DIGITAL COMMUNICATION AND CONTROL CIRCUITS FOR 60GHz FULLY INTEGRATED CMOS DIGITAL RADIO

A Thesis
Presented to
The Academic Faculty

by

Gopal B. Iyer

In Partial Fulfillment
of Requirements for the Degree
Master of Science in
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2010

DIGITAL COMMUNICATION AND CONTROL CIRCUITS FOR 60GHz FULLY INTEGRATED CMOS DIGITAL RADIO

Approved by:

Dr. Joy Laskar, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Saibal Mukhopadhyay
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Manos Tentzeris
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 2nd April 2010

I dedicate all my research work and the culmination, this thesis:

to my Mom, Dad,

my sisters, Sheela and Sheetal

my brother-in-law, Vinayak,

and to the apple of my eye, my nephew, Vivek ☺

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Joy Laskar for his inspiring leadership and guidance throughout the course of my research project. I would also like to express my gratitude to Dr. Saibal Mukhopadhyay and Dr. Manos Tentzeris for taking the time and serving on my reading committee.

I wish to acknowledge Dr Stephane Pinel and Dr. Bevin Perumana for providing excellent technical guidance in helping me complete my research work. In particular I would like to thank Dr. Bevin Perumana, for mentoring me in the art of Mixed Signal Design. I wish to thank Dr. Padmanava Sen and Dr. Saikat Sarkar for their constant support and valuable friendship.

I take this opportunity to thank all the team members of the Millimeter-Wave Applications Group, who have been a part of this Digital Radio Transceiver project. I thank Chan-Kim and Jin-Keyong Kim from ETRI, Korea, for their help during the design of PHY and MAC modules. I would like to thank Ashwin Muppalla, Jacob Leemaster, Matthew Leung and David Yeh for their support during the SPI testing. I thank my colleagues and seniors from Tata Honeywell and LSI Logic, who have mentored me during my stay there. I thank all my friends for sharing my special moments and making this journey exciting. I specially thank Saumya Venkatram for sharing this journey with me and being a constant support.

None of this would have ever been possible without the unwavering support and blessing of my parents, sister and brother-in-law, the grace of my Maharaj, Shri Brahmachaitanya Maharaj Gondavlekar and Lord Sri Rama and Shri Hanuman.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ABBREVIATION / SYMBOL	MEANING
CMOS	Complementary Metal Oxide Semiconductor
RF	Radio Frequency
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
MIMO	Multiple Input Multiple Output
PHY	Physical Layer of the Open System Interconnect Model
MAC	Media Access Control
ASIC	Application Specific Integrated Circuit
SoC	System on Chip
I/O	Input – Output
LVDS	Low Voltage Differential Signaling
SLVS	Scalable Low Voltage Signaling
ESD	Electro-Static Discharge
HBM	Human Body Model
MM	Machine Model
CDM	Charge Device Model
RTL	Register Transfer Logic
VHDL	Very High Speed Integrated Circuit Hardware Description Language
TLU	Table Look Up
STA	Static Timing Analysis
HFNS	High Fanout Net Synthesis
CTS	Clock Tree Synthesis
SDC	Synopsys Design Constraints
SDF	Standard Delay Format
SI	Signal Integrity

ABBREVIATION / SYMBOL	MEANING
SPEF	Standard Parasitic Exchange Format
DRC	Design Rule Check
LVS	Layout versus Schematic
OCV	On-Chip variation
ECO	Engineering Change Order
CMP	Chemical Mechanical Planarization
SAR ADC	Successive Approximation Register Analogue to Digital Converter
DAC	Digital to Analogue Converter
AGC	Automatic Gain Control
VGA	Variable Gain Amplifier
VCO	Voltage Controlled Oscillator
RFID	Radio Frequency Identification
UWB	Ultra Wide Band
HDMI	High Definition Multimedia Interface
PCI	Peripheral Component Interconnect
MIPI	Mobile Industry Processor Interface
GbE	Gigabit Ethernet
HBT	Heterojunction Bipolar Transistor
PHMET	Pseudomorphic High Electron Mobility Transistor
CMOS	Complementary Metal Oxide Semiconductor
RF	Radio Frequency

SUMMARY

Emerging “bandwidth hungry” applications such as high definition video distribution and ultra fast multimedia side-loading have extended the need for multi-gigabit wireless solutions beyond the reach of conventional WLAN technology or even more recently emerging UWB and MIMO systems. The availability of 7GHz of unlicensed bandwidth in the 60GHz spectrum [1] represents a unique opportunity to address such data-throughput requirements. The 60GHz Integrated CMOS digital radio chipset comprises of PHY and MAC layers, RF transceiver, High-Speed Digital Interface and an underlying Serial Communication Fabric.

To have a complete communication solution compliant with the latest ECMA-369 [3], ISO/DIS 13156 and IEEE 802.15.3c standards [4], we build a digital implementation of MAC and PHY with over a million gates. The Serial Peripheral Interface (SPI) serves as the bridge between the higher layers in the communication stack (PAL-MAC) [5] [6] and the lower layers like PHY-RF Front End. The MAC module can setup the communication link on the fly by tuning parameters such as operating channel, channel bonding and bandwidth, data rates, error correction mechanisms, handshaking mechanisms, etc, by using the SPI to communicate with internal components. The SPI interface plays a crucial rule in not only this, but also during the testing and debug phase. Operation of each of the RF modules is monitored through the serial interface using local SPI slaves which are hooked up to the 4-wire serial bus running all through the chip. The SPI host controller emulates an embedded protocol analyzer. For calibration and fine tuning purposes, digital settings can also be loaded onto these modules through the SPI interface. R-2R DACs are used to convert these commands into analog voltages which then provide a tunable bias to the RF and mixed-signal modules. Other key functions of this serial communication and control interface are: Initialization of all of the RF and mixed signal modules, DC calibration of data converter [7], PLL and other mixed-signal modules [8], data acquisition, parametric tuning for digital modules such as linear equalizer, Gain Control loops (AGC, VGA) [9], etc.

Ultra high speed digital Input-Output buffers are used to provide an external data interface to the radio chipset. These high speed I/Os are also used in the Gbps (gigabit-per-second) link for data transfer between the RF transceiver chip and the PHY-MAC baseband chip. The IOs are expected to comply with different signaling standards such as LVDS [10],

SLVS200, SLVS400 [11], etc. A robust system involves a meticulous pad ring design with proper power domains and power cuts. Full-chip integration of the digital PHY, MAC, peripheral logic and IO ring is done in a semi-custom fashion.

1 INTRODUCTION

1.1 Motivation

Over the past decade, advances in wireless communication have significantly altered the networking and connectivity space. This transformation has been two-pronged. On one hand, advances in semiconductor manufacturing coupled with innovative design techniques have pushed forward the bandwidth envelope. On the other hand, the overcrowded electromagnetic spectrum compels us to develop newer data transmission and modulation techniques.

Recent advances in semiconductor technology have shrunk the CMOS devices from 180nm down to 32nm and 22nm. These developments have brought in a lot of positive changes like higher operating frequencies and much smaller silicon real estate (die areas). However, these advantages come with a flip side, namely, the static power consumption has increased manifold. Most recent trends indicate that there is growing emphasis to develop newer devices and innovative design techniques to minimize the power consumption. With active device f_T's (unity current gain transit frequency) of over 100GHz, it is now possible to have Radio Frequency Front End circuitry (signal conditioning, filtering and amplification) along with digital baseband on the same silicon die. This technology improvement has created newer market segments with a variety of high bandwidth applications. This is the "*technology driven*" perspective.

Another perspective is the *"user need"* based. Figure 1 depicts a comparative study of various communication standards that were prevalent at the turn of the century. As seen, traditional mobile communication standards such as Global System for Mobile (GSM), Code Division Multiple Access (CDMA) and Wideband Code Division Multiple Access (WCDMA), have throughputs of a few megabits per second. The Ultra Wide Band (UWB) standard supports data rates of 1Gbps. All these are limited to transfer of voice, data and images. Streaming of video needs much higher bandwidths and only wired communication standards were capable of supporting it.

Standard	Frequency	Data rate	Distance
CDMA	824-849 MHz 869-894 MHz	Less than 1 Mbps	Greater than 1mile
GSM	880-915 MHz 925-960 MHz	Less than 1 Mbps	Greater than 1mile
GPS	1.575 GHz	Very slow	Greater than 1mile
WCDMA	Around 2 GHz	2 Mbps	Greater than 1mile
Bluetooth	2.402-2.408 GHz	3 Mbps	10 meters
802.11b/g	2.4-2.483 GHz	11 Mbps @802.11b 54 Mbps@ 802.11g	35 meters (indoor)
802.11a	5.15-5.35 GHz 5.47-5.725 GHz 5.725-5.875 GHz	54 Mbps	30 meters (indoor)
UWB	3-10 GHz	Up to1 Gbps	3-10 meters depending on the transmission speed

Figure 1: Comparison of various communication standards

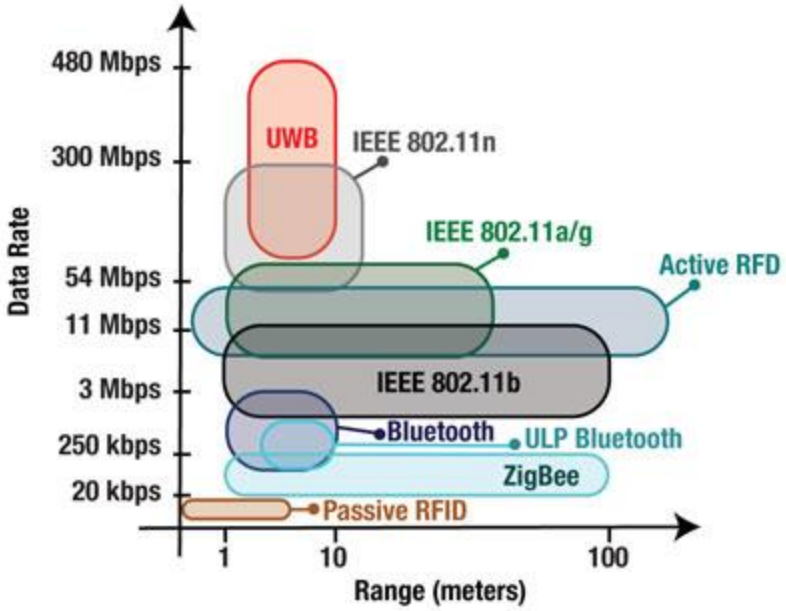


Figure 2: Wireless standards spectrum

Transfer of huge amounts of data would also need high bandwidth communication. Figure 3 shows the growing size of the average computer hard disk, which will soon move into the terabyte regime. As this trend grows, we would soon need systems for faster data transfer between these devices [12]. A variety of multi-gigabit application interfaces and entertainment connectivity solutions such as HDMI, USB2/3, MIPI, GbE, PCI, etc… can be address with the single chip 60GHz solution. This is the "*user need*" driven motive to explore 60GHz Communication system.

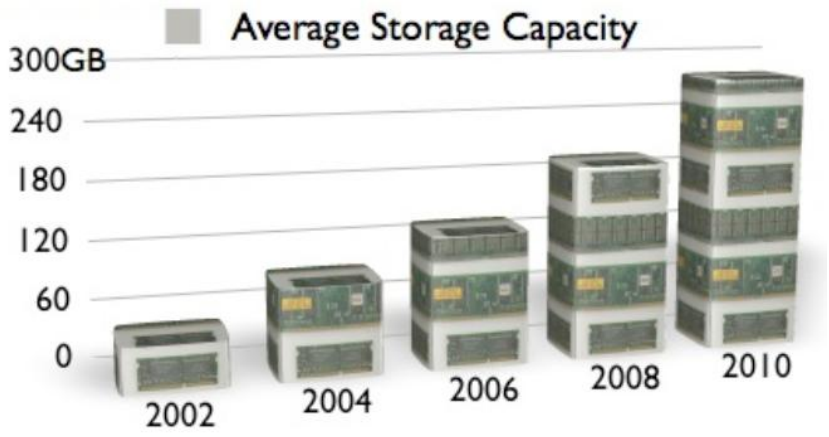


Figure 3: Average storage capacity of hard disks

1.2 Choice of 60GHz for the Digital Radio

A wide variety of reasons influenced the choice of developing a 60GHz digital radio, some of which are highlighted below. Typical transit frequency (f_T) curves for Silicon Germanium (SiGe) and Complementary Metal Oxide Semiconductor (CMOS) devices at various technology nodes are presented in Figure 4. At the 90nm node, the CMOS f_T is over 140GHz, making it an ideal choice for the development of 60GHz radio transceiver.

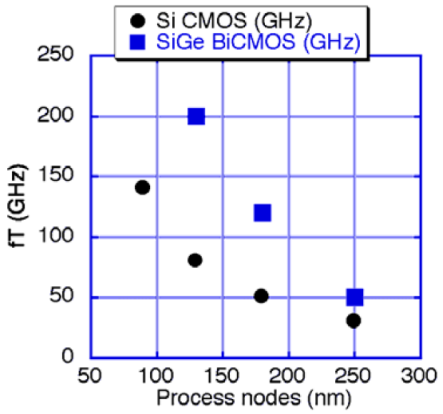


Figure 4: Transit frequencies for SiGe and CMOS at various technology nodes

From the mid 90's, the defense sector has applications in the higher frequency band (77GHz) using III-V compound semiconductors (GaAs, HBT, PHMET) [13] [14].The digital post processing for these applications was realized on vanilla CMOS processes, mainly for cost benefits. Using the III-V semiconductors for high volume production design was not feasible and this was one of the major thrusts for CMOS RF front-ends. The interest in utilizing the unlicensed band at 60GHz for high-speed short-range communication was exploited. A unique feature of this frequency band is the high oxygen absorption level (~15dB/km). High absorption levels prohibited long range communication. However they were advantageous from the frequency reuse perspective. This meant that two communication links, located in close proximity of each other could operate simultaneously. This wireless network was standardized by the IEEE 802.15 Working Group for Wireless Personal Area Networks (WPAN).

The Federal Communications Commission (FCC) in the United States, allocate the 59-64GHz frequency band for general unlicensed use. Japan standardized the 59-66GHZ band for high-speed data communication. In Europe, the 59-62GHz band has been allocated for Wireless Local Area Network (WLAN) and mobile broadband systems use the 62-63GHz and 65-66GHz bands. Since propagation losses increase with the operating frequency, we need high gain antennas for longer reach. Higher the gain of the antenna, narrower the beam width and this helps minimize the beam spread. Overall, the combined effect of oxygen absorption, narrower beam widths, and higher frequency reuse positions the 60GHz band [15] most appropriate for short range high data rate communications.

This 60GHz unlicensed band has a sufficiently large bandwidth of about 7GHz and is capable of providing multi-gigabit throughput using simple modulation schemes like ASK (Amplitude Shift Keying) and BPSK (Binary Phase Shift Keying). More complex schemes such as QPSK (Quadrature Phase Shift Keying), QAM (Quadrature Amplitude Modulation) and OFDM (Orthogonal Frequency Division Multiplexing) support data-rates as high as 10Gbps. Thus we can ensure a seamless integration with multimedia standards.

1.3 Digital Control and backend

The design of 60GHz CMOS Digital Radio Transceiver is extremely complex with many RF, Analog and digital modules juxtaposed on the chip. Each of these RF and Analog mixed-signal modules need to be tuned for optimal performance. Doing so would ensure lower power consumption, higher spectral efficiency and higher data rates with lower BERs (Bit Error Rates). Hence it is crucial for us to ensure that all these modules get tuned and calibrated appropriately. This gives rise to the need for a reliable debug and tuning mechanism over a robust communication channel. Amongst the available serial interfaces, the SPI [16] communication protocol suits our needs best. This duplex serial communication link has been customized for high speed operation.

The digital backend on this transceiver forms the backbone of the wireless communication channel. The digital modem, PHY (physical communication layer of the OSI 7 layer software stack) [17] and the MAC (Media Access Control sub-part of the Link Layer), form the digital content on this chip. The Radio link is serial and hence the digital modem also operates on the serial data stream. Thus it is designed to be fairly simple with only the necessary functionality built into it. For ease of test, the transceiver was designed in a two-chip solution. Later revisions were fully integrated single silicon die CMOS transceiver solutions. The serial data stream is sent off-chip using high-speed digital I/Os. The separate test-chip for PHY and MAC modules interfaces with this high speed serial link. The PHY and MAC make up a million gate design which was implemented using the ASIC physical design flow. Special attention was given while building the I/O ring with extra attention to the pad ring, power cuts and pad assignments.

1.4 Organization of Thesis

The thesis is divided into seven chapters. After introducing the research objective of the 60GHz digital radio and its design challenges in chapter 1, chapter 2 talks about the physical design flow in ASIC design. It starts with register transfer level (RTL) design and briefly touches upon the entire flow: RTL coding, synthesis, simulation, layout, verifications. The PHY and MAC designs were developed using this flow. For version-1 they were part of a separate chip (ease of testing). In the later versions, they were integrated with the RF front-end on a single chip. Screenshots used to illustrate the digital flow are some of the intermediate steps during the MAC-PHY Physical layout development. Chapter 3 discusses the various serial communication interfaces used for on-chip and off-chip data acquisition. The chapter concludes with the choice of Serial Peripheral Interface (SPI) as the chosen serial communication protocol for the 60GHz digital radio system.

Details of the serial interface sub-system are explained in chapter 4. Two designs for the SPI were realized in 90nm CMOS process. The differences in the two designs and the motivation to switch from one to the other, is provided. Detailed timing diagrams for the different modes of operation are presented. Chapter 5 talks about the measurement setup developed for using this serial communication interface. The testing phase began with standard off-the-shelf protocol analyzers and bit-stream signaling sources, and went on to custom-designed modules.

Digital I/Os (with standard 50Ω termination) were designed to bring out the SPI ports. For bringing out the baseband signal we developed a multi-gigabit digital I/O interface. Chapter 6 discusses these designs. This chapter briefly touches upon the importance of ESD protection in the ASIC development flow. It discusses the major ESD failure mechanisms and outlines a few design techniques to mitigate that risk. A special I/O ring had to be developed for the MAC-PHY standalone chip (version-1) which forms the last part of this chapter. The chapter concludes with some measurement results. Chapter 7 concludes this thesis by summarizing the work performed and suggesting future improvements.

2 PHYSICAL DESIGN FLOW FOR ASIC IMPLEMENTATION

The *Physical Design* of any full-custom ASIC [18] can follow either the top-down design or the bottom-up approach. Figure 5 shows an overview of the physical design flow for ASIC (Application Specific Integrated Circuit) implementation from RTL to GDSII, in the top-down approach. The following subsections detail the individual steps mentioned. The step-wise physical design flow [19], highlighting the corresponding tool used at each step, is shown in Figure 6.

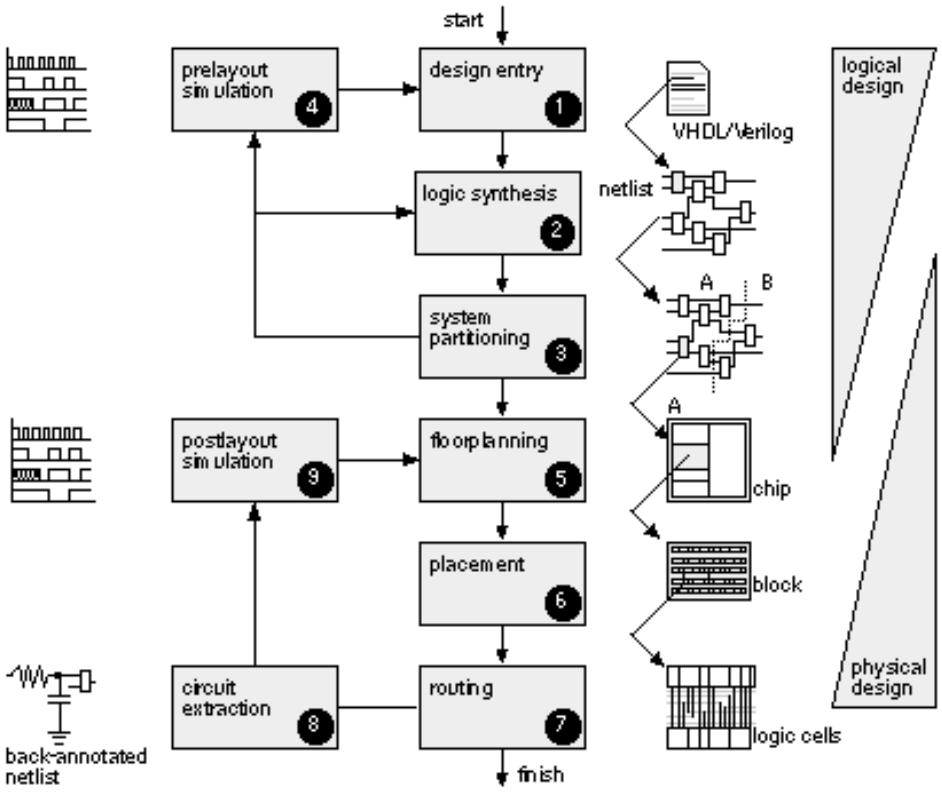


Figure 5: Designing SoC - From Logic design to Physical design

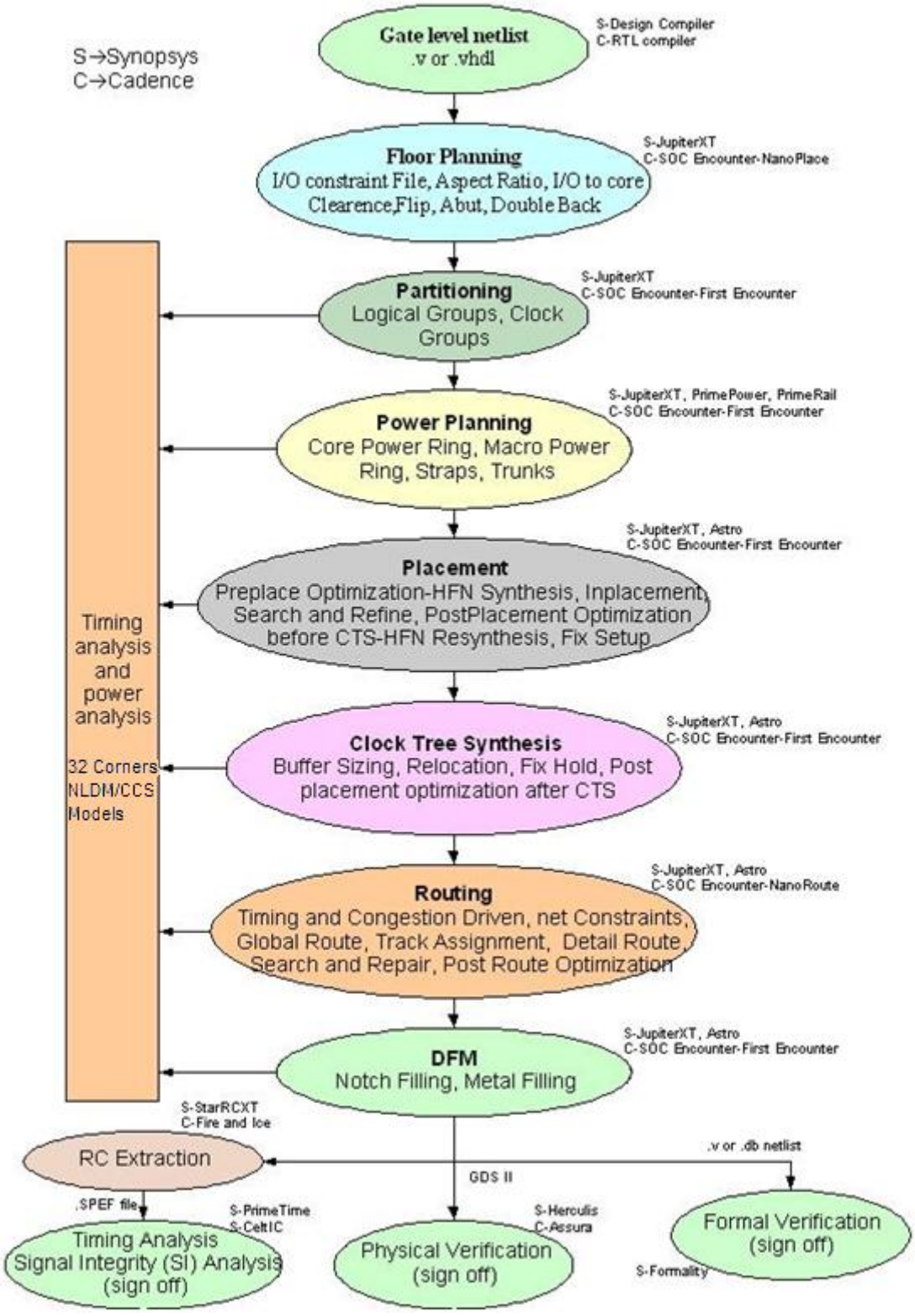


Figure 6: System-On-Chip Design Flow

2.1 Design Entry

The physical design flow for ASIC implementation begins with design entry in logical format as shown in Figure 5. The system is described in a symbolic or descriptive language and a simulation is run to test the functionality. This description resembles a flowchart and hence is technology independent. The system operation is tested at an abstract level. High-level architectural changes need to be incorporated at this stage.

2.2 Logic Synthesis

Generally, all the digital cells are developed in each of the fabrication houses (foundries) and licensed out as "Digital Standard Cell Libraries." The standard cells come in different descriptions with each one called a "view". During the synthesis stage we make use of the Verilog [20] (behavioral) view and the timing-power view. The logical description of the design is mapped on to standard cells (digital gates) through this process of synthesis. Synopsys Design Compiler [21] and Cadence BuildGates [22] are some of the tools that we used for synthesis. These tools support schematic viewers for viewing the gate level design and the connectivity. At this stage, we are ready to perform a Static Timing Analysis (STA) [23] on the design.

2.3 Logical Verification:

The synthesized gate level Netlist is read into a simulation tool like “ModelSim” from Mentor Graphics [24]. A detailed testbench is created in the same descriptive language (Verilog or VHDL [25]). This testbench has a sequence of test vectors that are applied to the unit under test (our design). A waveform viewer is used to view the test results. The following set of images highlight the procedure.

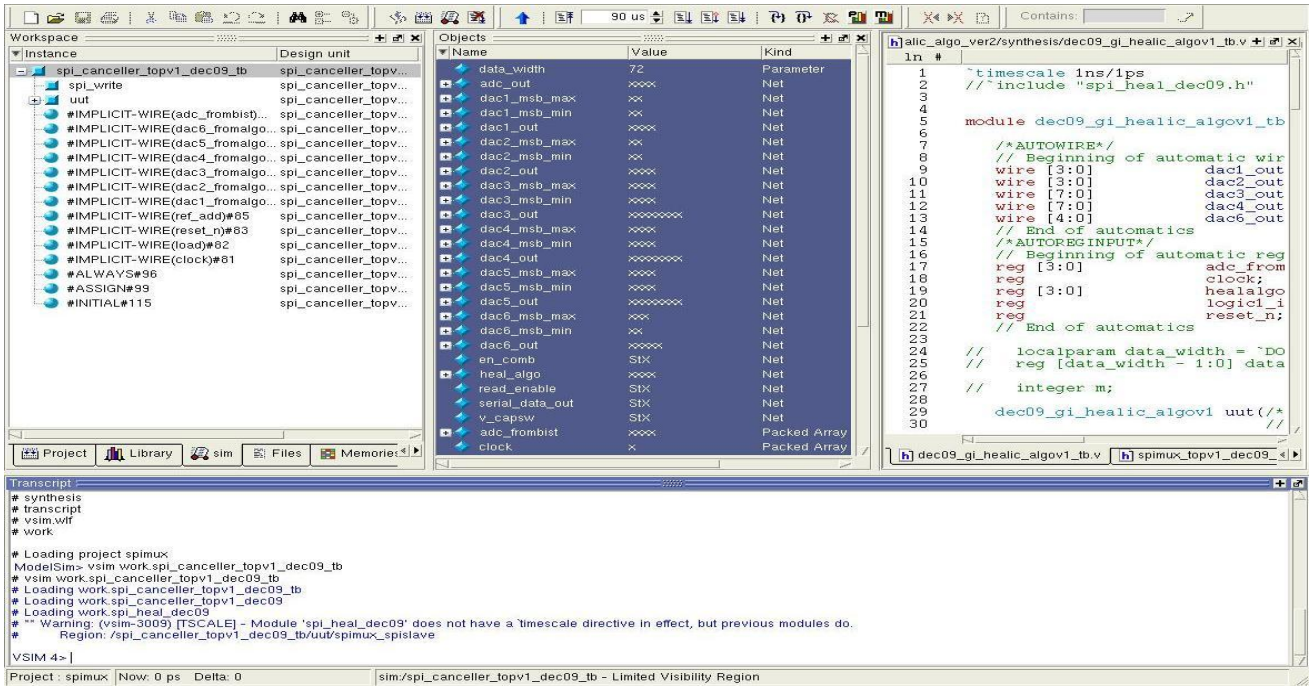


Figure 7: ModelSim Workspace

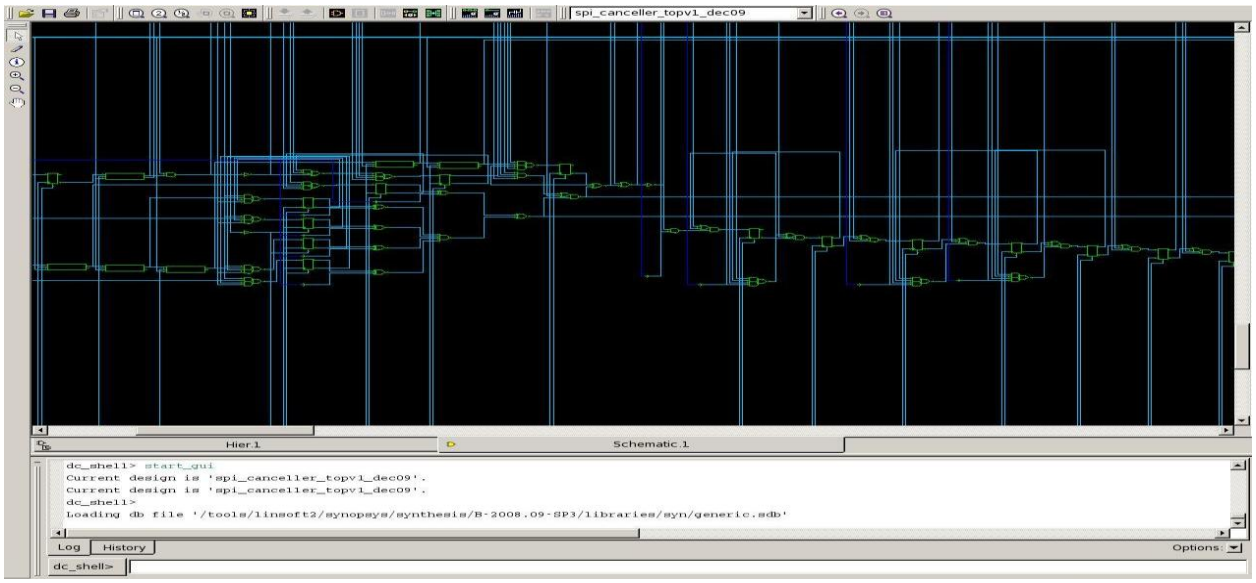


Figure 8: ModelSim Schematic

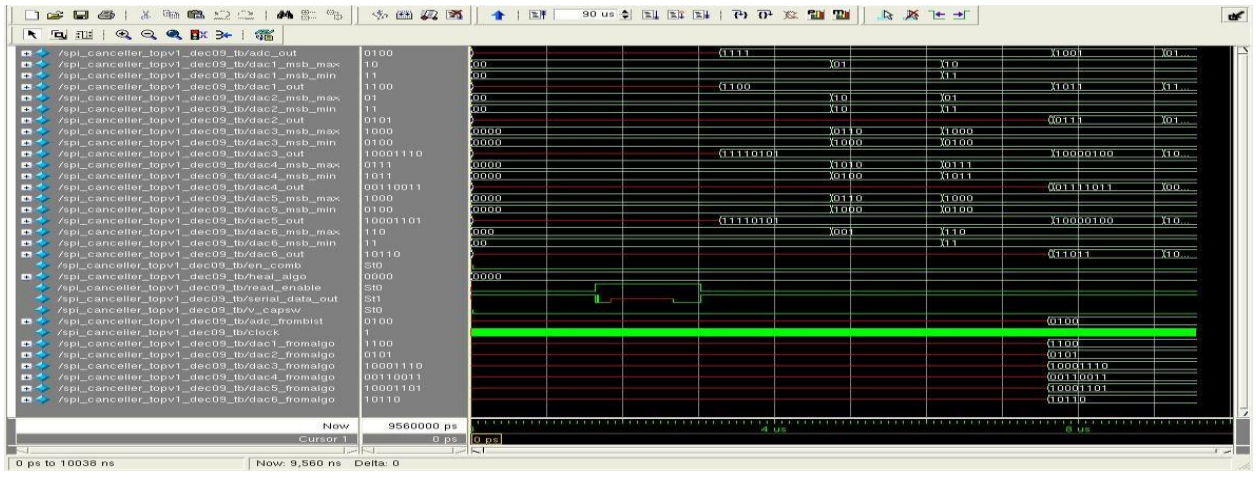


Figure 9: ModelSim Waveform

2.4 Static Timing Analysis

All the standard cells are characterized and Look-Up Tables (LUT) for timing and power data has been generated. The timing views for the standard cells are read into a static timing analysis tool like PrimeTime from Synopsys [23] and the design is linked. We then define the boundary conditions and the timing constraints for the design through the Synopsys Design Constraint (SDC) file [26]. A detailed outline of the timing analysis flow is shown below:

i. Preparing the Analysis Environment

The standard cell libraries are read in first. Then the Netlist (design) is read in. The link paths are setup and the design is linked. The appropriate wire load models are read in and the operating conditions are set.

ii. Preparatory timing and clocking checks

The most important step here is to define the system clock. Min-max values for the clock uncertainties and latencies are set. We can also define the clock duty cycle and transition times. Setup, hold and other clock gating checks are verified as part of this step. This timing verification is done at multiple stages in the flow, starting with post-synthesis, then post-layout and finally during the timing sign-off phase. During the post-synthesis phase, the wire delay is only an estimate. However we have real extracted data during the other two phases. The wire delay is extracted in a format called the Standard Delay Format (SDF) [27] and is back-annotated on the design.

iii. Checking design integrity

When the design has been setup correctly, the design integrity can be checked by commands such as *report_design*, *report_reference* and so on.

iv. Running timing analysis

Executing the command *'check_timing'* will use all the timing constraints and run the timing engine on the design. A thorough check on setup times, hold times and transitions time will be performed. Warnings and errors will be thrown up for each and every check. Some of these could be exceptions such as false paths, multi-cycle paths, clock domain crossings, etc. These errors need to be resolved at this stage. Timing slacks [28] can be reported for further detailed analysis. Some of the advanced timing issues like reset recovery and removal, clock synchronization, etc are explained in the references [29].

v. Boundary timing characterization

Timing characterization can be hierarchical with sub-designs and parent design. The partitioning could be in design compiler (synthesis) or within PrimeTime (for ease of timing analysis). Some of the key aspects related to timing hierarchy are: input and output arrival delays, input and output port loads, constant logic values on inputs, logic propagation, annotated delays and parasitic, timing exceptions and so on.

vi. Fixing timing violations

Timing violations need to be fixed on a case-by-case basis. Some violations could be false, such as false paths. Like the reset paths on flip-flops. Some paths may span multiple clock periods and need to be constrained as multi-cycle paths. Then there are IO paths that need to be properly constrained with commands such as '*set_input_delay'*.

```
Inferred memory devices in process
        in routine spi_heal_dec09 line 105 in file
                '/home/st/giyer6/BG/healic/dec09_gi_spimux/temp_synthesis/spi_heal_dec09.v'.
=================================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
=================================================================================
|   load_sync_1_reg   | Flip-flop |   1   |  N  | N  | Y  | N  | N  | N  | N  |
|   load_sync_2_reg   | Flip-flop |   1   |  N  | N  | Y  | N  | N  | N  | N  |
=================================================================================

Inferred memory devices in process
        in routine spi_heal_dec09 line 118 in file
                '/home/st/giyer6/BG/healic/dec09_gi_spimux/temp_synthesis/spi_heal_dec09.v'.
=================================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
=================================================================================
|      idr2_reg       | Flip-flop |  72   |  Y  | N  | Y  | N  | N  | N  | N  |
=================================================================================

Inferred memory devices in process
        in routine spi_heal_dec09 line 138 in file
                '/home/st/giyer6/BG/healic/dec09_gi_spimux/temp_synthesis/spi_heal_dec09.v'.
=================================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
=================================================================================
|      pdout_reg      | Flip-flop |  64   |  Y  | N  | Y  | N  | N  | N  | N  |
|      pdout_reg      | Flip-flop |  64   |  Y  | N  | Y  | N  | N  | N  | N  |
=================================================================================

Inferred memory devices in process
        in routine spi_heal_dec09 line 152 in file
                '/home/st/giyer6/BG/healic/dec09_gi_spimux/temp_synthesis/spi_heal_dec09.v'.
=================================================================================
|     Register Name     |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
=================================================================================
| serial_data_count_reg | Flip-flop |   8   |  Y  | N  | Y  | N  | N  | N  | N  |
|     pdin_reg_reg      | Flip-flop |  64   |  Y  | N  | Y  | N  | N  | N  | N  |
|     pdin_reg_reg      | Flip-flop |   1   |  N  | N  | Y  | N  | N  | N  | N  |
|     pdin_reg_reg      | Flip-flop |   1   |  N  | N  | N  | Y  | N  | N  | N  |
=================================================================================
```

Figure 10: DC Shell Execution

2.5 Physical Design

Once the design passes timing checks, it is ready to be taken into layout. We use the SOC Encounter tool form Cadence Design Systems [30] for physical design. An outline of the physical design flow is presented here:

i. Data Preparation

Before the design is taken into the physical layout tool, the reference libraries and constraints are setup. These steps include: setting up the path of the technology file, reading in the physical (.lef) and timing (.db) libraries, reading in the timing constraints file (.sdc), I/O assignments (.tdf or .io) and setting up the TLU+ model files for parasitic RC extraction.

ii. Design Check

The '*checkDesign'* command checks the design integrity with respect to the following:

- Physical library
- Timing library
- Netlist
- I/O constraint
- Power and Ground nets and pins
- Tie-highs and tie-lows

iii. Read Design

The Netlist must be read into the design. While doing so, the *SoC Encounter* can be instructed to preserve logical hierarchies in the design. This will show up as soft-macros during the placement stage and helps group relevant cells (logic).

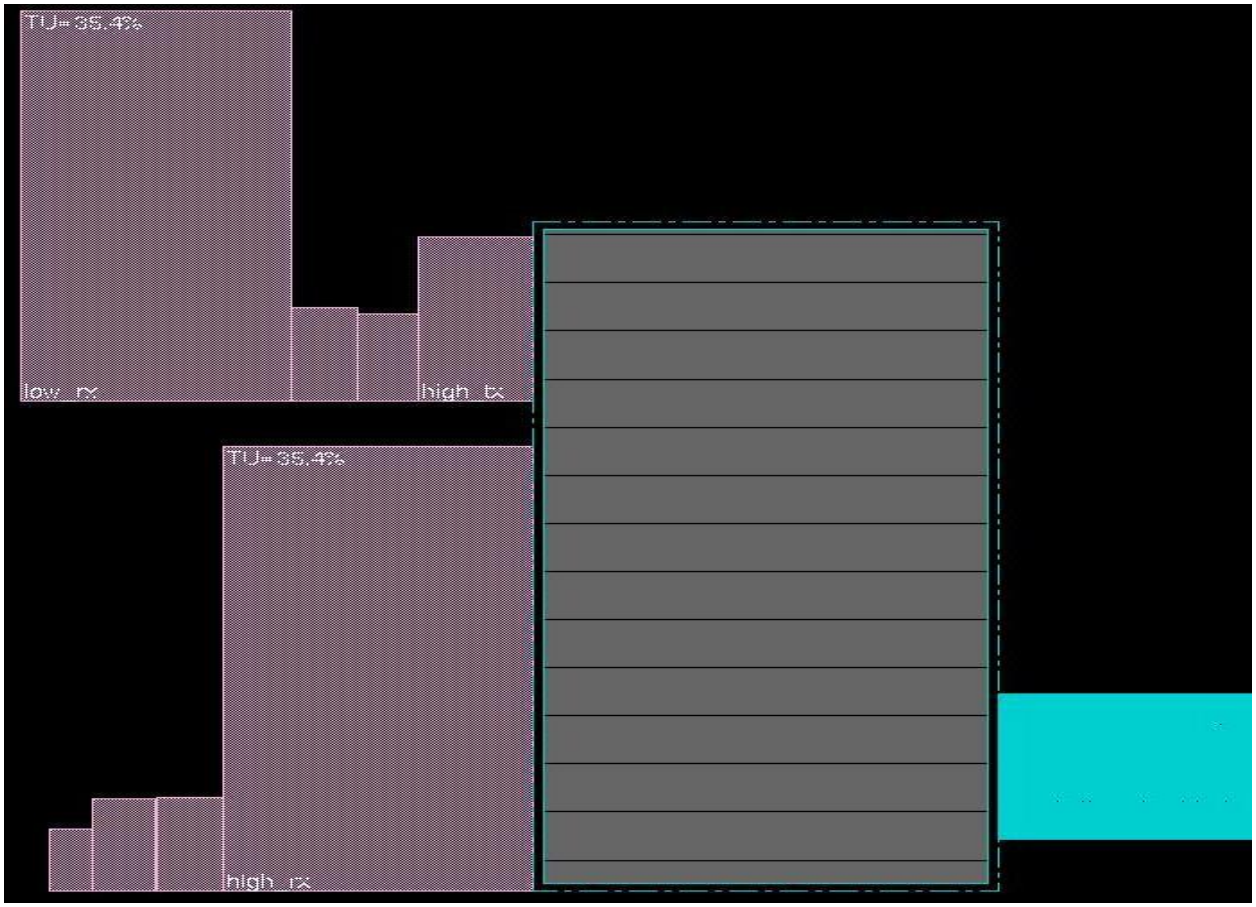


Figure 11: Design netlist read into SoC Encounter

The Netlist is made unique as part of the reading process. A unique Netlist is a pre-requisite for performing Clock Tree Synthesis (CTS), scan chain reordering and Timing optimization and closure related functions (command: uniquifyNetlist).

