizontal clock

6.2.4 Vertical clock

The simulation result for a vertical clock with all features is shown in figure 6.13 The coarse rise edge and fall edge are configured at 16 and 34 respectively.Further the start offset is shown and the pattern is repeated 4 times to get the final vertical clock sequence.

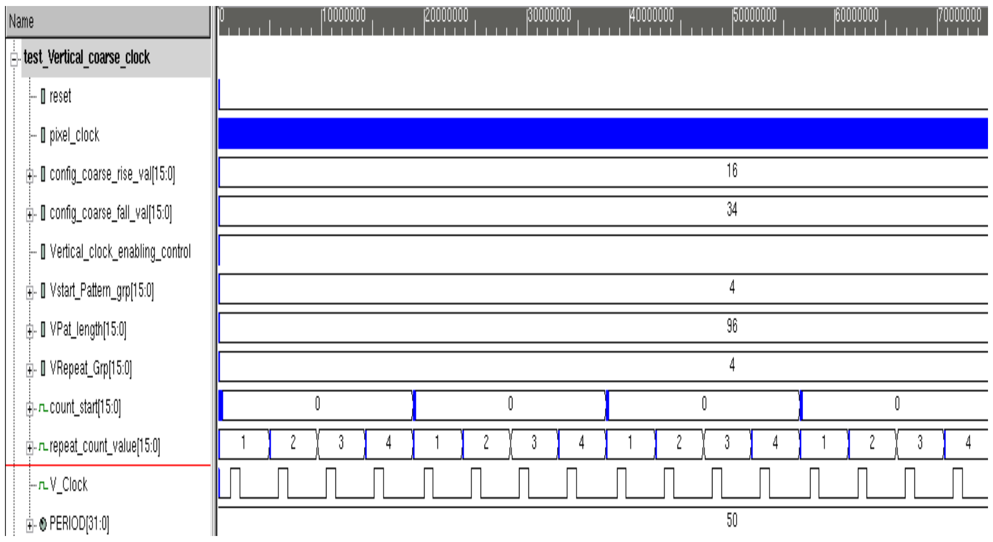


Figure 6.13: Vertical clock sequence

6.2.5 Line and Frame synchronization pulses

The simulation result in figure 6.14 shows the clock pulse obtained for line length of 20 and frame length of 2.

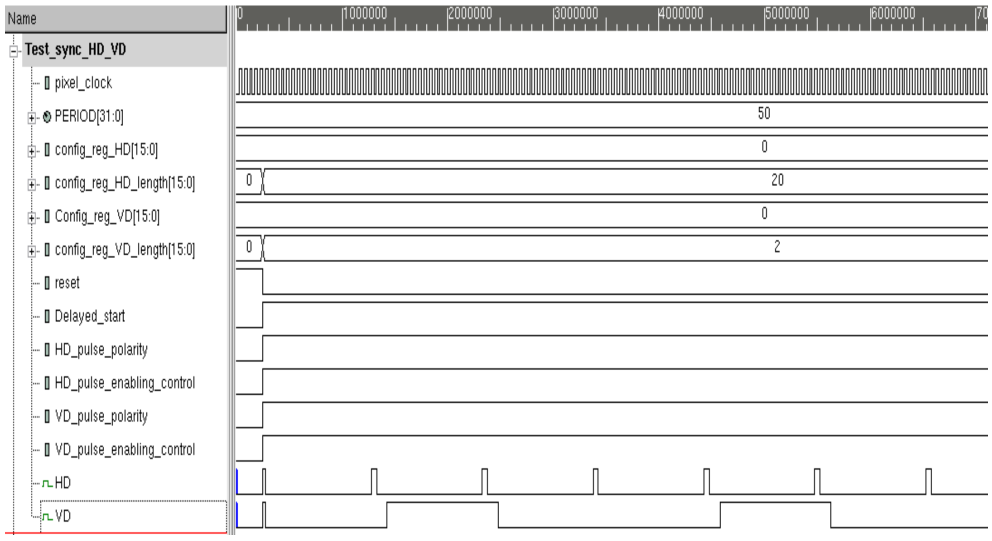


Figure 6.14: Line and Frame synchronization pulses

6.2.6 Exposure Clock

The simulation result in figure 6.15 shows the exposure clock with 4 toggles.

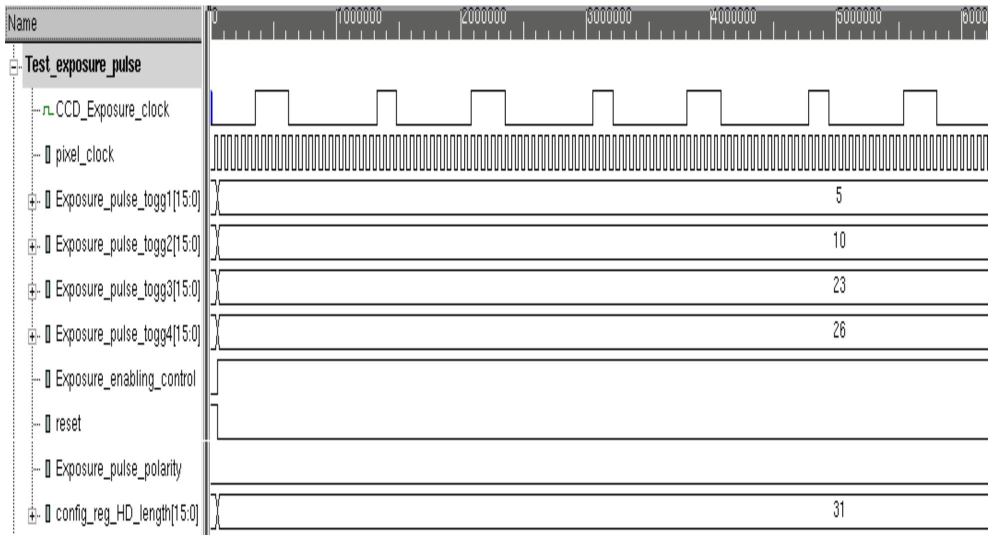


Figure 6.15: Exposure clock pulses

6.3 Summary

In this chapter post synthesis simulation and STA results are shown.

Chapter 7

Verification

7.1 Directed v/s Constrained Random Verification

In Directed case verification approach, the test cases generated by stimulus are applied to DUV, responses are collected and checked. Specific features of the design are verified by each test case. So as the design complexity increases it becomes difficult to create test cases which verify each and every feature of the design. Also the simulation becomes harder and time consuming. Thus in this approach there is a possibility that corner cases are missed.

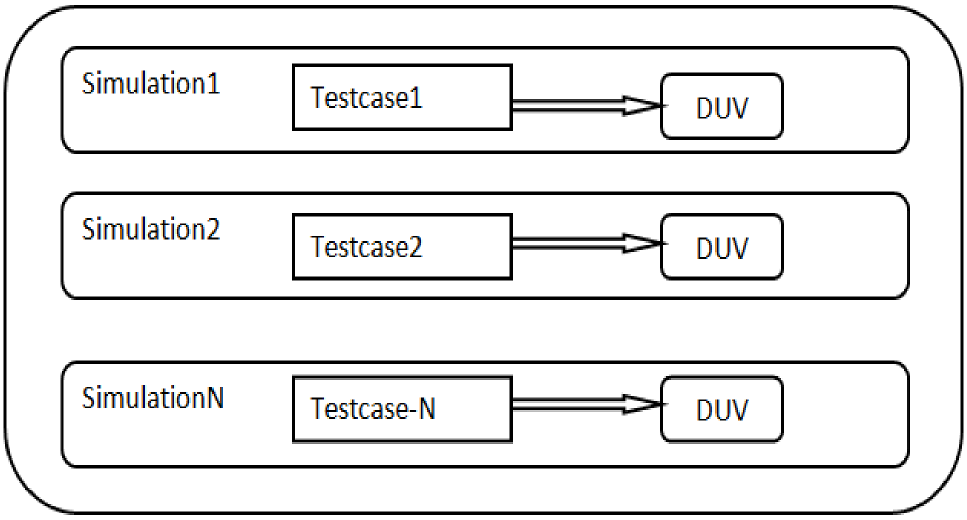


Figure 7.1: Directed Verification Approach

In Constrained Random verification approach, the test cases are generated automatically and randomly. Also generating completely random scenarios is meaningless, so constraints are given which control the random generation of test cases. Using constraint, the solution space is defined, and randomization picks up scenarios randomly from the solution space. Thus Constraint random verification reduces manual effort and code for individual tests.