Another important check at this stage is the check for "assign" statements. The assign statement emulates a wire in RTL/Verilog. This is not the case with a physical design tool. In Verilog, the two ends of the logical wire can be connected to two different net names. However in physical layout as well as in reality, there exists only one unique name for that net. Hence the assign statements need to be cleanup by modifying the Netlist (*setDoAssign on*) or inserting buffers

(*setDoAssign on –buffer buffer_name*). A little care needs to be taken by manually ensuring that the timing constraints through that net are retained.

iv. Design Partitioning

To minimize the run-time and memory requirements, large designs are partitioned either in a top-down method or a bottom-up method. Both approaches involve the use blackboxes. When working with blackboxes, they need to be defined early in the flow. The logical boundary needs to be specified with intermediate ports/pins. Timing sync (synchronization) points need to be specified. Separate power domains need to be defined. When the blackbox is taken into the placement stage, the top level power mesh needs to be pushed down into the macro. The I/O and timing information needs to be pushed down into the macro. After that, the blackbox macro would be laid out as a separate design. To integrate the blackbox back into the top level design, special commands need to be used; '*loadBlackBoxNetlist'*, *'convertBlackBoxToFence'*, *'assignPtnPin'*, *'alignPtnClone'* and so on. At this stage, the partitioning on the macro can be undone and the subsequent macro can be merged with the top design if needed. The designer needs to make this engineering judgment which is primarily influenced by the timing criticality of the design. For a detailed timing closure, a flat design is preferred and hence the purpose of Unpartitioning.

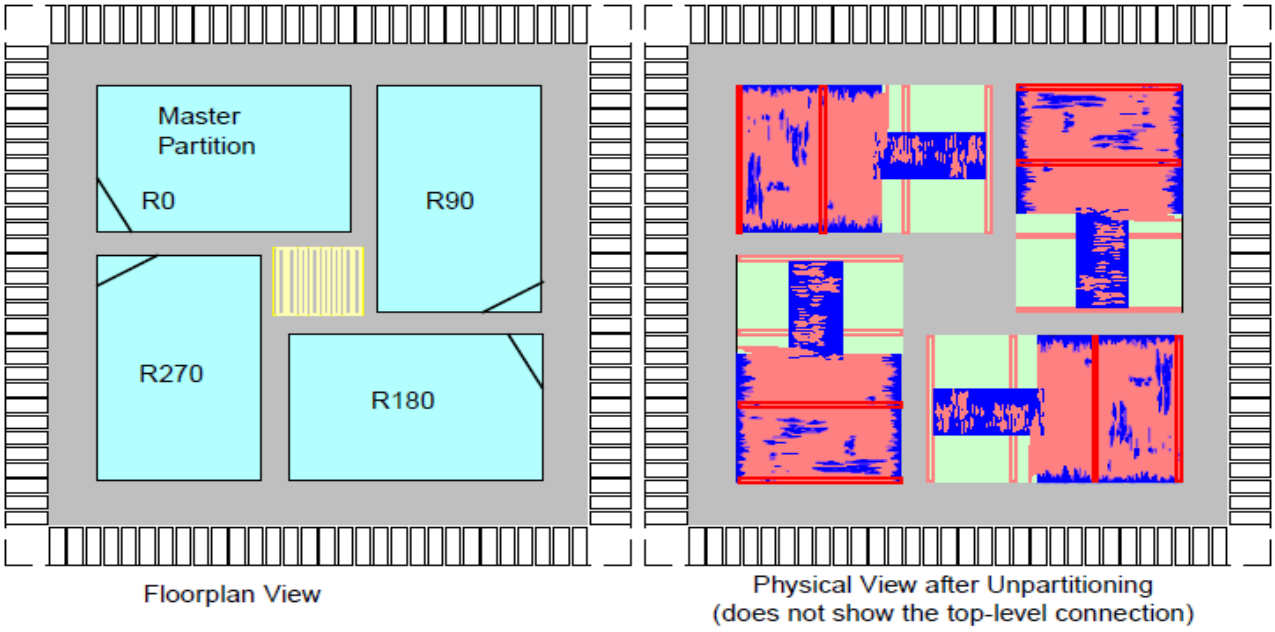


Figure 12: SoC Encounter Views for Floorplan (Macro Placement)

Figure 12 shows the blackboxes in different orientations. The fig on right shows the macros after unpartitioning. The top level wirings are not shown here. In the *Floorplan view*, we can see placement blockage at the center of the chip. [31].

v. Floorplanning and Placement

After reading in the Netlist, the design moves into the floorplan stage. Various options exist for defining the floorplan; rectilinear floorplan, core-to-io spacing, aspect ratio, cell placement utilization, number of rows and columns and so on. All this information can be saved and re-read during the design iteration. The design connectivity needs to be studied and memories, macros and blockages need to be properly placed. This is very crucial to avoid routing congestion at a later time which might trickle down to timing closure problems. Common layout practice is to place placement halos around memories. The halo is generally concentric with the inner-halo being a hard placement blockage and the outer-halo allowing decap (decoupling capacitor cells) and buffer placement.

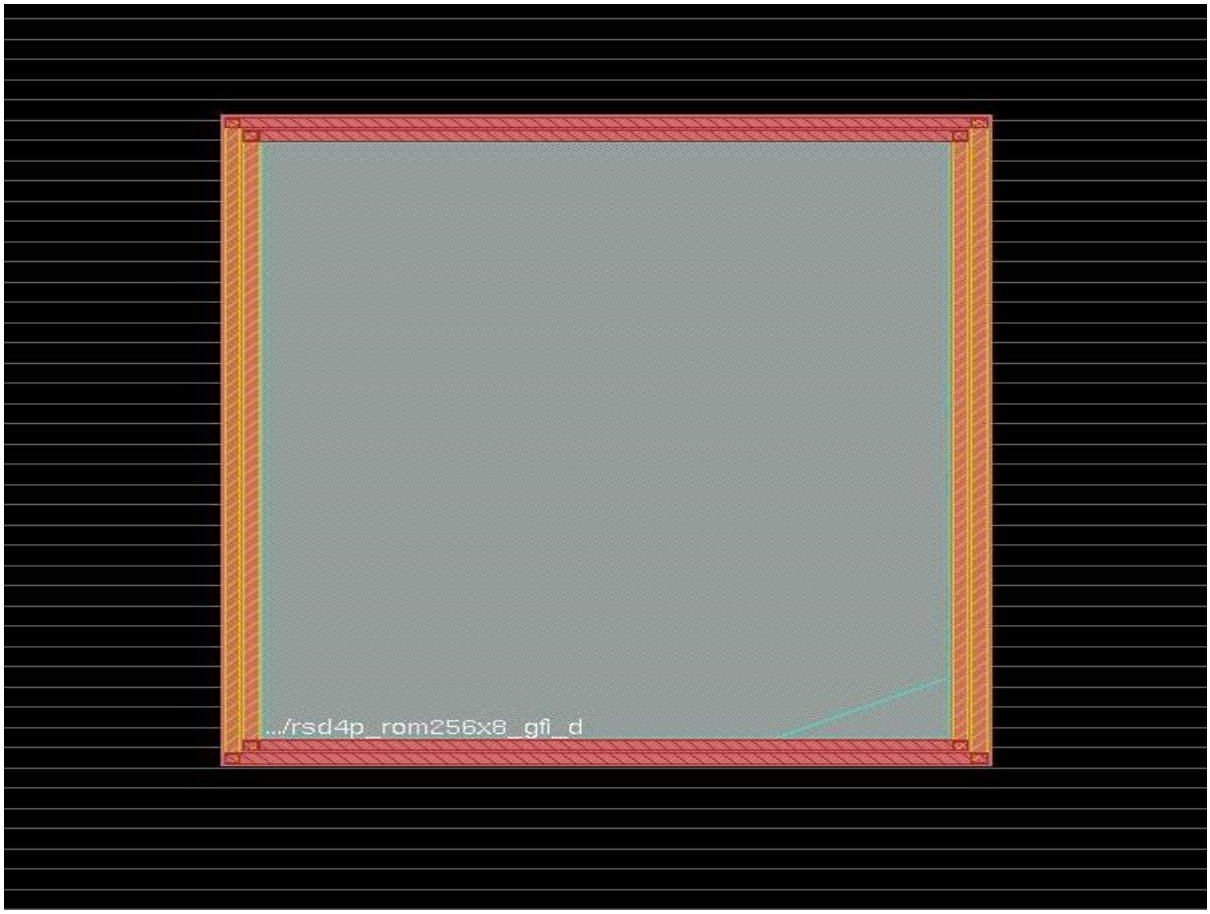


Figure 13: Placing memories with power termination rings

The design is now ready for timing driven placement of standard cells. To save up on the layout area, the standard cell rows should be flipped and abutted. The cells can be grouped by logical partitions and optimized for timing. It is a general practice to “glue” related cells together in a group and to color them. Thus the layout can be optimized in a partitioned design scenario. Pre-routes are created in the next step. Power ring and stripe widths are decided by the power and delay requirements.

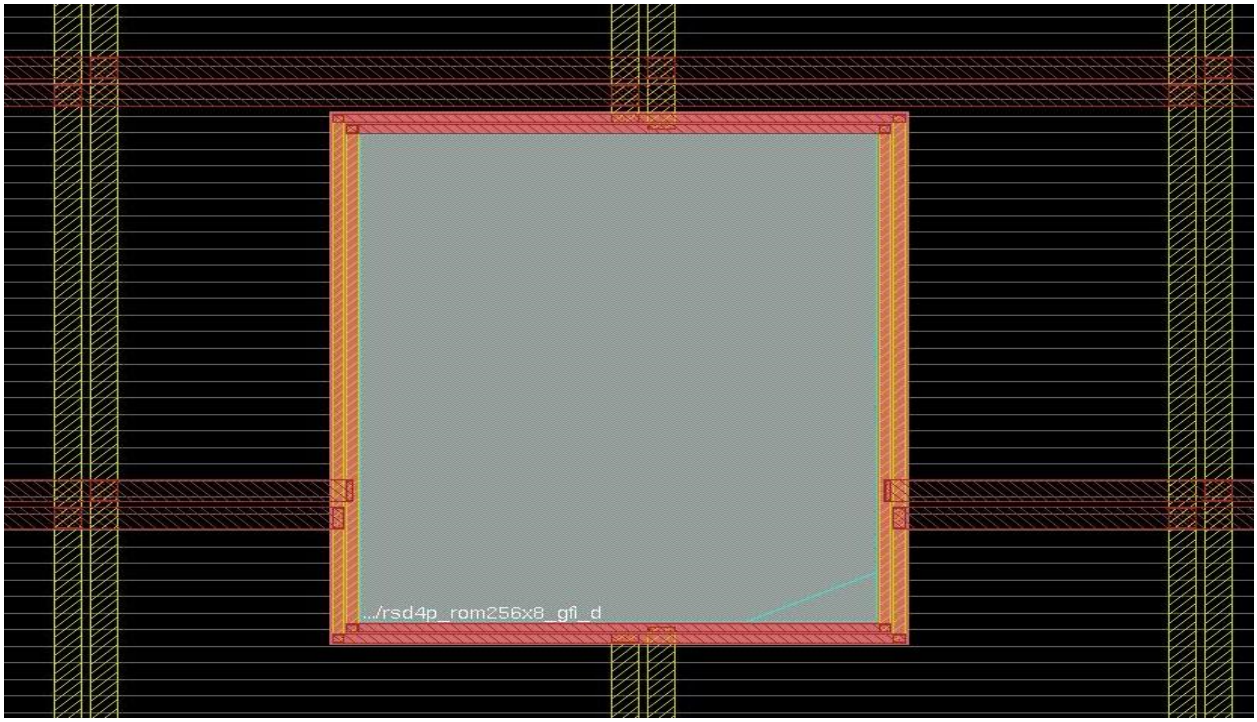


Figure 14: Power ring and stripe

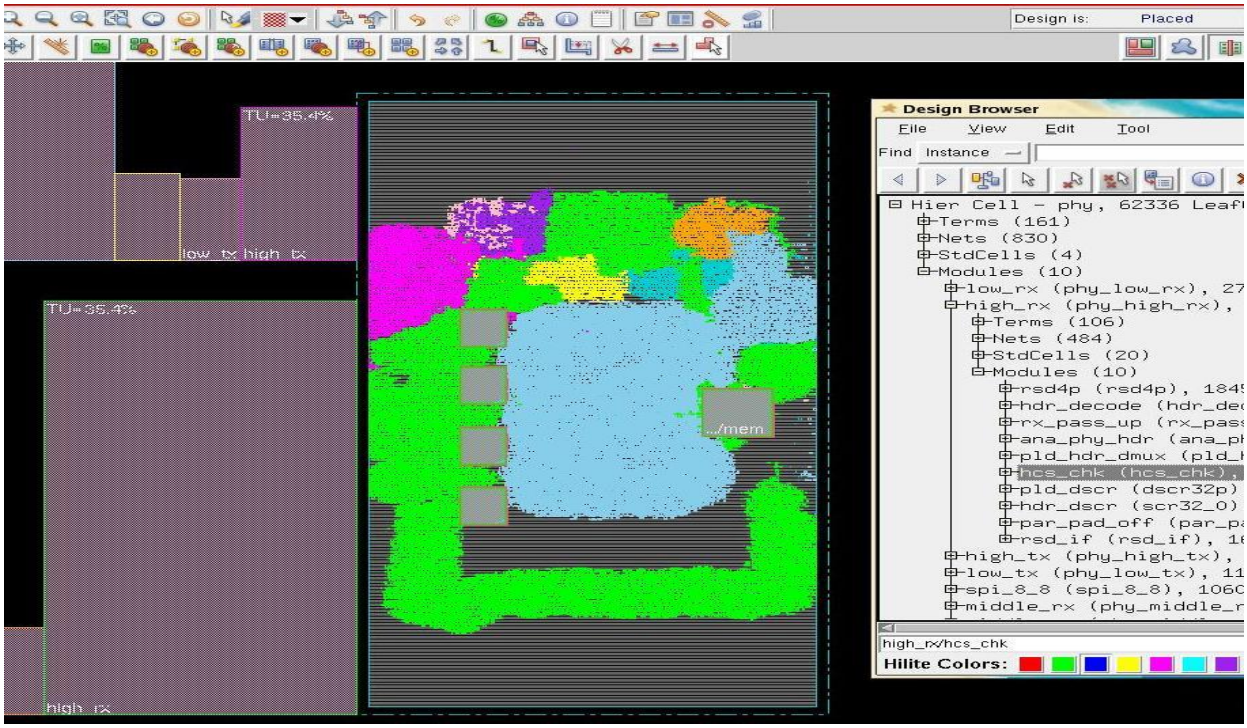


Figure 15: Standard cell groups in colors

The pins are placed during this stage. Care needs to be taken when placing the Clock and reset pins. Clock pin placement decides the H-tree structure during clock tree synthesis stage. Reset pin is also important due to the large fan-out on that net.

Certain process nodes mandate the use of tap cells to supply power to the Nwell and to ground the p-substrate. The taps need to be placed in a repetitive fashion and must lie within some specific distance from each other. Tie-off cells are used on inputs which are tied off to VDD or GND, as an additional ESD (Electro-Static Discharge) protection.

vi. Synthesizing the clock tree

The goal of Clock Tree Synthesis (CTS) [32] is to minimize clock skew and insertion delay. The process of CTS involves the usage of the following list of files:

- The timing constraints file (.sdc)
- The clock specification file (.ctstch)
- The capacitance table and
- The LEF (physical library) for clock buffers and delay cells.

Table 1 enumerates the structure of the clock tree specification file.

Table 1: Clock Tree Specification File

AutoCTSRootPin	<*clock pin name*>
NoGating	Rising
SetDPinAsSync	YES
SetIoPinAsSync	YES
PostOpt	YES
OptAddBuffer	YES
Buffer	<buffers in library>
MaxDelay	200 ps
MinDelay	0 ps
MaxSkew	50 ps
End	

Most of the attributes specified above are self explanatory. The attribute "NoGating" sets clock gating as a sink. Hence the CTS engine stops tracing through the gates and considers those inputs as rising edge triggered flip-flop clock pins. Assign a "NO" value to have a gated clock tree synthesis. We also specify to run a post CTS optimization to confirm that the timings are within bounds. Some details about the timing related terminology (skew, slew, etc) is presented later.

The timing constraints for the design need to be setup. Common practice suggests addition of some design margin into these constraints to account for routing bottle necks later in the flow. Encounter timing engine will trace the clock net throughout the design and propagate it through combinational logic. The tool is smart enough to figure out the correct phase of the clock at every flip-flop.

For gated clock tree synthesis, the clock tracing stops at:

1. A clock pin
2. An asynchronous set/reset pin
3. A user specified leaf pin (clock sink) or an exclude pin
4. An input pin without a timing arc to an output pin.

The designer can check whether the clock specifications were properly applied on the design by executing the '*ckSynthesis –check*' command and fixing any errors. The next step after this is to synthesize the clock tree on the design. The setup and hold times are reported in the command console. CTS added cells and affected nets are dumped into reports that can be read from a web browser or text editor. Special (detailed) analysis for delay variation and OCV (On Chip variation) [33] can be performed using *'setAnalysisMode'* and *'setTimingDerate'* commands. Clock reconvergence issues are discussed in a later section.

vii. Routing

SoC Encounter tool divides the core area in smaller regions called g-cells (global routing cells). This is similar to the small delta areas we consider when performing surface integrals. The entire core region is divided into routing channels which are alternating and orthogonal. Each g-cell will have about 10-20 routing resources which get assigned to the nets in the design. This helps in analyzing the congestion in the design. If some areas seem to have very high congestion numbers then we need to go back to the placement stage to reorient or to move cells around. There exist other tricks such as creating density screens, strategically placing routing blocks and so on.

Before we are ready to route the design, we need to set the correct layer stack information. Extra spacing, double via, extended via and other options need to be set. Special routes are generally done before the entire design gets routed. Clock nets would be an example of special routes.

Routing is divided into three stages: global route, detailed route and eco route. The designer needs to ensure that the design is placed and power routing is complete. Once the Nano Router has run, various reports are generated. Depending on the complexity and size of the design, there could be opens and shorts as well as DRC (Design Rule Check) violations such as *vias* under wide power nets, etc, which need to be fixed. These can be fixed with ECO routing. It is customary to run a timing check and timing optimization loop before fixing these errors. Since the routing is complete, the timing optimization engine can perform setup as well as hold fixes. This step is also called as the Post Route Optimization step. Though the timing engines used while routing and timing optimization are identical, their accuracy settings are different, with the most accurate one being sign-off timing checks.

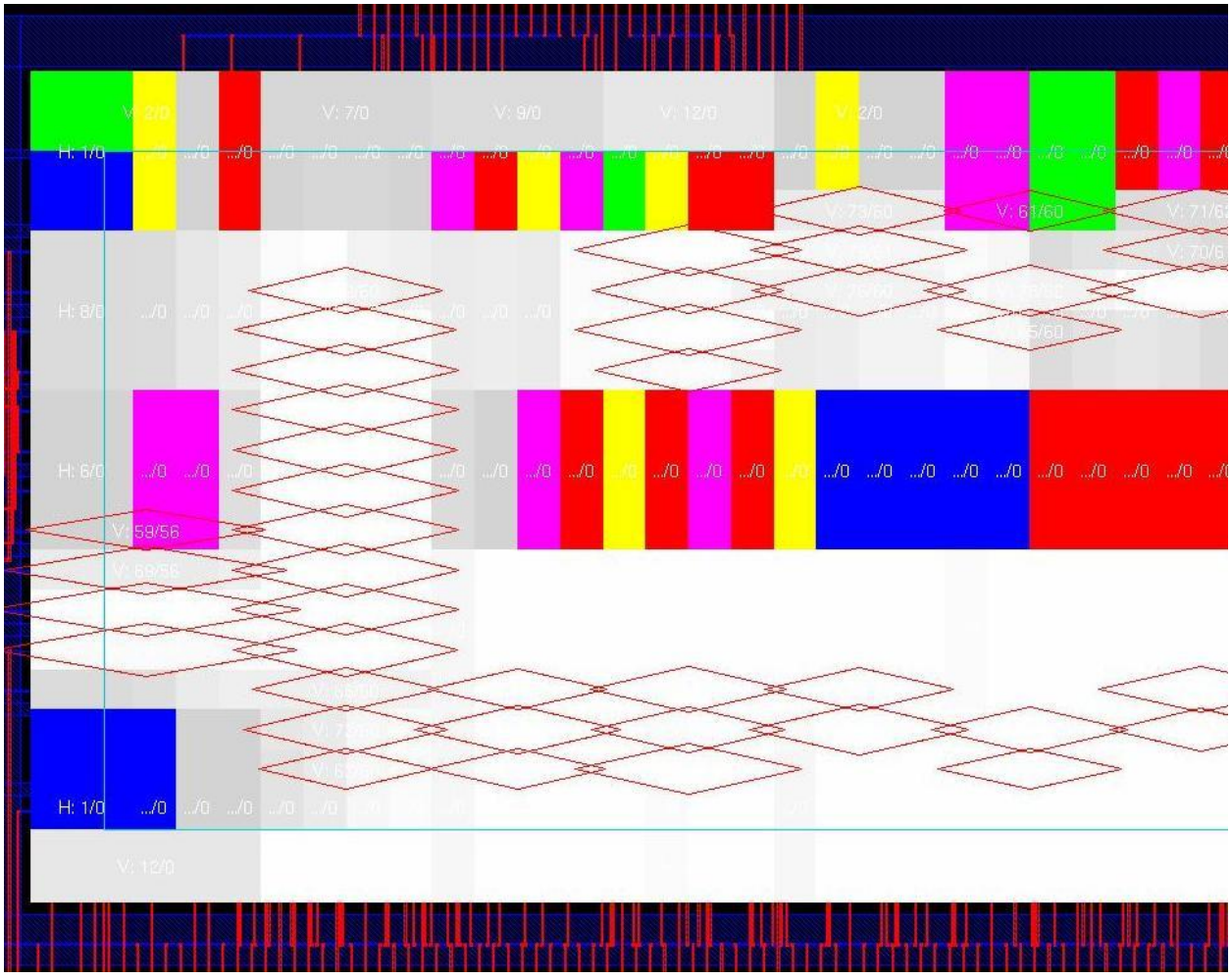


Figure 16: Congestion map and tracks used

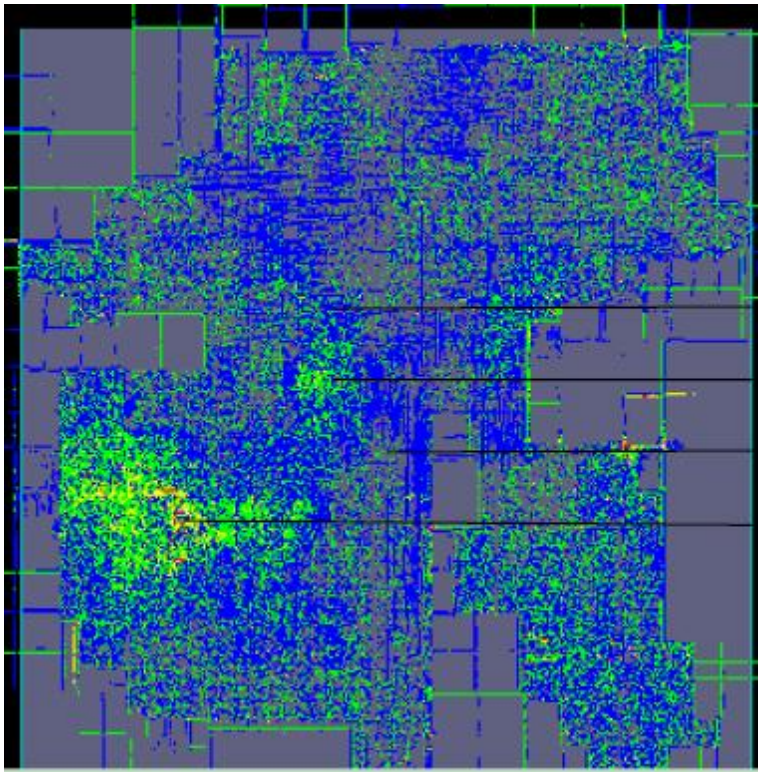


Figure 17: Showing congestion

The designer should ensure that the special routes are not messed up during detailed route. Special routes are clock routes (generally on higher metal layers to minimize parasitic caps) and analog routes (differential routing with shielding). This would be a good time to verify the geometry (DRCs) and connectivity (LVS).

viii. Design finishing

Post route optimization stage will fix all setup violations and most hold violations. Any setup violations left might need an RTL level fix, in which case, an ECO flow would be necessary. Hold violations can generally be fixed by insertion of buffers and rerouting. Sometimes hold is not fixed by the tool due to lack of space. Freeing up the placement a little helps. At the 90nm and finer technology nodes there is a minimum metal density needed for proper CMP (Chemical Mechanical Planarization). To have uniform metal thickness all over the chip, with varying thicknesses of dielectric layers, we need passive (floating chunks of metal) or active (connected to a particular net) metal fill. Making the topology of the metal layers more uniform, we can minimize the variations in metal density.

Typically metal fills are targeted to achieve around 40% density, since at that density, the impact on added capacitance is minimal and it is easier to converge on the timing of the design. Adding rectangular metal fill chunks instead of square ones, yields the same density with fewer pieces. There is statistical data to support this claim. The command *'verifyMetalDensity'* can be executed, to check which metals need filling and what locations need it the most. Unconnected metal fill adds lesser parasitic capacitance than connected fill (tie-off). Special care must be taken when adding fill over pre-laid out macros. Since the macros are already timing closed, the impact of the newly added metal fill should be kept at a minimum.

Certain process nodes/foundries would enforce via cut density checks in their DRC suite. This requirement can be met with double via insertion. Doubling the vias is beneficial from the DFM (Design For Manufacturability) standpoint too. We then run an exhaustive DRC and LVS check on the entire design to verify that there are no new spacing violations or shorts created due to the added metal fill.

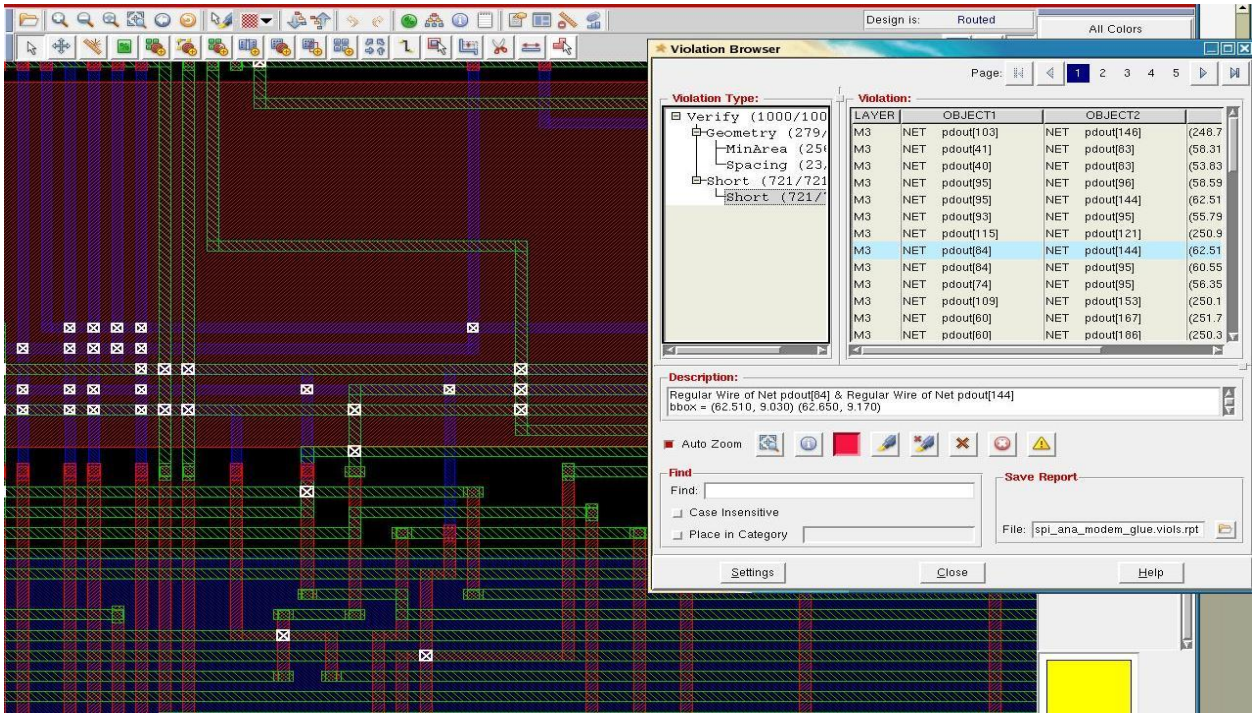


Figure 18: SoC Encounter – DRC

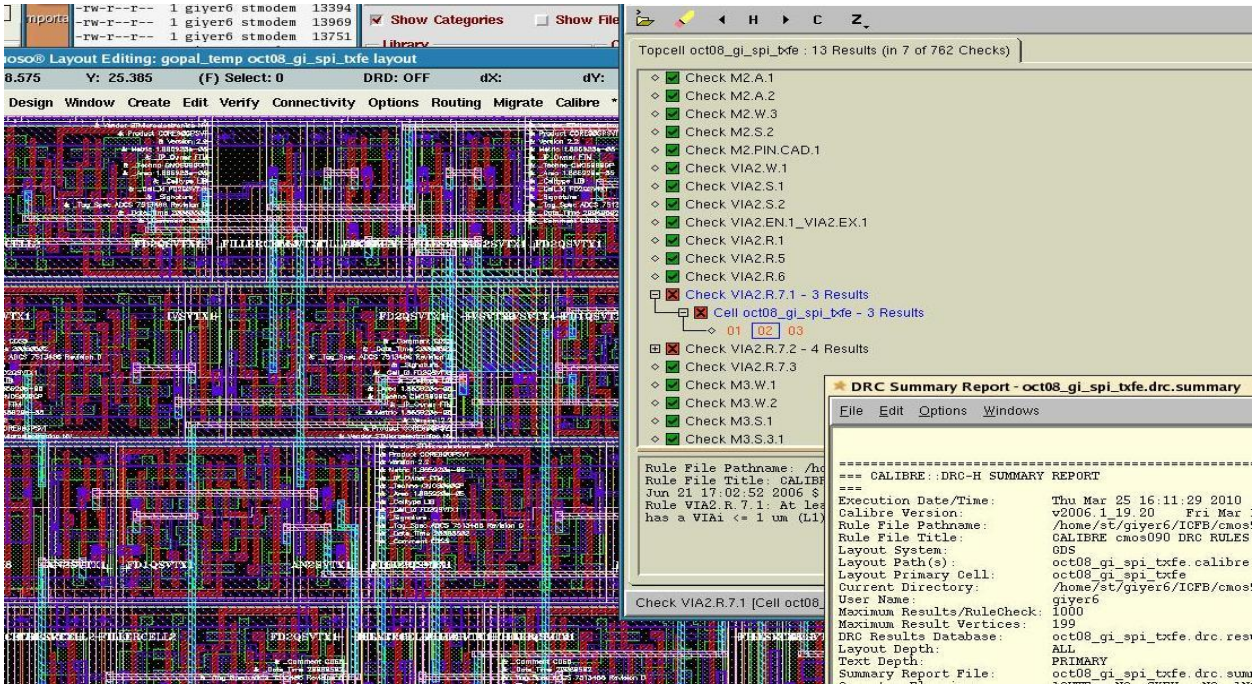


Figure 19: Calibre DRC

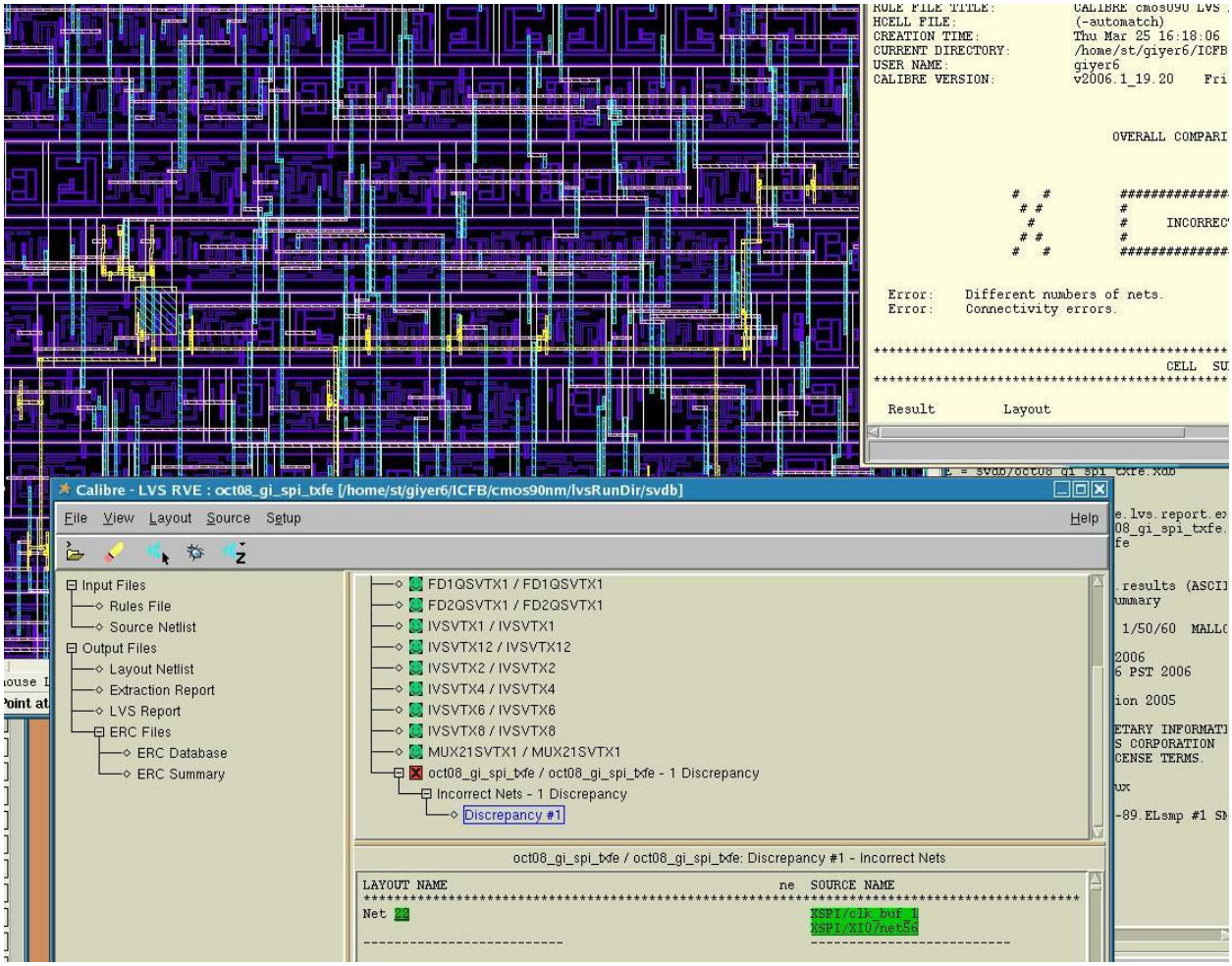


Figure 20: Calibre LVS

ix. RC Extraction, Delay calculation and Timing analysis

Routing adds significant parasitic capacitance to the design nets. This will alter the timing on different paths and that impact needs to be studied. The parasitic RC extraction flow, followed by delay calculations, SDF (Standard Delay Format delay file) generation and timing analysis will address this need.

x. Engineering Change Order (ECO) flow

An automatic ECO is run when the timing optimization step is executed. However, some of the timing fixes need manual intervention, which is when a manual ECO becomes essential. As an engineering practice, it is wise to save the design and the Netlist prior to the execution of manual ECO. The following are some of the types of ECOs:

a) Adding a Buffer

For example; when we need to intentionally skew the clock to make use of positive setup margin.

b) Changing a Cell

When the tool deems it unnecessary to up/down size a cell, yet the design calls for it. For instance when driving a large capacitive load on output ports.

c) Deleting buffers

When certain buffers cause extra delay and the tool is unable to optimize that path, those buffers might have to be deleted manually.

The Netlist changes can be viewed in a schematic browser before running ECO placement, ECO routing and ECO finishing steps (timing analysis and DRC fixes).

Routing only fixes, though do not impact the Netlist directly, can be broadly classified as a type of ECO. Avoiding Crosstalk for instance would need re-routing and sometimes might call for driver cell fixing, buffer insertion or other ECOs.

xi. Design Sign-off

Power analysis on sub-90nm designs is a must. Encounter has a built-in power analysis tool. However, for detailed power analysis – based on switching patterns and test vectors (vcd input files), designers often rely on Cadence Voltage Storm. At cutting edge technology nodes power is tightly couple with delay and hence analyzing the instantaneous power drop all over the chip becomes a necessary design sign-off metric.

Design-for-Yield and Design-for-Manufacturability steps are a part of the sign-off flow. Multi-mode Multi-corner timing analysis and optimization will take into account On-Chip Variation (OCV) effects. To optimize run-times and processing memory requirements, this timing analysis is best done outside of SoC Encounter. Major EDA vendors like Synopsys, Cadence and Mentor Graphics have a suite of toolsets to aid formal verification. However, upcoming EDA vendors like Calypto Design Services [34], etc offer quite competitive products.

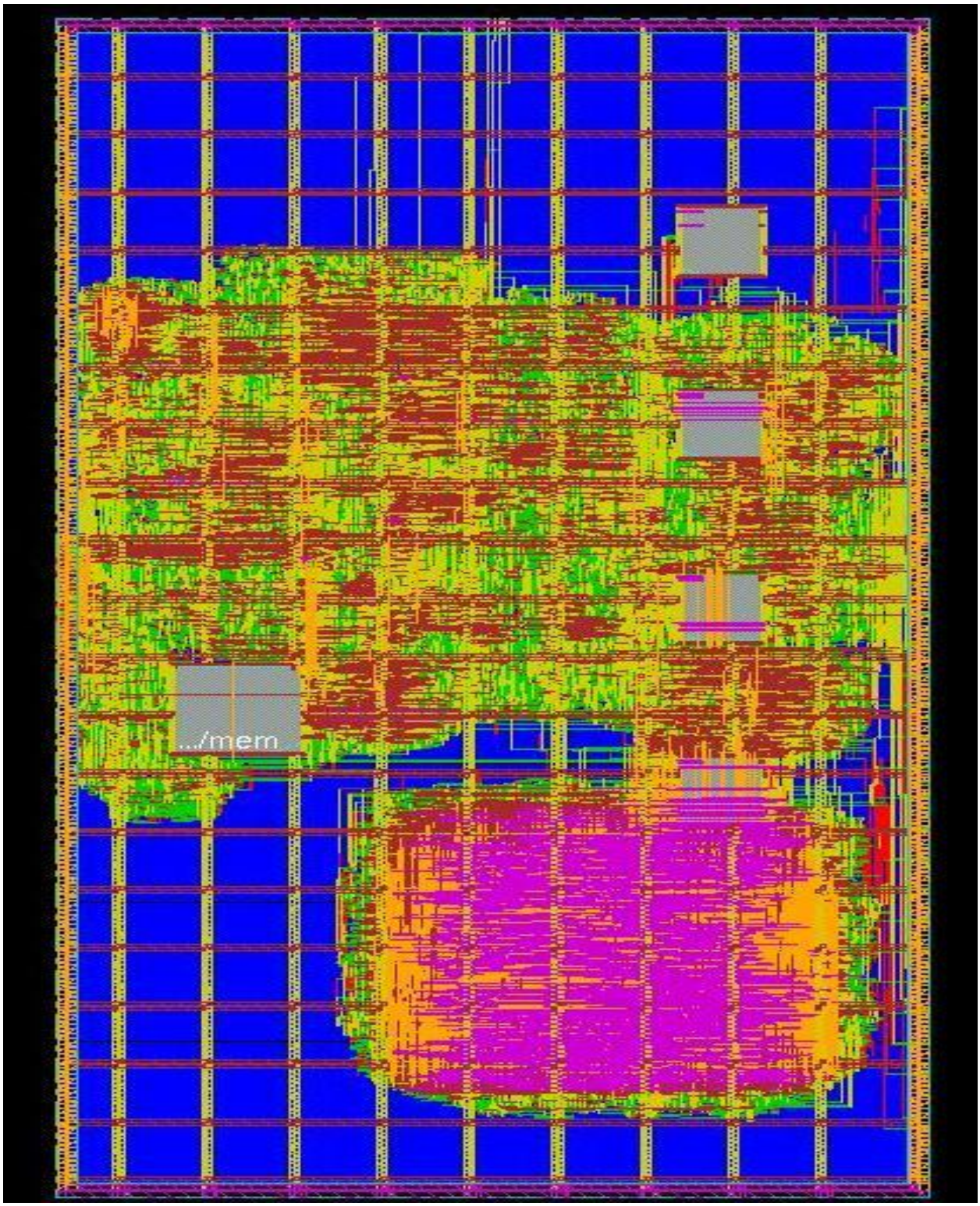


Figure 21: Design Completion

3 SERIAL COMMUNICATION INTERFACES

Various serial communication interfaces such as JTAG [35], SPI [36] and I2C [37] were evaluated for the Gigabit Digital Radio application. The following sections explain these serial interfaces in greater detail.

3.1 JTAG

JTAG (Joint Action Test Group) was initially developed as an inexpensive means of testing printed circuit boards. To support this, a device would contain additional circuitry that would allow, at a minimum, the reading and writing of the I/O (boundary) pins. Internally, the device had one big, long serial shift register with parallel load capability. The input and output of the serial shift register (Test Data In - TDI, and Test Data Out - TDO) would be externally connected to other chips on the board to form a series of serially connected shift registers called a scan chain. By using various JTAG instruction operations built into the chip it was possible to load the scan chain with bit patterns called test vectors that would allow a pin on a device to be driven high or low. Using interconnects on the PC board; it would then be possible to read this value through an input pin on another device somewhere else in the scan chain. The chain could then be serially shifted back into a JTAG tester and examined for correctness. In this manner it would be possible to check interconnects between devices as part of an inexpensive manufacturing test. This was great for things like Ball Grid Array packages where it was not possible to verify connectivity between pin and pad through direct access. The JTAG approach became quite popular for testing and evolved over time to handle other functions as well. When programmable devices started to appear they initially had their own proprietary programming interfaces. It wasn't long before manufacturers realized they could enhance the JTAG circuitry with additional User Defined instructions (something that was planned for in the JTAG spec). These instructions would allow the programming information to be shifted into the device and programmed using the same JTAG pins that were already being used for testing. Reducing pins on a device was usually a "good thing" so programming via JTAG starting being added to a number of devices. The JTAG circuitry does add a good bit of overhead to the part so it is not

very efficient for small parts, but for larger devices it can be extremely cost effective. Not only does it (normally) allow you to read and write the boundary pins of a device but the internal chain can also include access to otherwise inaccessible information that can be extremely valuable to the chip designer. And, another big achievement - it has been used extensively to add debugging functions to the device, such as the ability to single step, read and write internal registers and so on. Figure 22 shows the typical JTAG connection between multiple devices in a system scenario. Figure 23 shows the timing diagram.

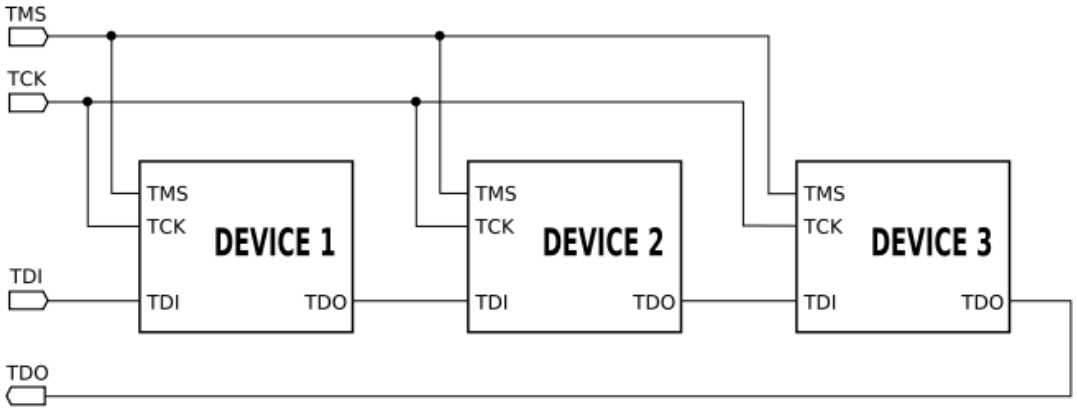


Figure 22: JTAG Interface

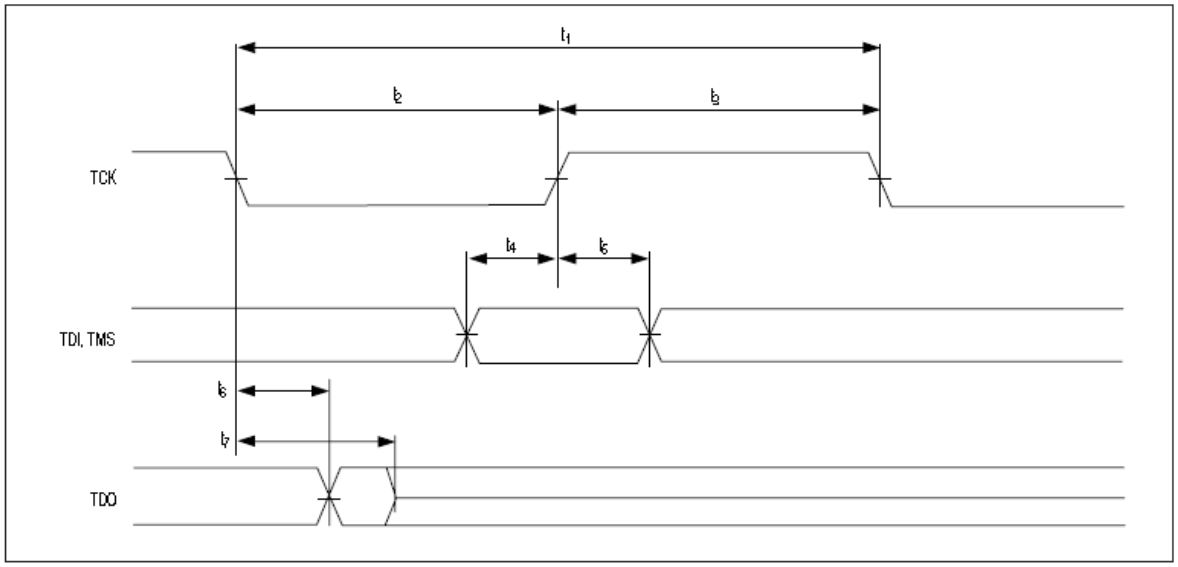


Figure 23: JTAG Timing Diagram

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
TCK Clock Period	t_1			1000		ns
TCK Clock High/Low Time	t_2, t_3	(Note 8)	50	500		ns
TCK to TDI, TMS Setup Time	t_4		15			ns
TCK to TDI, TMS Hold Time	t_5		10			ns
TCK to TDO Delay	t_6				50	ns
TCK to TDO High-Z Delay	t_7				50	ns

Figure 24: Timing numbers for DS4550 [37]

3.2 SPI

The Serial Peripheral Interface (SPI) was originally designed as a means of connecting peripherals to a central processor using a minimum of interconnect pins. Typically, this was used where access to the peripheral was infrequent or the transfer rate was unimportant. One of the initial popular uses was small non-volatile memories that could be used to store configuration information. Today the interface is used on a wide range of products from real time clocks to UARTs. The interface does share some of the characteristics of the JTAG interface; it communicates serially over separate input and output lines, and thus contains an internal serial shift register. Although not mandatory in either case, JTAG normally connects the internal shift registers of multiple devices in one long serial shift register chain where SPI normally connects multiple peripherals in parallel. In other words, with SPI the pin that transfers data from a controller (master) to a peripheral (slave) can be connected to the input pin of multiple devices. The pin is called Master Out Slave In (MOSI) which is a really nice way of removing any ambiguity regarding the direction of data transfer versus a specific chips viewpoint. Conversely, multiple outputs from the various peripheral chips (Master In Slave Out or MISO) are connected together going back to the controller. These outputs are always tristated unless a device is specifically selected for data transfer by the master and is in the process of transferring data. Like JTAG, SPI has evolved to handle other functions as well, such as low speed communication between processors where one acts as the master and one as the slave during any given transfer. Also, like JTAG, since SPI transfers data into and out of a device, its usage can be extended to programming that device.

In the standard SPI protocol, the data is shifted in on the rising edge of the clock and is shifted out on the falling edge of the clock. Figure 25 shows a SPI based system with a single master and multiple slaves. Each of the slaves needs a separate *'chip_select'* signal. However, in pin limited systems, so many connections come at a premium. To work around this problem, the slaves can be daisy-chained as shown in Figure 26. All the slaves can be controlled using a single chip select pin. However the disadvantage in doing so is the delay in accessing some of the slaves. Access to every slave will go through each of the other slave devices and hence slows down the entire communication protocol. A typical timing diagram is shown in Figure 27.

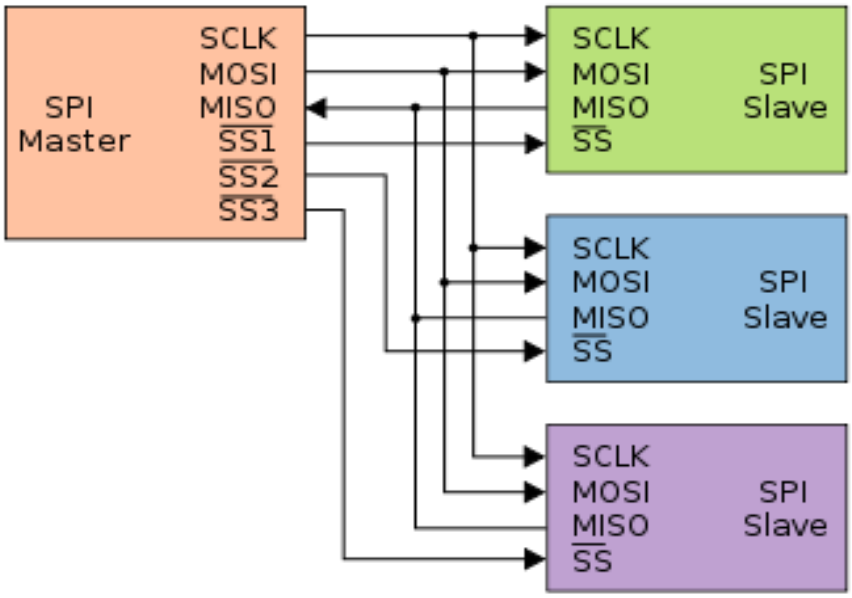


Figure 25: SPI Subsystem depicting Master-Slave Configuration with individual slave select

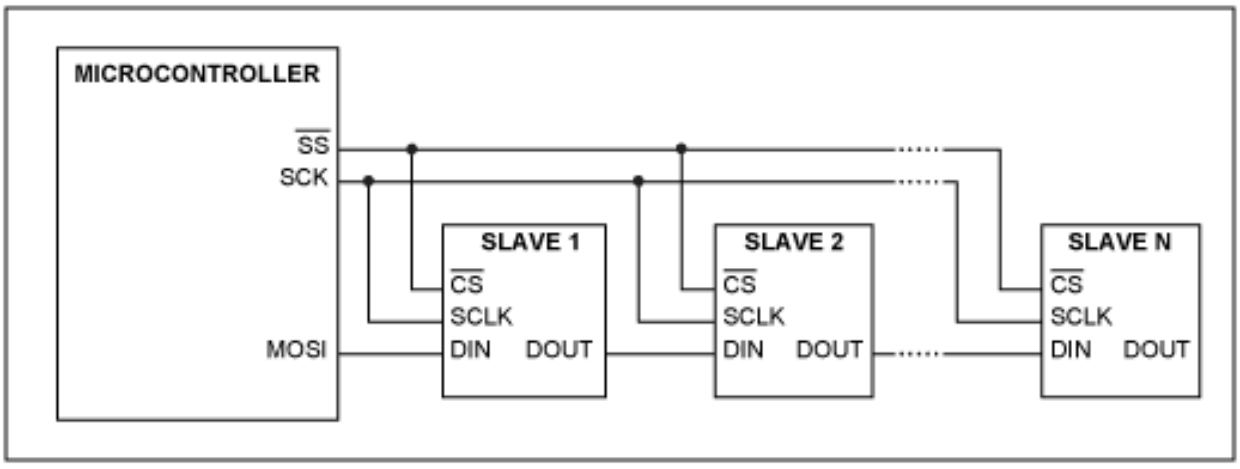


Figure 26: SPI Subsystem in Daisy-Chain Format

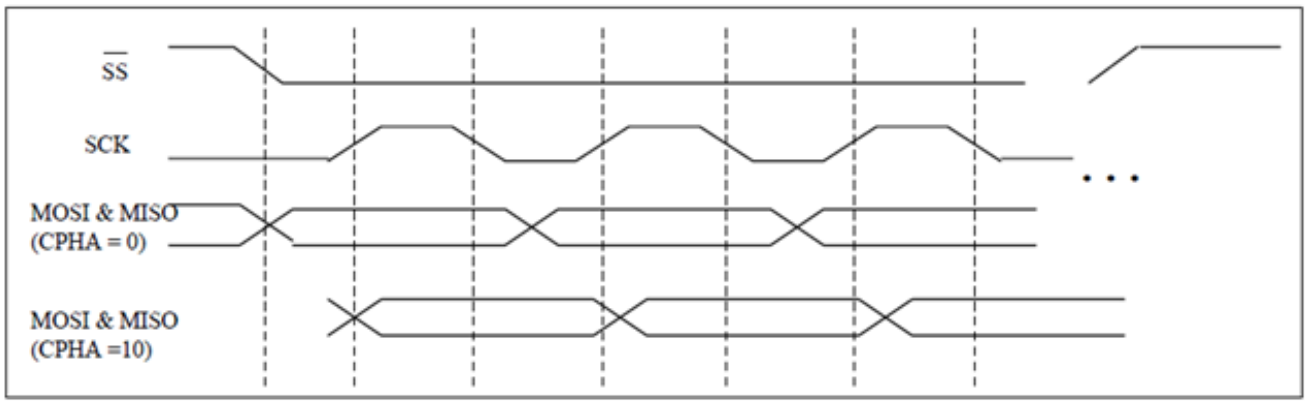


Figure 27: SPI sample timing diagram

3.3 I^2C

I²C is an acronym for Inter Integrated Circuit bus. I²C is a 2-wire serial interface standard defined by Philips Semiconductor in the early 1980's. Its purpose was to provide an easy way to connect a CPU to peripheral chips in a TV-set. The BUS physically consists of 2 active wires and a ground connection. The active wires, SDA and SCL, are both bidirectional. Where SDA is the Serial Data line and SCL is the Serial Clock line.

The key advantage of this interface is that only two lines (clock and data) are required for full duplexed communication between multiple devices. The interface typically runs at a fairly low speed (100 kHz to 400 kHz). With I2C, each IC on the bus has a unique address. Chips can act as a receiver and/or transmitter depending on its functionality.

The I²C-bus is developed by Philips to maximize hardware efficiency and circuit simplicity. The I²C interface is a simple master/slave type interface. Simplicity of the I²C system is primarily due to the bidirectional 2-wire design, a serial data line (SDA) and serial clock line (SCL), and to the protocol format. Bi-directional communication is facilitated through the use of wire and connection (the lines are either active-low or passive high). The I²C Bus protocol also allows collision detection, clock synchronization and hand-shaking for multi-master systems. The clock is always generated by the master, but the slave may hold it low to generate a wait state. In most systems the microcontroller is the master and the external peripheral devices are slaves.

The maximum number of devices connected to the I²C bus is dictated by the maximum allowable capacitance on the lines, 400pF, and the protocol's addressing limit of 16k; typical device capacitance is 10pF. The I²C protocol has 127 addresses available. The original vision was to assign addresses by device function, but when Philips began to sell microcontrollers for I²C, the address could be programmed, eliminating the need for a Philips-assigned address.

A device that controls signal transfers on the line in addition to controlling the clock frequency is the master and a device that is controlled by the master is the slave. The master can transmit or receive signals to or from a slave, respectively, or control signal transfers between

two slaves, where one is the transmitter and the other is the receiver. I²C bus support more than one master connected to one bus.

The I²C bus is an innovative hardware interface which provides the software designer the flexibility to create a truly multi-master environment. It is possible to combine several masters, in addition to several slaves, onto an I²C-bus to form a multi-master system. If more than one master simultaneously tries to control the line, an arbitration procedure decides which master gets priority.

To begin communication, the bus master (typically a microcontroller) places the address of the device with which it intends to communicate (the slave) on the bus. All ICs monitor the bus to determine if the master device is sending their address. Only the device with the correct address communicates with the master.

What is notable about the I²C architecture is that the Slaves can hold the SCL signal low until the slowest slave is ready for data transfer. However, the clock is always generated by the Master and is an input into the slaves.

The I²C protocol also allows for multiple masters in the system. Each master senses the SDA port and does its own bus arbitration.

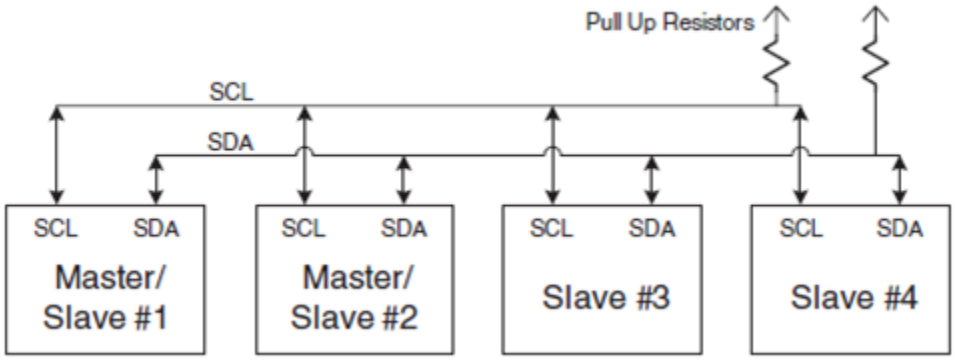


Figure 28: Typical I2C Bus System

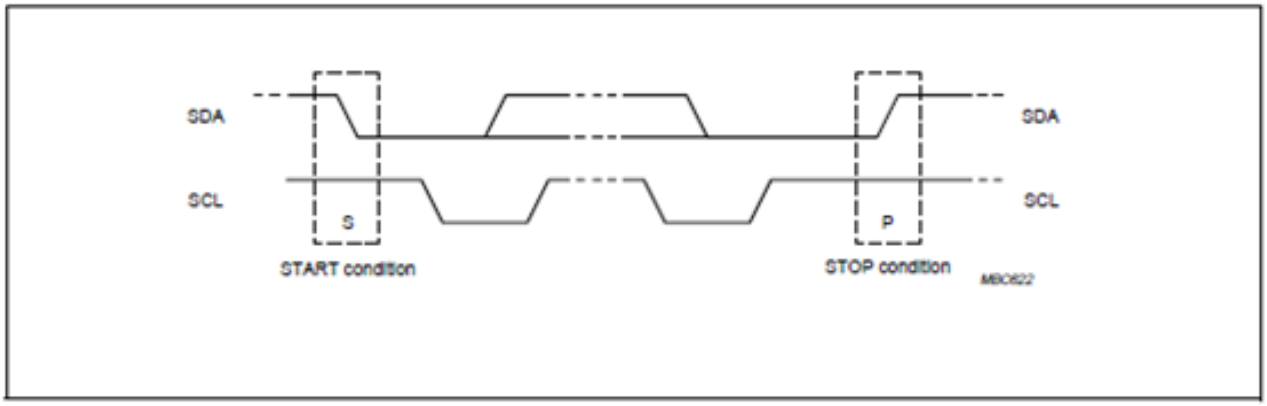


Figure 29: START and STOP Conditions

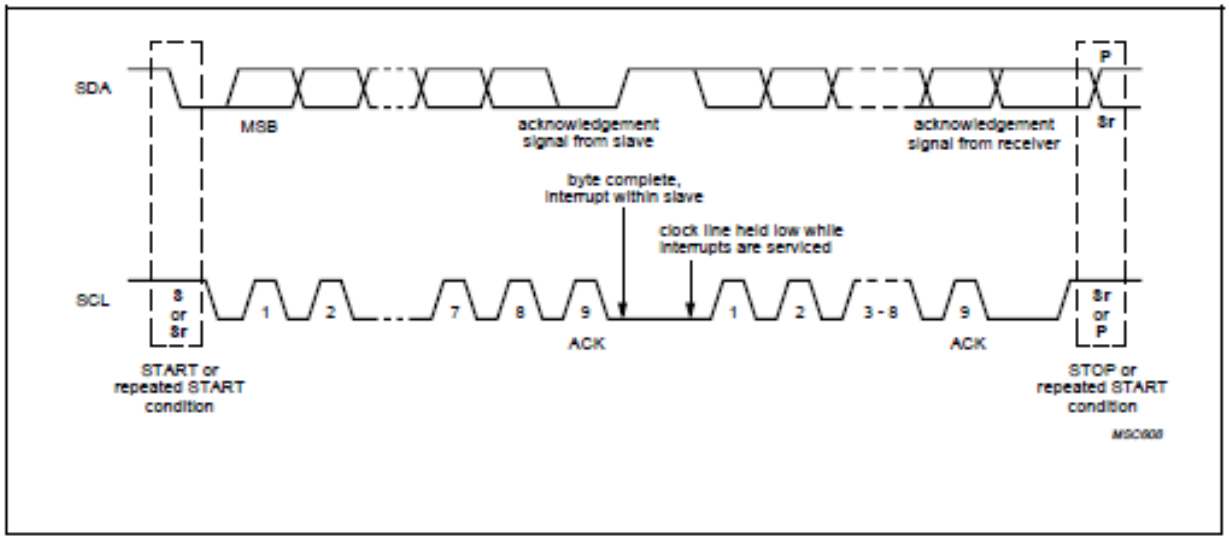


Figure 30: Data Transfer on I2C Bus

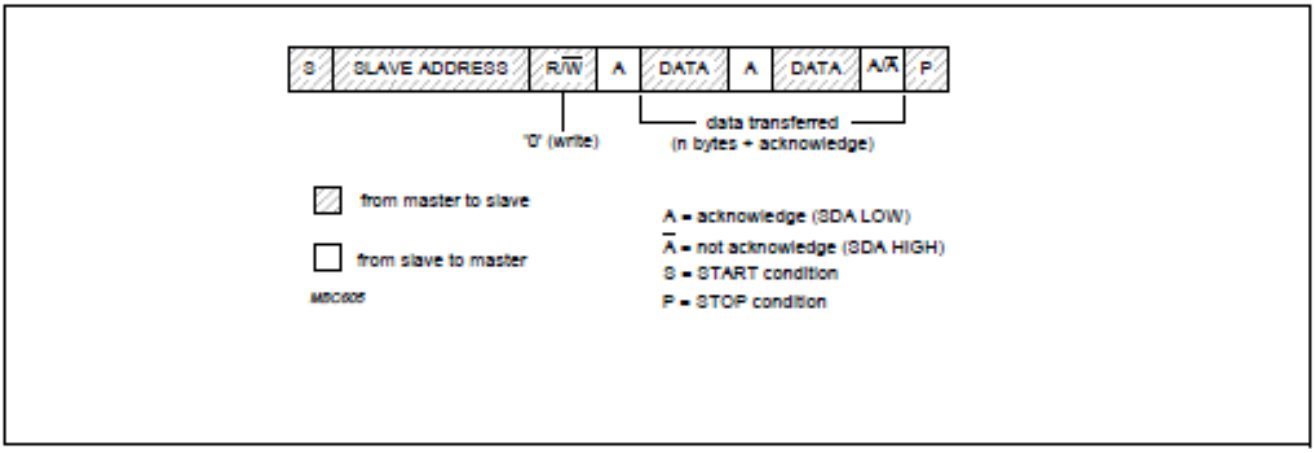


Figure 31: A master-transmitter addressing a slave receiver with a 7-bit address.

The transfer direction is not changed.

3.4 Choice of SPI as the chosen serial communication interface

A comparison of the three serial interfaces is shown in **Table 2**.

Table 2: Overview of SPI, I2C and JTAG Serial Communication Protocols

Protocol	Architecture	Multi-Master	Data Rate	Flyby Data transfer	Full Duplex
SPI	4-wire	No	1Mbps	No	Yes
I2C	Shared data and clock signals	Yes	400Kbps	Yes	No
JTAG	Daisy Chain Data Signals	No	100MHz	N/A	N/A

As is seen in the comparison table, SPI is the only protocol where the data rate can be scaled well into 100s of Mega bits per second. The Digital RF CMOS radio needs over 10 slaves which would mean we need an equal number of chip select lines. In a SoC (system-on-chip) scenario where the silicon real estate is at a premium, having these many chip select lines is not always an option. Hence we develop a new serial communication interface which is a blend of the SPI and I^2C protocols. It has a blend of the SPI signaling protocol along with the I^2C address based approach juxtaposed with some newer features, which push the operating speed over 750 Mbps. Even higher data rates are possible at the cost of burning more power. The next chapter looks at this system in detail.

4 THE SPI SUB-SYSTEM FOR CMOS DIGITAL RADIO CUM MODEM

4.1 Block Description

The block diagram of a typical SPI Slave is shown in Figure 32.

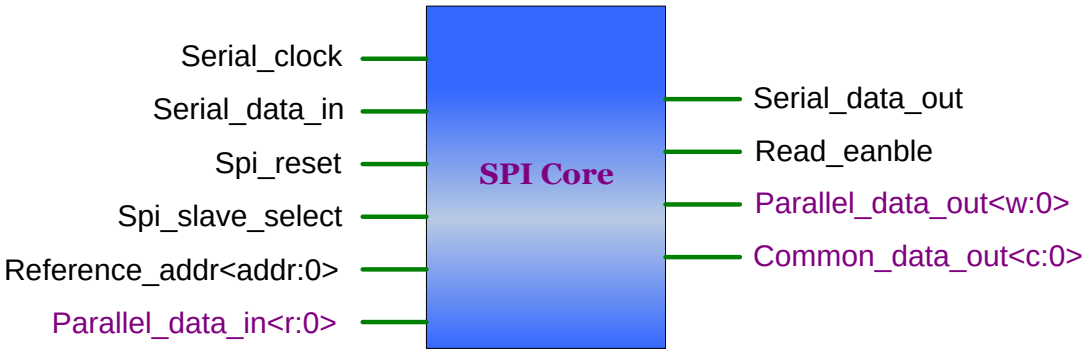


Figure 32: Typical SPI Slave

Table 3 harbors the pin description of the serial communication block.

Table 3: Pin Description for SPI Slave

Signal	**Function**
Serial Data In	SPI Data is serially pushed into the SPI block
Serial Data Out	SPI data is read out serially on this pin
Clock	SPI clock pin. All registers are triggered on the rising edge of the clock. Clock pulses should be sent for each data bit being pushed in/read out. This would imply that we need to send clock pulses even after the read load signal has been sent. (as many as the width of the data being read out)
Slave Select	This signal is pulsed after the data has been written into the SPI module (write mode) and after the control word has been written into the SPI (read mode). The rising edge of this pulse will transfer the previously sent serial data onto the SPI parallel bus.
Reset	Resets all the registers inside the SPI module. This is an active low signal.
Read Enable	This active high signal signals the availability of the serial data on the *Serial_Data_Out* pin. This pin is not available outside the chip.
Reference Address	4 bit address hardwired for individual SPIs (a1-a4)
SPI Read Bus	A parallel bus input into the SPI block.
SPI Write Bus	A parallel output of the SPI block. This data changes during the individual SPI write sequence.
Common Control Word Bus	A 14 bit parallel output of the SPI bus. This data changes during the common SPI write sequence.
Common Control Mode	A ‘1’ indicates Common word operation and a ‘0’ indicates the individual SPI operation.
Reset Out	This is an active low output from the SPI block (*resetb_out*). It is zero in the default mode. This output is intended to be used as *Power On Reset* and users can use this to reset their register files. This signal goes low when, either the global reset (rstb) goes low or a reset sequence has been exercised on the SPI block.

Since we intend to use only one chip select line to control every slave device, an address based system is needed. As part of the transmitted data sequence, the first few bits represent the address bits. The data bits follow these address bits. This serial stream is shifted into every slave. Within every slave, the received address is matched with a hard-wired address and when a match is found, the data is loaded into that particular slave. The rest of the slaves discard the serially shifted data. A typical hookup of the SPI master-slave system is shown in Figure 33.

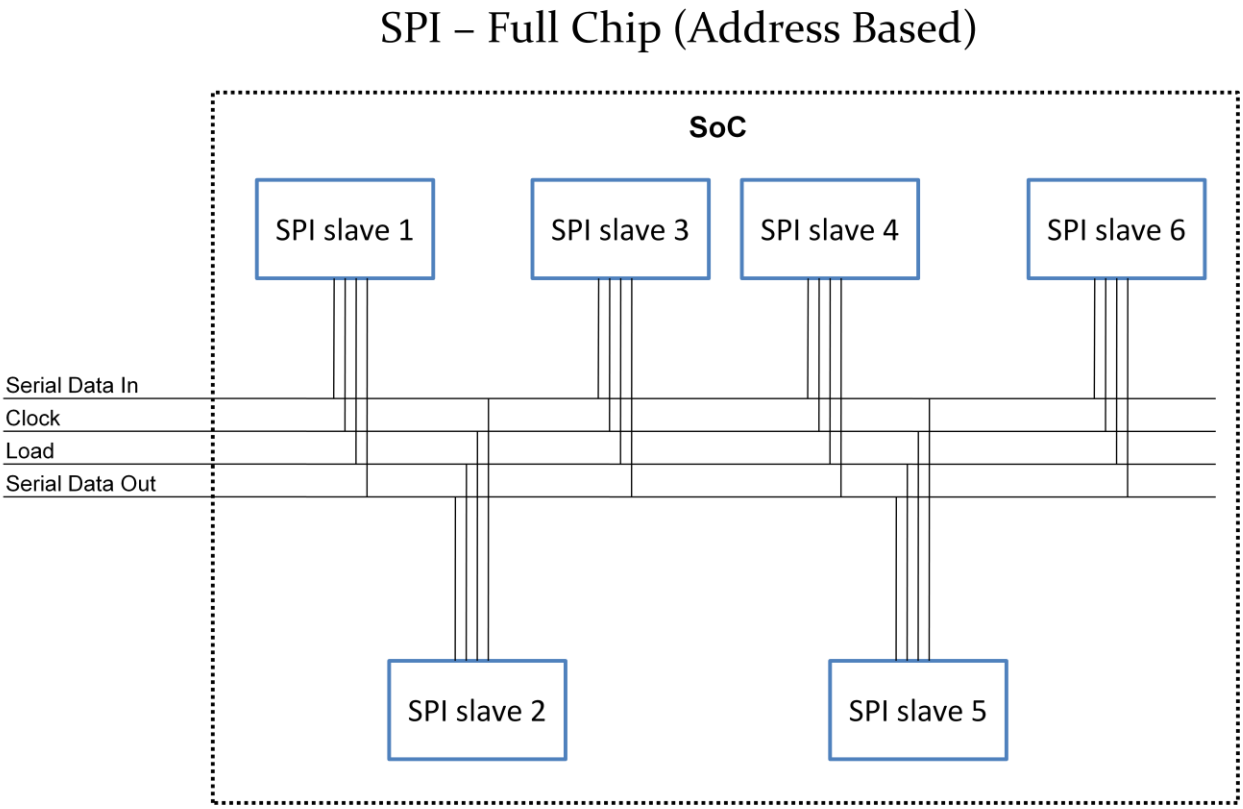


Figure 33: Typical SPI Subsystem with Multiple slaves

The outline the Version 1 of the SPI sub-system is presented in the coming section which is followed up by a further detailing of the Version 2 SPI and its various modes of operation.

4.2 Version 1 of the SPI sub-system

4.2.1 Overview

In 2008, the digital flow using a place and route tool was not completely setup. It was being used for the cell layout only. Rest of the design was done in the Cadence Virtuoso ICFB environment. As seen in chapter2, functional simulations were not performed on the design, but detailed transistor level spice simulations were. This was a very time consuming process and for the lack of sufficient time, exhaustive testing on the sub-system was traded off.

The standard data sequence was as follows:

Addr<0:4>	Read bit	Write bit	Long Bit	Write_bits<w:0>

4.2.2 Modes of Operation

- Write Mode: Write bit is high. The data in the serial stream is loaded into the spi_write bus. Only the short word is loaded.
- Long Mode: Long Bit is high. This means all the write bits will get loaded. If this is 0 only the short word (length specified in table) is loaded.
- Read Mode: Read bit is high. The parallel word from the spi_read bus is loaded onto the *dout* line serially.

For V1 of the SPI slave, two different write modes were designed based on the system requirements. The RF modules in the Digital Radio chip, need tuning or parameter updates fairly quickly and repetitively. For that reason, these bits were clubbed together as a “short write” and could be accessed easily (typically 12-16 bits). The rest of the bits (typically 120 bits or more) were part of the extended write or “long write” which took much longer to access.

4.2.3 Cell Naming Convention

Each of the SPI slaves is given a name which indicates the read and write widths as shown in Figure 34.

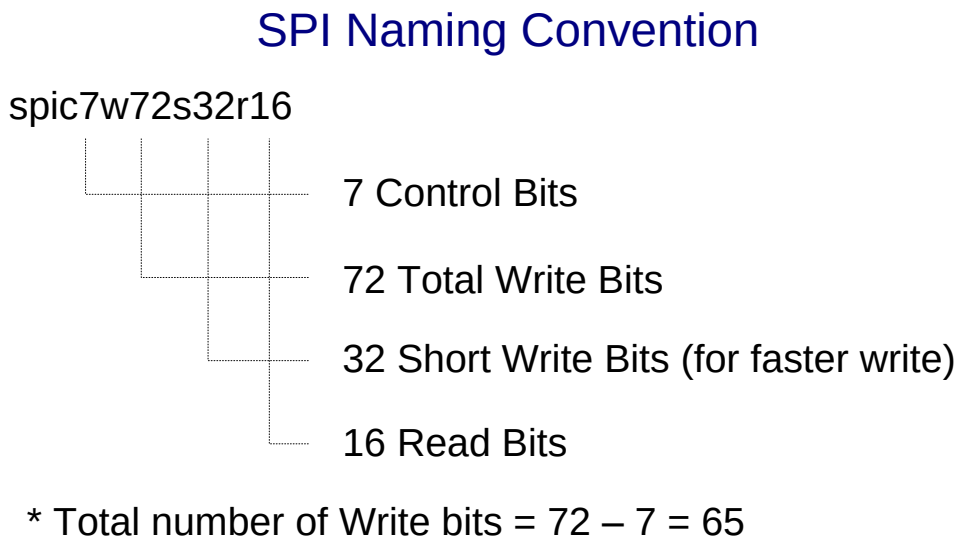


Figure 34: SPI slave naming convention

4.2.4 Typical SPI sub-system address space

Table 4 shows the SPI slave address space used on October 2008 ST90nm CMOS Digital Radio chipset.

Table 4: SPI Subsystem Address Space

User/Block	Bits					Cell Name	Address
	Total	Control	Write	Short Wr	Read		refadd[0:3]
Power Amplifier	16	6 (4a+1r+1w+0L)	10	0	8	Spic6w16s0r8	“0000”
LNA+Mixer	32	7 (4a+1r+1w+1L)	57	16	8	Spic7w64s16r8	“0001”
Analog Modem	128	7 (4a+1r+1w+1L)	121	32	16	Spic7w72s32r16	“0010”
RF Amplifier	48	7 (4a+1r+1w+1L)	41	16	8	Spic7w48s16r8	“0011”
VCO	72	7 (4a+1r+1w+1L)	65	8	8	Spic7w72s8r8	“0100”
Data Converters	8	4 (4a+0r+0w+0L)	4	0	0	Spic4w8s0r0	“0101”
Digital Modem	56	7 (4a+1r+1w+1L)	49	16	8	Spic7w56s16r8	“0110”

4.2.5 Top Level SPI System

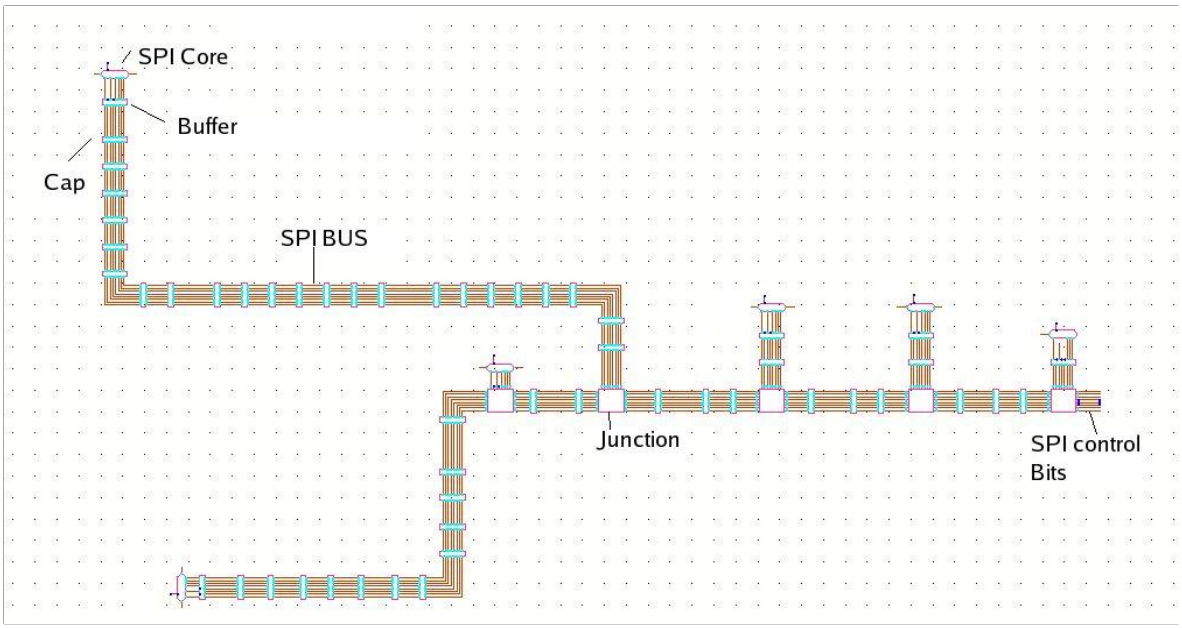


Figure 35: Top Level SPI System

The SPI control bits from the input consist of the following signals: *datain, clock, load, read_enable and reset*. These will be routed to each individual SPI through the top level bus with buffers and capacitors inserted at regular intervals. Since the design was entirely done in the cadence schematic entry environment, buffering on the data bus had to be done at this stage (pre-layout). The assumption being made was that the delay impact would be uniform over all the signals. The SPI digital bus had deep penetration inside the RF core, crosstalk was a major concern. To mitigate this risk, sufficient decoupling capacitance was added all through the digital bus. This however mandated an increased use of buffering to keep the slew rates within bounds. A ground shield was laid out on both sides of the data bus.