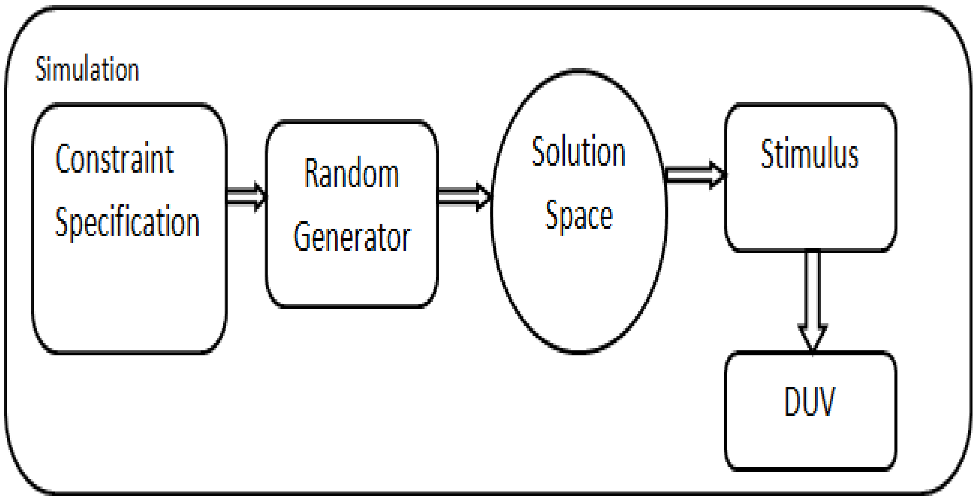


Figure 7.2: Constrained Random Verification Approach

7.2 Basic Architecture of Coverage Driven Constrained Random Verification

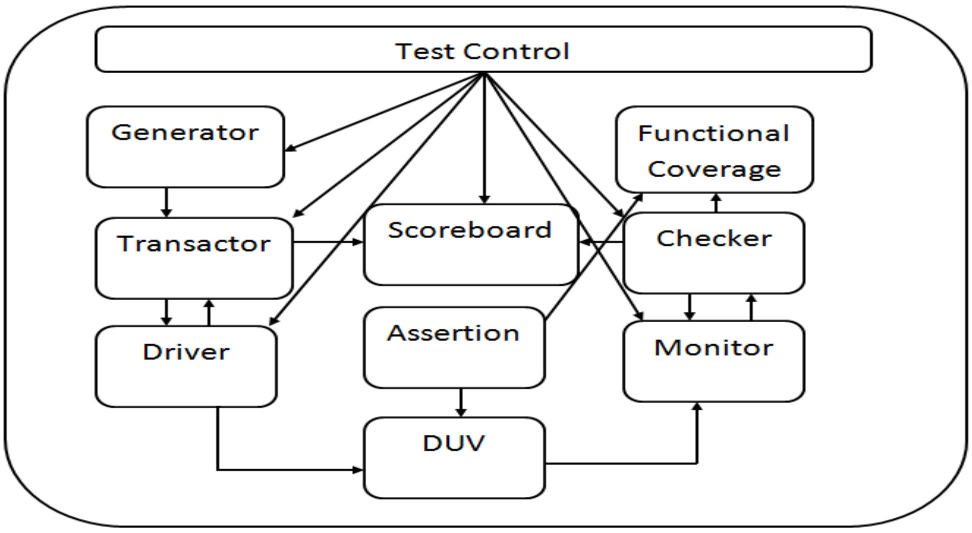


Figure 7.3: Basic Architecture of Constrained Random Verification Environment

The goal of verification is to achieve 100% code coverage and functional coverage by writing deterministic and random tests.The brief description of these verification architecture components is discussed below:

1. Stimulus Generator
Stimulus is generated automatically using System Verilog randomization based on the design specifications and sent to DUV by driver. Stimulus is also processed in other verification components. SystemVerilog high-level data structures helps in storing and processing of stimulus in an efficient way. Also , constraints are defined here. Error injection is a

mechanism in which the DUV is verified by sending error input stimulus. Generally it is also taken care in this module. Generator should be able to generate every possible scenario.

2. Transactor
Transactor does the high level operations like burst-operations into individual commands. This high level stimulus is converted into low level data using packing. This low level data is just a array of bits or bytes. Packing is an operation in which the high level stimulus values scalars, strings, array elements and struct are concatenated in the specified manner. This layer also provides necessary information to coverage model about the stimulus generated.

3. Driver
The driver translates the operations produced by the generator into the actual inputs, as defined in the specification of the designs interface, for the design under verification.

4. Monitor
Monitor, also referred as Receiver, reports the protocol violation and identifies all the transactions. Monitors are two types, Passive and active. Passive monitors do not drive any signals. Active monitors can drive the DUV signals.

5. Assertion
Assertions are a necessary compliment to transaction based testing as they describe the pin level, cycle by cycle, protocols of the design. Assertions are also used for functional coverage.

6. Checker
The monitor only monitors the interface protocol. It doesn't check whether the data is same as expected data or not. Checker converts the low level data to high level data and validates it. This operation of converting low level data to high level data is called Unpacking which is reverse of packing operation. Data is compared against the expected values. The comparison state is sent to scoreboard.

7. Scoreboard
Scoreboard stores the expected DUV output and compares with the output of the DUV in checker module. Dynamic data types and Dynamic memory allocation makes it much easier to write a scoreboard in SystemVerilog.

8. Coverage
This component has all the coverage related to the functional coverage groups.

9. Environment

Environment contains the instances of all verification component and component connectivity is also done. Steps required for execution of each component is done here.

10. Tests
Tests contain the code to control the TestBench features and can communicate with all the TestBench components. Once the TestBench is in place, one needs to focus on writing tests to verify that the device behaves according to specification.

7.3 Verification Environment for Timing Generator

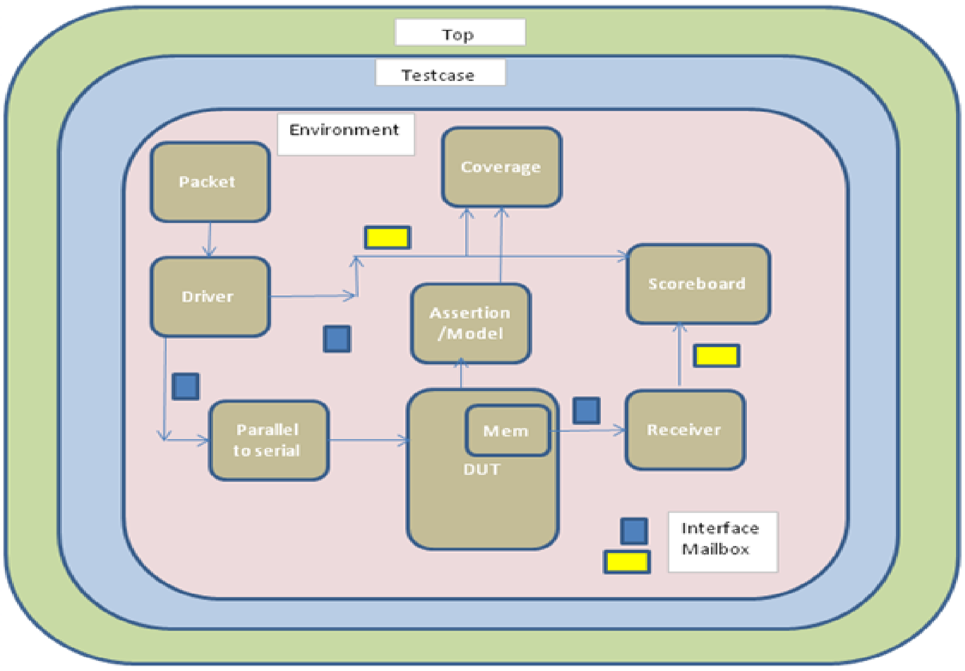


Figure 7.4: Verification Environment for Timing Generator

1. Packet: This class contains declaration of all memory registers.These registers are declared random. Constraints are specified here.