4.2.6 SPI Core Design (for spic6w16s0r8)

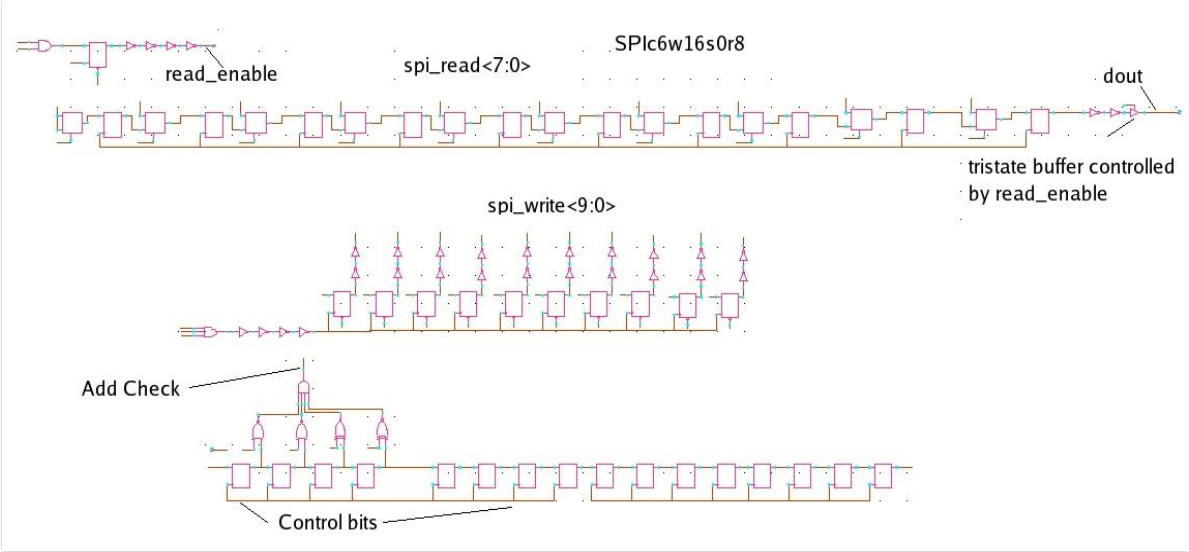


Figure 36: SPI Core Design

As shown in Figure 37, the first few bits comprise of the address bits. An *'add_check'* signal is generated in the address decoder block, which is used as a gating signal for loading the data bits. The register space is made up of two levels of registers clocked on different clocks as seen in Figure 37.

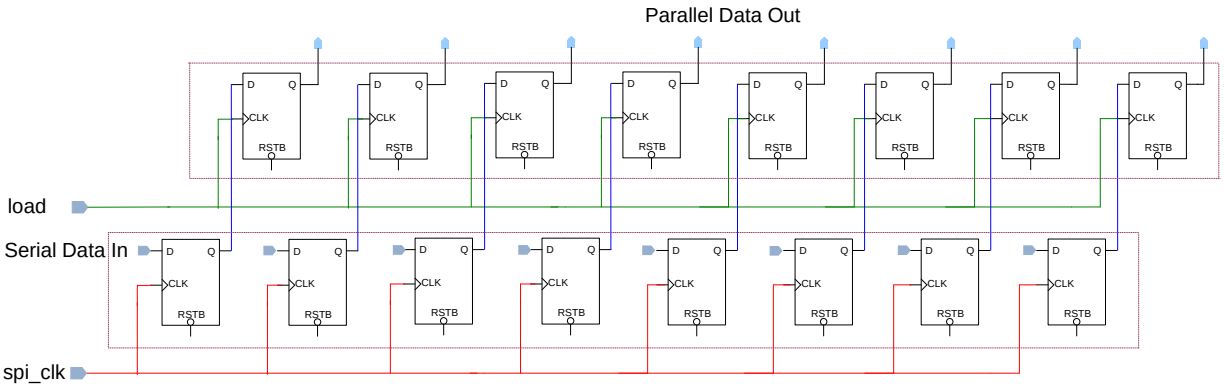


Figure 37: Two level of registers clocked by spi_clk and load

The read_enable signal stays high when serial data is being shifted out of the SPI. This is used in the junction blocks shown in [fig] for bus arbitration. All of the serial data out ports are wired-or since only one of them would be operational at any given point in time. Hence, tri-state buffers are used before wire-or'ing two signals, with read_enable providing the gating check.

4.2.7 SPI Junction Circuitry

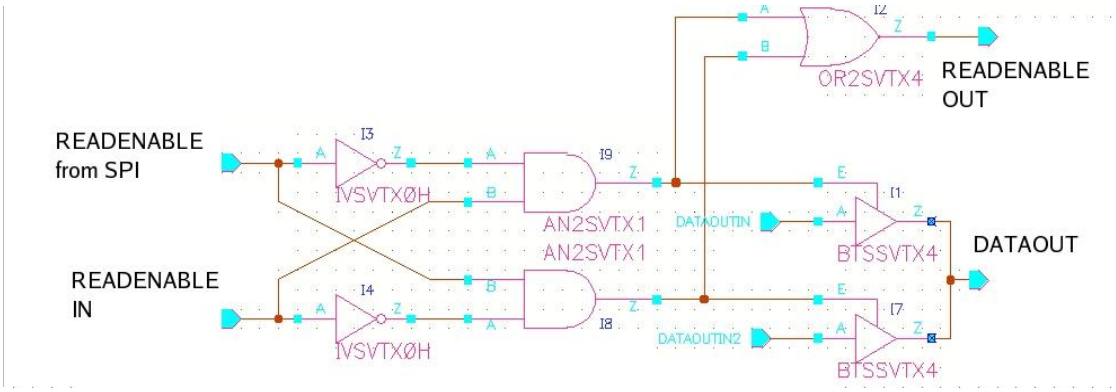


Figure 38: SPI Junction Circuitry

The junction ensures that two SPI blocks do not contend for the dout line at the same time. In case of an error, if two SPI slaves do contend for the same serial out bus, both of them lose control of the bus.

4.2.8 SPI Layout

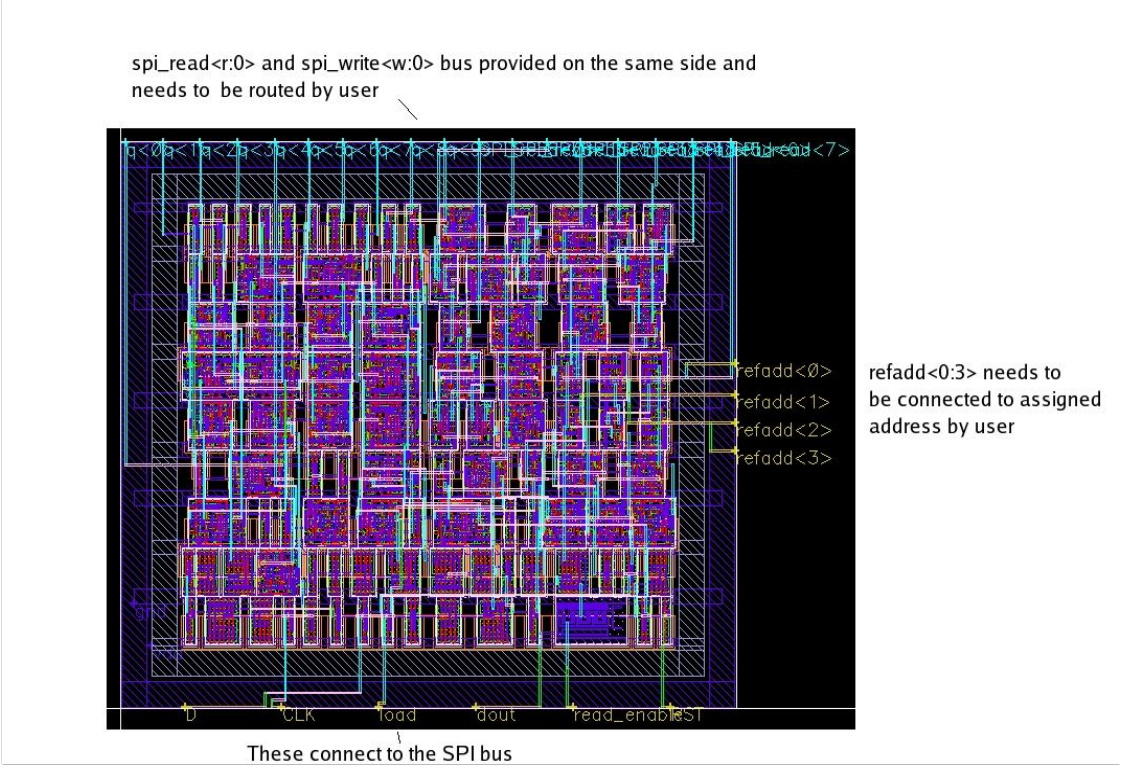


Figure 39: SPI Layout

Most slaves had an aspect ratio of 1 to 1.2. Inputs and Outputs were on opposite sides with static configuration signals on one side. The *ref_add* bits set the hard-wired address for the individual slave.

4.2.9 Read Operation

Special care needs to be taken during the read operation. Since the total bus length is over 2mm, it could take longer than a single clock period to shift the data out. Hence two successive read operations are avoided by interleaving a dummy write operation between consecutive reads. During the idle state and write state, there is no activity on the serial_data_out port. A logic high state is always maintained on this port during idle state. When a read operation is initiated, a preamble consisting of {1,0} is shifted out first. A 1 to 0 falling edge on the serial data out port indicates the start of the data frame. A post-amble of 3 logic zeros follows the data frame.

4.2.10 Typical RF Radio Initialization sequence

The two flowcharts in Figure 40 and Figure 41 highlight the SPI based RF Radio initialization sequence.

1. Chip Initialization

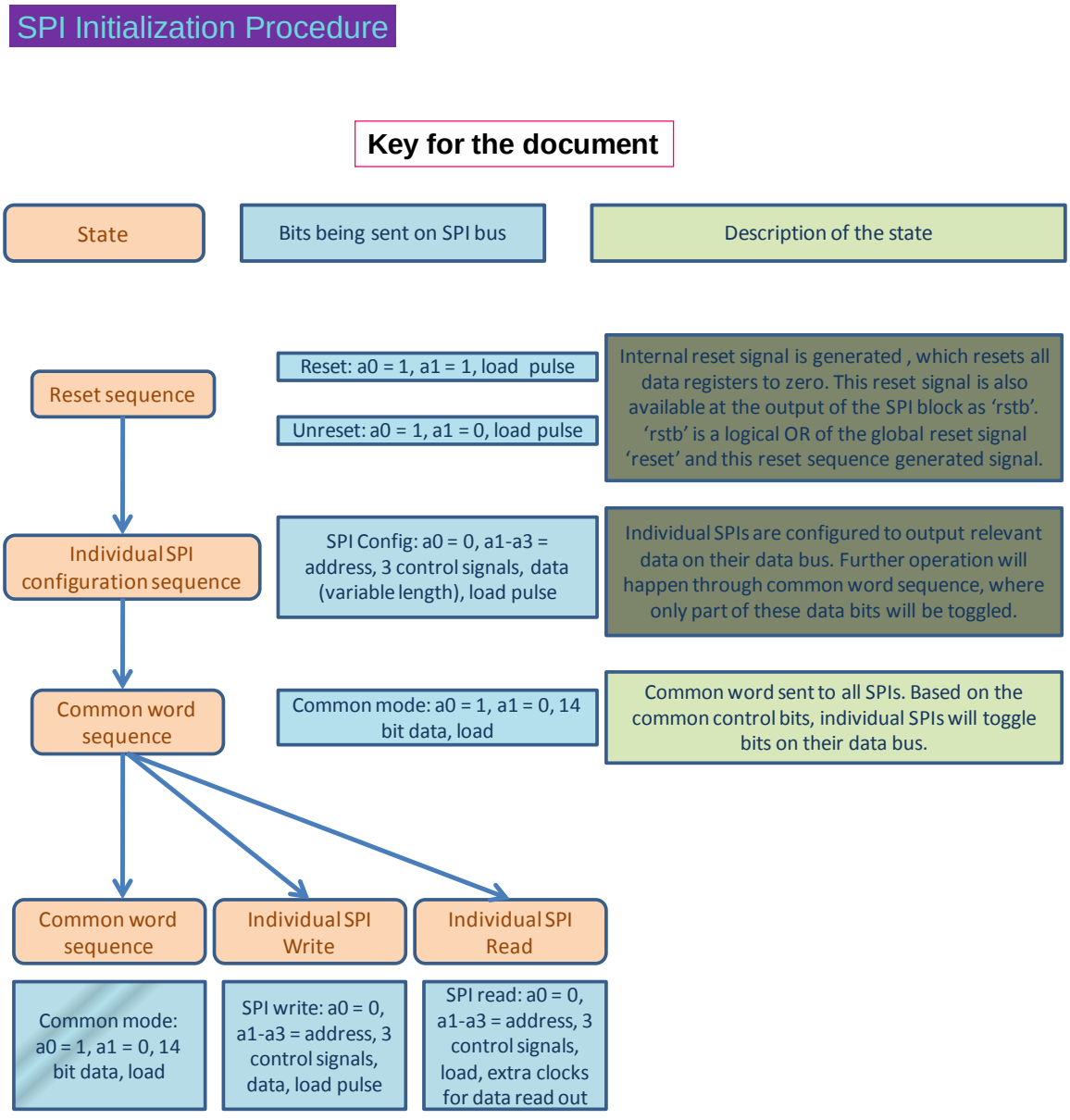


Figure 40: Chip Initialization

2. Time Domain Duplexing Operation

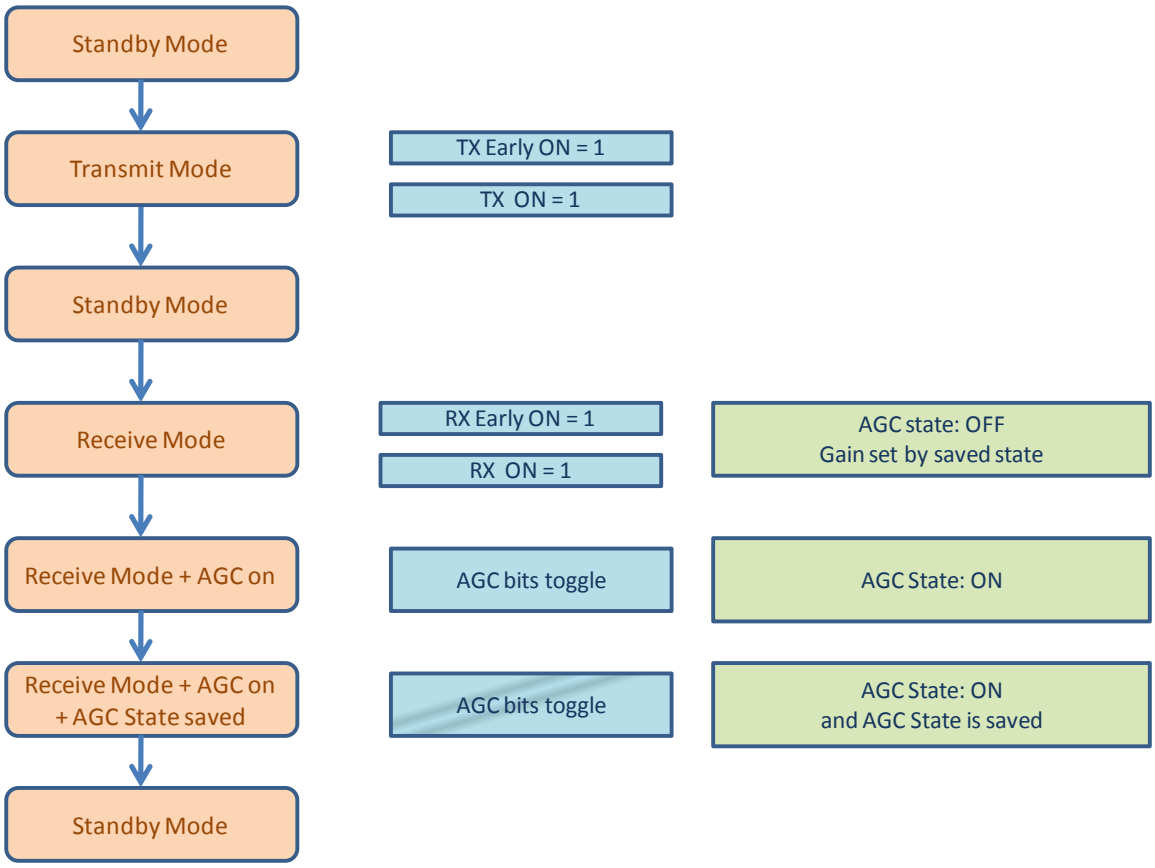


Figure 41: Time Domain Duplexing Operation

4.3 New Upgraded V2 SPI Sub-System

4.3.1 Motivation for the upgrade

The following reasons prompted the re-design of the SPI sub-system:

1. The design had to be ported to the standard digital design flow right from RTL to GDSII.

2. The entire design needed to be synchronized to a single clock (*spi_clk*). The version1 design had multiple clocks (*spi_clk, load*) with no clock-domain crossing checks. This led to some bits getting erroneously loaded or getting loaded with the wrong data. Such glitches were totally eliminated in this revision of the SPI system.

3. The RF radio got revised from 3 separate chips to one single chipset. The number of SPI slaves increased manifold and the addressing had to be changed. The concept of Control Word followed by a Data Word took birth.

4. The data word format changed from {Short word + Long word} to {normal data word + Common Mode word}. Thus was born the "Broadcast Mode" which enabled control of all the SPI slaves at one go.

5. During the tuning and calibration phase, only a few bits used to get updated. However with the older SPI system, the entire 200-odd bits had to be streamed in. A change to the register based system meant that data could be updated in short bursts and hence in lesser time. The whole data word was broken down into 32b registers for both read and write operations.

6. This 32b register based data access is also in compliance with the standard serial communication interfaces found on microcontrollers. It can be used to transfer data between any two devices, say a microprocessor and memory unit.

7. Bugs were found when doing a parallel load into certain SPI slaves. Additionally, some of the bits in one SPI slave got flipped when writing into another SPI slave. For the lack of exhaustive testing, this erroneous behavior could not be explained. With the new digital flow setup, running functional simulations at different stages (RTL, post-synthesis and post-layout) was fairly straightforward. Many more test cases were created for the exhaustive testing which led to fixing of some minor bugs. All the modes were tested with different combinations being tried out. Probability of an error was reduced to less than 1 bit in 100,000 (measured data).

8. Since the 3-chip implementation of the digital radio was now a single chip solution, the classification of SPI slaves had to be modified. The address space was reassigned with the two most significant bits now representing the layer in the OSI 7-layer protocol stack [17] – RF, MAC, PHY and PAL.

9. Two consecutive read operations were not feasible in the first version of the sub-system. With the new design, it is now possible to perform consecutive read or write operations and in fact any permutation of read/write access is supported.

10. A bug was found with the software reset functionality. The slaves were reset through software just fine; however the *unreset* command had a glitch wherein the common mode indicator bit got reset. Staying in the common mode even during software reset was important for the RF blocks. This was fixed in the next version of the SPI slaves. The common mode indicator flag in the new system gets reset only on hard-reset and on common mode exit command.

4.3.2 Block description

The basic block diagram of an individual SPI slave is shown in [fig].

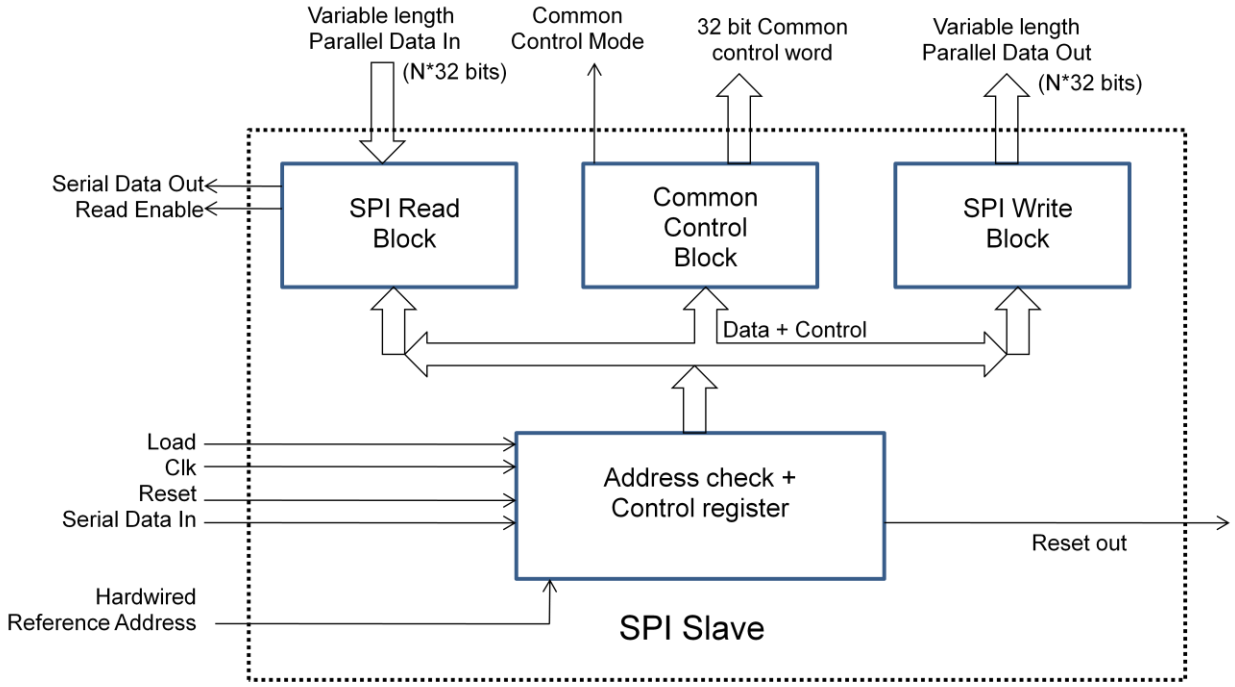


Figure 42: SPI Block Diagram - Individual Slave

Since the new SPI slave is a single clock domain design, the load signal is now converted from a positive going pulse at the end of the data transmission, to a low going enable signal indicating transmission. It stays low throughout the data transmission and is illustrated in the timing diagrams below. Rest of the SPI interface comply with the standard 4-wire SPI protocol (courtesy: Motorola), namely*: reset_n, spi_clk, sdi and sdo*. Read enable signal is used for bus arbitration on the read path. A *reset_n_out* signal is generated as a software reset and is explained in 4.3.7.1below.

The data interface is split into three blocks: read bits, parallel write bits and common mode control bits.

a) Read block: The read block is split into 32 bit registers and each one of these registers can be read out individually. The number of registers is a programmable parameter in the RTL Verilog file. The read block latches the data on the read (pdin) pins on each clock rising edge. A predefined preamble and post amble is attached to the data before serially shifting it out.
b) Parallel Write Block: This block contains the regular parallel write registers, each one 32bits wide.
c) Common Mode Control Block: This block contains one 32b register that is used for common mode control. The common mode is another name for broadcasting data. The [sec 5.3.5] below will talk more about common mode.

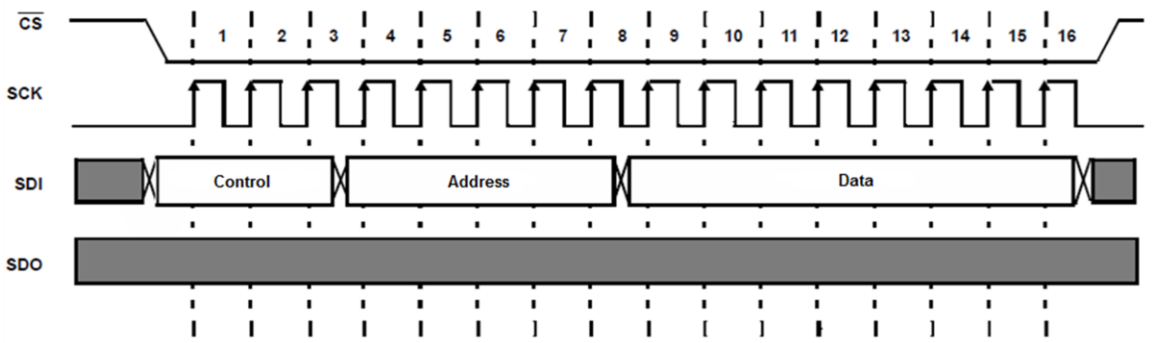
4.3.3 Simplistic Timing Diagram

A simple timing diagram for the SPI serial communication is show in [fig]. The first step is the SPI master pulling the Chip select line low. This signal is common for all of the address based SPI slaves. One dummy clock cycle follows this action. Hence, during the first clock cycle (after CS_N goes low) all the internal counters and state machines get initialized. Valid transmission begins with the second clock cycle. The 14b control word is sent first followed by the 32b data. After the 45^{th} clock cycle, a few dummy data bits are padded. The number of bits is programmable and is set through a parameter in the RTL source code. On the next falling edge of the clock, the chip select line is pulled up to logic high.

Since this system is half-duplex, data transmission can occur only in one direction at any given time. The clock is always sent from the master to the slave. The Master should always change the data during the falling edge of the clock. This allows the slave device, sufficient time to latch the data on the rising edge of the clock.

When a read command is issued, the control word is sent to the slave. On the following clock rising edge (16^{th} clock cycle)

SPI Timing Diagram



1 Dummy Cycle after CS goes low

12 bits Control Word + Address

32 bits Data

Figure 43: SPI Timing Diagram (Version 2 implementation)

4.3.4 Control Word Format

The 12b control word is shown in Figure 44. The first bit is the read/write bit which indicates what operation is being exercised. According to this bit, the appropriate state machine kicks in. This is followed by a 2bit block address to choose between the RF, PHY, MAC and PAL blocks. The same slave is used for the RF, PHY and MAC blocks. However the PAL slave is actually a memory cell. That mode of communication is used by the off-chip microprocessor (PAL) to initialize the memory contents. When these two bits are "11", there is no data width limit (32bits for all other cases). The whole memory content is streamed continuously in a single burst. The MAC serves as the link between the PAL and memory and hosts the Direct Memory Access (DMA) controller. Depending on which type of memory is being accessed, the DMA controller knows the data width (8bits/16bits). For RF, PHY and MAC blocks, the next five fields in the control word represent the sub-address or the slave address. The Figure 44 illustrates the sub-addresses for the RF block. The last 4 bits of the control word tell what of the 32b registers is being accessed.

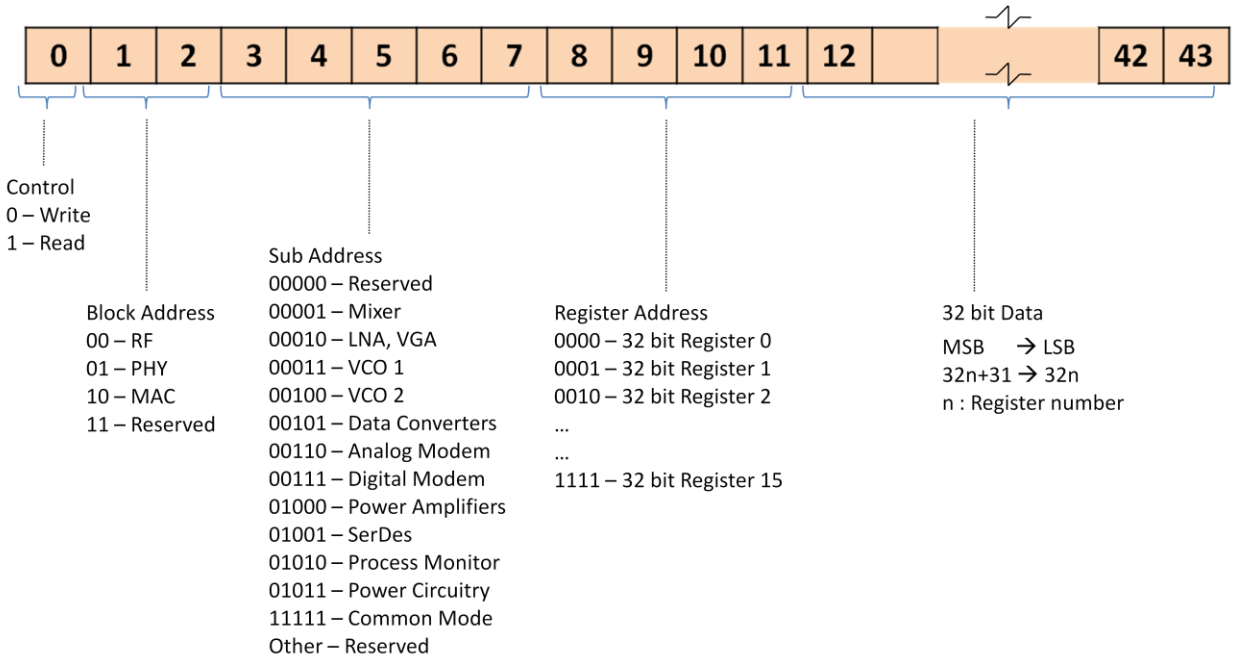


Figure 44: SPI Timing Diagram - Control Word

4.3.5 Common Mode

The Common Mode is also known as the Broadcast mode for the SPI sub-system. In this mode, all the slave devices are controlled, all in one go. Global level changes are applied to all of the RF sub-modules, such as channel change, modulation change, gain setting, etc. The following table will better illustrate this point.

Table 5: Common Mode Control Word

Bit Number	**Function**	**Pin Details**
0	CM Indicator	1 = Indicates we are in Common Mode
1	CM Reset	1 = Common Mode Reset
2	Tx ON	1 = Transmitter ON
3	Rx ON	1 = Receiver ON
4	Tx Early ON	1 = Early ON sent to Transmitter
5	Rx Early ON	1 = Early ON sent to Receiver
6	Channel Select 4	See Channel Table in point 5 below which is as per ECMA standards 'Channel Select 4' is reserved for future
7	Channel Select 3	
8	Channel Select 2	
9	Channel Select 1	
10	Channel Select 0	
11	Mode Select 3	Modulation Schemes such as: OOK, BPSK, QPSK, MSK, 16-QAM, 64-QAM and so on.
12	Mode Select 2	
13	Mode Select 1	
14	Mode Select 0	
15	Data Rate Select 2	000 = 1.728Gbps 001 = 864Mbps 010 = 1.485Gbps 011 = 3.456Gbps 100 = 2.97Gbps 101 = Reserved 110 = Reserved 111 = Reserved
16	Data Rate Select 1	
17	Data Rate Select 0	
18	AGC 2	000 = OFF 001 = ON 010 = Save State (AGC ON) 011 = Load Saved State(OFF) 1xx = Reserved
19	AGC 1	
20	AGC 0	
21-31	Reserved	Bits reserved for future use

The mode information is as shown in Table 6.

Table 6: Mode Selection Table

Mod3	Mod2	Mod1	Mod0	Mode
0	0	0	0	Various Modulation Schemes such as: OOK BPSK BFSK QPSK 16-QAM 64-QAM PFM MSK and so on
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Table 7: Channel Change Table [3]

CH_CHANGE		BAND_ID	Lower Freq	Center Freq	Upper Freq
32	10				
00	00	1	57.240	58.320	59.400
00	01	2	59.400	60.480	61.560
00	10	3	61.560	62.640	63.720
00	11	4	63.720	64.800	65.880
01	00	5	57.240	59.400	61.560
01	01	6	59.400	61.560	63.720
01	10	7	61.560	63.720	65.880
10	00	8	57.240	60.480	63.720
10	01	9	59.400	62.640	65.880
11	00	10	57.240	61.560	65.880

Software reset is also a part of the common mode control. Four other operations; Transmitter/Receiver Early ON, Tx/Rx ON set the transceiver chip in either transmit or receive mode. The Early On mode is needed to give the on-chip VCOs (Voltage Controlled Oscillators) sufficient time to startup. Any jitter in the output frequency will directly impact the overall Bit Error Rate (BER).

The Common Mode Control Block within each slave is as shown in Figure 45. The common mode control word is latched in a separate register. Thus, each slave has parallel data (multiple 32b registers) and common mode data (one 32b register). The data from these two registers are processed together with some combinational logic. The parallel data is also available for direct use (in the RF module). The multiplexed outputs are extra set of signals which are fed into the RF module. This way, the RF module's functionality can be overridden and tuned in the common mode/broadcast mode. There may be some static signals that are generated from other digital blocks, RF modules or are generated off-chip. One example being, the *Tx_On* signal. It is fed into the SPI slaves from an off-chip control (highest priority) or from a local SPI slave (next priority) or set through parallel data (lowest priority).

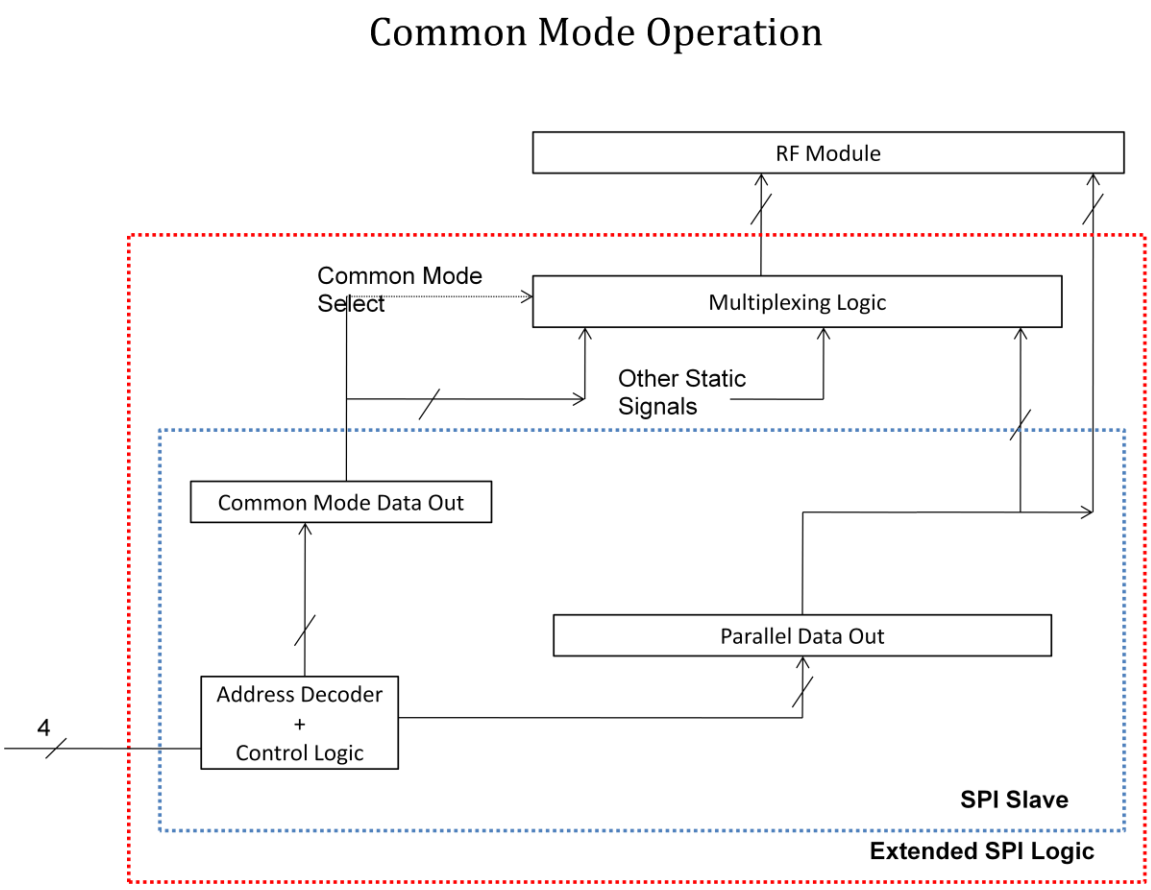


Figure 45: SPI Slave - Common Mode Operation

4.3.6 Software Reset

A new feature in this serial communication module is the ability to generate a software reset. To save up on an extra bonding pad the reset pad might not always be bonded out. However, the digital content on the chip is significant and it is best to initialize the digital logic to a known state. The state machines could go into a blind state and flip-flops could turn metastable; both cases being severely detrimental to the performance of our chip.

Hence software reset functionality was developed and the active low reset was used to reset majority of the digital content on the chip. Special SPI commands were reserved for this purpose. When this command (address) is broadcast, each of the slaves generate this reset signal. Peripheral logic such as SAR ADC digital core, ADC error correction algorithms, counters and gain control state machines, etc can make use of this reset pulse. Just to be safe, an unreset

command was also reserved. These functions were tested to be working fine on the second version design.

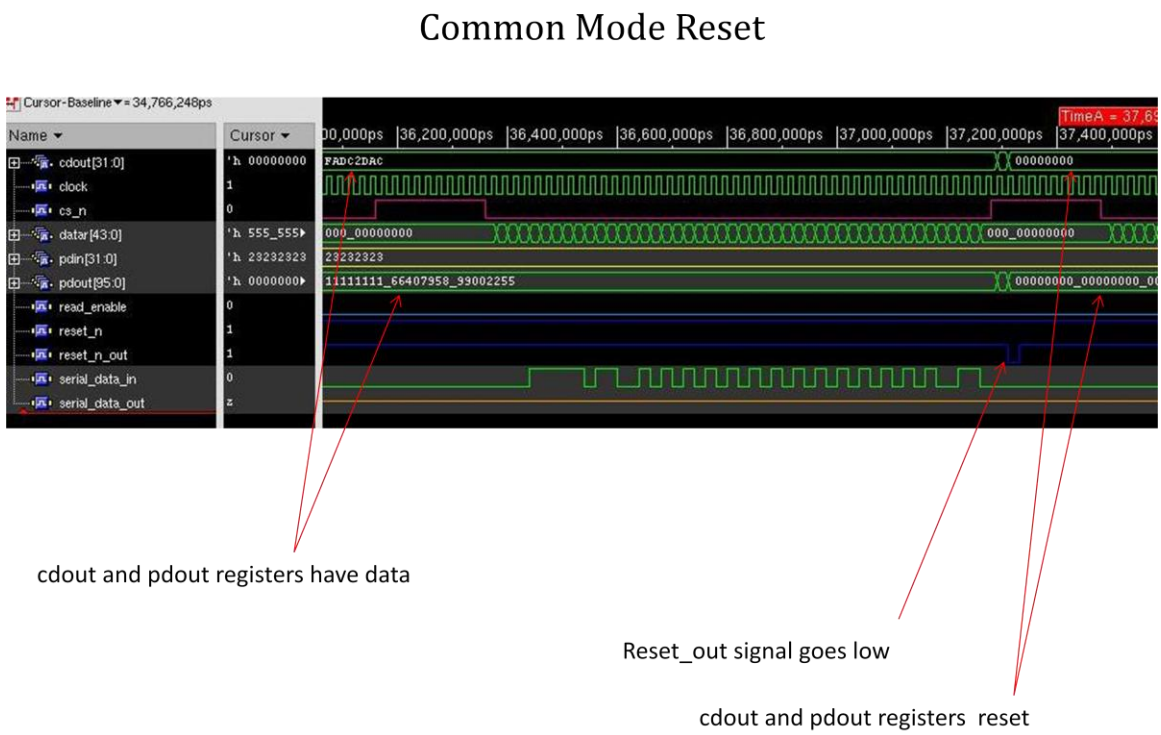


Figure 46: SPI Slave - Common Mode Reset

As seen in the Figure 46, serial data is shifted into each of the SPI slaves. The *reset_n* signal is the off-chip reset signal; which is held high. After the serial data is latched into the slave (*cs_n* going high), the software reset command is detected and the *reset_n_out* signal is generated as a low going pulse. All the internal registers are also reset as can be seen by observing *cdout* and *pdout* signals.

4.3.7 Detailed Timing Diagrams

The following screenshots illustrate the timing sequences for SPI communication. Different modes are illustrated. These functional simulations were done using ModelSim on post-layout Verilog netlists. The post layout timing was verified in PrimeTime.

4.3.7.1 Reset and Write Sequence

The first step after turning the power on is always applying a reset pulse. This initializes all the state machines in a known state and all the registers get pre-loaded with a known value. Metastability in flip-flops should be avoided not only because the operation needs to be predictable but also to keep the power consumption within bounds. A simple SPI parallel write sequence is the executed and the timing diagram for the same is shown in [fig]. The data bits are sent in a particular order: 12b Control word followed by a 32b data word. The control word is comprised of the read/write bit followed by block address, slave address and register address. The serially transmitted data gets loaded into the appropriate register while the sdo (*serial_data_out*) line remains tristated.

Figure 47: SPI Parallel Write

4.3.7.2 What happens when extra bits are shifted in?

When the data is serially shifted in, the chip_select (CS_N) signal indicates the start of transmission and end of transmission information. A falling edge on this line indicates the start of data transmission and a rising edge indicates the end. The CS_N signal is expected to be held low all through the serial shifting process. What if the same SPI master is used to communicate with various SPI slaves? Some slaves could be built to receive 32b data while others could support higher data widths (say 128b). With the same SPI master, the chip select line would be held low for longer, providing excessive clocks. Each SPI slave is immune to the length of the CS_N low-period. For a 32b slave, the first 32 bits of data get latched on the rising edge of the CS_N signal as shown in Figure 48.

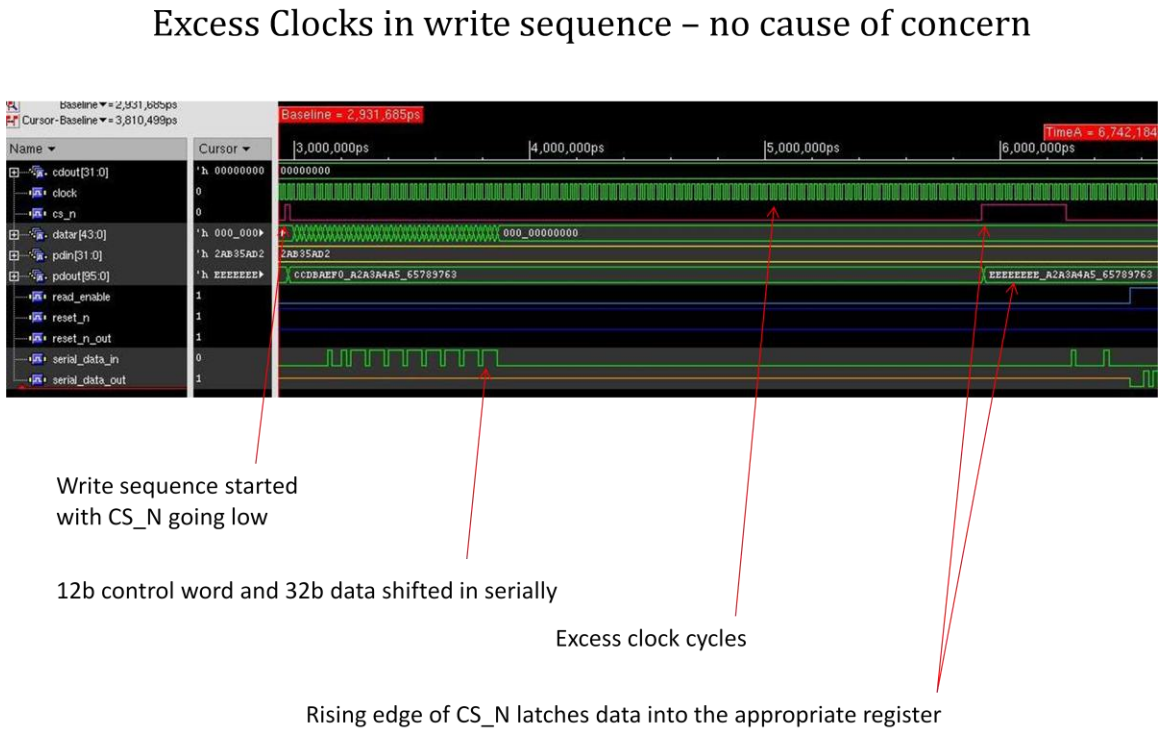


Figure 48: SPI Write Operation - Excess Clocks (Redundant Data)

4.3.7.3 Common Mode Write Sequence

As discussed in Section 4.3.5, the Common mode is used as a broadcast control mode to control all the SPI slaves in one go. The common mode address assigned in this case was "0011111" which can be seen in Figure 49. The 32b data gets loaded in the common mode register and appears on the 'cdout' port. Parallel data register (internal) stays unaffected by this change. However as seen in Figure 45, the multiplexing logic could change the data on 'pdout' port.

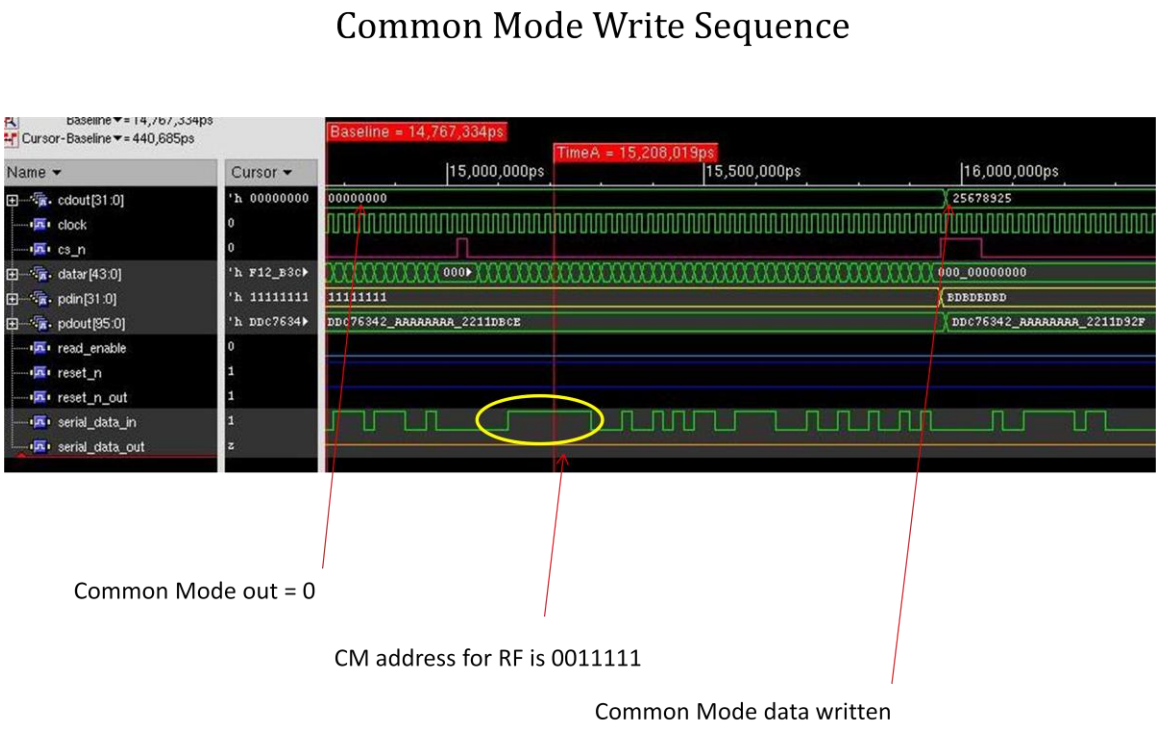


Figure 49: SPI Common Write Sequence

4.3.7.4 Read Sequence

When the SPI masters sets the read/write bit in read mode, serial data is shifted out of the slave device. Data on the ‘pdin’ port is latched internally on every clock cycle. When a read command is executed, this latched data is shifted out on the ‘sdo’ port. To avoiding bus contention a read enable flag is set. It indicates valid data on the ‘sdo’ line. Since the serial data out lines from all slaves are wire-or’ed, bus arbitration logic is mandatory. The *read_enable* pulse is used to avoid contention as shown in Figure 61. 3 logic high bits are padded to the data and hence 35 clock cycles are needed for a successful read operation. A word of caution that this serial interface is half duplex and the clock is always provided by the master SPI.

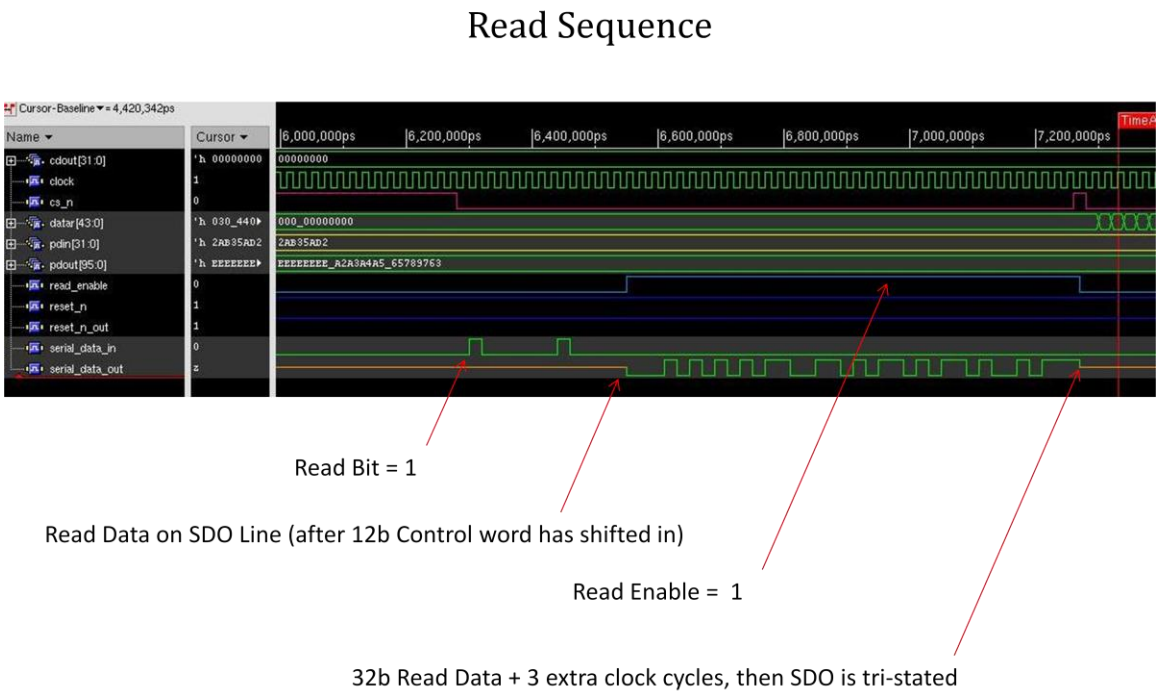


Figure 50: SPI Slave - Read Sequence

4.3.7.5 Accessing the wrong register

When the sub-block address is wrong, the SPI slave does not respond at all. Hence the read and write commands will get ignored. Common Mode is characterized by the special broadcast address and hence that would also be ignored when a wrong address is shifted in. When the slave address is correct but the register address is wrong, the write operation will again be ignored. No change in the parallel data port happens. However when a read operation is executed with the wrong register address, all 1's are serially shifted out as shown in Figure 51. There is no particular reason in choosing all 1's over all 0's or a tri-stated output.

Read Sequence with Wrong Register Address

Figure 51: SPI Slave - Read Sequence with Wrong Register Address

4.3.8 Physical Layout

Cadence SoC Encounter is used as the physical design tool. The steps followed during the physical layout have already been outlined in Chapter 2 of the thesis. A number of SPI slaves were developed using the same flow. Depending on which module uses the SPI slave, a custom wrapper a.k.a. glue logic was developed around the SPI slave. Once the individual slave modules are designed using the digital flow, the top level integration and hook-up is done on Cadence Virtuoso [38].

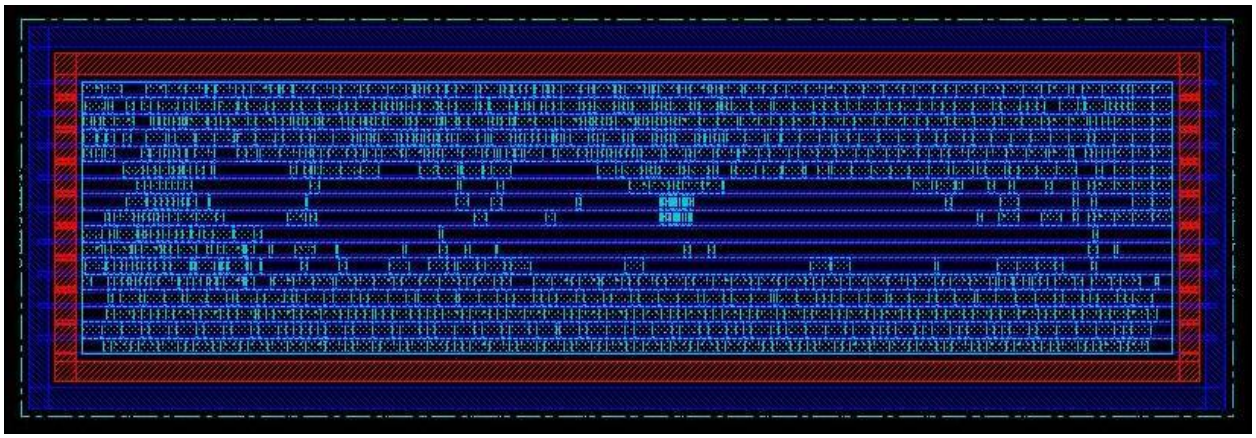


Figure 52: Physical Design of SPI Slave - Placement Stage

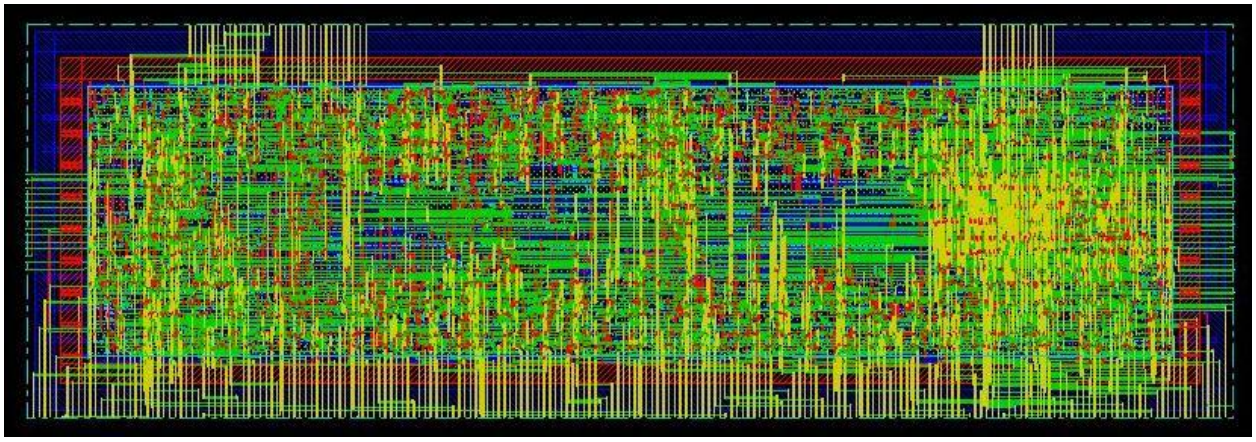


Figure 53: Completed Physical Design of SPI Slave

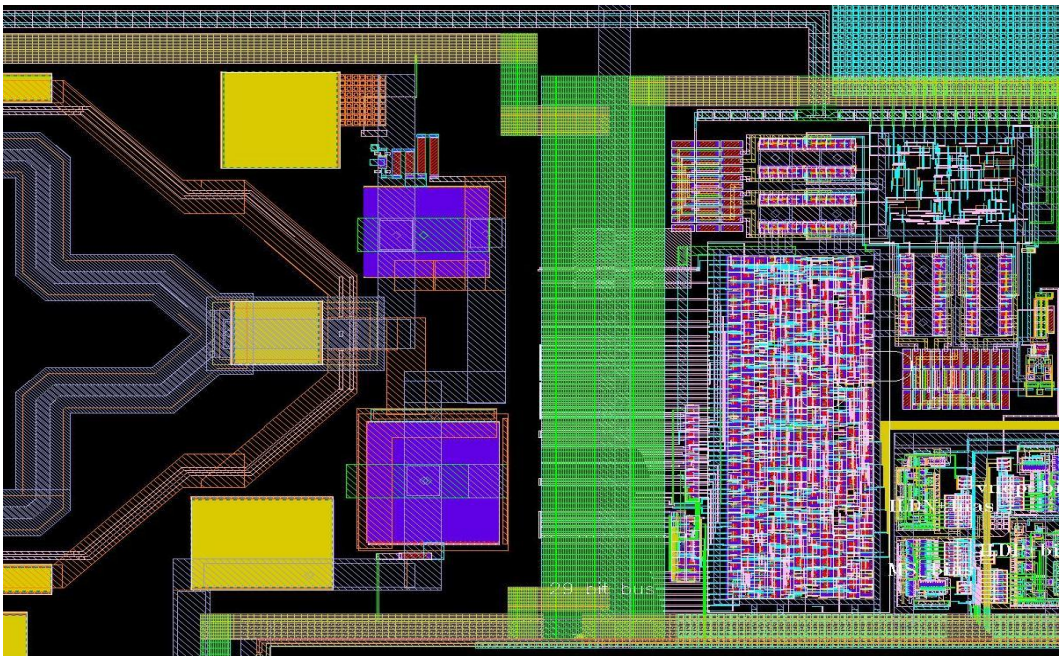


Figure 54: SPI Integrated with an RF Module (VCO)

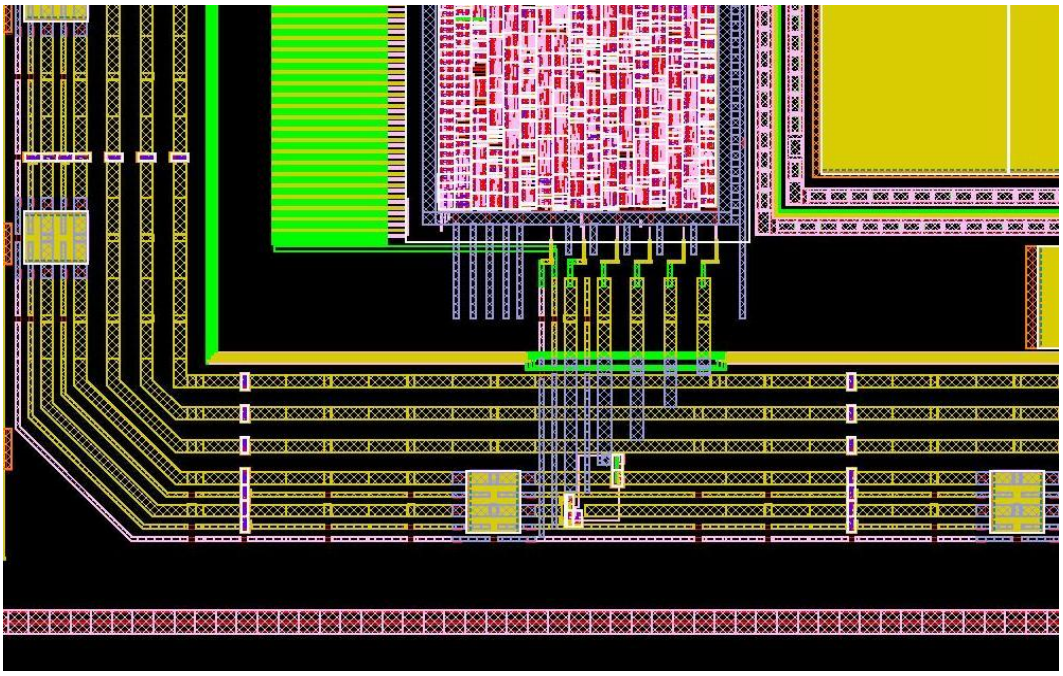


Figure 55: SPI Bus arbiter Junction

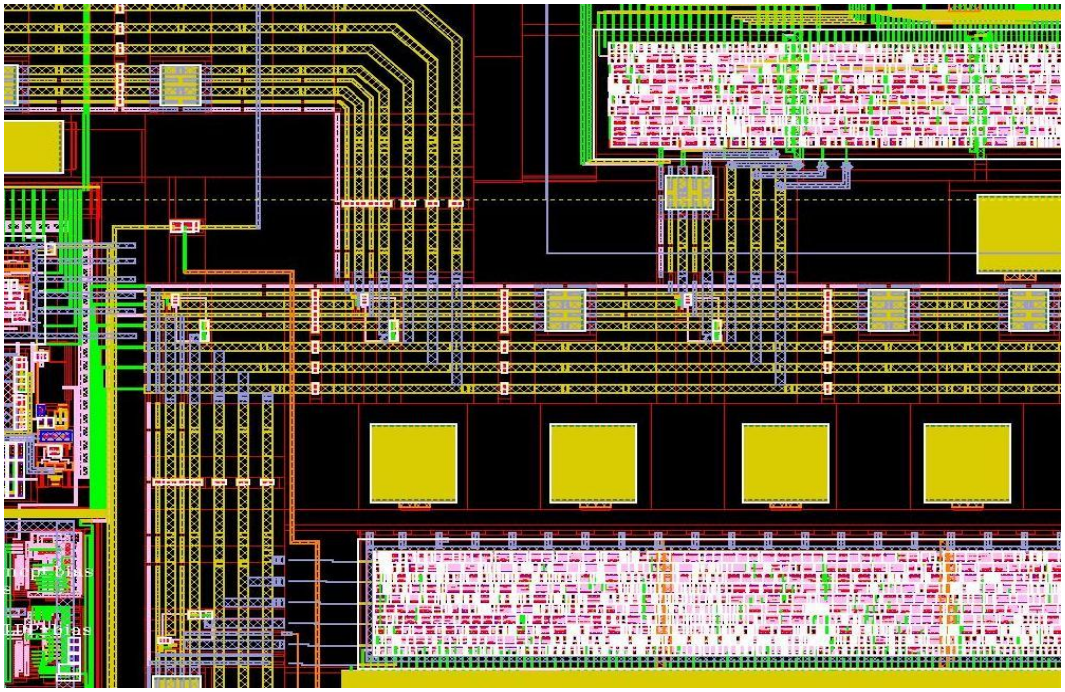


Figure 56: SPI Bus

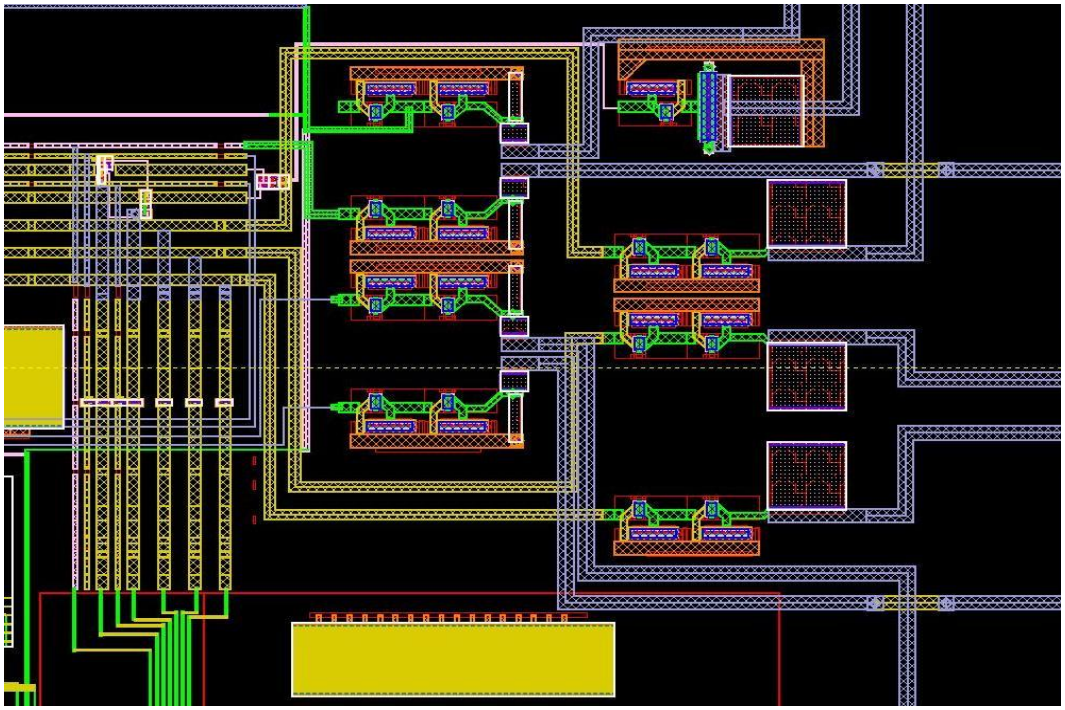


Figure 57: SPI I/O circuit and pad interface

4.3.9 MAC-RF Adapter and SPI Bypass

The initial version of the RF Radio Transceiver was a two-chip solution. The RF Analog Front End (AFE) was one chip and the digital PHY and MAC layers comprised of the second chip. Serial high speed data (I – In-phase and Q – Quadrature) is exchanged between the RF and PHY blocks. The PHY processes serial data, checks CRC, detects the preamble and combines the data in 32b format. Hence the PHY-MAC interface is a parallel 32b interface. The SPI serial communication system is shared across both chips and is controlled by the off-chip PAL (Application layer) processor. The 4-wire serial bus continues on from one chip to the other as shown in Figure 58.

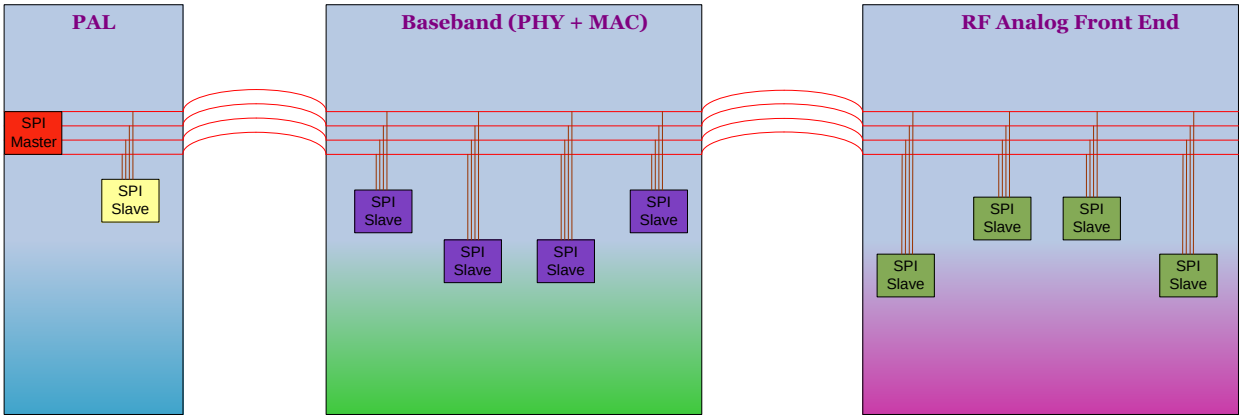


Figure 58: Common SPI Bus on PAL -> Baseband (PHY/MAC) -> RF AFE interface

Individual SPI slaves are used for local calibration, tuning and operational settings. The common mode or broadcast mode is used to tune all the RF blocks all at one go. These changes are initiated by the MAC block and hence there is a need for a separate SPI master. The MAC-RF Adapter serves exactly this purpose. It is the interface between the MAC control loop and the RF sub-system.

As seen in Figure 59, the MAC module provides the clock to the SPI Adapter. The MAC provides other control signals to the adapter namely, Channel Change, Modulation Change, Data Rate change, Channel Bonding, Gain setting and Tx/Rx switch. These commands are processed and sent over the serial link as part of the common mode control. Hence the SPI adapter acts as a slave to the MAC state machine and behaves as a pseudo-SPI master for the RF SPI slaves. The Adapter logic is an interrupt driven state machine and triggers any operation when the MAC initiates some command. On completion of the action, an *action_complete* flag is raised. This is

the handshaking between the two state machines. The SPI Adapter also has a local SPI slave that is used to override some of its functionality. Every individual command that the MAC sends to the SPI adapter can be overridden by user defined commands with the help of the in-built (local) SPI slave.

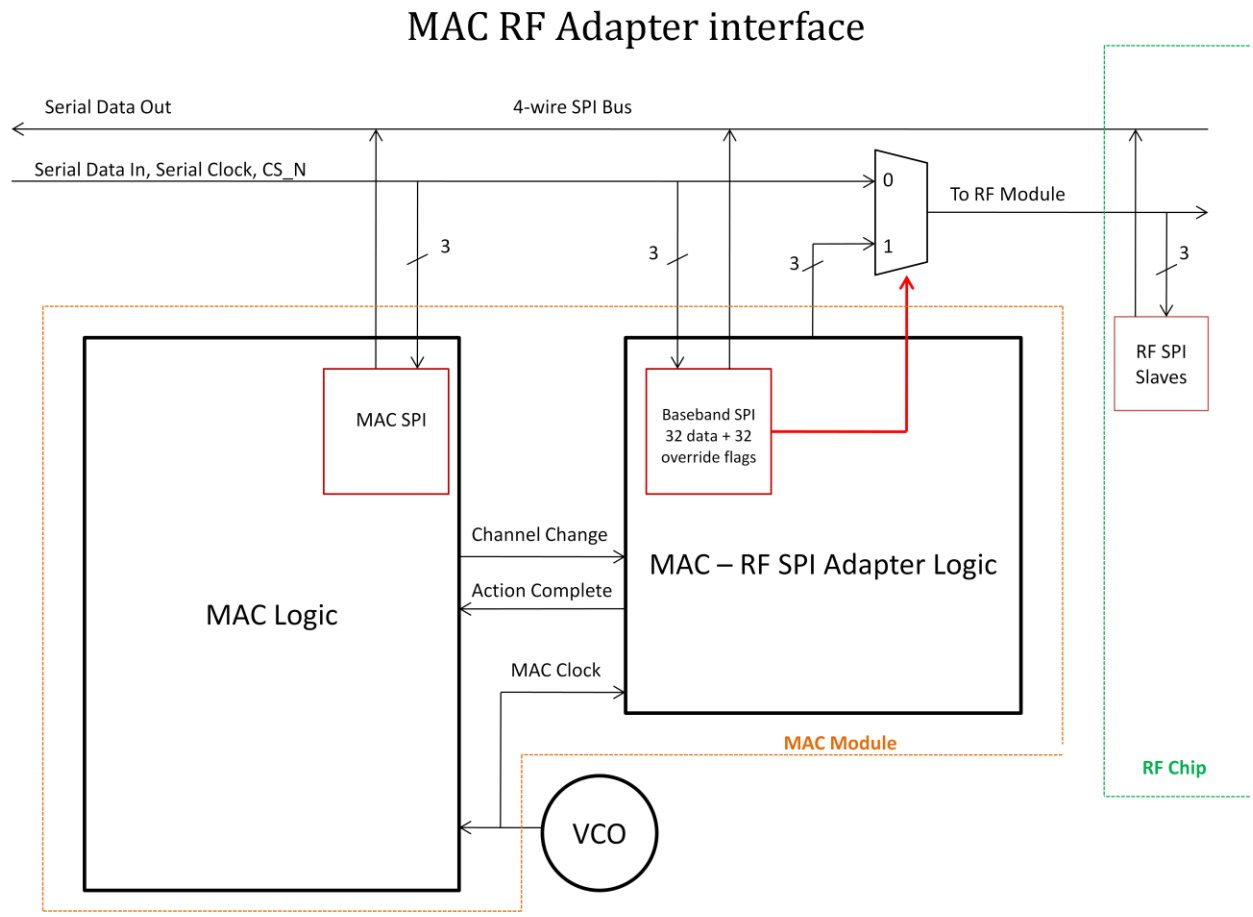


Figure 59: SPI sub-system inside MAC

The MAC-RF SPI Adapter is initially in the bypass mode. Hence the external SPI master has full control of every SPI slave on the chipset. On completion of this phase, the control of the bus is handed over to the MAC-RF Adapter SPI.

As seen in the Figure 60, there is a multiplexer outside the MAC module which allows the PAL processor SPI master to access to the RF SPI slaves. This is the default mode. Once the MAC module takes over the control of the RF front-end, it sends information to the Adapter logic on the Channel Change lines. This is translated into a Common Mode Control Word and is broadcast to the RF SPI slaves. When in this mode, the external multiplexer is set to port 1 allowing complete access of the RF slaves to the MAC. The MAC provides the clock to the Adapter and allows the Adapter to behave as a pseudo-SPI master.

In case the PAL processor wishes to take back the control of the RF SPI slaves, it can communicate with the Baseband SPI (Figure 60) and flip the multiplexer over to port '0'. Thus the MAC control can be bypassed for test purposes.

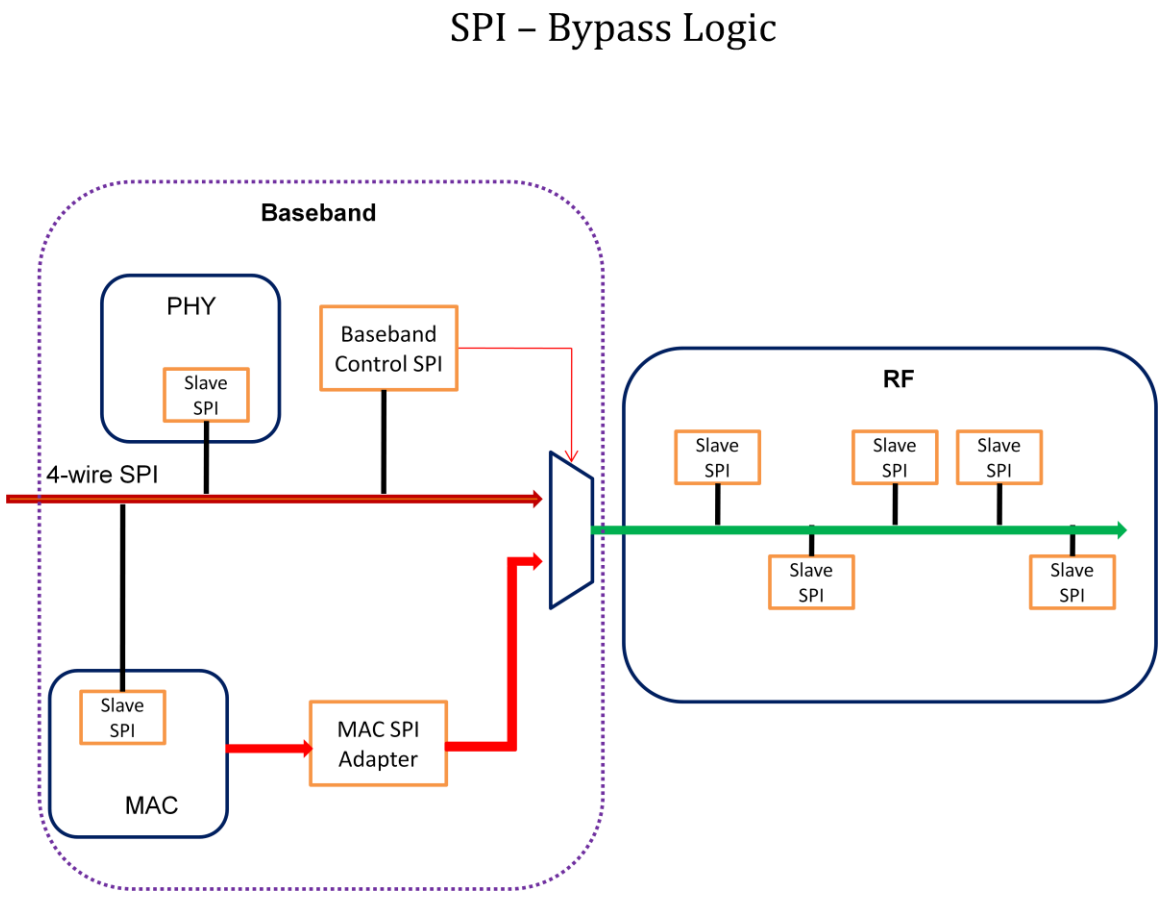


Figure 60: SPI Bypass Logic

4.3.10 An in-depth look at the Common Mode

The SPI implementation in our design supports clock rates in excess of 750MHz. One of the primary reasons to design for such high speeds is for providing flexibility to change the settings on-the-fly. The control sequence is briefly described here:

1. A power-on-reset is sent to the entire chipset. The PAL microprocessor, Baseband logic (MAC + PHY) and the RF Front-end all get reset.

2. All the slave SPI devices are initialized using the master SPI (outside the chipset). Multiple 32b writes and some 32b reads are exercised on the SPI slaves.

3. The SPI slaves in the Baseband chip are first initialized and then the slaves on RF front-end chip are initialized. The SPI bus passes through the Baseband chip through to the RF chip.

4. An SPI RF-Adapter module has been incorporated with the MAC module in the Baseband chip. This block acts as a pseudo-SPI-master for the RF front end SPI slaves. The MAC issues commands to the RF blocks (VCOs, CDR PLLs, Transmitter chain, Receiver Chain, etc…) though the SPI RF-Adapter, on the SPI communication channel.

5. After all modules have been initialized properly, the Baseband SPI slave hands over the control the SPI RF-Adapter block. This is then the pseudo-master for the RF frontend slaves. The MAC module now issues commands related to the communication channel, data rate and the modulation scheme. These commands affect most of the RF blocks if not all. There is a need for a common set of commands that would affect each and every RF slave. Hence the creation of the ‘Common Mode Control’.

As shown in Figure 45, each RF SPI slave has two sets of register outputs:

i. *cdout* – Common Data Out which indicates the "Common Mode Command". A broadcast on the SPI bus sends this command to every slave (common mode address = 0011111 for all slaves).

ii. *pdout* – Parallel Data Out which holds the data written to each slave device. These values get coded during the SPI initialization phase. .

As shown in Figure 45; when in Common Mode the *common mode* command influences the parallel data out. This is indicated by the multiplexing logic. This combinatorial logic would vary from block to block within the RF frontend. Using this broadcast scheme, the MAC module can simultaneously change the knobs inside each and every RF block.

4.3.11 Sub-circuits

The Read path bus arbitration logic is shown in Figure 61. *Read_enable* is an output signal from the SPI slave. This signal is high when serial data is being shifted out of the slave as seen in Figure 50. SDI is the serial data out of the SPI slave and is the Serial_Data_In input into the SPI master. When no data is being read out, this signal stays tristated (Hi-Z) and hence tristate buffers are used on this path.

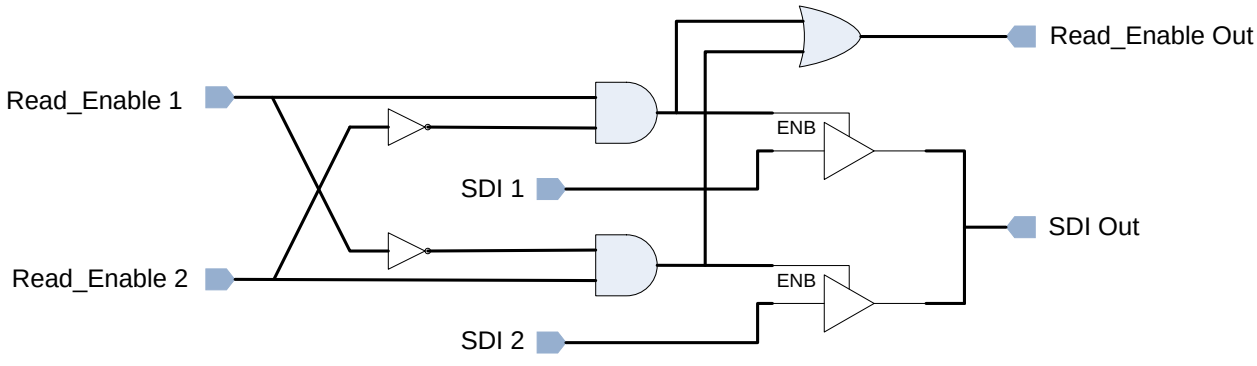


Figure 61: Bus Arbitration Logic on SPI Read Path

The Self Reset circuit was developed for the first design of the SPI slave. The initial version of the SPI slave had two clocks: *spi_clk and load*. The serial shifting occurs on *'spi_clk'* edges and the evaluation and decoding occurs on *'load'* clock edge. Based on the command word settings, the parallel data is loaded into the short_write_register, long_write_register or the common_mode_write_register. Thus, we needed three separate enable / internal clocking pulses to accomplish this task. A *reset_cum_pulse* generator logic was developed for this purpose and is shown in Figure 62. The *add_check* signal is the address match flag and ensures that a particular SPI slave is selected for evaluation. The *external_reset* signal is connected to the *'power on reset'* pulse.

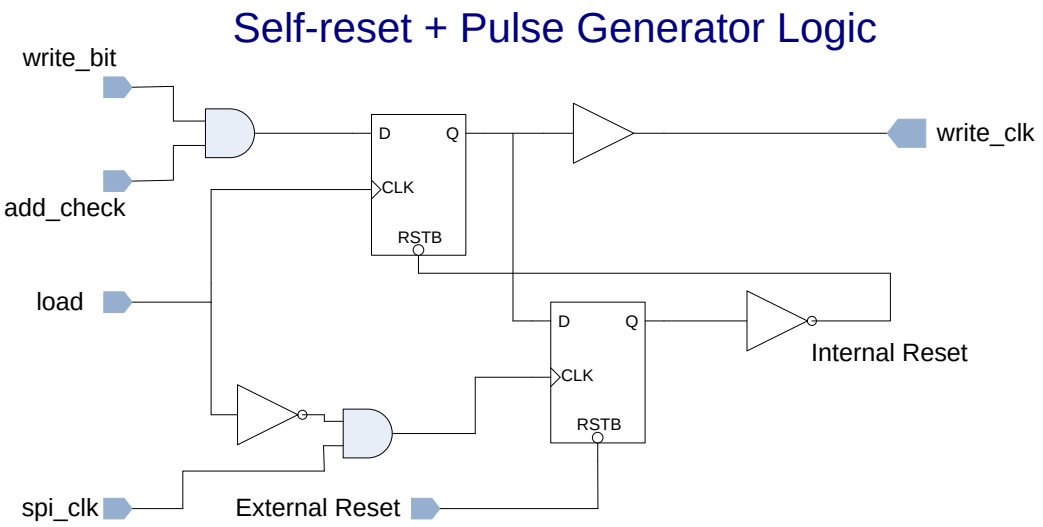


Figure 62: Self-Reset and Pulse Generator Logic developed for the first version of SPI slave (Multiple clock domains)

5 SPI TEST AND MEASUREMENT SETUP

Just as the SPI Master-Slave sub-system went through multiple revisions, the test mechanism too was developed in stages. The first stage involved using an off-the-shelf solution from Byte Paradigm – the GP22050 Protocol Analyzer [39]. The hardware dongle is shown in Figure 63 and the GUI used to test the SPI sub-system can be seen in Figure 64.