2. Driver: This class will randomize the packet and will send it to parallel to serial module via interface.

3. Parallel to Serial: This module will convert the parallel data available from driver into serial so as to pass it to DUV.

4. DUV: It is an entity for Timing Generator. (Design)

5. Assertion: It is a model which checks that whether the CCD clocks generated by DUT

and by this model is same or not.

6. Receiver: This class receives the packet from driver and from memory to check that memory registers are written with the same values as in the packet.

7. Scoreboard: This class keeps record of checks passed or failed.

8. Coverage:This class has cover groups consisting of different cover points written to check the functional coverage.

9. Interface: It provides communication between class and module allowing smooth migration from abstract system level design to RTL design.

10. Top: This module has the instances of testcase, DUV and assertion.

11. Testcase: The program construct serves as a clear separator between design and testbench.It creates a scope that encapsulates programwide data, tasks, and functions.

12 Environment: This class contains the instantiation of all verification components and provides connectivity between them. Also the data is initialized here.

7.4 Verification Plan

7.4.1 Verification checks for SPI

1. Check when por = 0 data_reg is initialized to 0.
2. Check when por =1
   If load =0 then serial data is written in data_reg
   If load =1 then no change in value of data_reg.
3. Check for por = 0 and load = 0 ,1.

7.4.2 Verification checks for SPI to Memory Interface

1. Check when por =0 data4_out and wadd are initialized
2. Check when por =1
   If load =0 then data_out and wadd are loaded with data and address respectively from data_in
      If MSB of data_in =1 then data_out is written
      If MSB of data_in =0 then address is written

3. Check that all addresses are written when load =0 and por =1
4. Check that output register values does not change when load =1 and por = 0 or 1

7.4.3 Verification checks for Memory

1. Check when rstb = 0 All memory registers are initialized to 0
2. Check when rstb =1
   If wea = 1 then write memory registers
   If load =0 then read memory registers
3. Check for all addresses ,both read and write
4. Check for start and end of write operation
5. Check for start and end of read operation

7.4.4 Verification checks for Clock divider

1. Check for reset =1 and reset =0
2. Check for even divisor_in including 0 and 20
3. Check for odd divisor_in
4. Check for divisor_in > 20

7.4.5 Verification checks for Line and frame Synchronization pulse

1. Check for all config_reg_HD from 0 to 65535
2. Check for all config_reg_VD from 0 to 65535
3. Check for all config_reg_HD_length from 0 to 65535
4. Check for all config_reg_VD_length from 0 to 65535
5. Check for HD pulse polarity and enabling control
6. Check for VD pulse polarity and enabling control

7.4.6 Verification checks for Exposure shuttering control

1. Check for exposure pulse with 2 toggle values ranging from 0 to 65535 within a line for line length 0 to 65535
2. Check for exposure pulse with 4 toggle values ranging from 0 to 65535 within a line for line length 0 to 65535
3. Check for exposure pulse polarity 0 and 1
4. Check for exposure pulse enable 0 and 1

7.4.7 Verification checks for Horizontal Clock

1. Check for all rise edge values from 0 to 511
2. Check for all fall edge values from 0 to 511
3. Check for all period values (9 bits of config_reg1) from 0 to 511
4. Check for cross of above 3 with rise edge<fall edge<period value
5. Check for the clock enable 0 and 1
6. Check for clock polarity 0 and 1

7.4.8 Verification checks for Fine clock generation

1. Check For fix = 0
   Check for all phasein1 from 0 to 360
   Check for all phasein2 from 0 to 360
2. Check for For fix =1
   Check for all phasein1 from 0 to 360
   Check for all phasein2 from 0 to 360
3. Check for clock enable for fix =1 and 0
4. Check for clock polarity for fix =1 and 0

7.4.9 Verification checks for Horizontal Masking Signal Generation

1. Check for all masking with 2 toggles from 0 to 65535 within HD_length 0 to 65535 with toggle1<toggle2<HD_length
2. Check for all masking with 4 toggles from 0 to 65535 within HD_length 0 to 65535 with toggle1<toggle2<toggle3<toggle4<HD_length
3. Check for all masking with 6 toggles from 0 to 65535 within HD_length 0 to 65535 with toggle1<toggle2<toggle3<toggle4<toggle5<toggle6<HD_length 4. Check for Masking_signal_enable =0 and 1

7.4.10 Verification checks for Toggle Masking

1. Check for all Masking toggle position from 0 to 65535
2. Check for HD_length from 0 to 65535
3. Check for cross of above 2 with :
   Masking_toggle_position < HD_length
   Masking_toggle_position should be within toggle values 1 to 6
4. Check for Masking_toggle_enable =0 and 1

7.4.11 Verification checks for Vertical coarse clock

1. Check for all coarse rise values from 0 to 65535
2. Check for all coarse fall values from 0 to 65535
3. Check for cross of above 2 with coarse rise value < coarse fall value
4. Check for pattern length from group A,B,C,D and 32 patterns within each group (total 128)
5. Check for pattern start position from group A,B,C,D and 32 patterns within each group (total 128)
6. Check for pattern repeat from group A,B,C,D and 32 patterns within each group (total 128)
7. Check for clock enabling control =1 and 0

7.4.12 Verification checks for Vertical Masking Signal Generation

1. Check for all masking with 2 toggles from 0 to 65535 within HD_length 0 to 65535 with toggle1<toggle2<HD_length
2. Check for all masking with 4 toggles from 0 to 65535 within HD_length 0 to 65535 with toggle1<toggle2<toggle3<toggle4<HD_length
3. Check for all masking with 6 toggles from 0 to 65535 within HD_length 0 to 65535 with toggle1<toggle2<toggle3<toggle4<toggle5<toggle6<HD_length
4. Check for Masking_signal_enable =0 and 1

7.4.13 Verification checks for CLPOB signal generation

1. Check for CLPOB with 2 toggles from 0 to 65535 within HD_length 0 to 65535 with toggle1<toggle2<HD_length 2. Check for CLPOB with 4 toggles from 0 to 65535 within HD_length 0 t0 65535 with toggle1<toggle2<toggle3<toggle4<HD_length
3. Check for CLPOB with 6 toggles from 0 to 65535 within HD_length 0 to 65535 with toggle1<toggle2<toggle3<toggle4<toggle5<toggle6<HD_length
4. Check for CLPOB_pulse_enabling_control=0 and 1
5. Check for CLPOB_signal_polarity =0 and 1

7.5 Coverage Plan

For a given pixel clock frequency:
1. All memory registers are initialized to 0 when por =0
2. Memory registers are written when load =0
3. Memory registers are read into corresponding blocks when load goes from 0 to 1
4. Horizontal_clock: All coarse rise edge value from 0 to 510 with rise_val <= fall_val

5. Horizontal_clock: All coarse fall edge value from 1 to 511 with fall_val <= config_reg1(9 downto 0)

6. Cross for 4 and 5 with the specified constraints

7. Horizontal_clock: Config_reg1 (9 downto 0) (period value) from 0 to 511

8. Fine clock generation: Phasein1 for 0 to 360 with 32 bins

9. Fine clock generation: Phasein2 for 0 to 360 with 32 bins

10. Cross for 8 and 9

11. Fine clock generation: Phasein1 < phasein2 when coarse_rise_edge = coarse_fall_edge

12. Horizontal_blanking: 2 masking toggles with toogle3,toggle4,toggle5 and toggle 6 =0
    toggle1 can take values from 0 to 65534 with toggle1<toggle2
    toggle2 can take values from 1 to 65535 with toggle2<HD_length
    cross of toggle1 and toggle2

13. Horizontal_blanking: 4 masking toggles with toggle5 and toggle6 =0
    toggle1 can take values from 0 to 65534 with toggle1<toggle2
    toggle2 can take values from 1 to 65535 with toggle2<toggle3
    toggle3 can take values from 2 to 65535 with toggle3<toggle4
    toggle4 can take values from 3 to 65535 with toggle4<HD_length
    cross of toggle1,toggle2,toggle3 and toggle4