Number of digital channels	: 16
Max. frequency	: 50 MHz
Embedded memory	: 16 kByte
Maximum throughput	: 100 MByte/s
Maximum sustained throughput	: 11 MByte/s
Maximum data depth per run	: 100 MByte

Figure 63: GP-22050 from Byte Paradigm used as the Embedded Protocol Analyzer

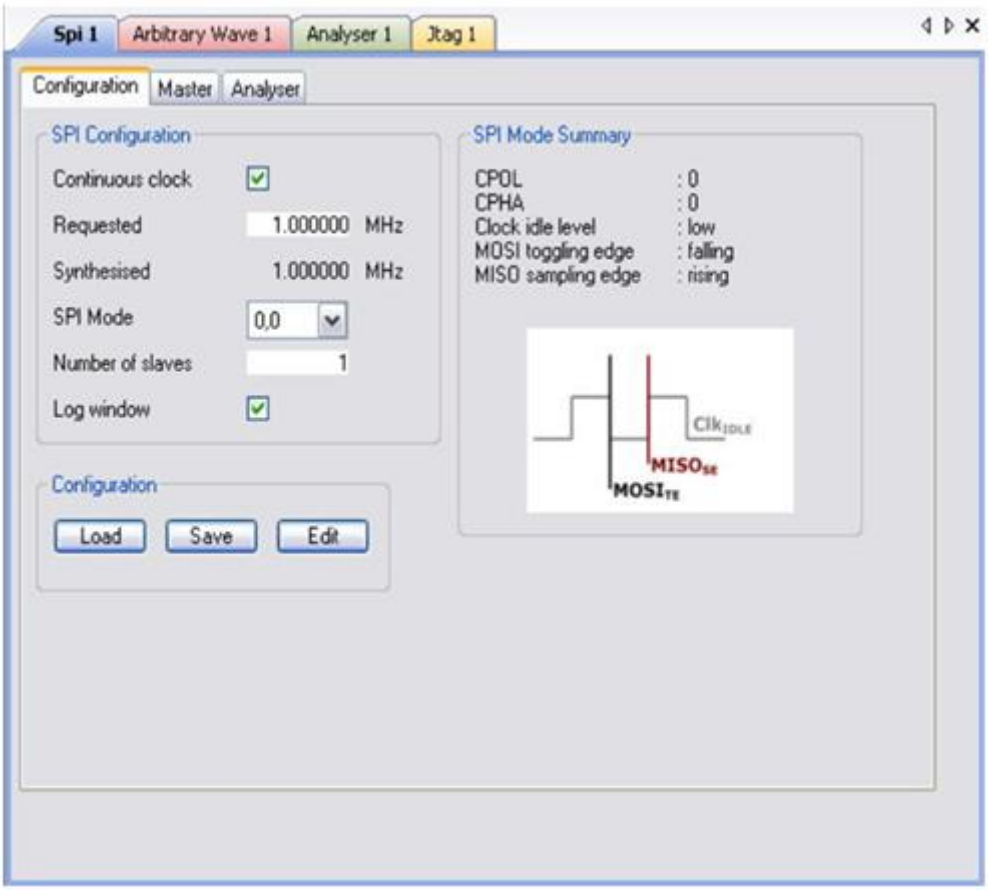
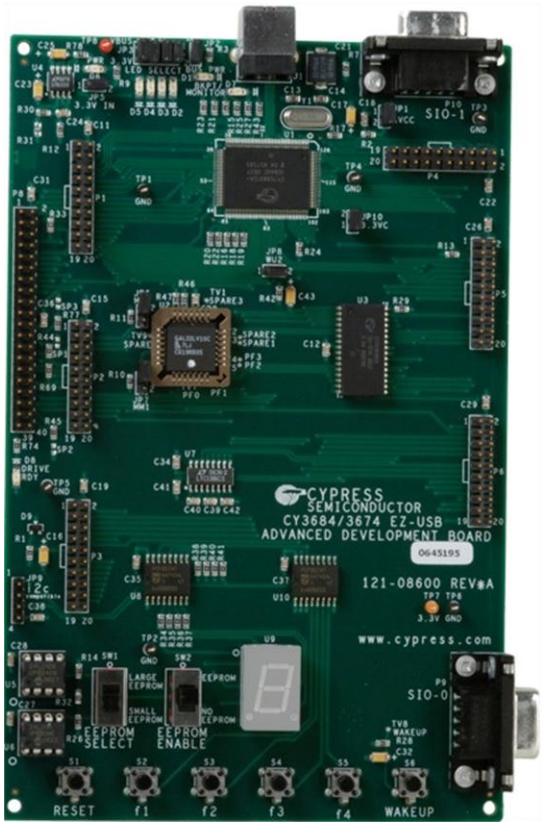


Figure 64: User Interface (8PI-Controller) for GP-22050

The next logical step in development of test and analysis tools was to build a self-contained system. As a result of which, it was imperative to move to a Microcontroller based solution: Cypress Advanced Development Board [40] shown in Figure 65. The user interface was prepared in Matlab and can be seen in Figure 66. The microcontroller board can be configured either as an SPI master or as a slave. A sample waveform when the board acts as an SPI master is shown in Figure 67.

Testing: Cypress Advance Development Board



Features:

Microcontroller: CY7C68013A

Development Kit: CY3684

Memory Size: 16KB

# of Total Pins: 128

# of I/Os: 40

Interface Options: 8/16 bit Data bus
DMA
GPIO
I2C
UART

Package: VFBGA

Application: USB High Speed Peripherals

Figure 65: CY3684 Development Board from Cypress Semiconductor

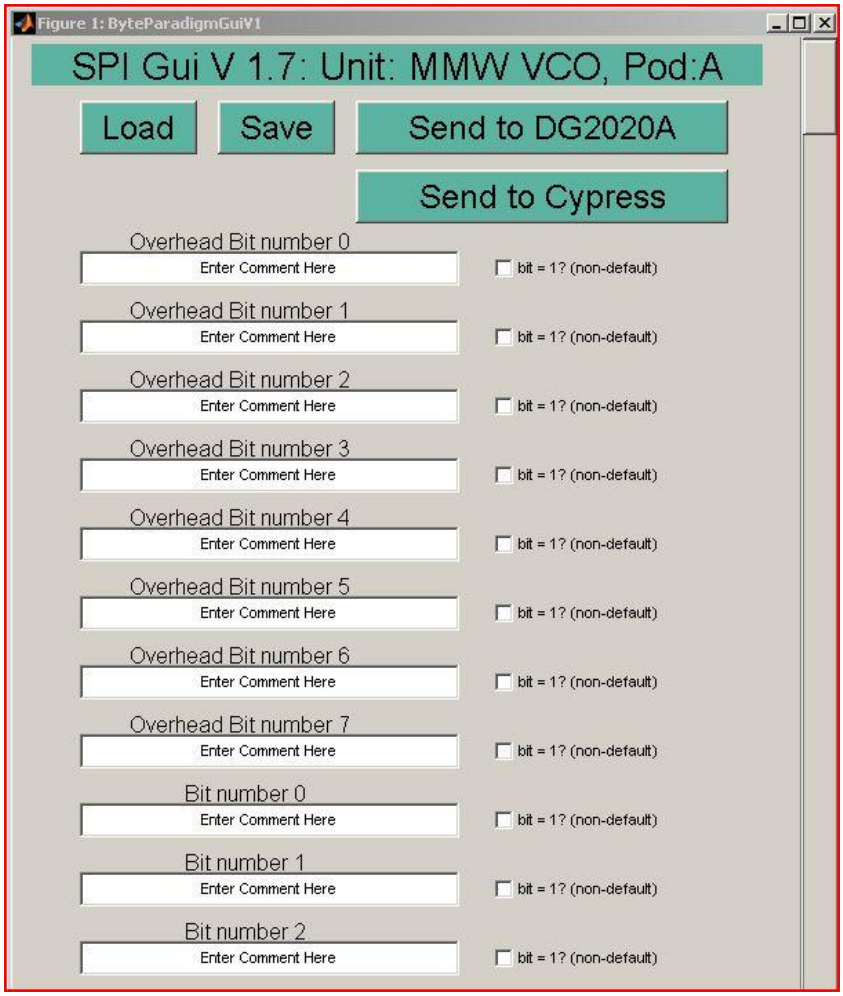


Figure 66: Graphical User Interface in Matlab for Cypress ADB

Figure 67: Sample waveforms for Cypress Microcontroller based SPI Master

To make a self-contained system and to help prepare an evaluation kit for the Digital CMOS Radio solution, a mini-board was developed as seen in Figure 68. The board could directly be plugged into the USB [41] or FireWire [42] port of the computer. A working demo of the RF CMOS Digital Radio can be seen in Figure 69.

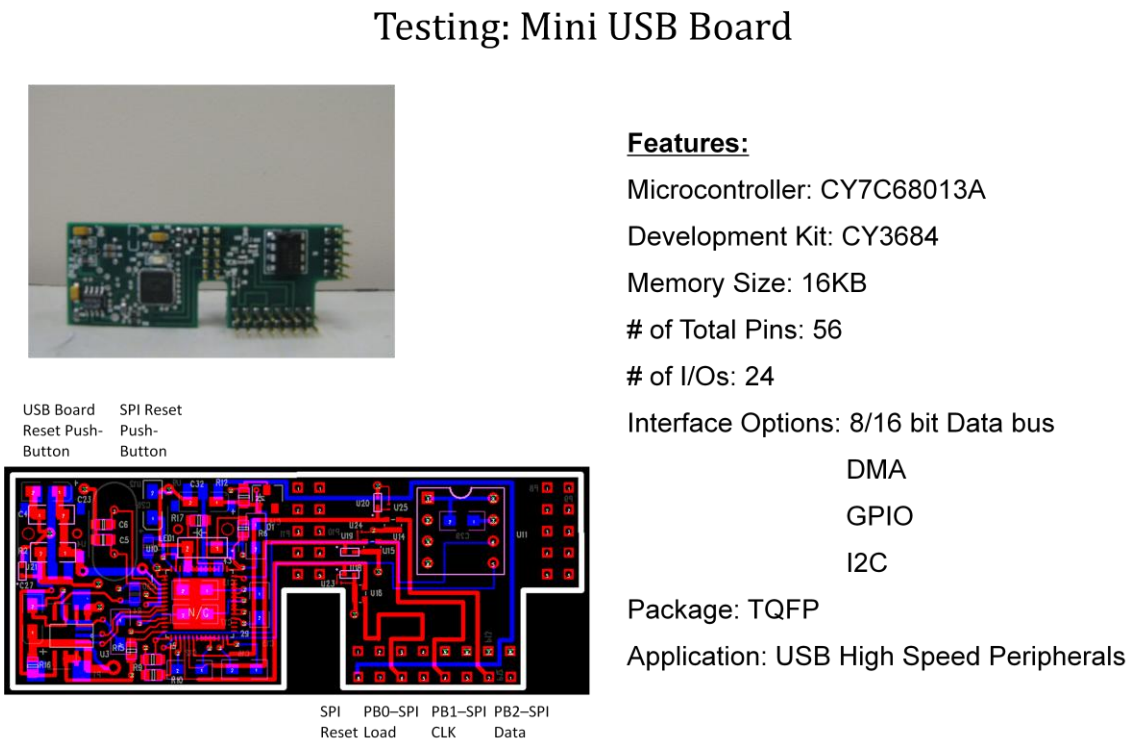
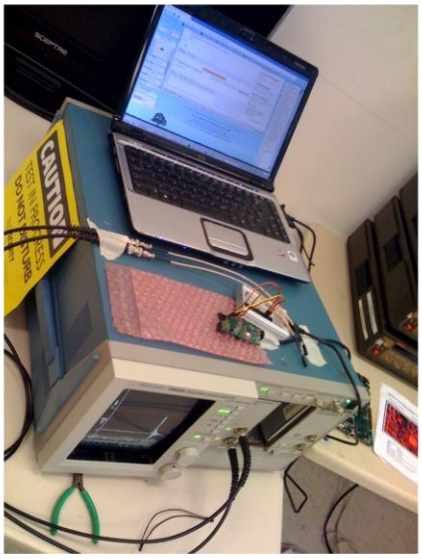


Figure 68: Mini-board for self contained RF CMOS Digital Radio evaluation kit

Mini USB Board - Demo



- **Working Demo**
- **Verified in SPI control of on-wafer test structures**

Figure 69: Working demo for mini-USB based solution

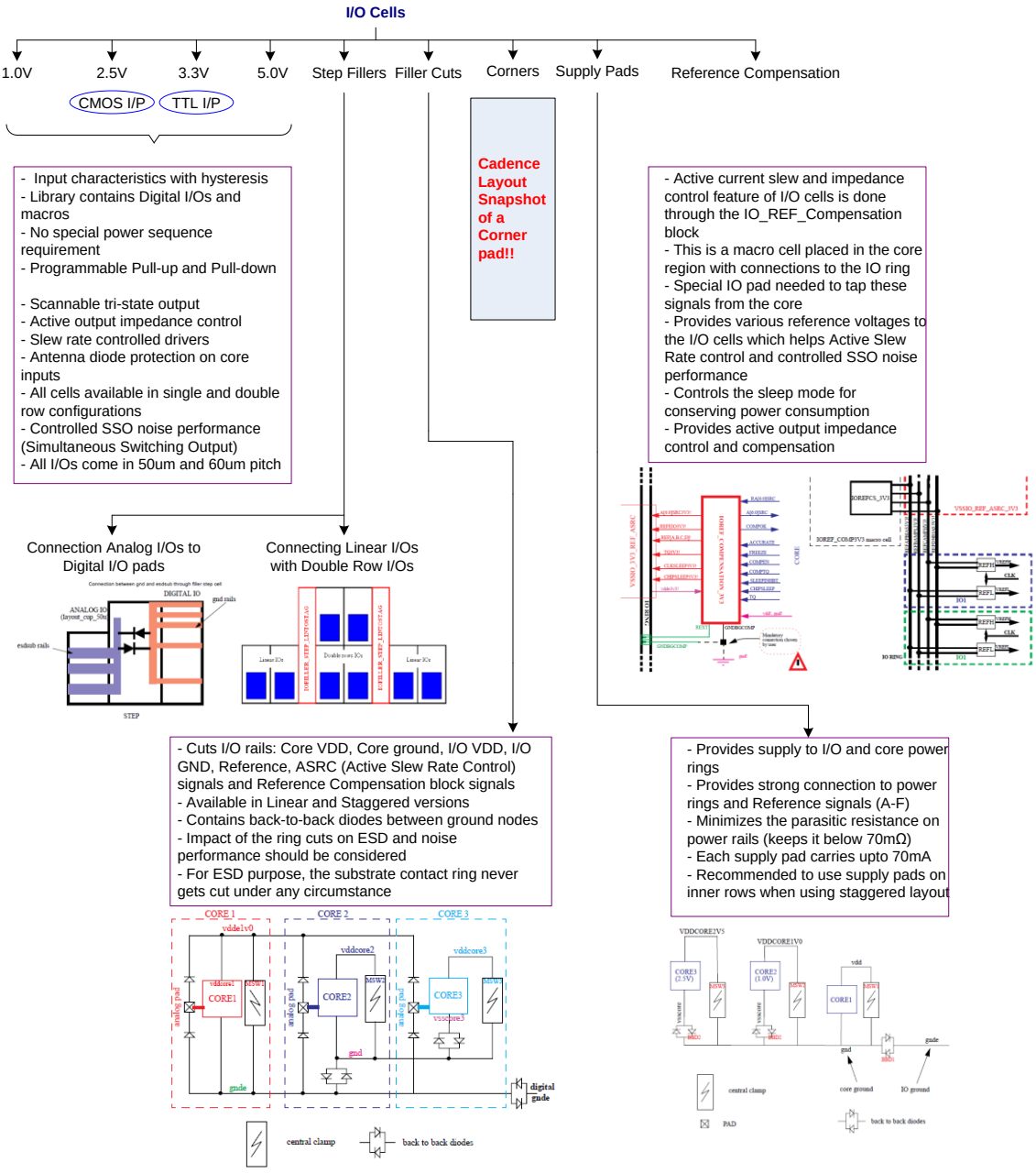
6 ULTRA HIGH SPEED DIGITAL I/O DESIGN

6.1 ST90nm Standard Digital IO Evaluation

This chapter briefly outlines the IO cells available in the design kit provided by one of the vendors (ST Microelectronics [43]). Similar IO cell libraries are available from other vendors (TSMC [44], IBM [45], Samsung [46] and so on).

6.2 *Classification of I/O cells*

The IO cells are divided as shown in the chart Figure 70. The figure is self-explanatory and highlights the most important features of the IO library.



* Reference: STMicro IO datasheets for 90nm node (CMOS090DK)

Figure 70: Classification of IO Cells

6.3 I/O Performance Evaluation

Some test scenarios were created to test the performance of various I/O cells. A sample test was to tie two bi-directional digital I/Os back-to-back. Simulation results for the current drawn by the bi-directional cells are shown for the following two cases:

i) Bi-Di driving another Bi-Di with 10pF of wire-load (Figure 72).

ii) Bi-Di driving a 200Ω resistor // 1pF, for max drive capability (Figure 71).

It is seen that when the Bi-Di cell is driving another Bi-Di cell, the average current consumption per IO is around 1.1mA at 108MHz. This amounts to 1.32mW average power consumption with 50% switching rate of incoming data. This switching rate is too high and is the worst case. Usually switching rates are in the 10% range.

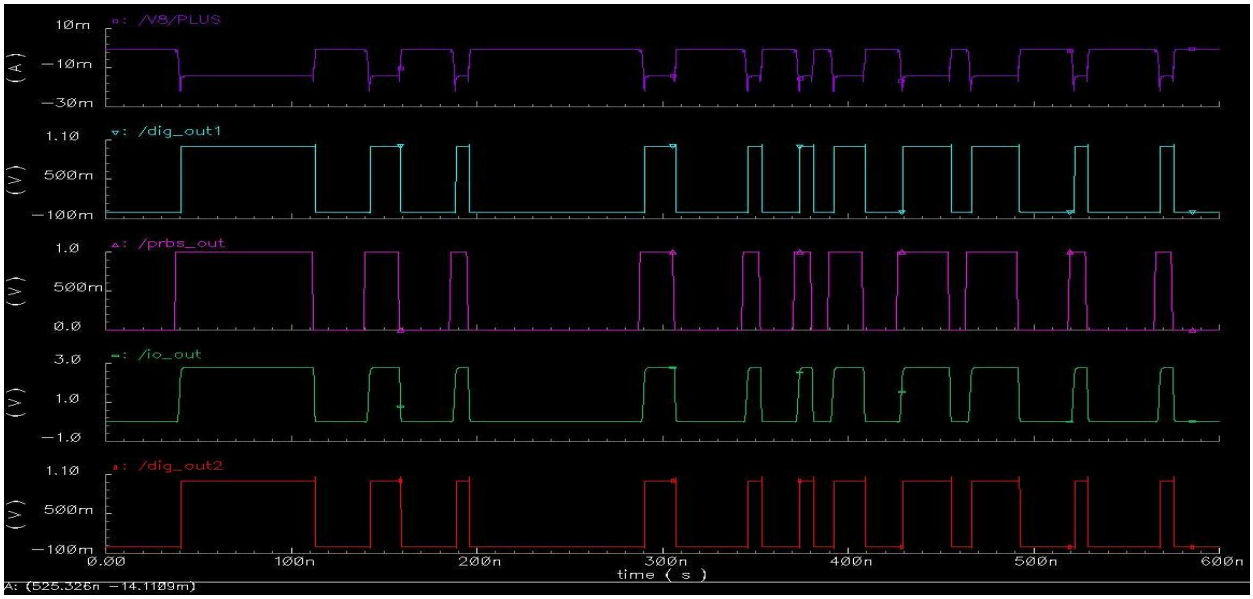


Figure 71: Simulation results for Max loading (200ohms//1pF). The first waveform shows the current consumption in two IOs = +/- 14.5mA

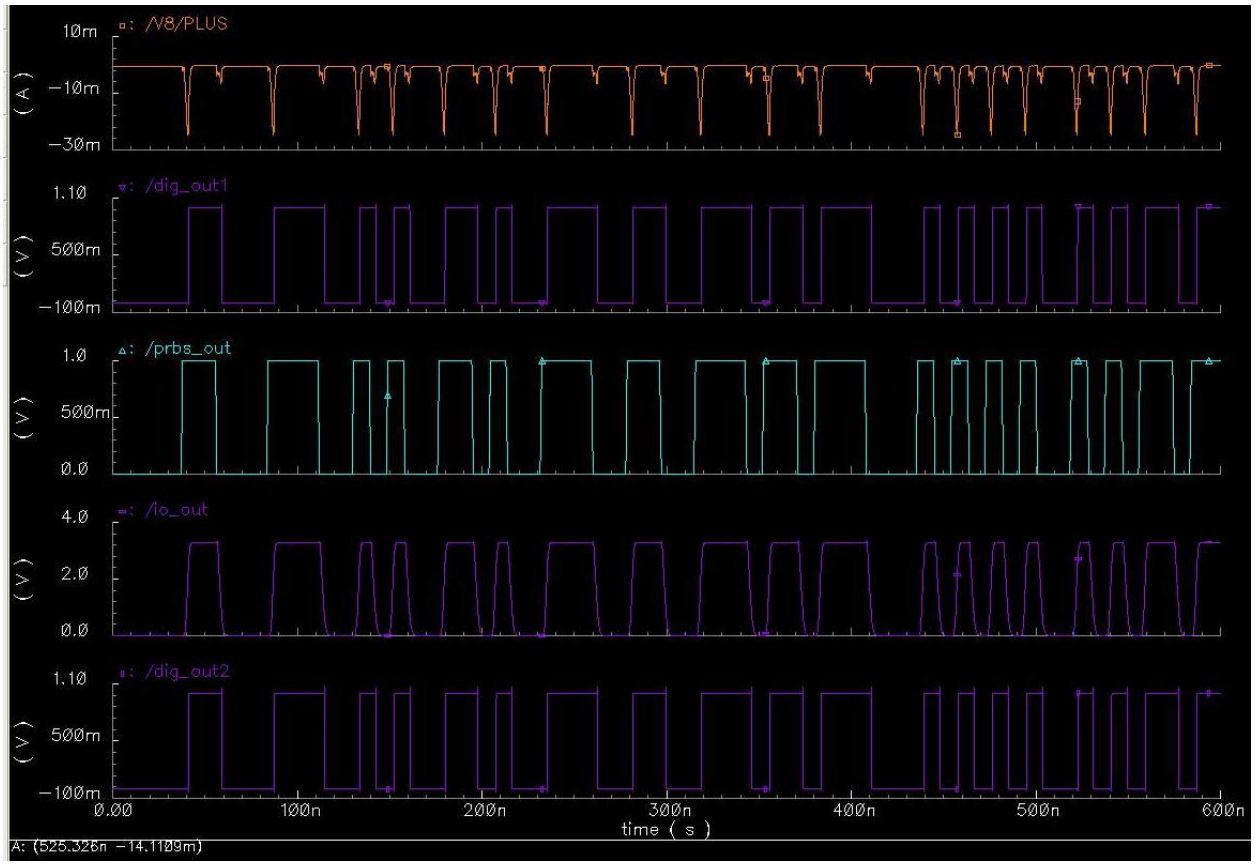


Figure 72: Simulation results when one Bidi cell is driving another Bidi cell (10pf wire load). First waveform shows that the average current consumption is about 2.135mA for two Bidi cells.

6.4 Electro-Static Discharge

6.4.1.1 What is ESD?

The word electrostatic discharge brings to our mind the memories of lightening striking the top of a building or sparks that form between our fingertips and metallic surfaces like doorknobs on dry winter mornings. The sparks are the caused due to the ionization in the air pocket between the charged human body (here palm of the hand) and the zero-potential metallic surface (here doorknob). These scenarios highlight the importance of studying the phenomenon of high voltage discharges. We focus on EOS (Electrical Over Stress) and ESD (which is a subset) occurring on semiconductor chips in particular CMOS chips.

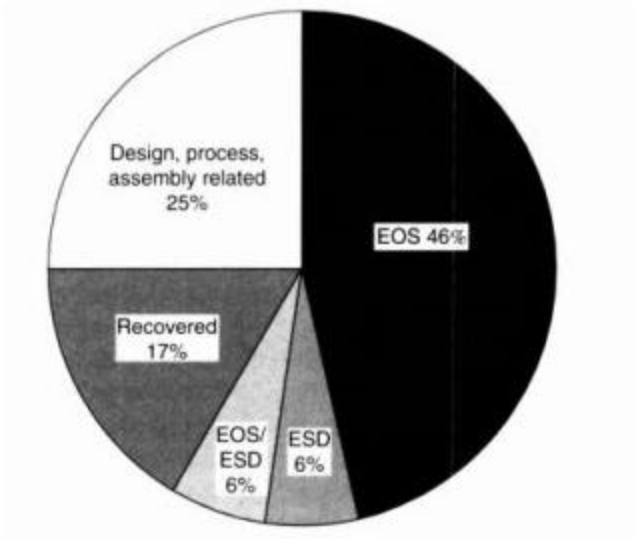


Figure 73: Distribution of failure models in Silicon ICs. ESD accounts for approximately 10% with EOS responsible for close to 5% of the failures. [47]

In many cases EOS classified failures could be classified as ESD failures, which would make its percentage even higher. [48]. Charvaka Duvvury (Texas Instruments) defines ESD as the transient discharge of static charge, which can arise from human handling or contact with machines [49]. The next sub-section briefly covers the major type of ESD related failure mechanisms plaguing the chip design community and their approach to counter it. Some of those

steps have been incorporated into the STMicro IO library cells and others taken care of manually.

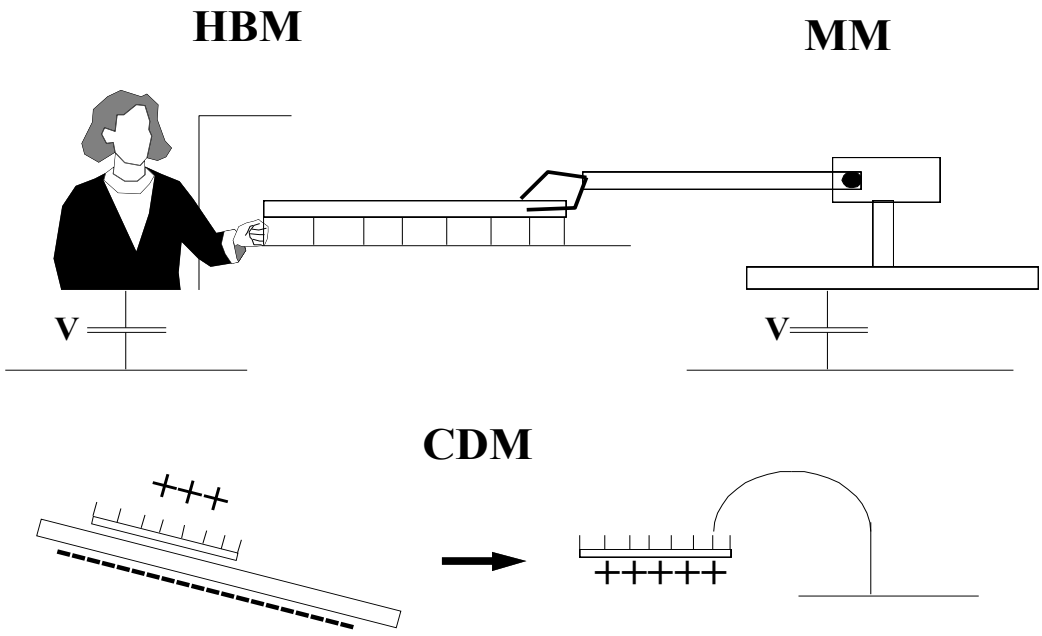


Figure 74: Three dominant ESD mechanisms, namely the Human Body Model, Machine Model and Charge Device Model

6.4.1.2 ESD Failure Mechanisms - Human Body Model

HBM is modeled on a person with a static charge touching a pin on a sample, while another pin is grounded. It is intended to address the hazard of charged person handling parts. This is one of the oldest failure mechanisms and the test methods used to test it are very mature. However, pin-to-pin hazards from charged people are less relevant for high pin-count BGA packages.

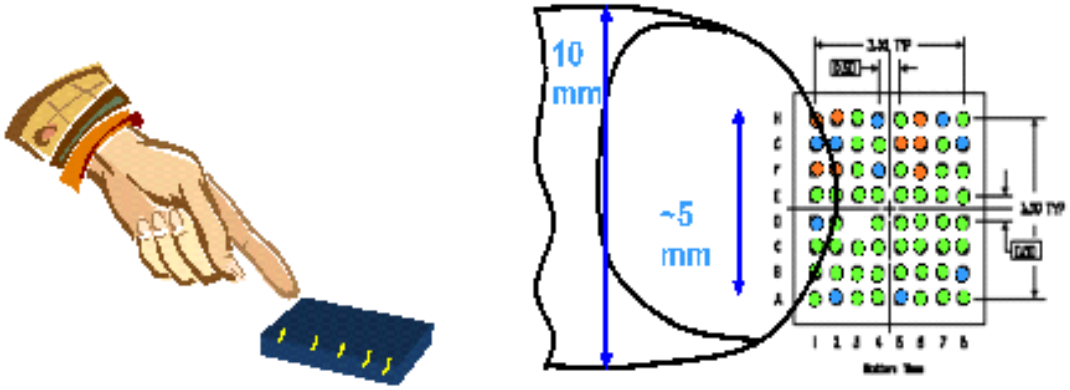


Figure 75: ESD Failure - Human handling the chip [56]

Hence the current design methodologies and testing procedures are built to test the robustness of the ESD protection devices and not to precisely model the field hazards.

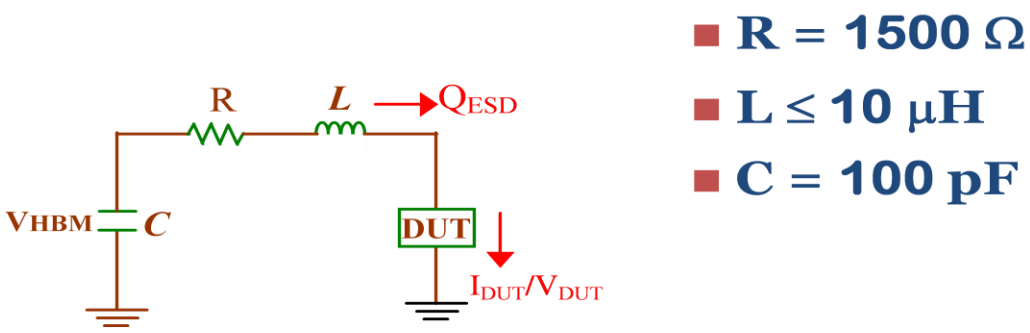
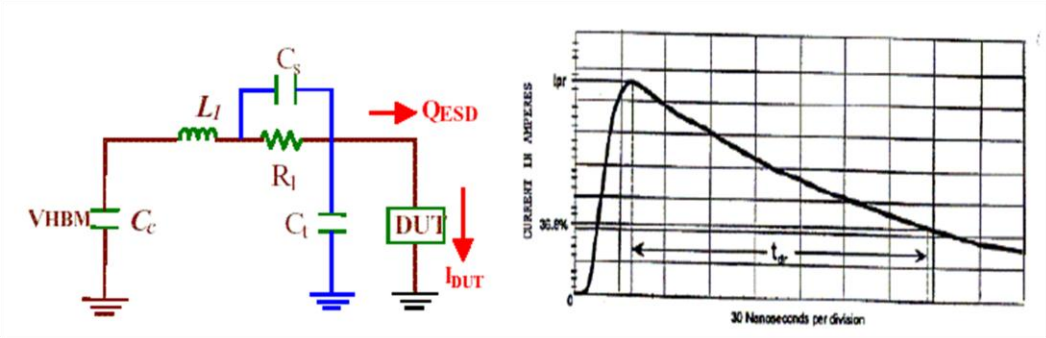


Figure 76: Simplified HMB Equivalent Circuit [56]



$R_1 = 1500\ \Omega$, $5\ \mu H \leq L_1 \leq 10\ \mu H$, $C_c = 100\ pF$

Cc: Discharge cap R1: HBM resistor L1: HBM inductance
Cs: parasitic stray cap of R1 Ct: Parasitic cap of the test board

- C_t causes the peak current I_p to decrease by up to 20% and increases rise time.
- C_s causes the rise time to decrease and the I_p to increase

Figure 77: HBM Test circuit [56]

6.4.1.3 Machine Model

Machine model is modeled on an isolated and charged conductive objective touching a pin on a sample, while the charge is discharged through another pin that is grounded. Generally, HBM tests encompass all of the MM tests in them and hence MM is getting obsolete with JEDEC [50].

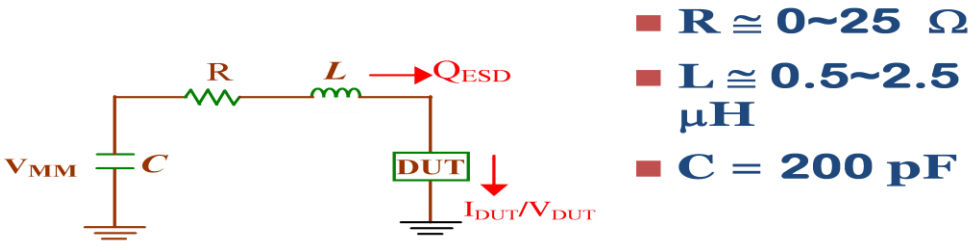
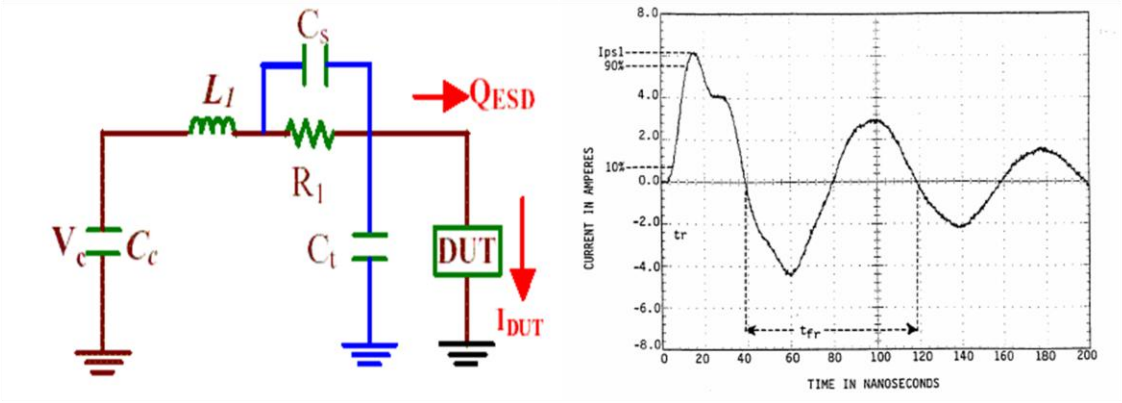


Figure 78: Machine Model Equivalent Circuit [56]



R_1 = 20 ~30 Ω, 1.5 μH≤ L_1 ≤ 3.5 μH, C_c = 200 pF

Cc: Discharge cap R1: MM resistor L1: MM inductance
Cs: parasitic stray cap of R1 Ct: Parasitic cap of the test board

Figure 79: Machine Mode Test Circuit [56]

6.4.1.4 Charge Device Model

The Charge Device Model models automated handling of parts. An ESD event that occurs when a device acquires charge through some triboelectric (frictional) or electrostatic induction process and then abruptly discharged to a grounded object. In the field, the part picks up a charge through motion in the assembly or board mounting process or by encountering an electric field, and then one pin strikes a metallic ground, discharging the part suddenly. Usual steps in CDM testing would be 100V to 250V.

A field induced CDM simulator is used to test for CDM zapping and the procedure is:

1. The sample is placed on a field charging electrode, its potential is raised by applying the test voltage to the electrode, all pins floating.
2. A POGO pin, which is tied to a ground plane through a 1 ohm resistor then touches a package pin, letting the part neutralize back to 0V (discharge).
3. Each pin will be zapped three times at both positive and negative polarity – total of 6 zaps.

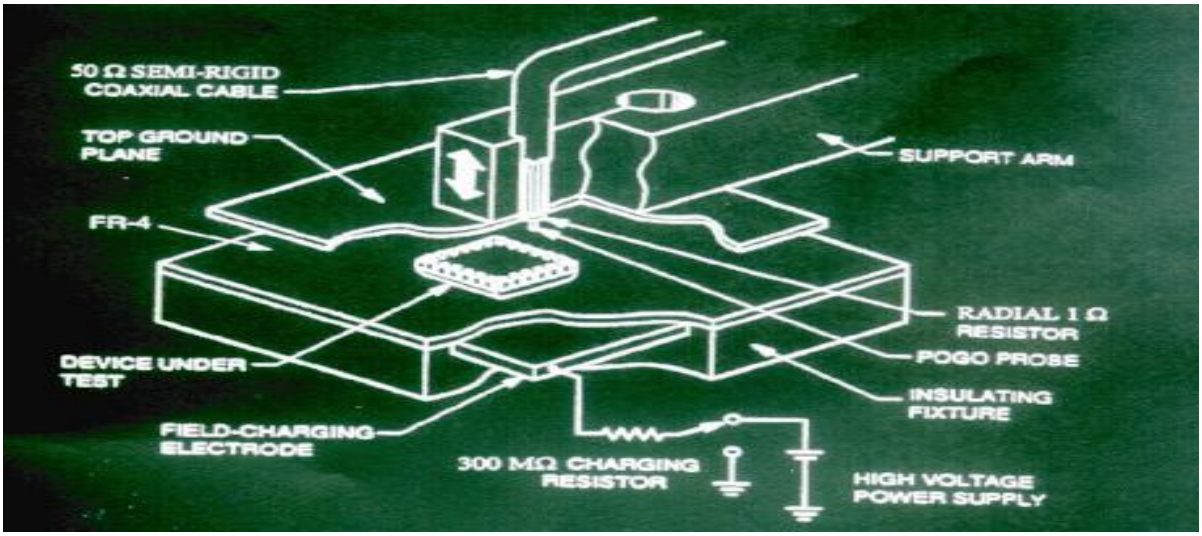


Figure 80: Field induced CDM Simulator [56]

.

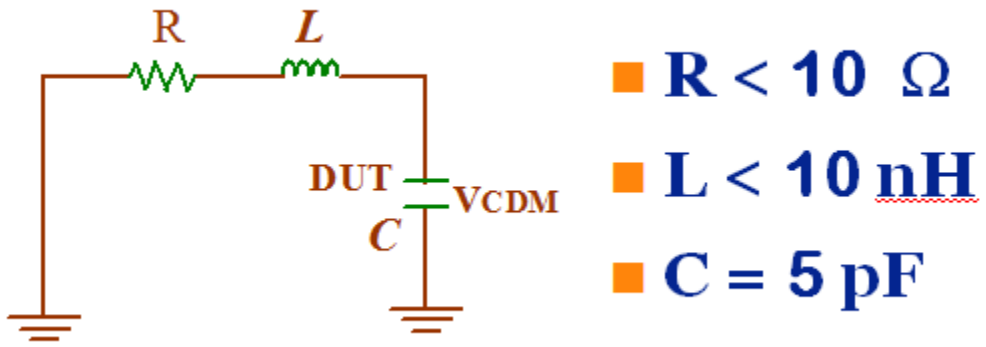
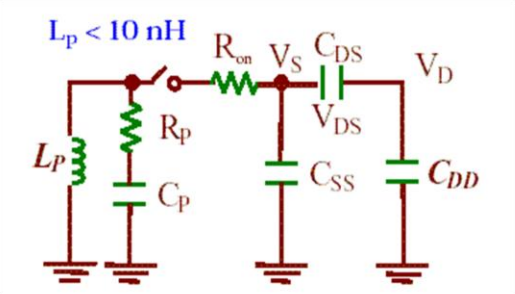
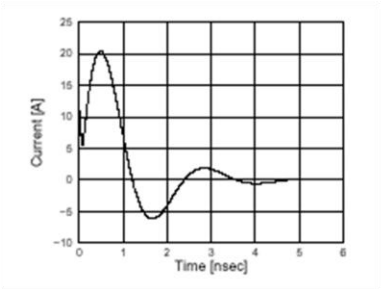


Figure 81: CDM Simplified Circuit Model [56]



VSS pin is selected as a discharge pin in this model.

1000V CDM current waveform

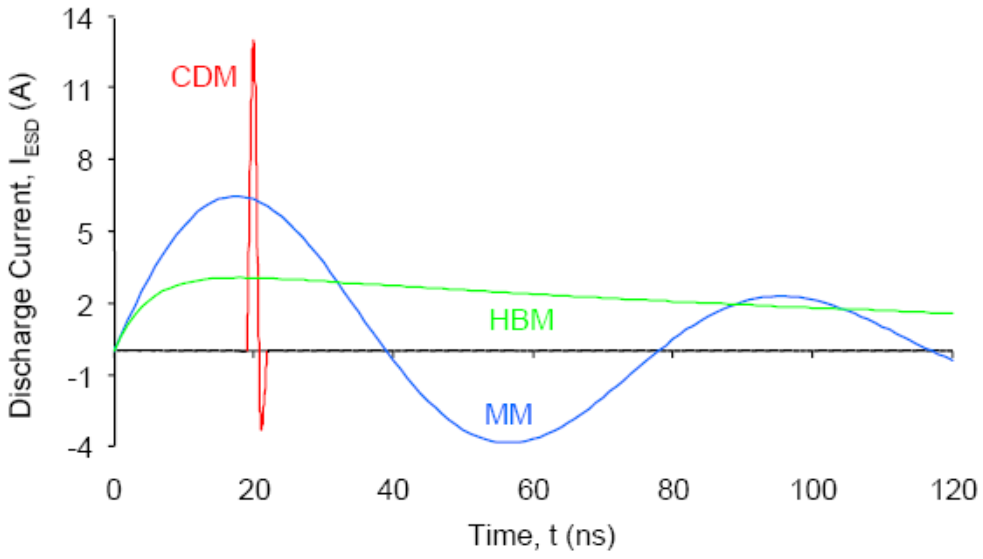


C_{DD} : Package capacitance of VDD
C_{SS} : Package capacitance of VSS
C_{DS} : VDD to VSS chip capacitance
R_P : PAD resistance
C_P : PAD capacitance
L_P : CDM discharge inductance
R_{on} : On-resistance of ESD protection device

$$C_{eq} = C_P + C_{SS} + \frac{C_{DD} * C_{DS}}{C_{DD} + C_{DS}}$$

$$I_{peak} = \frac{V_{CDM}}{\sqrt{L_P/C_{eq}}}$$

Figure 82: CDM Test Circuit [56]



IRPS Tutorial 2002 C. Duvvury

Figure 83: ESD Stress Waveform Comparison [56]

6.4.1.5 State-of-the-art ESD protection techniques

Different structures such as ESD protection diodes, diffusion resistors, GGNMOS (Grounded Gated NMOS) etc. are used as ESD protection devices. These are part of the IO cells and serve as the first level of protection against electrical overstress. The second layer of protection is included inside the IPs in form of CDM clamps, ESD inverting buffers, gate poly resistors and so on.

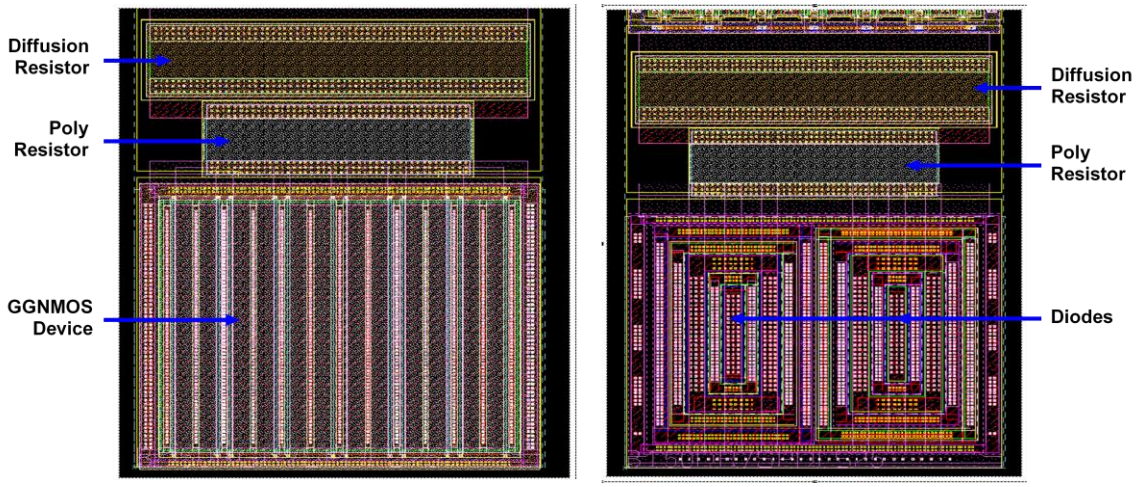


Figure 84: Some standard structures for ESD protection [54] [55]

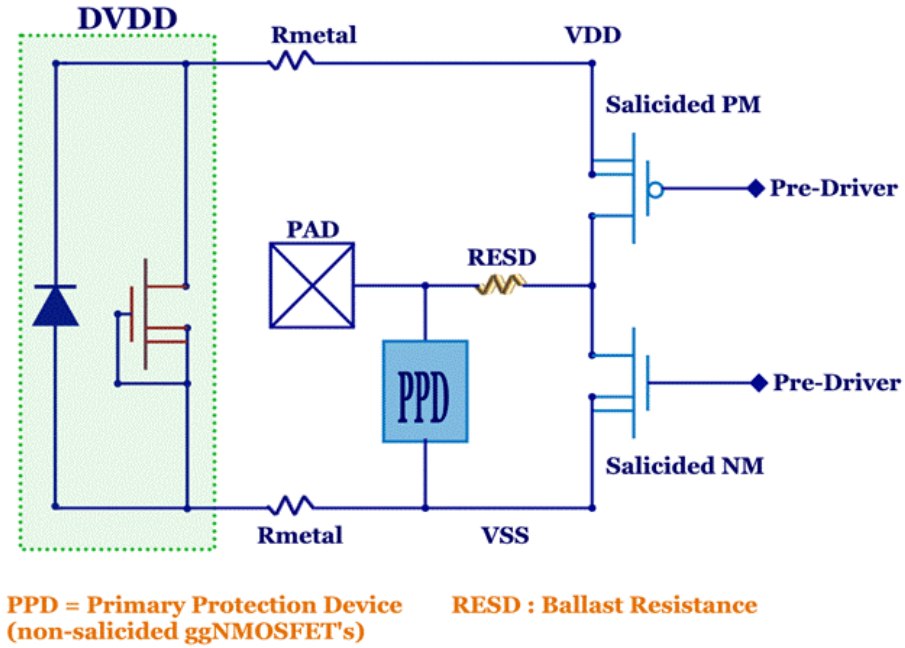


Figure 85: Sample ESD protected Output Buffer (Simplified Schematic)

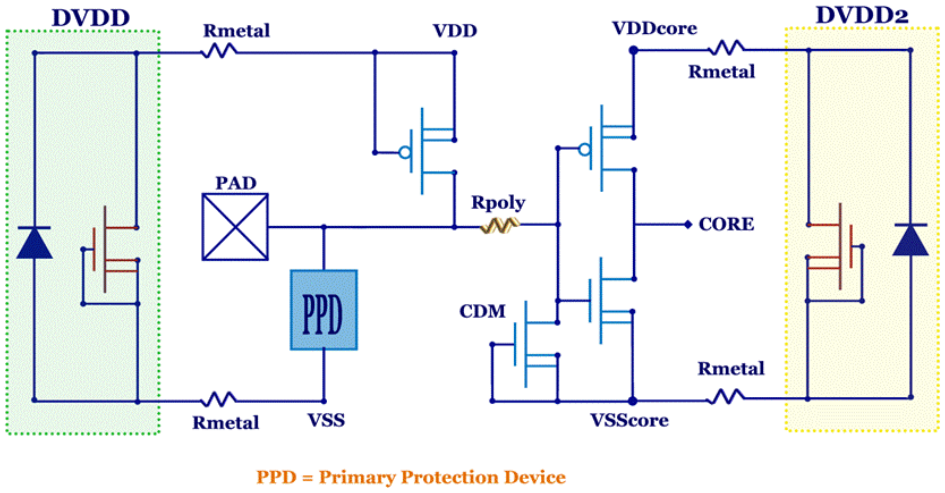


Figure 86: Sample ESD protected Input Buffer (Simplified Schematic)

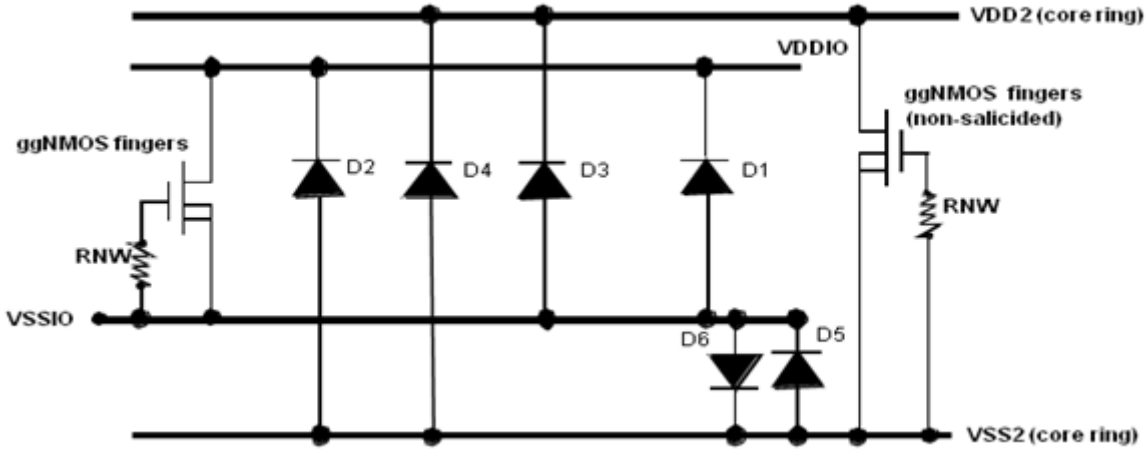


Figure 87: ESD Protection in Power Pad. VDDIO is typically 3.3V I/O Supply and VDD2 is typically 1.0V Core supply. The two grounds are connected through back-2-back ESD diodes. [55]

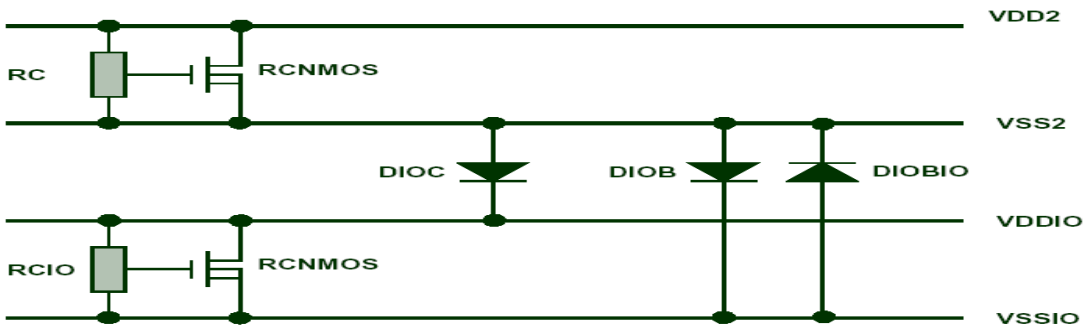


Figure 88: RC-Triggered NMOS devices could be used for primary protection on supply rails. The trigger voltage could be programmable by varying the resistance and capacitance values. Special ESD implant should be used on the transistors to reduce the threshold voltage. [55]

A sample HBM test based (ESD zapping) failure mechanism is highlighted in Figure 89. [51]. VDD1 and VDD2 are two power pads which are part of separate power domains. The worst case scenario would be a case wherein the two ground planes are separate. It is general practice to have one common ground/substrate plane for a Pwell process. The figure below shows all possible current paths in a possible ESD event. Notes in the figure show the methodology to be followed to size the output and input driver devices so as to mitigate the ESD impact. If sized appropriately, most of the current would be carried by the GGNMOS and reverse-biased diode, thus acting as primary protection devices and avoiding permanent damage.

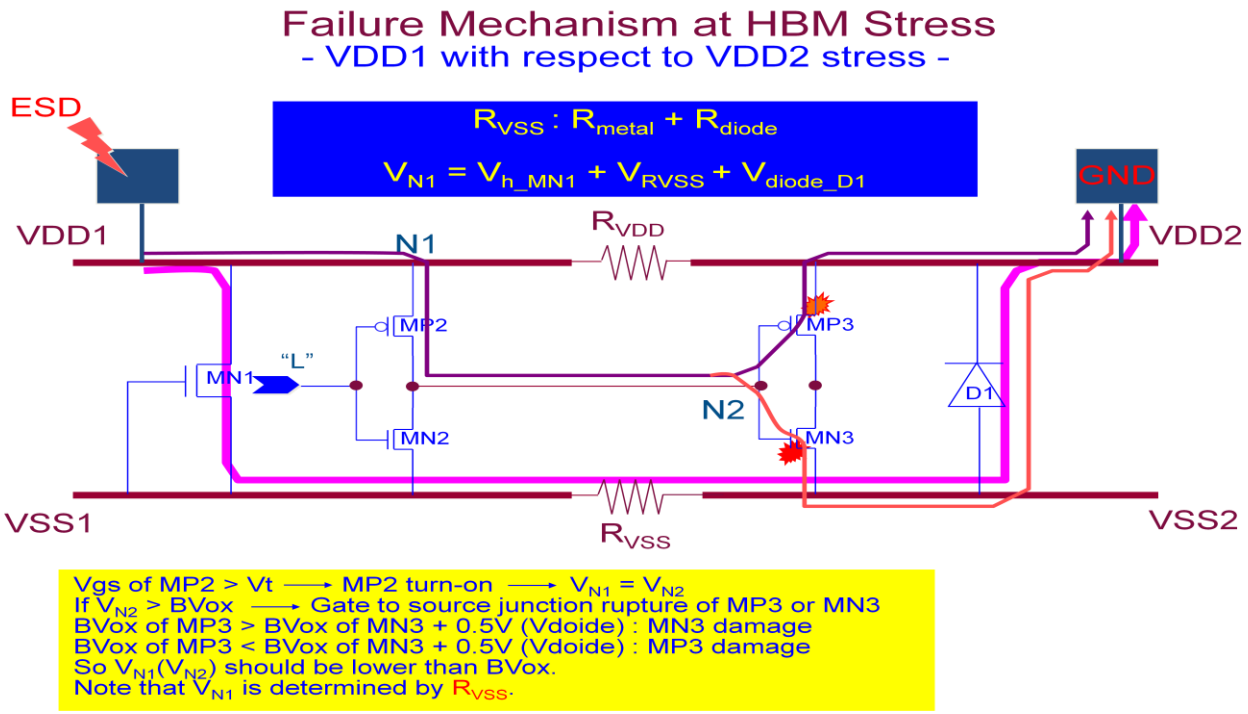


Figure 89: Failure Mechanism at HBM Stress [55]

6.4.1.6 Special ESD layout techniques

Special layout techniques might be used on the input and output driver devices to increase the source resistance and hence make them the non-preferred ESD current paths. The next few figures show some such techniques [52].

a. Every layer including metal, poly and active (n- or p-diffusion), is shaped as 45-degree polygons until the signal path changes layers (through a contact or via). By avoiding 90-degree corners, we avoid charge concentration and power surges.

b. Contacts are spaced a little away from the poly/gate to increase the active region resistance on source and drain sides. This reduces the voltage drop across the gate and increases the ESD protection characteristics of the transistor. If available, an additional layer could be used in the region between the contact and the gate poly to increase the resistivity of that region (RPO/Salicide block).

c. Distance between gate and contacts on the power side should be minimal.

d. Place multiple contacts and place them symmetrically. As seen in Figure 90, the electric field between source and drain forms symmetrical arcs. Distances *Ra1+Rg1+Ra11 = Ra2+Rg2+Ra12* ensure that the current from an ESD strike is evenly distributed evenly along the width of the transistor.

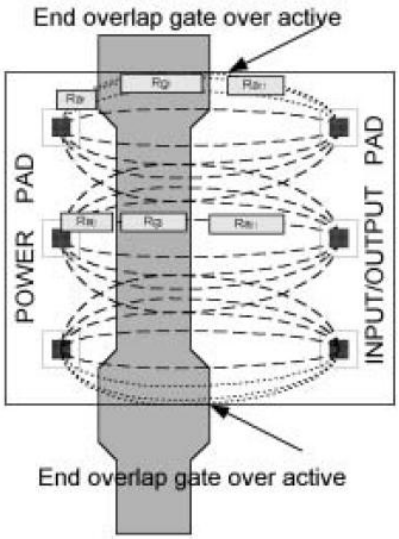


Figure 90: Layout of Output Buffer: End Overlaps [52]

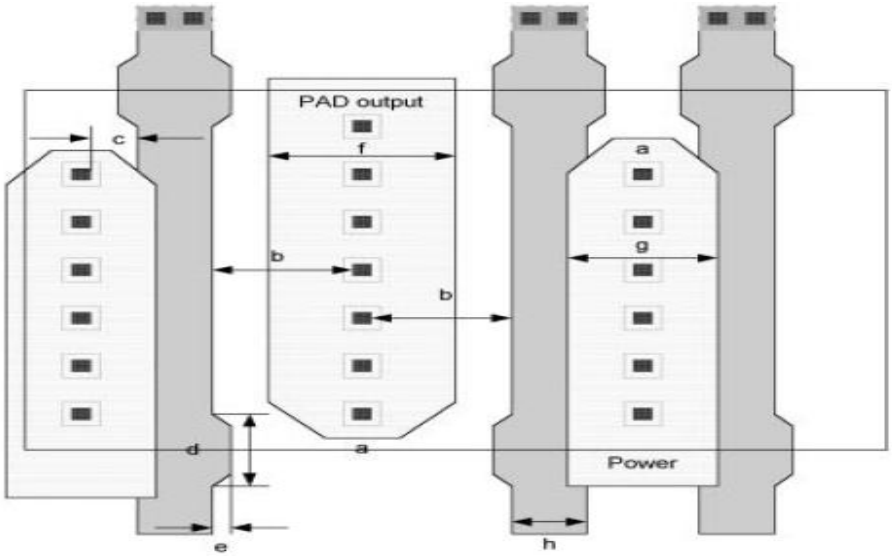


Figure 91: Layout of Output Buffer: Transistor Design [52]

e. ESD protection resistors along with Grounded Gate NMOS devices form the typical protection circuitry for an input buffer. The layout of an Input ESD protection resistor with variable setting (selectable through Focused Ion Beam (FIB) machine) is shown in Figure 92.

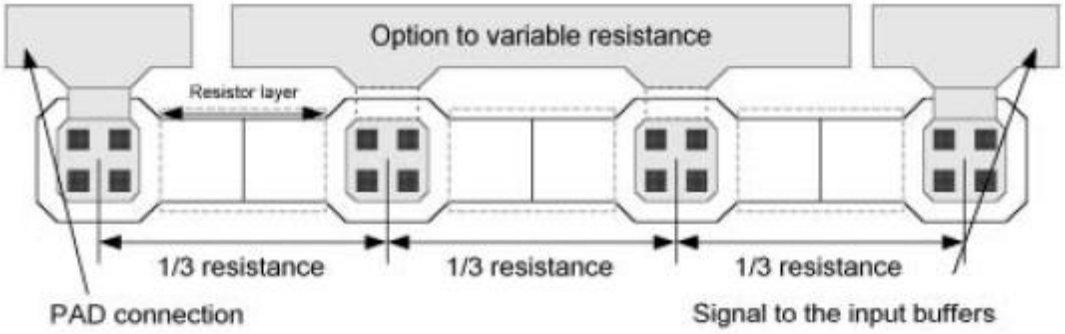


Figure 92: Input ESD Protection Resistor [52]

6.4.2 SLVS Output Buffer

6.4.2.1 General Description

The high-speed digital IOs implemented here are based on differential signaling similar to the LVDS signaling shown in Figure 93. The serial data interface can support LVDS, SLVS-200, and SLVS-400 at up to 4.32Gsps.

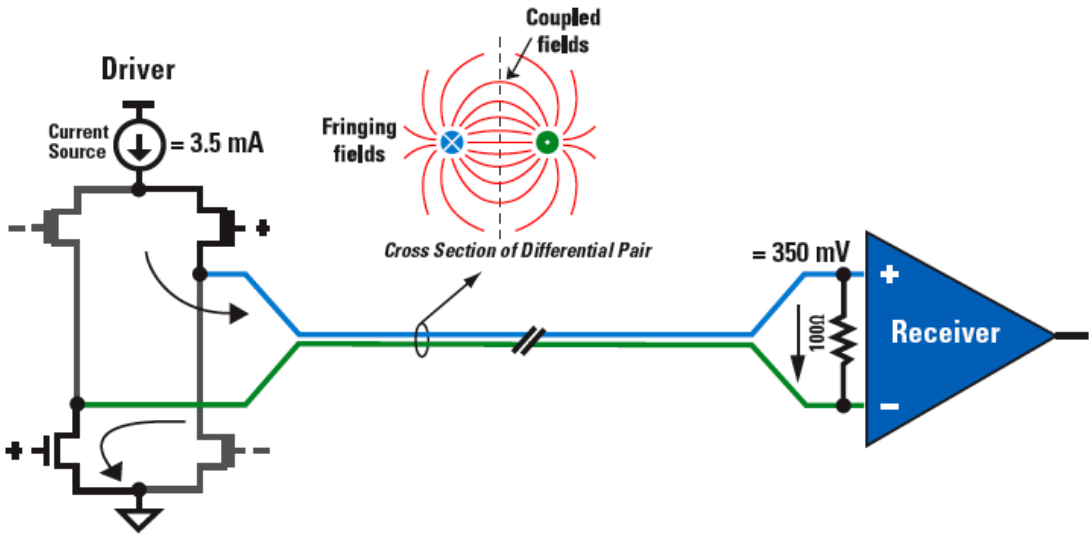


Figure 93: LVDS signaling [10]

SLVS uses differential signaling with resistive line termination both in the transmitter and the receiver. While differential signaling ensures more robustness against noise, line terminations at both ends eliminate reflections leading to improved signal integrity. The voltage swing of a single line is 200 mV (400 mV differential swing). This signaling referred to as SLVS-200 is based on the JESD8-13 SLVS-400 standard [53] and is a good compromise between power consumption, noise immunity, and VDD requirements. Hence, SLVS-200 is the primary standard chosen for implementing the high-speed serial interface though LVDS and SLVS-400 are also supported.

The pulse-shaped outputs, being purely analog in nature, require analog 50 Ω output buffers.

6.4.2.2 SLVS/LVDS Output Buffer Swings

The SLVS/LVDS Output buffer has 5-level programmable swings as shown in Table 8.

Table 8: Output buffer voltage swings (into 50ohms)

Output Setting {c1,c0_bar}	Differential Peak (mV)	Standard
01	106	-
00	203	SLVS-200
11	358	LVDS
10	408	SLVS-400
Max. Swing Mode FSM = logic 1	700	-

6.4.2.3 Functional Schematic Diagram

In its simplest implementation, an SLVS buffer switches the direction of 2mA current through an external resistive termination (50Ω), depending on the data pattern.

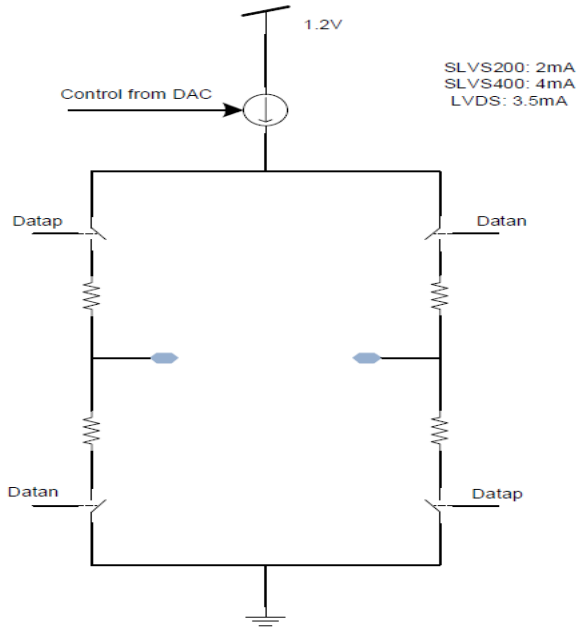


Figure 94: Output Driver section for a simple SLVS implementation

The reference current source shown above can be modified to support various standards such as SLVS200, SLVS400 and LVDS. The exact signal swings into a 100Ω differential termination have been illustrated in the previous sub-section. The current DAC setting is done using a 2bit decoder. A large buffer chain boosts the incoming data before switching the transmission gates on the output section. These transmission gates are huge since they carry a peak current of over 6.5mA (Fail Safe Mode). Under all cases, these gates offer 16Ω of impedance. To match this to a 50Ω termination, we have added a series resistance of 34Ω.

A special feature to disable the IO cell when it is not in use; has been provided. The Fail Safe Mode feature (FSM = logic '1') will maximize the reference current; thus giving us the maximum swing at the output.

6.4.2.4 Visio Schematic

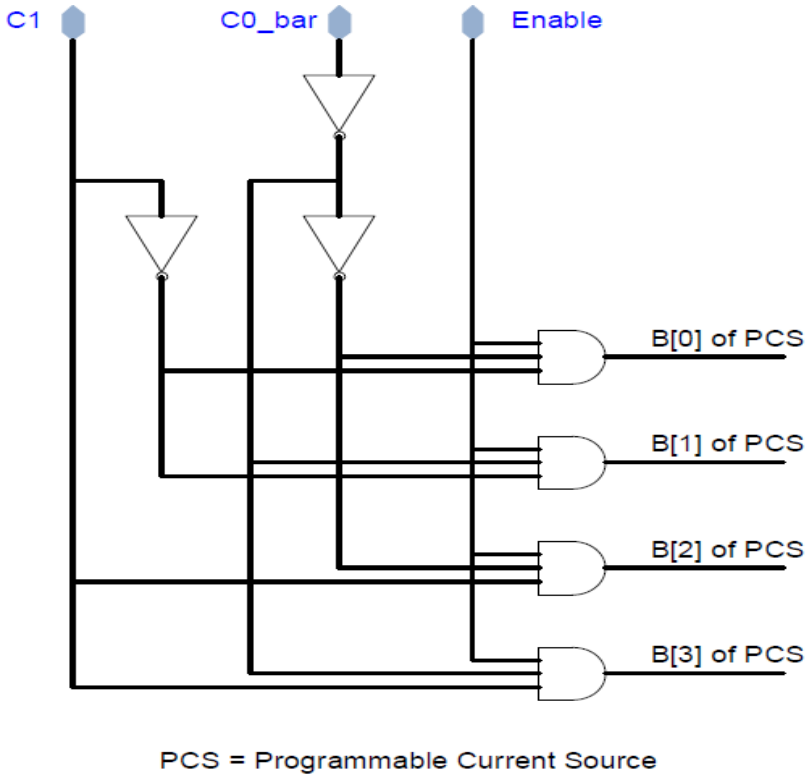


Figure 95: {C1, C0_bar} 2:4 decoder for DAC settings

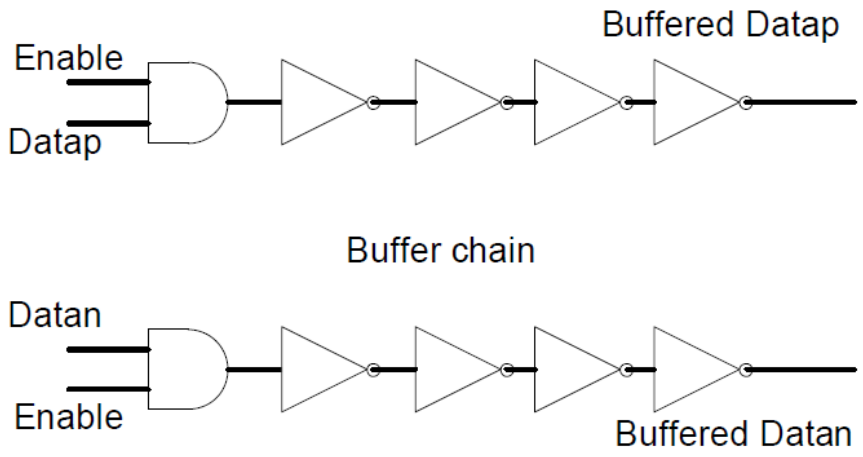


Figure 96: Circuit to buffer Datap, Datan; which drive large transmission gates in output driver section

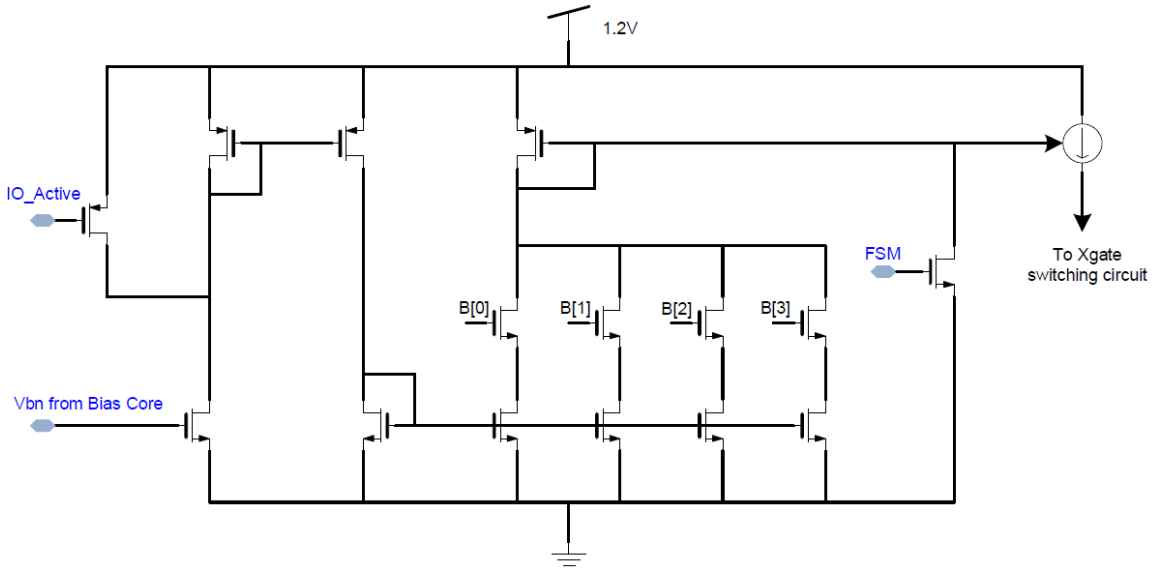


Figure 97: Current DAC for setting Output Driver Constant Current Source, as per the IO standard being supported

6.4.2.5 Layout

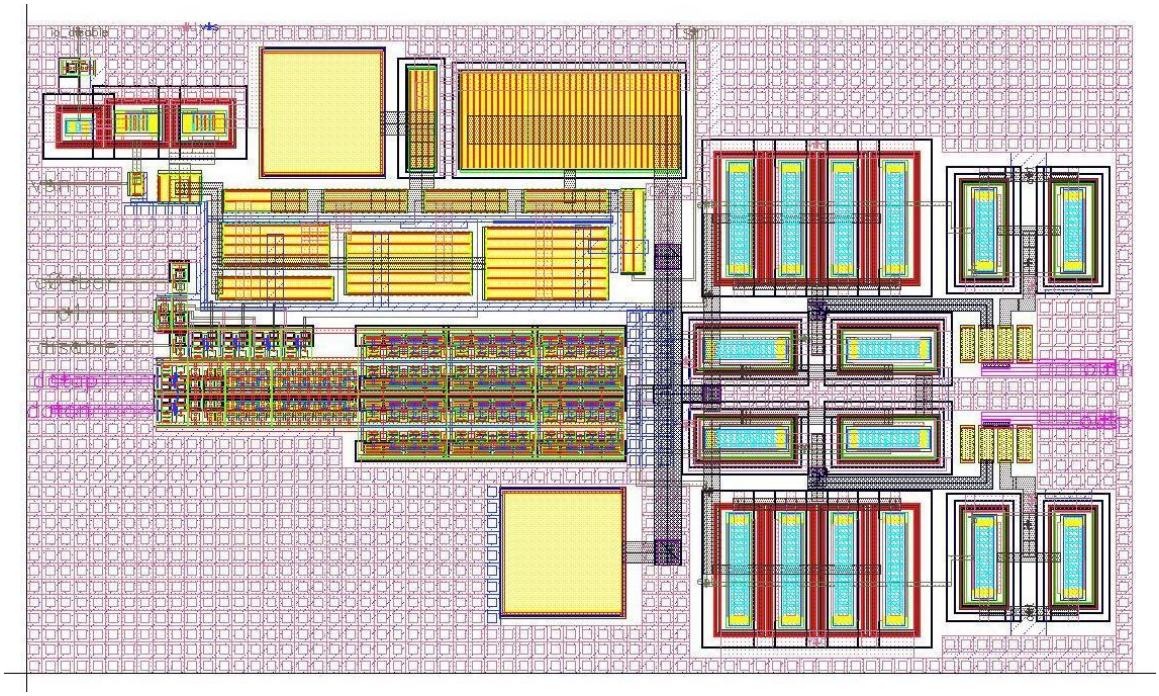


Figure 98: Layout of the SLVS Output Buffer

6.4.2.6 Performance Summary

Table 9: SLVS Output Buffer Performance Summary

C1	C0_BAR	F_{MAX} (Data rate)	V_{OUT} (peak-to-peak)	Power Consumption
0	1	5GHz	+/- 40mV	10.364mW
0	0	5GHz	+/- 45mV	10.749mW
1	1	5GHz	+/- 160mV	13.175mW
1	0	5GHz	+/- 170mV	13.525mW
FSM = Logic 1		5GHz	+/- 350mV	20.211mW

6.4.2.7 Component List

Table 10: SLVS Output Buffer Component List

Block Name	Test Bed
SLVS Output Buffer	Back-to-Back Transient Simulation Testbench
Inverter (drive strength X1)	
Inverter (drive strength X2)	
Inverter (drive strength X4)	
Inverter (drive strength X8)	
Non-inverting Buffer (drive strength X12)	
3 input AND gate	
Transmission Gate with PMOS and NMOS	
Transmission Gate with only NMOS	

6.4.3 SLVS Input Buffer

6.4.3.1 General Description

The SLVS output buffer generates a switching current of typically 2mA pk-pk amplitude. This current flows through the internal termination of 50Ω (inside this digital input buffer). A total swing of 200mV_{p-p} is developed which is then passed through an inverter chain. These inverters are appropriately biased, such that the input differential swing swings around the inverter switching threshold (V_M). To ensure that we have biased the inverters near the switching threshold under all conditions (irrespective of variations in process corners, temperature or operating voltages), we use a special biasing technique.

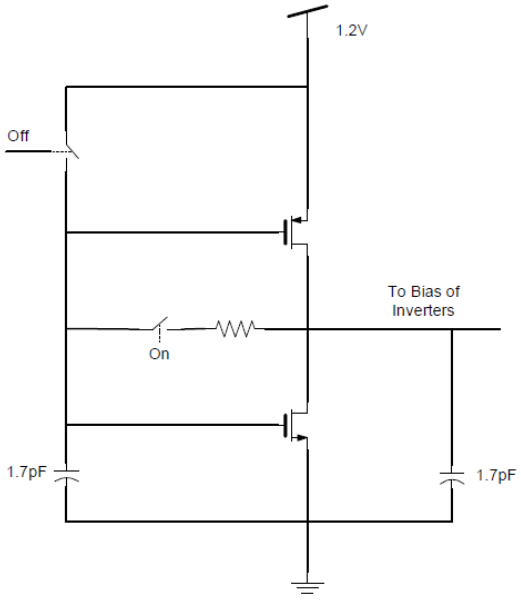


Figure 99: Bias generator for Inverters

An identical inverter is used to generate the bias voltage. This inverter has a switch that would short its output, back to its input; hence forcing the inverter to settle at V_M, its switching threshold. This bias voltage sets the input differential voltage to swing around VM. We get a much larger swing at the output of the first set of inverters on data path. After the signal is sufficiently boosted through the second set of inverters, it is fed into a buffer chain to drive a

load of 20fF on each node. Simulation results with 4.4GHz data rate and complete loading, is shown in the next section.

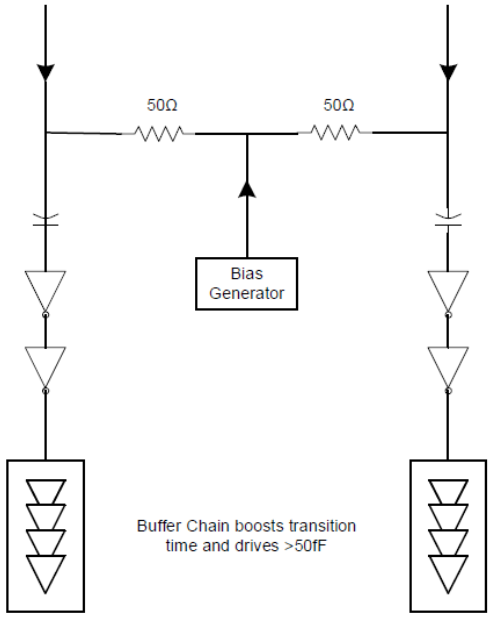


Figure 100: SLVS Digital Input Buffer functional block diagram

6.4.3.2 Visio Schematic

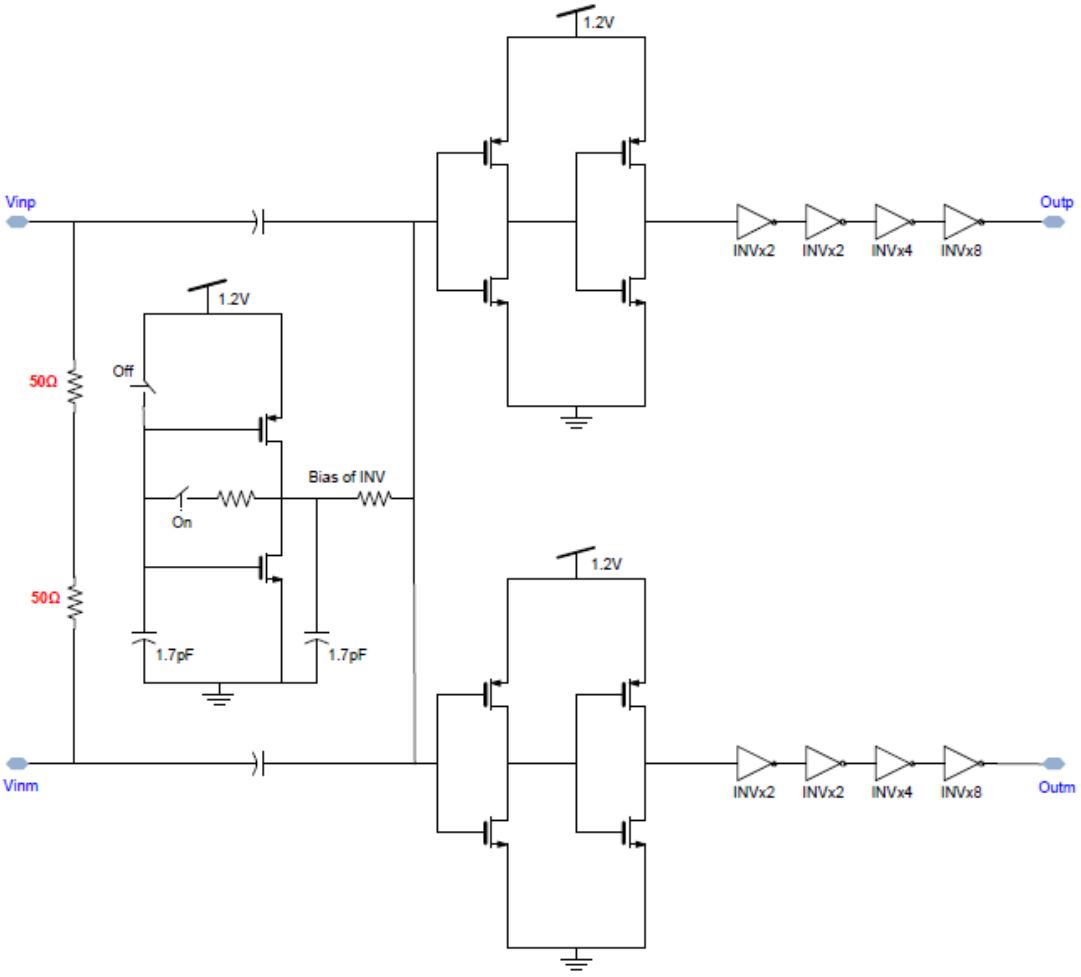


Figure 101: SLVS Digital Input Buffer

6.4.3.3 Layout

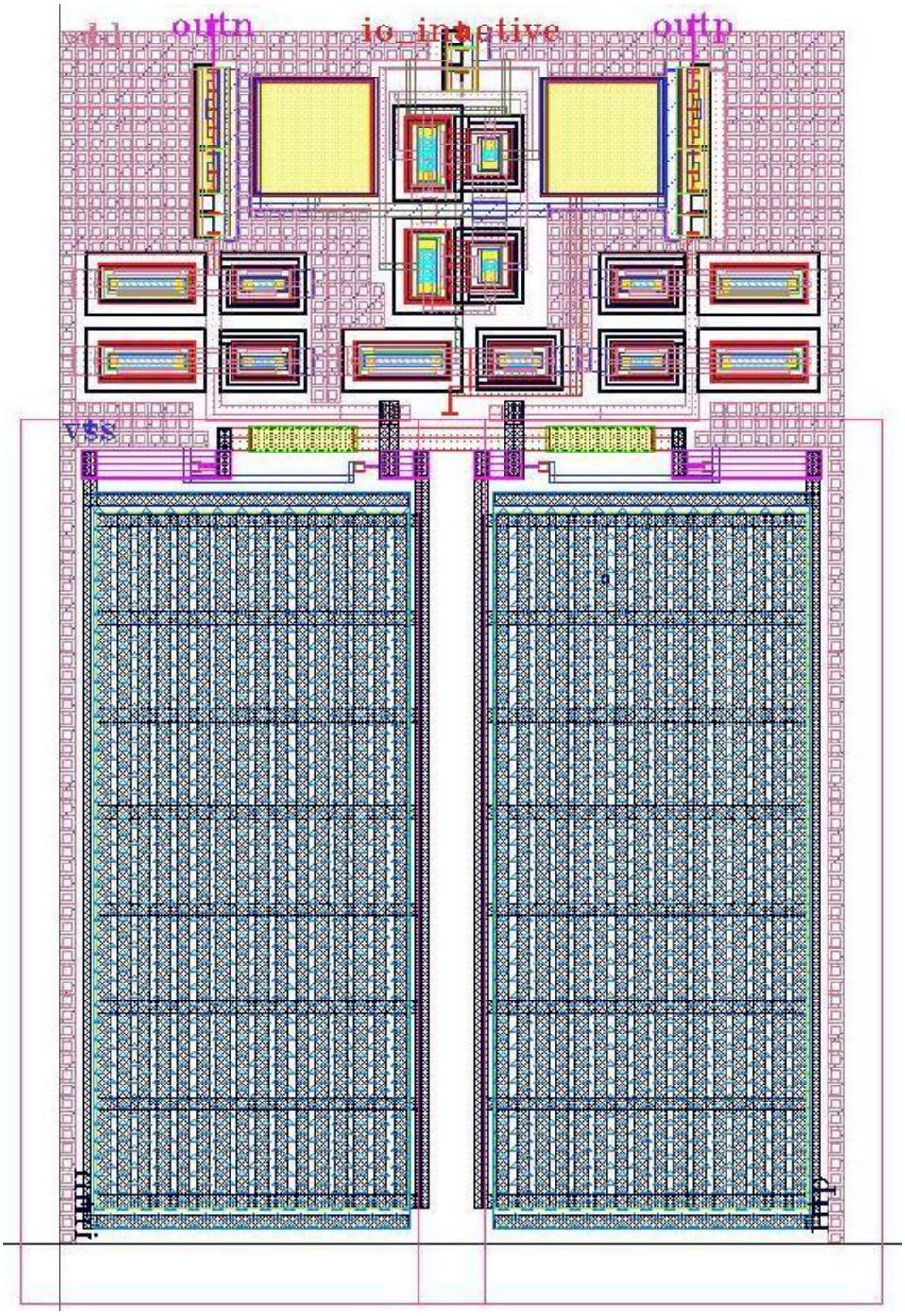


Figure 102: Layout of SLVS Input Buffer

6.4.3.4 Performance Summary

Table 11: SLVS Input Buffer Performance Summary

F_{IN} – Cc extracted simulations	**5GHz***
Input Amplitude	10mV peak-peak
Power Consumption	3.373mW

** Verified operation up to 4.4Gbps in measurements*

6.4.3.5 Component List

Table 12: SLVS Input Buffer Component List

Block Name	**Test bed**
SLVS Input Buffer	Back-to-Back Transient Simulation Test bench
Inverter (drive strength X1)	
Inverter (drive strength X2)	
Inverter (drive strength X4)	
Inverter (drive strength X8)	
Transmission Gate switch	

6.4.4 SLVS Input-Output Buffers back-to-back Simulation Results

A test schematic was created with the SLVS output buffer driving a long interconnect, which in turn drives an SLVS input buffer. The test schematic is as shown below in Figure 103.