14. Horizontal_blanking: 6 masking toggles
    toggle1 can take values from 0 to 65534 with toggle1<toggle2
    toggle2 can take values from 1 to 65535 with toggle2<toggle3
    toggle3 can take values from 2 to 65535 with toggle3<toggle4
    toggle4 can take values from 3 to 65535 with toggle4<toggle5
    toggle5 can take values from 4 to 65535 with toggle5<toggle6
    toggle6 can take values from 5 to 65535 with toggle6<HD_length
    cross of toggle1,toggle2,toggle3 , toggle4,toggle5 and toggle6

15. All HD length from 0 to 65535

16. Horizontal_masking: Masking toggle position from 1 to 65535
    2 toggle : masking_toggle_position between (toggle1,toggle2)
    4 toggle: masking_toggle_position between (toggle1,toggle2) or (toggle3,toggle4)
    6 toggle: masking_toggle_position between (toggle1,toggle2) or (toggle3,toggle4) or (toggle5,toggle6)

17. Horizontal_clock: masking toggle enable =1 and 0

18. Horizontal_clock: masking signal enable =1 and 0

19. Horizontal_clock: enable =1 and 0

20. Horizontal_clock: polarity =1 and 0

21. Vertical clock: All coarse rise edge value from 0 to 65534 with rise_val <= fall_val

22. Vertical clock: All coarse fall edge value from 1 to 65535 with fall_val <= VPat_length

23. Cross for 21 and 22 with specified constraints.

24. Vertical clock: All VPat_length from 1 to 65535 with VPat_length < HD_length
25. Vertical clock: All Vstart_pattern_grp with (Vstart_pattern_grp +rise_val < VPat_length)
26. Vertical clock: All VRepeat_grp with VRepeat = (HD_length/VPat_length)
27. Vertical clock: enable =1 and 0
28. Vertical clock: polarity =1 and 0
29. Vertical _blanking: 2 masking toggles with toogle3,toggle4,toggle5 and toggle 6 =0
    toggle1 can take values from 0 to 65534 with toggle1<toggle2
    toggle2 can take values from 1 to 65535 with toggle2< VPat_length
    cross of toggle1 and toggle 2
30. Vertical _blanking: 4 masking toggles with toggle5 and toggle 6 =0
    toggle1 can take values from 0 to 65534 with toggle1<toggle2
    toggle2 can take values from 1 to 65535 with toggle2<toggle3
    toggle3 can take values from 2 to 65535 with toggle3<toggle4
    toggle4 can take values from 3 to 65535 with toggle4< VPat_length
    cross of toggle1,toggle2,toggle3 and toggle4
31. Vertical _blanking: 6 masking toggles
    toggle1 can take values from 0 to 65534 with toggle1<toggle2
    toggle2 can take values from 1 to 65535 with toggle2<toggle3
    toggle3 can take values from 2 to 65535 with toggle3<toggle4
    toggle4 can take values from 3 to 65535 with toggle4<toggle5
    toggle5 can take values from 4 to 65535 with toggle5<toggle6
    toggle6 can take values from 5 to 65535 with toggle6< VPat_length
    cross of toggle1,toggle2,toggle3 , toggle4,toggle5 and toggle6
32. Vertical _blanking: Masking_signal_enable_V =1 and 0
33. Line and Frame Synchronization Pulse:
    config_reg_VD_length from 1 to 65535
    config_reg_HD_length from 1 to 65535
    HD_pulse_polarity =1 and 0
    VD_pulse_polarity=1 and 0
    HD_pulse_enabling_control=1 and 0
    VD_pulse_enabling_control=1 and 0
    Cross of HD_pulse_polarity and HD_pulse_enabling_control
    Cross of VD_pulse_polarity and VD_pulse_enabling_control
34. Exposure clock : 4 toggles
    toggle1 can take values from 0 to 65534 with toggle1<toggle2
    toggle2 can take values from 1 to 65535 with toggle2<toggle3
    toggle4 can take values from 2 to 65535 with toggle3<toggle4
    toggle4 can take values from 5 to 65535 with toggle4< config_reg_HD_length
    cross of toggle1,toggle2,toggle3 ,toggle4

Exposure_pulse_polarity = 0 and 1

Exposure_enabling_control = 0 and 1

Cross for Exposure_pulse_polarity and Exposure_enabling_control

35. CLPOB_signal_generation: 6 masking toggles

toggle1 can take values from 0 to 65534 with toggle1<toggle2

toggle2 can take values from 1 to 65535 with toggle2<toggle3

toggle3 can take values from 2 to 65535 with toggle3<toggle4

toggle4 can take values from 3 to 65535 with toggle4<toggle5

toggle5 can take values from 4 to 65535 with toggle5<toggle6

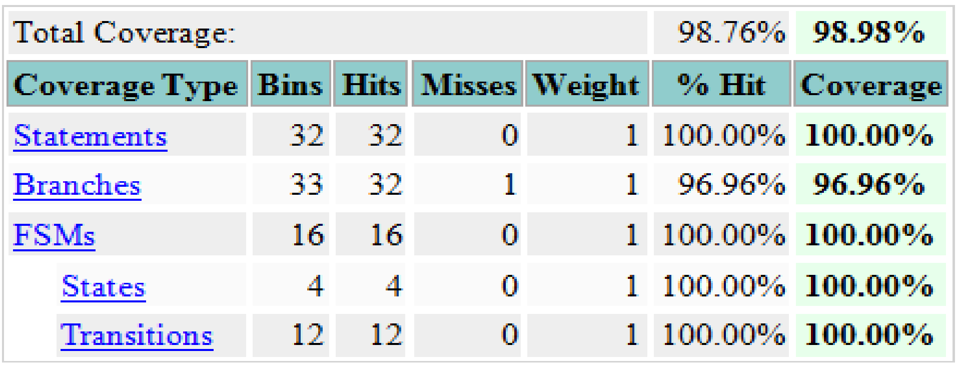
toggle6 can take values from 5 to 65535 with toggle6< config_reg_HD_length

cross of toggle1,toggle2,toggle3,toggle4,toggle5 and toggle6

7.6 Code Coverage Results

7.6.1 Horizontal coarse clock

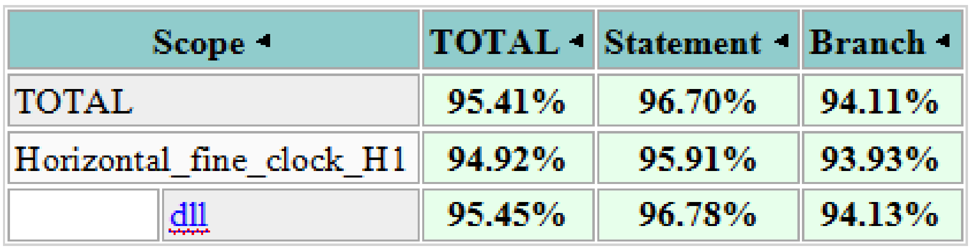
Table 7.1 Code coverage for horizontal coarse clock



Total Coverage:					98.76%	98.98%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	32	32	0	1	100.00%	**100.00%**
Branches	33	32	1	1	96.96%	**96.96%**
FSMs	16	16	0	1	100.00%	**100.00%**
States	4	4	0	1	100.00%	**100.00%**
Transitions	12	12	0	1	100.00%	**100.00%**

7.6.2 Fine clock generator

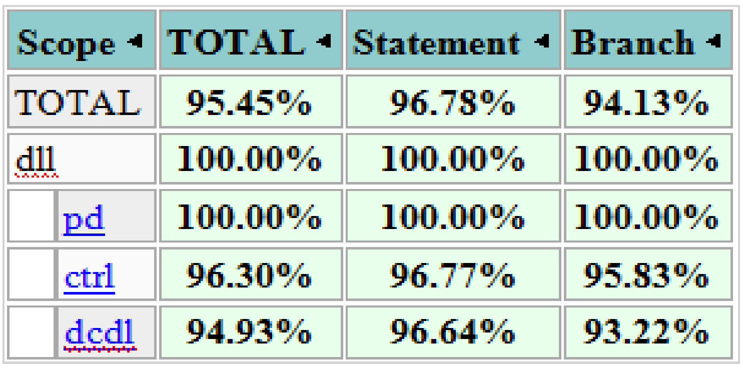
Table 7.2 Code coverage for Fine clock generator



Scope ◂		**TOTAL ◂**	**Statement ◂**	**Branch ◂**
TOTAL		**95.41%**	**96.70%**	**94.11%**
Horizontal_fine_clock_H1		**94.92%**	**95.91%**	**93.93%**
	dll	**95.45%**	**96.78%**	**94.13%**

7.6.3 Delay Lock Loop (DLL)

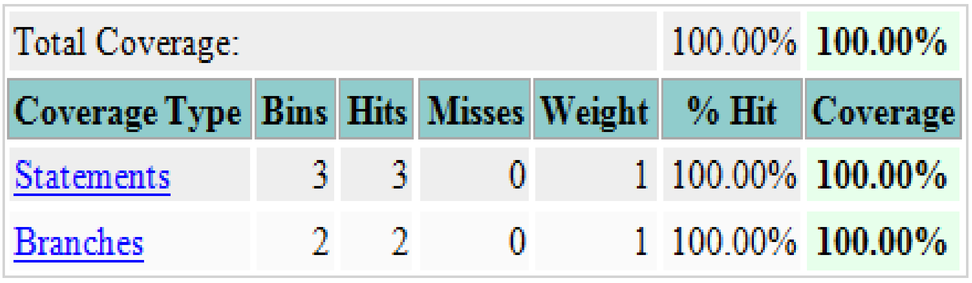
Table 7.3 Code coverage for Delay Lock Loop (DLL)



Scope ◂		TOTAL ◂	Statement ◂	Branch ◂
TOTAL		95.45%	96.78%	94.13%
dll		100.00%	100.00%	100.00%
	pd	100.00%	100.00%	100.00%
	ctrl	96.30%	96.77%	95.83%
	dcdl	94.93%	96.64%	93.22%

7.6.4 Phase Detector

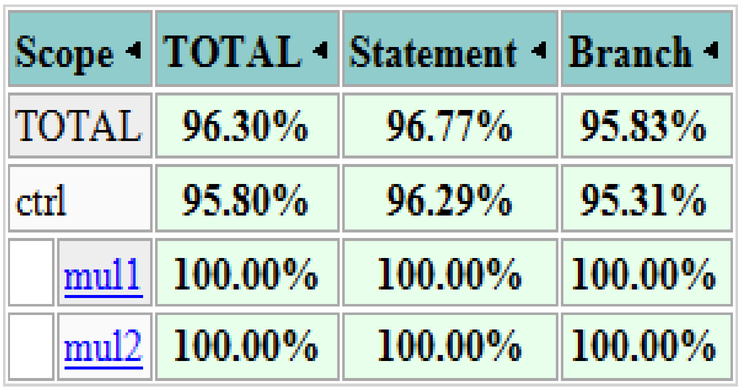
Table 7.4 Code coverage for Phase Detector



Total Coverage:					100.00%	100.00%
Coverage Type	Bins	Hits	Misses	Weight	% Hit	Coverage
Statements	3	3	0	1	100.00%	100.00%
Branches	2	2	0	1	100.00%	100.00%

7.6.5 Controller

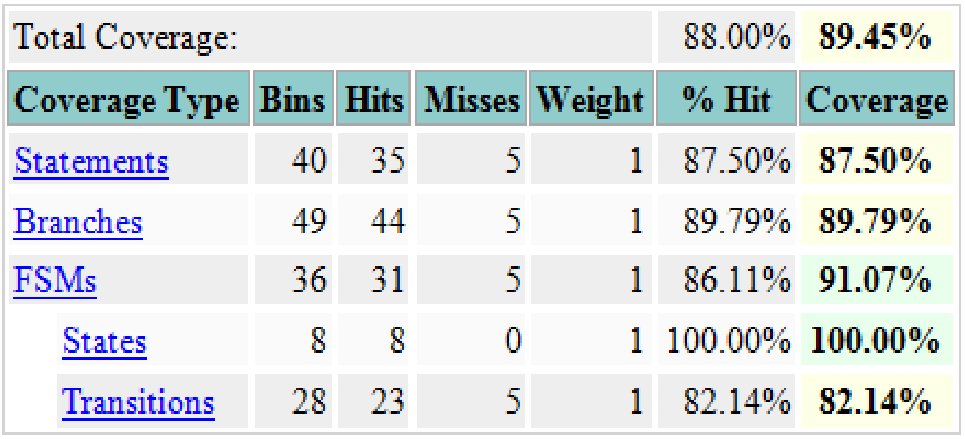
Table 7.5 Code coverage for Controller



Scope ◂		TOTAL ◂	Statement ◂	Branch ◂
TOTAL		96.30%	96.77%	95.83%
ctrl		95.80%	96.29%	95.31%
	mul1	100.00%	100.00%	100.00%
	mul2	100.00%	100.00%	100.00%

7.6.6 Horizontal Blanking

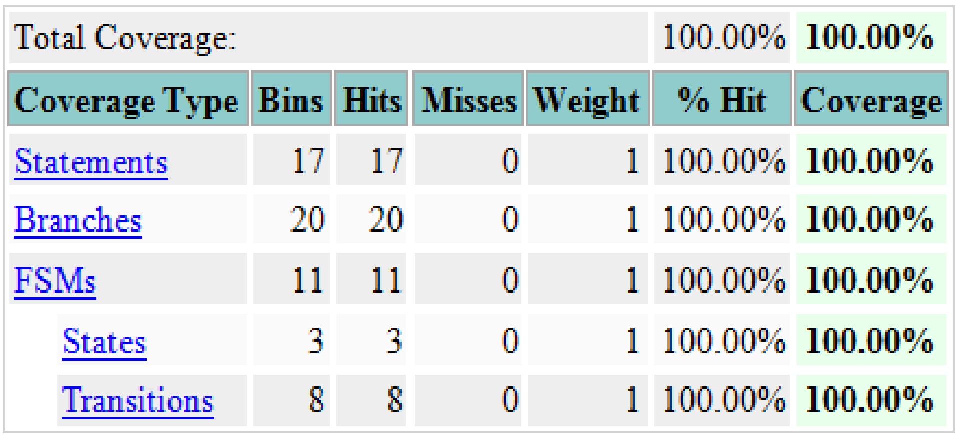
Table 7.6 Code coverage for Horizontal Blanking



Total Coverage:					88.00%	89.45%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	40	35	5	1	87.50%	**87.50%**
Branches	49	44	5	1	89.79%	**89.79%**
FSMs	36	31	5	1	86.11%	**91.07%**
States	8	8	0	1	100.00%	**100.00%**
Transitions	28	23	5	1	82.14%	**82.14%**

7.6.7 Horizontal Masking Toggle

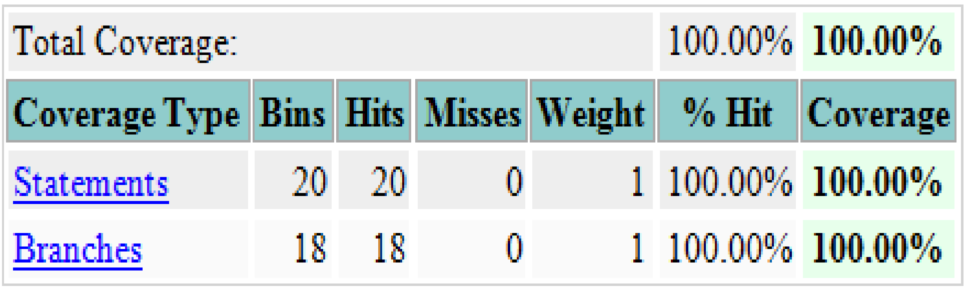
Table 7.7 Code coverage for Horizontal Masking Toggle



Total Coverage:					100.00%	100.00%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	17	17	0	1	100.00%	**100.00%**
Branches	20	20	0	1	100.00%	**100.00%**
FSMs	11	11	0	1	100.00%	**100.00%**
States	3	3	0	1	100.00%	**100.00%**
Transitions	8	8	0	1	100.00%	**100.00%**

7.6.8 Horizontal Masking

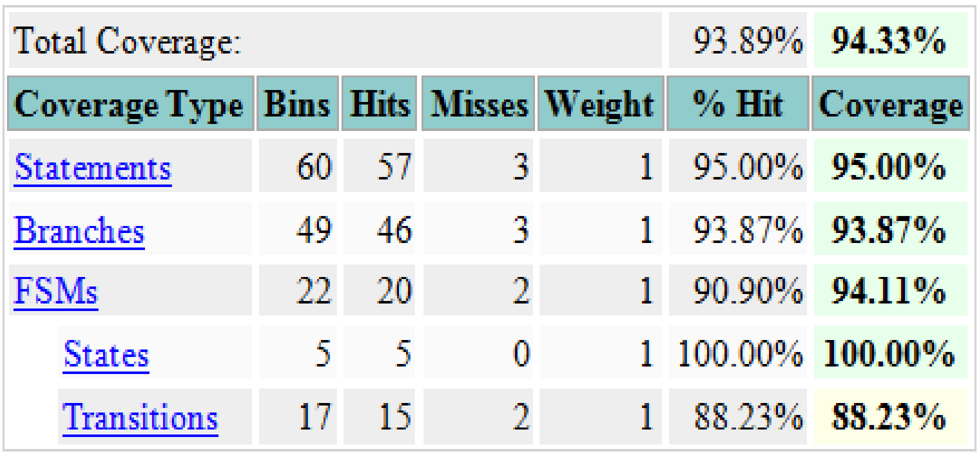
Table 7.8 Code coverage for Horizontal Masking



Total Coverage:					100.00%	100.00%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	20	20	0	1	100.00%	**100.00%**
Branches	18	18	0	1	100.00%	**100.00%**

7.6.9 Vertical coarse clock generation

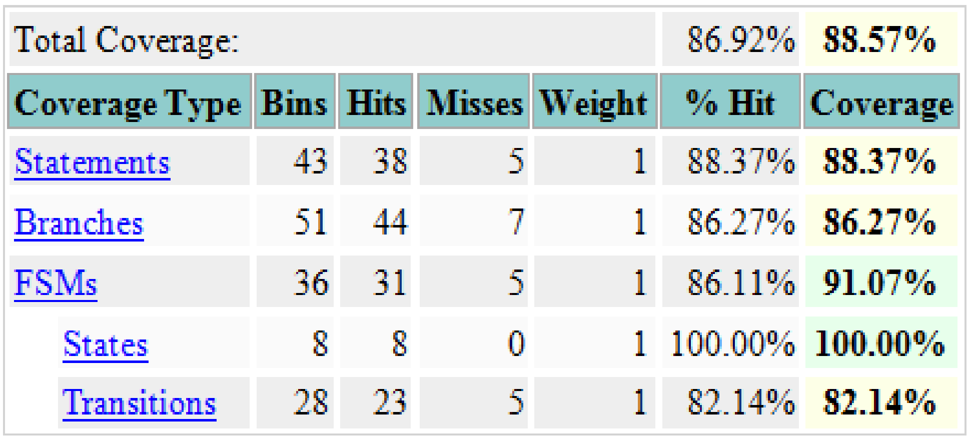
Table 7.9 Code coverage for Vertical coarse clock generation



Total Coverage:					93.89%	**94.33%**
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	60	57	3	1	95.00%	**95.00%**
Branches	49	46	3	1	93.87%	**93.87%**
FSMs	22	20	2	1	90.90%	**94.11%**
States	5	5	0	1	100.00%	**100.00%**
Transitions	17	15	2	1	88.23%	**88.23%**

7.6.10 Vertical Blanking

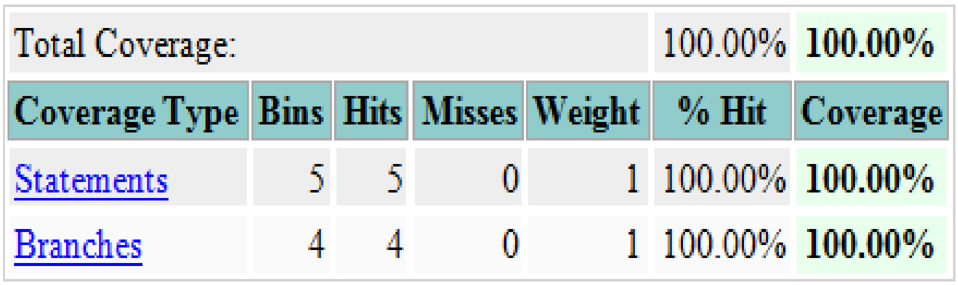
Table 7.10 Code coverage for Vertical Blanking



Total Coverage:					86.92%	**88.57%**
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	43	38	5	1	88.37%	**88.37%**
Branches	51	44	7	1	86.27%	**86.27%**
FSMs	36	31	5	1	86.11%	**91.07%**
States	8	8	0	1	100.00%	**100.00%**
Transitions	28	23	5	1	82.14%	**82.14%**

7.6.11 Vertical Masking

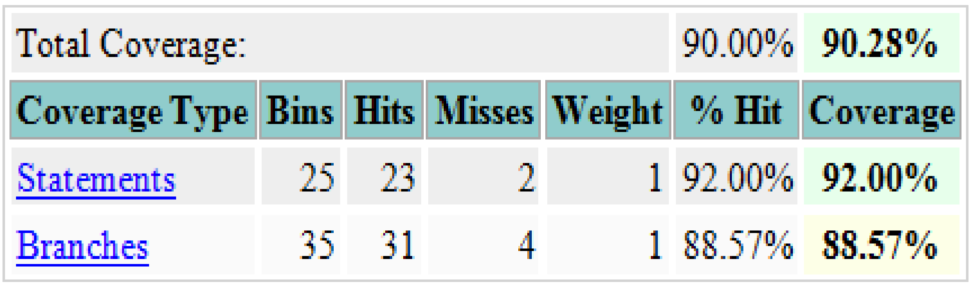
Table 7.11 Code coverage for Vertical Masking



Total Coverage:					100.00%	**100.00%**
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	5	5	0	1	100.00%	**100.00%**
Branches	4	4	0	1	100.00%	**100.00%**

7.6.12 Line and frame Sync pulse generation

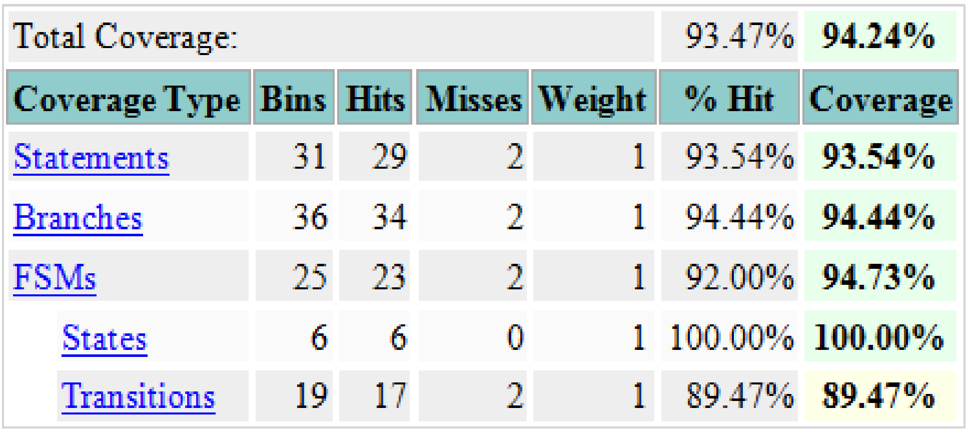
Table 7.12 Code coverage for Line and frame Sync pulse generation



Total Coverage:					90.00%	90.28%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	25	23	2	1	92.00%	**92.00%**
Branches	35	31	4	1	88.57%	**88.57%**

7.6.13 Exposure clock generation

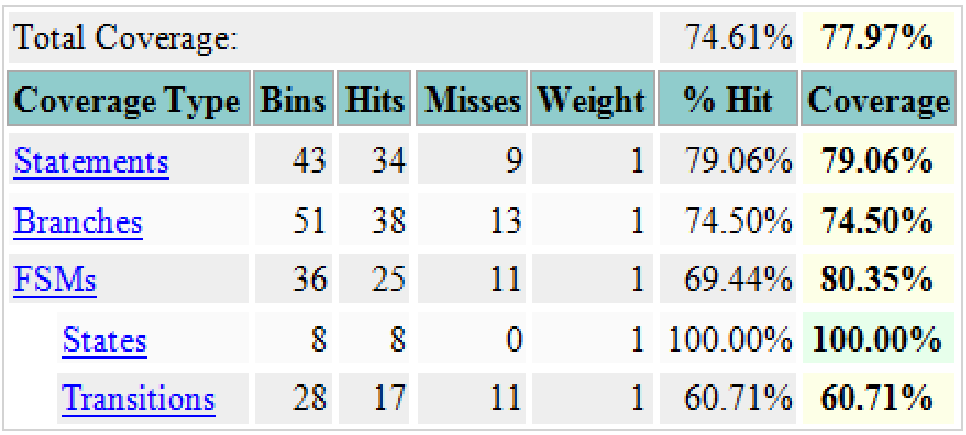
Table 7.13 Code coverage for Exposure clock generation



Total Coverage:					93.47%	94.24%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	31	29	2	1	93.54%	**93.54%**
Branches	36	34	2	1	94.44%	**94.44%**
FSMs	25	23	2	1	92.00%	**94.73%**
States	6	6	0	1	100.00%	**100.00%**
Transitions	19	17	2	1	89.47%	**89.47%**

7.6.14 CLPOB signal generation

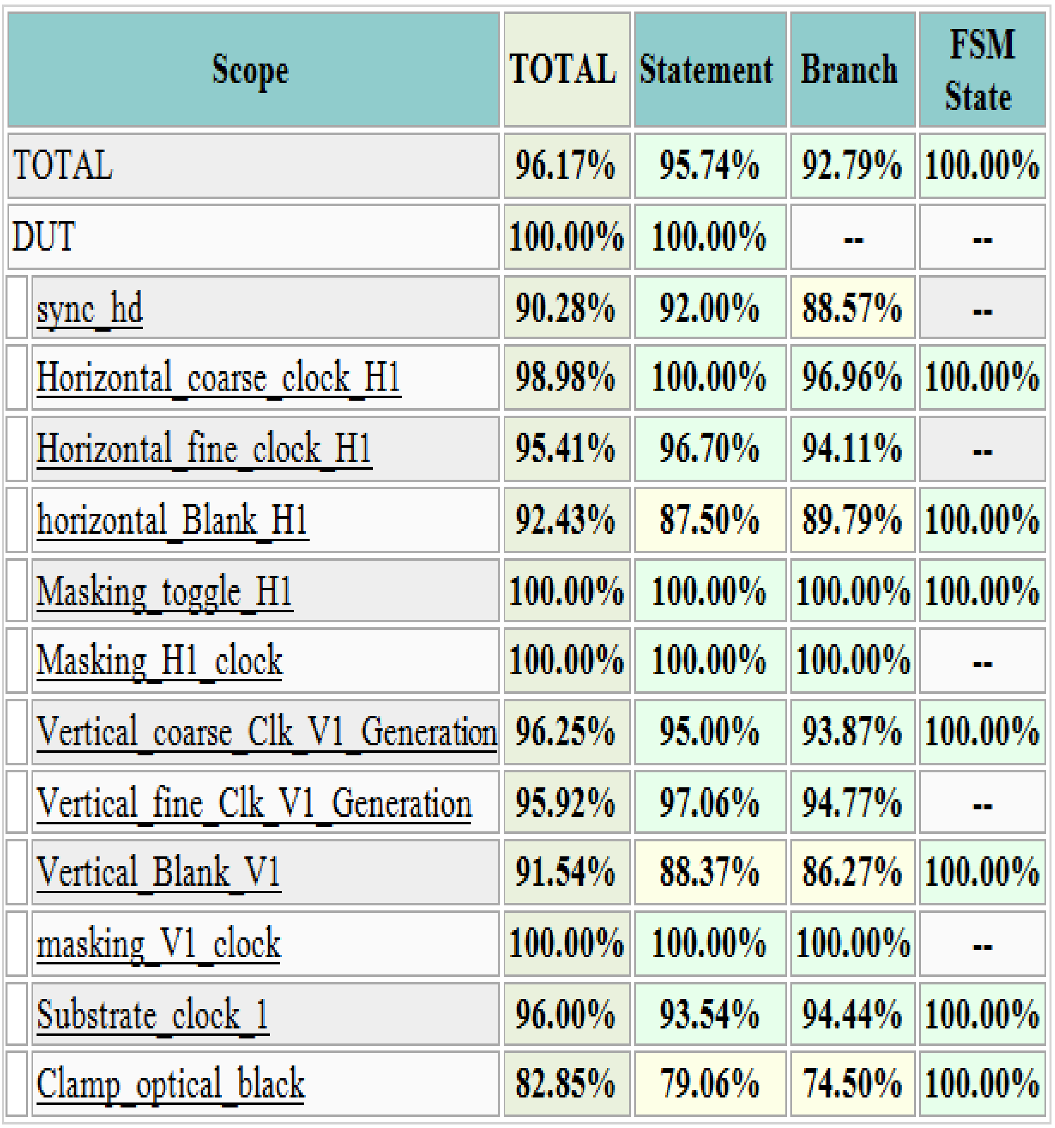
Table 7.14 Code coverage for CLPOB signal generation



Total Coverage:					74.61%	77.97%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	43	34	9	1	79.06%	**79.06%**
Branches	51	38	13	1	74.50%	**74.50%**
FSMs	36	25	11	1	69.44%	**80.35%**
States	8	8	0	1	100.00%	**100.00%**
Transitions	28	17	11	1	60.71%	**60.71%**

7.6.15 Summary of code coverage results

Table 7.15 Summary of code coverage results



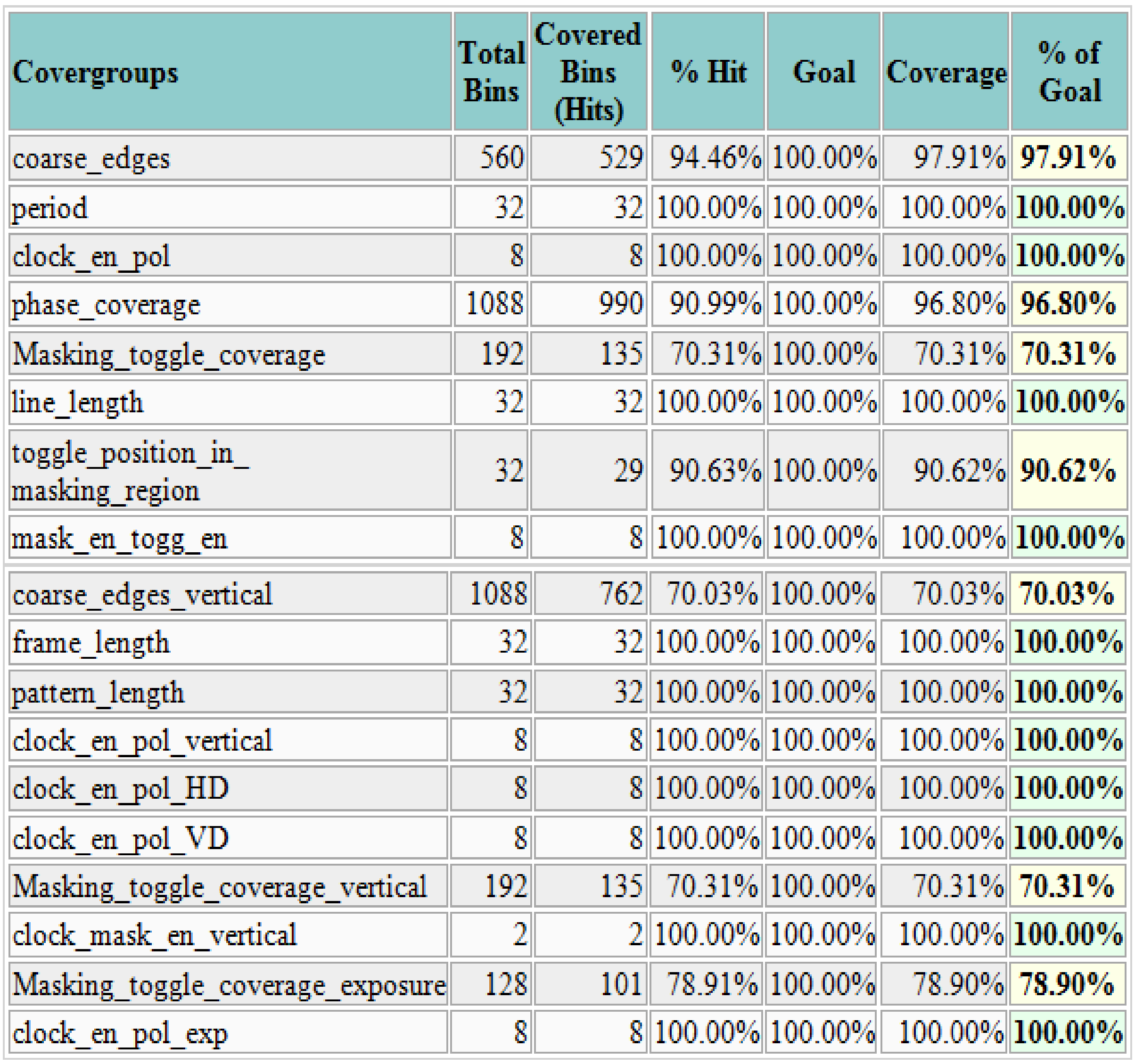
Scope	TOTAL	Statement	Branch	FSM State
TOTAL	96.17%	95.74%	92.79%	100.00%
DUT	100.00%	100.00%	--	--
sync_hd	90.28%	92.00%	88.57%	--
Horizontal_coarse_clock_H1	98.98%	100.00%	96.96%	100.00%
Horizontal_fine_clock_H1	95.41%	96.70%	94.11%	--
horizontal_Blank_H1	92.43%	87.50%	89.79%	100.00%
Masking_toggle_H1	100.00%	100.00%	100.00%	100.00%
Masking_H1_clock	100.00%	100.00%	100.00%	--
Vertical_coarse_Clk_V1_Generation	96.25%	95.00%	93.87%	100.00%
Vertical_fine_Clk_V1_Generation	95.92%	97.06%	94.77%	--
Vertical_Blank_V1	91.54%	88.37%	86.27%	100.00%
masking_V1_clock	100.00%	100.00%	100.00%	--
Substrate_clock_1	96.00%	93.54%	94.44%	100.00%
Clamp_optical_black	82.85%	79.06%	74.50%	100.00%

Thus, the code coverage of 96.17% is achieved as summarized in table 7.15.The total code coverage consisting of statement, branch and FSM states is expected to be 100%.It was analysed and found that additional test cases are required to achieve the same.

7.7 Functional coverage results

The detailed functional coverage for each cover point is tabulated below. The overall 93.04% is achieved.It is analyzed and found that additional test cases are needed to cover all the coverpoints, especially the cross coverpoints.

Table 7.16 Summary of Functional coverage results



Covergroups	Total Bins	Covered Bins (Hits)	% Hit	Goal	Coverage	% of Goal
coarse_edges	560	529	94.46%	100.00%	97.91%	**97.91%**
period	32	32	100.00%	100.00%	100.00%	**100.00%**
clock_en_pol	8	8	100.00%	100.00%	100.00%	**100.00%**
phase_coverage	1088	990	90.99%	100.00%	96.80%	**96.80%**
Masking_toggle_coverage	192	135	70.31%	100.00%	70.31%	**70.31%**
line_length	32	32	100.00%	100.00%	100.00%	**100.00%**
toggle_position_in_ masking_region	32	29	90.63%	100.00%	90.62%	**90.62%**
mask_en_togg_en	8	8	100.00%	100.00%	100.00%	**100.00%**
coarse_edges_vertical	1088	762	70.03%	100.00%	70.03%	**70.03%**
frame_length	32	32	100.00%	100.00%	100.00%	**100.00%**
pattern_length	32	32	100.00%	100.00%	100.00%	**100.00%**
clock_en_pol_vertical	8	8	100.00%	100.00%	100.00%	**100.00%**
clock_en_pol_HD	8	8	100.00%	100.00%	100.00%	**100.00%**
clock_en_pol_VD	8	8	100.00%	100.00%	100.00%	**100.00%**
Masking_toggle_coverage_vertical	192	135	70.31%	100.00%	70.31%	**70.31%**
clock_mask_en_vertical	2	2	100.00%	100.00%	100.00%	**100.00%**
Masking_toggle_coverage_exposure	128	101	78.91%	100.00%	78.90%	**78.90%**
clock_en_pol_exp	8	8	100.00%	100.00%	100.00%	**100.00%**

Chapter 8

Conclusion And Future Work

A front-end design of programmable and precision timing generator is completed.The design provides flexibility and supports various CCD architectures. Post synthesis results shows precise timing delay (approximately 620ps) using counter controlled DLL against requirement of sub-nano second precision. Functional verification is done with coverage driven constrained random stimulus generation and assertion based verification using system verilog.

Future work involves back-end design and testing of ASIC.

References

[1] ZHOU Jian-kang, CHEN Xin-hua, ZHOU Wang and SHEN Wei-min "Design and implementation of timing generator of frame transfer area-array CCD camera".Proc. of SPIE Vol. 6833, 68332K, (2007).

[2] Guoliang Si, Yunfei Li and YongfeiGuo ,"Timing Generator of Scientific Grade CCD Camera and Its Implementation Based on FPGA Technology".Proc. of SPIE Vol. 7658, 76585Y (2010).

[3] Marcus J and Nicholas R, "A single chip CCD waveform generator and sequencer".SPIE, vol.3355, pp. 547-559, 1998.

[4] Simon Tulloch,"Activity 1: Introduction to CCDs", "Activity 2: Use of CCD Cameras" and "Activity 3: Advanced CCD Techniques", article from Institute for Austronmony, Hawai. Source: www.ifa.hawaii.edu/ hodapp/UHH-ASTR-450

[5] Howell, S.B, "Handbook of CCD Astronomy", Cambridge University Press 2nd edition,April 2006.

[6] "CCD Fundamentals" by True sense imaging.Inc REVISION 1.0 PS-0046,3rd august 2012.

[7] "CCD Arrays, Camera Displays" by Gerald C. Holst, SPIE optical Engineering Press, 2nd edition,1998.

[8] "8k TDI CCD Image Sensor with 7x7um Pixel Size and Anti-Blooming", By Teledyne Dalsa, 4th April 2004.

[9] Analog Devices, AD9992,"12 bit CCD Signal Processor with Precision Timing Generator" by www.analog.com

[10] T. Xanthopouls (ed),"Clocking in Modern VLSI Systems, Integrated Circuits and Systems", DOI 10.1007/978-1-4419-0261-0-6, Springer Science +Business Media,LLC 2009.

[11] "A 45nm CMOS, low jitter, all-digital delayed locked loop with a circuit to dynamically vary phase to achieve fast lock",computer science master thesis by SoumyaShivakumar Begur,Northeastern university,October 2011.

[12] Cheng Jia, "A Delay Locked Loop for multiple clock phases/delays generation", Georgia Institute of Technology, 2005.

[13] Feng Lin, Jason Miller, Aaron Schoenfeld, Manny Ma, and R. Jacob Baker,"A Register controlled Symmetrical DLL for Double-Data-Rate DRAM", IEEE Journal of solid-state circuits, vol. 34, no.4, April 1999.

[14] Guang-KaaiDehng, Student Member, IEEE, June-Ming Hsu, Ching-Yuan Yang, Student Member,IEEE, and Shen-Iuan Liu, Member, IEEE, "Clock-Deskew Buffer Using a SAR Controlled Delay Locked Loop", IEEE Journal of solid-state circuits, vol. 35, no.8,August 2000.

[15] Nan Xing,Heesoo Song, Deog-KyoonJeong, and Suhwan Kim, "A PVT insensitive time to digital converter using fractional difference Vernier Delay Lines".

[16] John R. Shaw,Woodlawn, Canada, "Edge programmable timing signal generator" US Patent No-4,675,546 on 23rd june 1987.

[17] A. Sagha,"A novel high resolution Delay Locked loop", https://circle.ubc.ca/handle/2429/16709.

[18] Lee,T.H. et al. "A 2.5 V CMOS delay-locked loop for 18Mbit, 500 megabytes/s DRAM", IEEE Journal of Solid-State Circuits, vol.29, no.12, pp. 1491-6, Dec. 1994.

[19] www.testbench.in