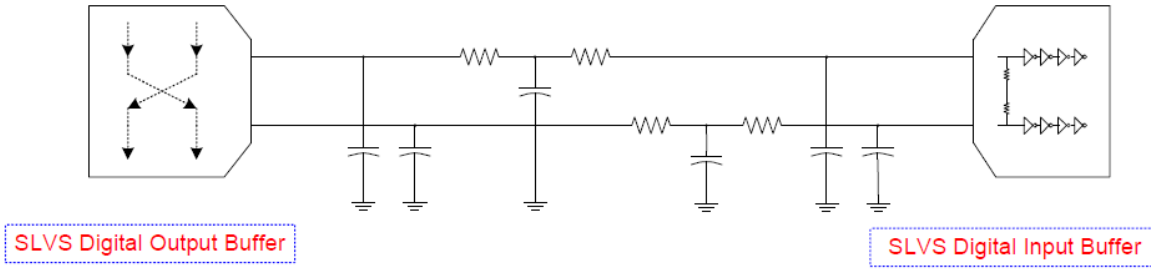


Figure 103: Test setup using SLVS I/Os back-to-back. Simulations show that speeds upto 5Gbps are easily supported

Conditions:

1. Data rate = 5.1Gbps
2. Output buffer setting = 400mVp-p (C1=1, C0_bar=0) (2mA drive)

The signal swing does not appear to be balanced for a truly differential output, yet we are able to recover the complete swing through the digital input buffer. We must also note that the input buffer is not truly differential but pseudo-differential.

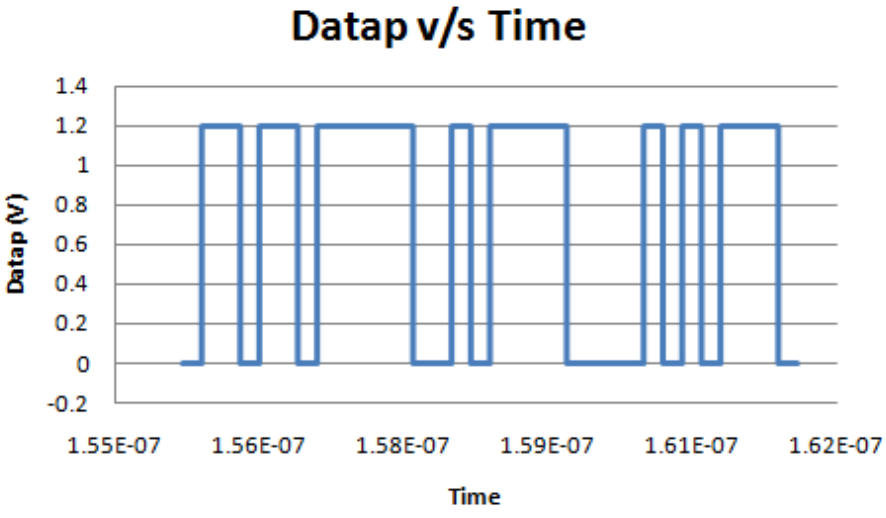


Figure 104: Plot of Datap – Input to SLVS Digital Output buffer

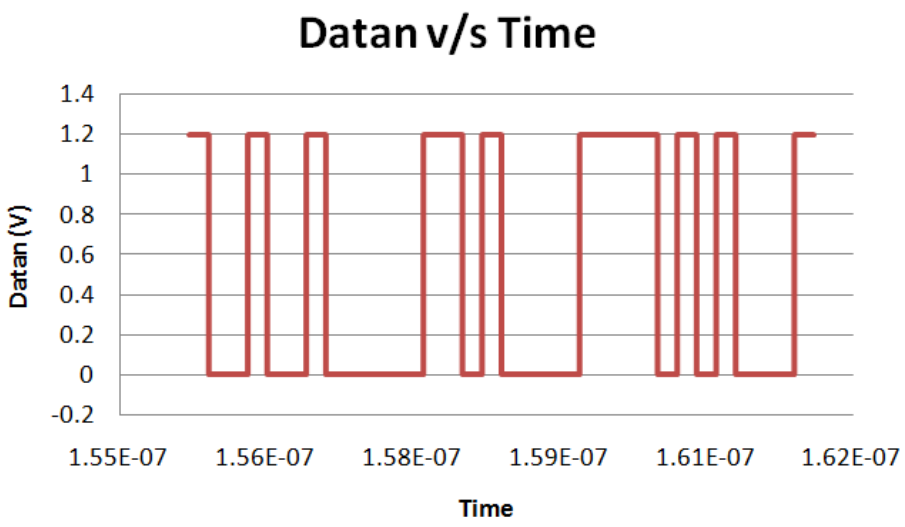


Figure 105: of Datan - Input to SLVS Digital Output buffer

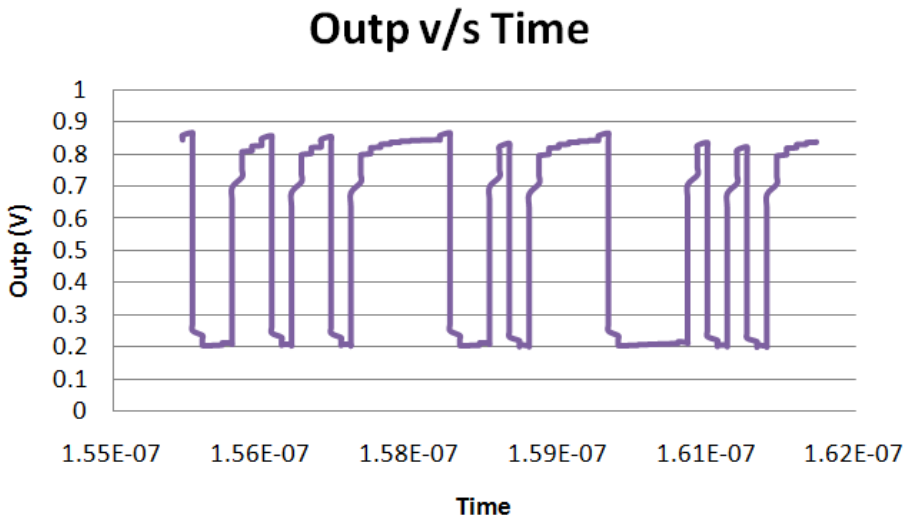


Figure 106: Plot of Outp – Output of SLVS Digital Output buffer

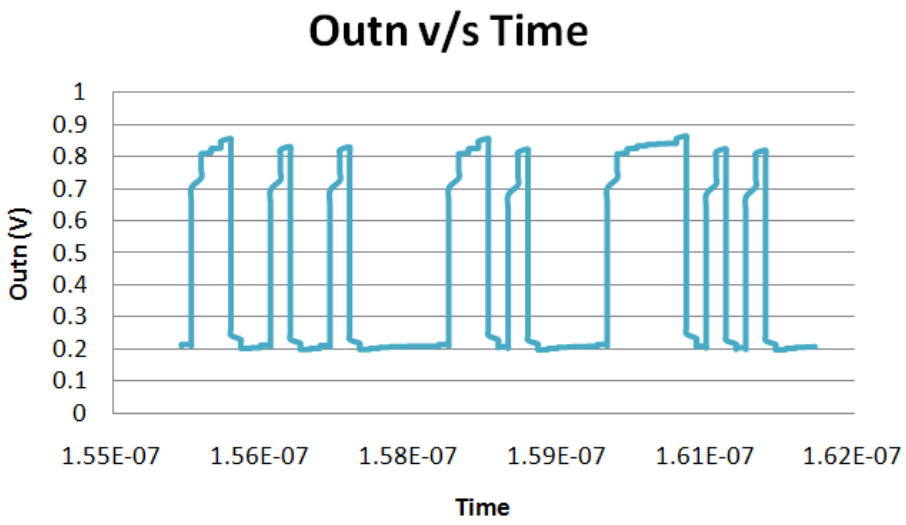


Figure 107: Plot of Outn - Output of SLVS Digital Output buffer

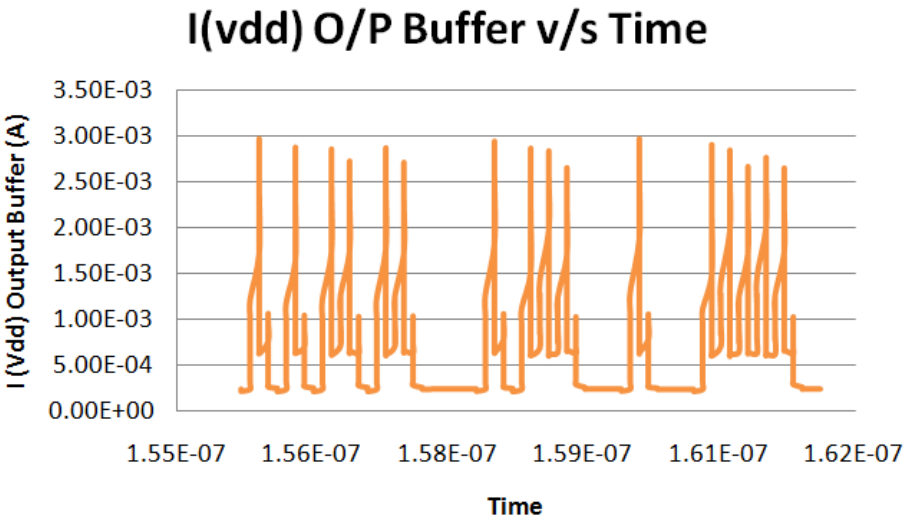


Figure 108: Power supply current drawn in SLVS Output buffer

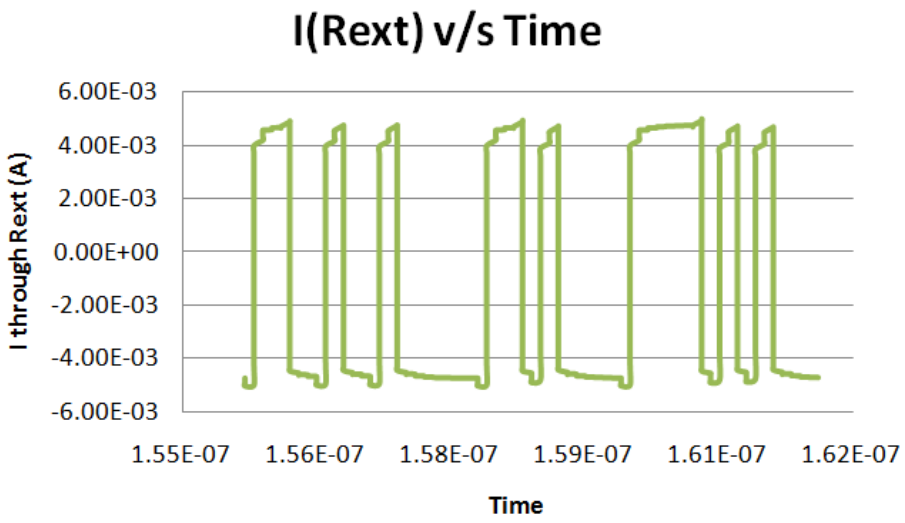


Figure 109: Current output of SLVS Digital Output buffer

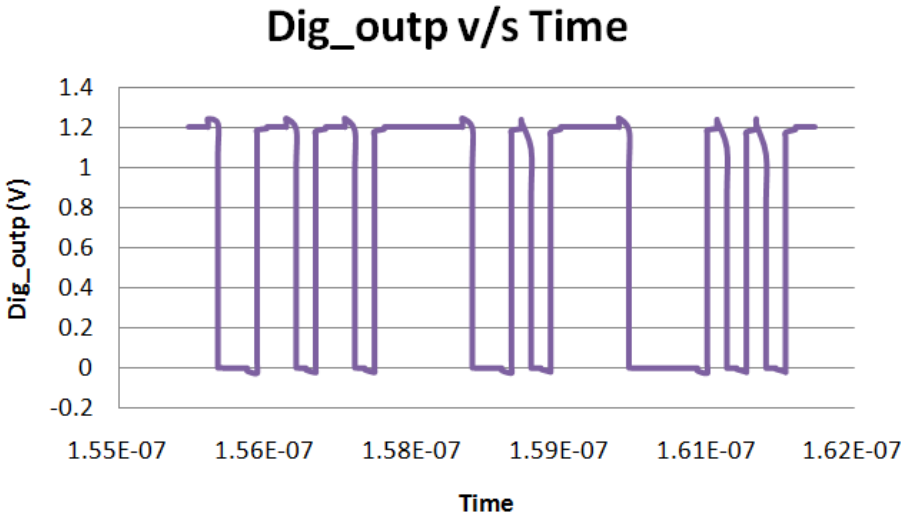


Figure 110: Digital output of SLVS Digital Input buffer

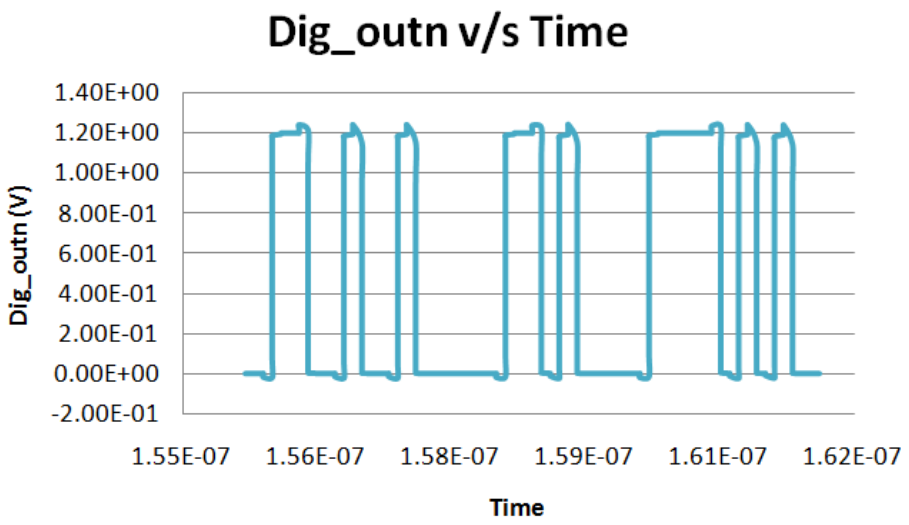


Figure 111: Digital output of SLVS Digital Input buffer

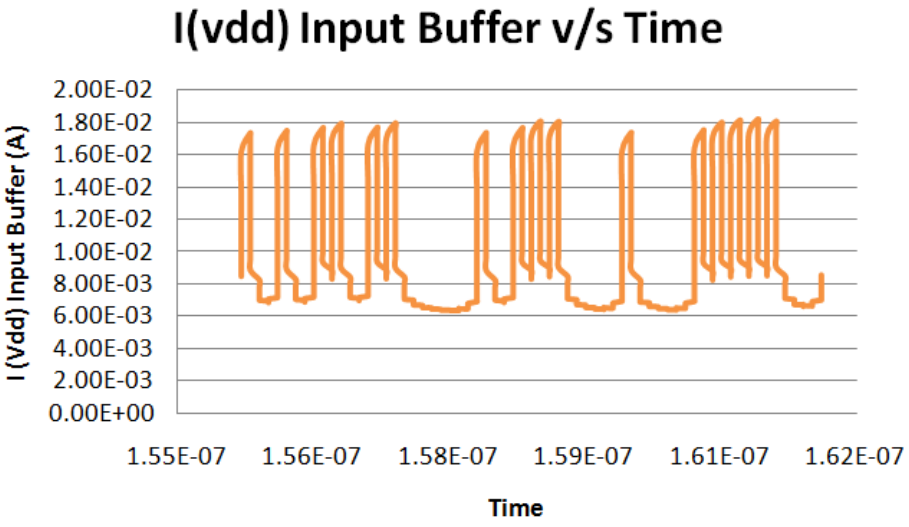


Figure 112: Power supply current drawn into SLVS Digital Input buffer

6.4.5 Measured Results

The designed LVDS-SLVS I/Os were integrated into the chip and used at data rates of up to 4.32GS/s. The proper operation of the output buffer can be seen from the eye-diagrams in [Figure 113 and Figure 114]. However, measuring the input buffer was not feasible. The entire loop was tested to be functioning properly, thus indirectly testing the input buffer at data rates of 1.728GS/s, 3.456GS/s and 4.32GS/s.

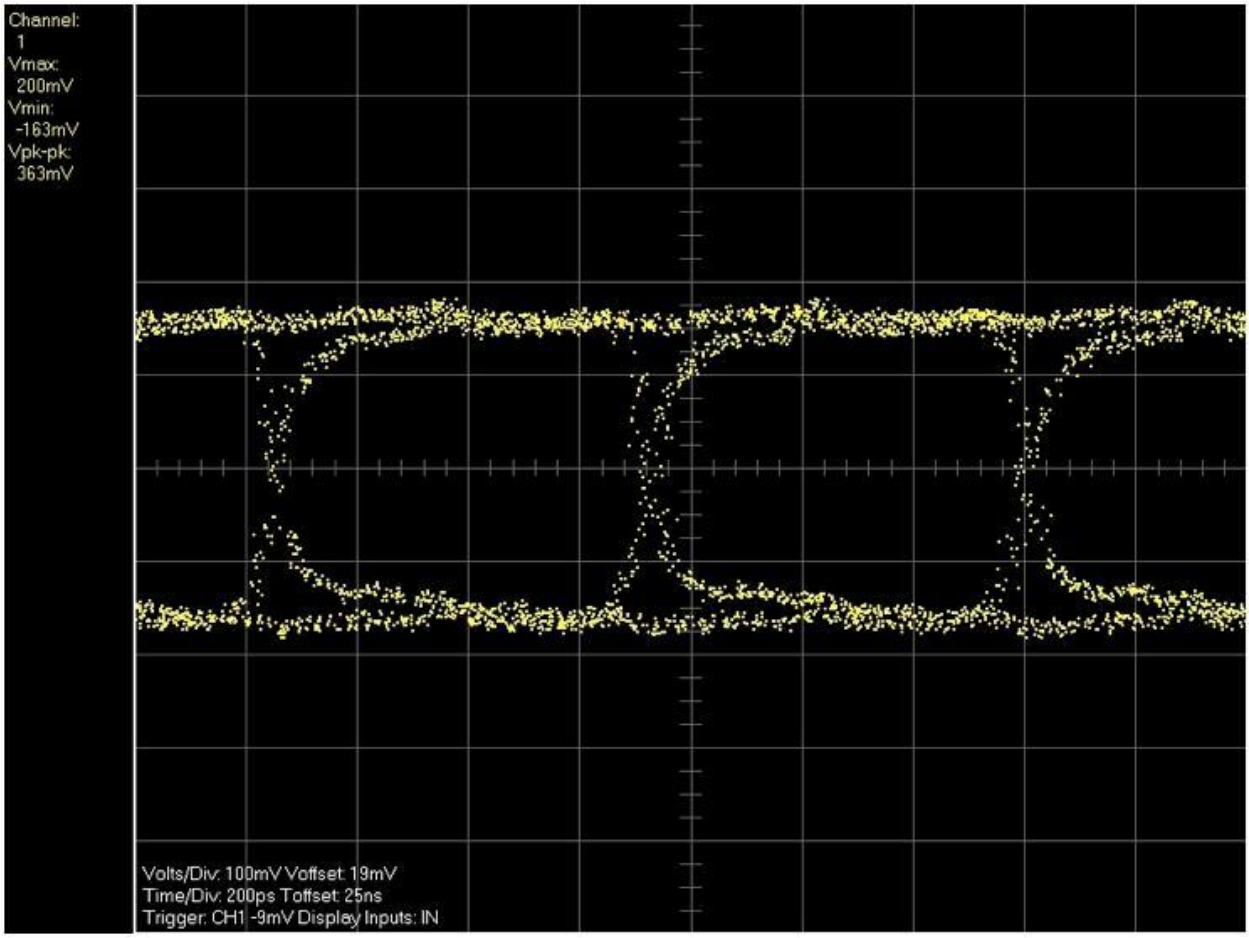


Figure 113: 1.728 Gbps eye Diagram

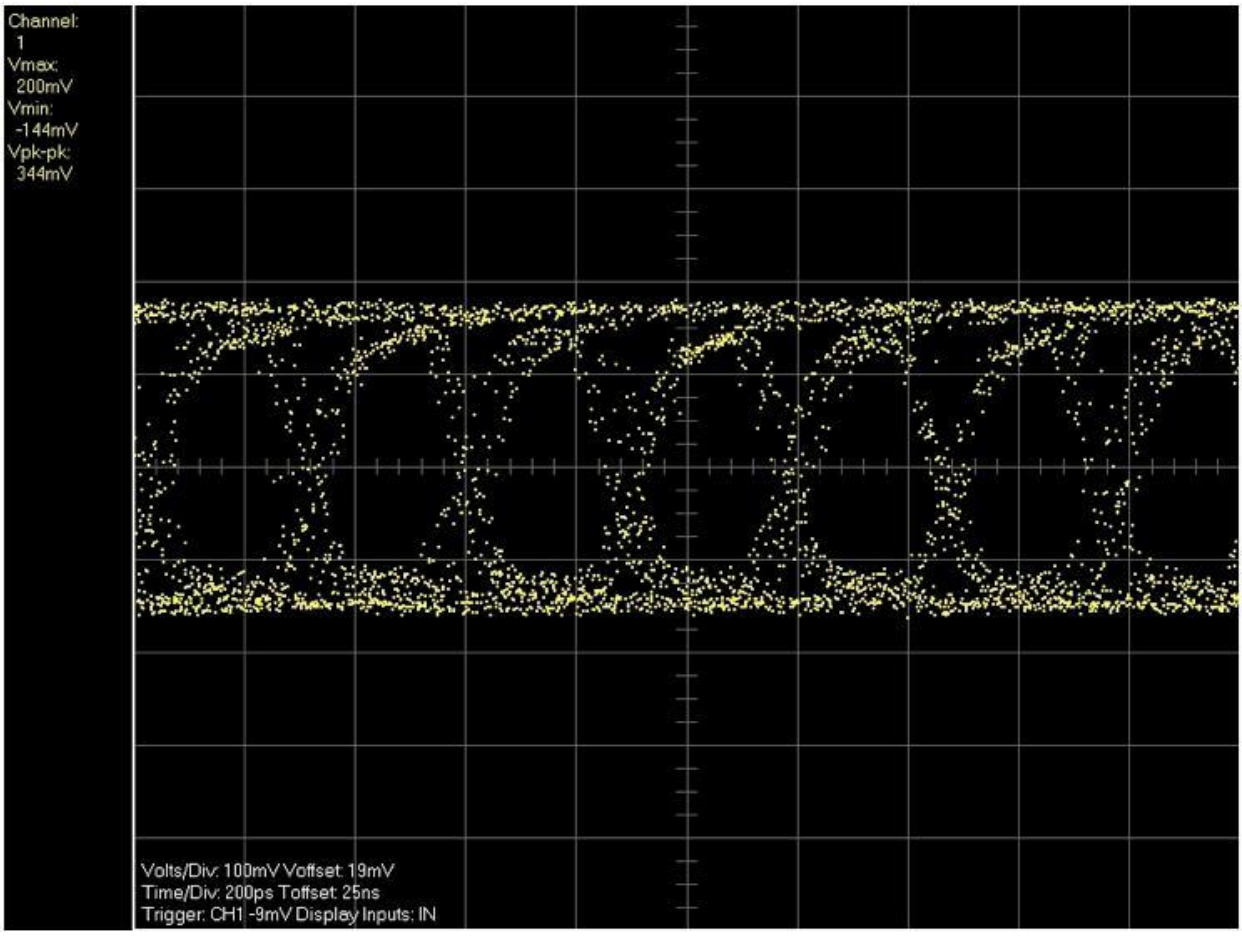


Figure 114: 3.456 Gbps eye Diagram

6.4.6 PHY-MAC Chip I/O Ring

During the first phase of design, a separate chip was designed for Baseband processing. This chip included the functional blocks of PHY (PHYsical layer) and MAC (Media Access Control from the Data Link Layer). These two blocks were part of a separate chip which needed its own I/O ring. The baseband chip included an on-chip VCO for clock generation. A few analog pads were needed to tune this VCO. The following screenshots outline the flow that was used during the I/O ring development.

1. The I/O pad requirement was estimated from the top level block diagram for the Baseband chip (PHY + MAC). Additional I/Os were needed for Clock Generation and Miscellaneous logic. Physical only I/O pads such as Power supply pads, ESD pads, corner pads etcetera were added to the top level Verilog netlist. Filler I/O pads are automatic inserted as part of the physical design flow.

2. Since the overall RF CMOS Radio (Figure 115) was developed as a multi-chip solution, block diagram of the parallel interface sub-section was also considered during the generation of the parallel interface partial I/O ring.

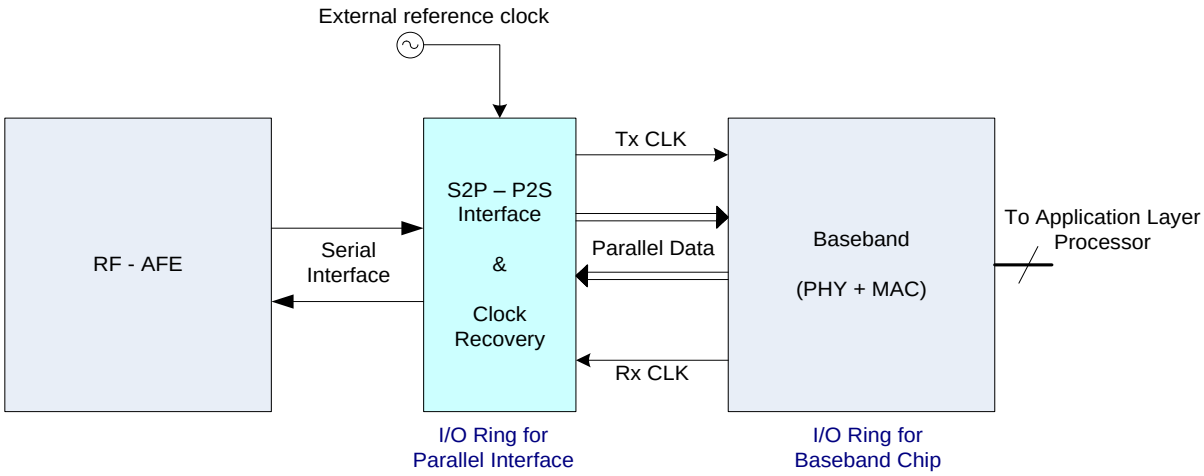


Figure 115: Multi-chip solution developed for the RF CMOS Radio (version-1)

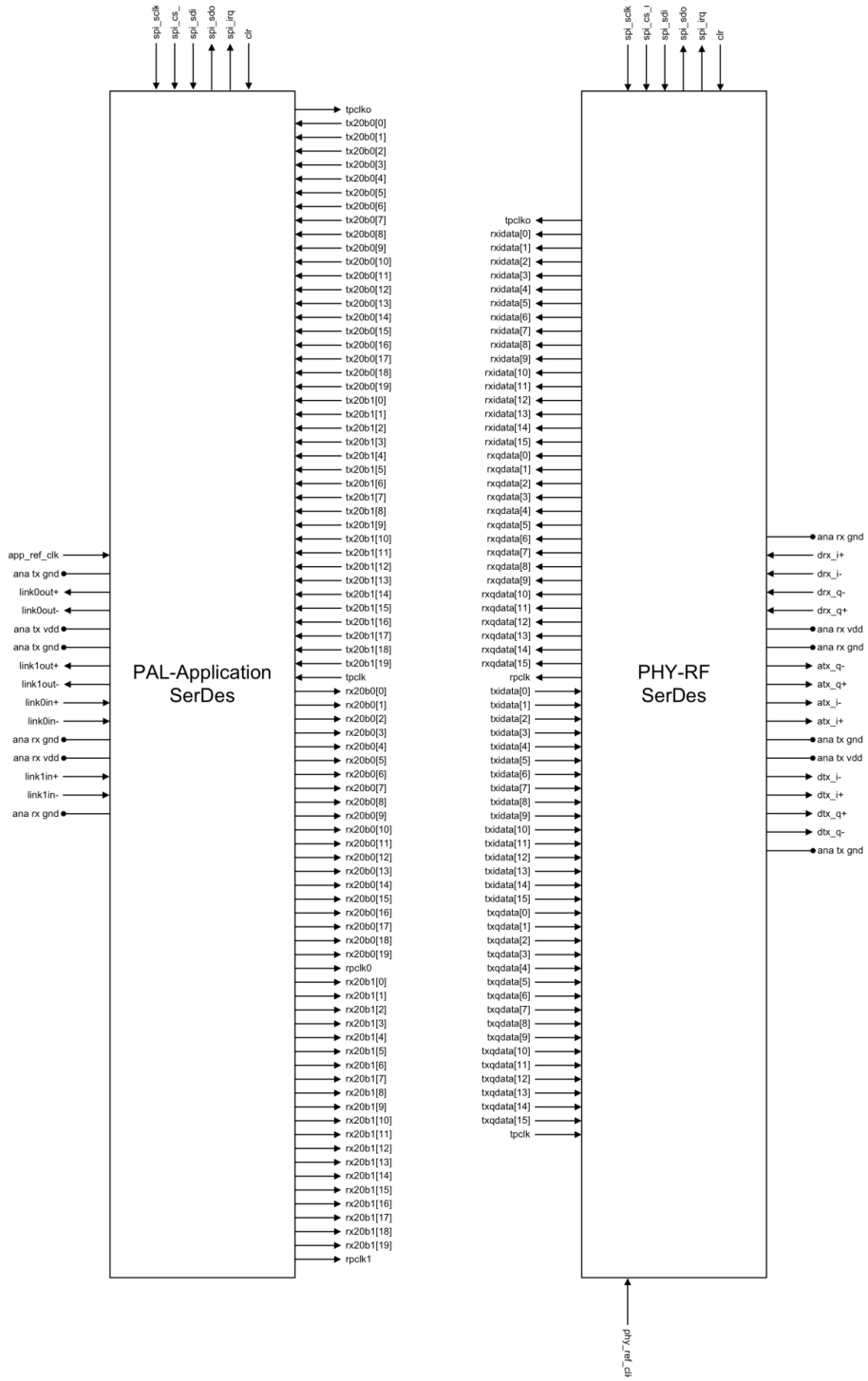


Figure 116: Pin locations for the Application-PAL interface block and the PHY-RF interface block

3. From the list of I/O signals, a spreadsheet representing the actual I/O pad assignment and the pad-cell to be used (from the STMicro IO Library) was developed.

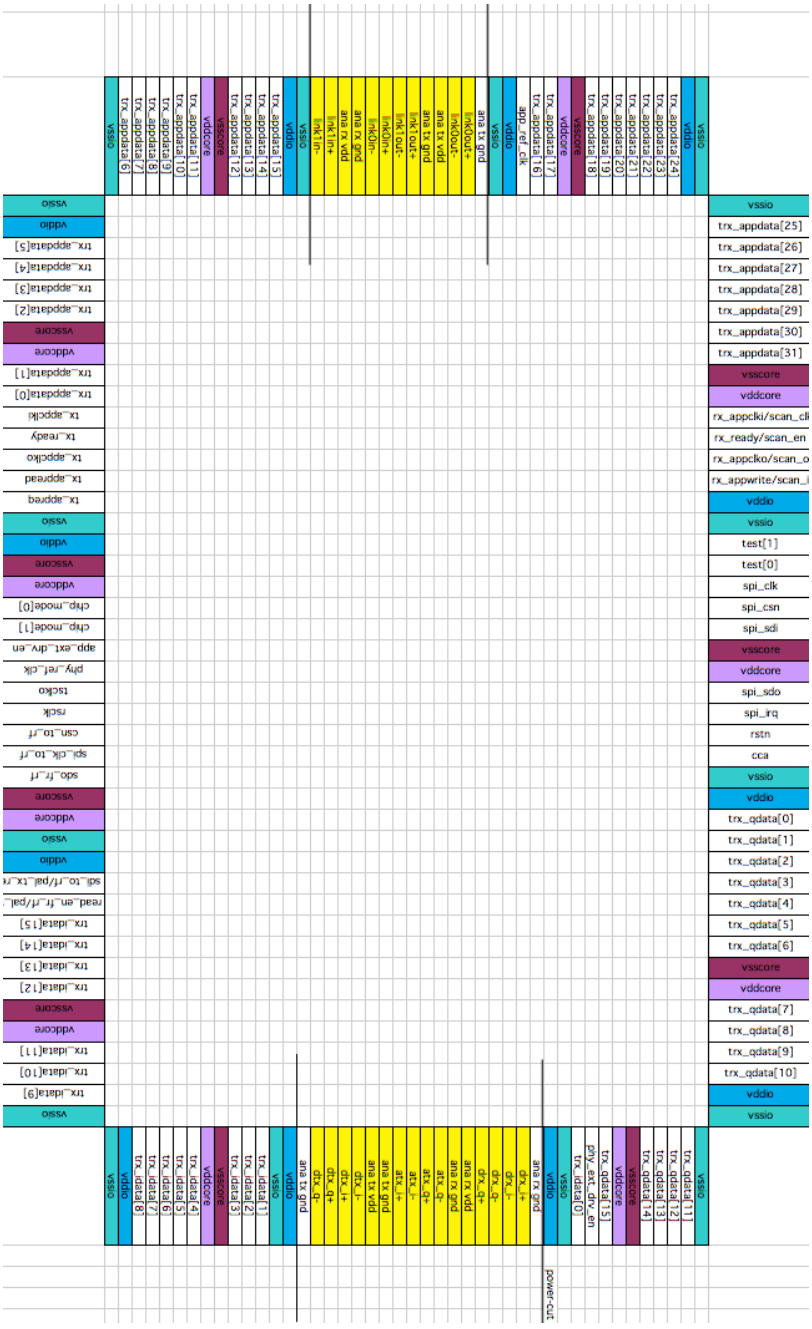


Figure 117: Full chip IO plan (Application on top & RF at bottom)

4. A fake bonding diagram for the various I/O ring sections was developed. CUP (Circuit Under Pad) I/O cells with staggered I/Os was used as seen in Figure 118.

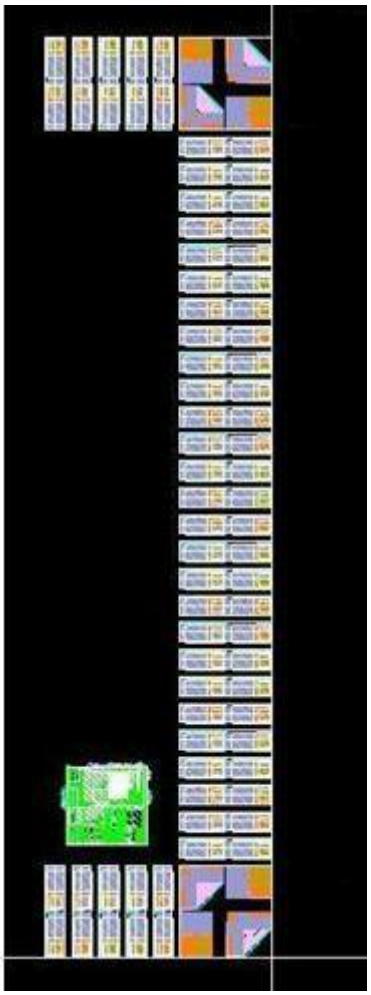


Figure 118: Preliminary bonding diagram for the Parallel I/O section

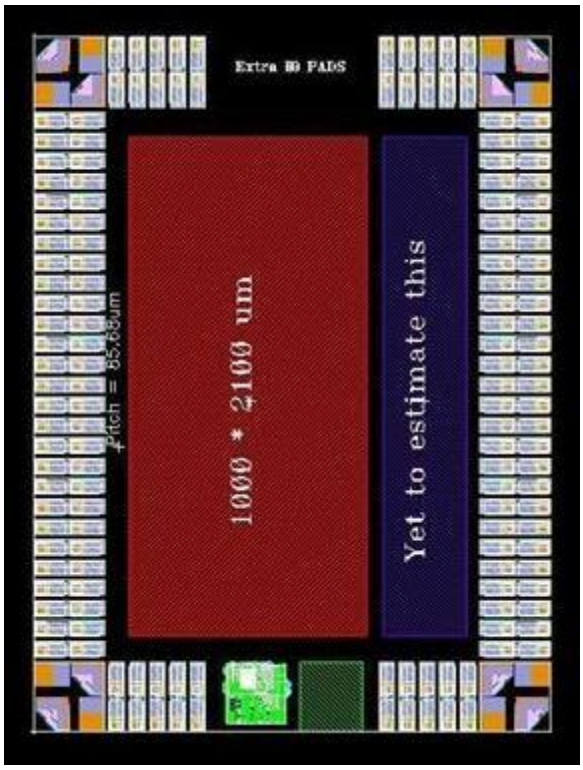


Figure 119: Preliminary bonding diagram for the Baseband Chip

5. In consultation with the Package development engineer, the bonding diagram was finalized. The full-chip layout with die photographs is show in Figure 120, Figure 121 and Figure 122.

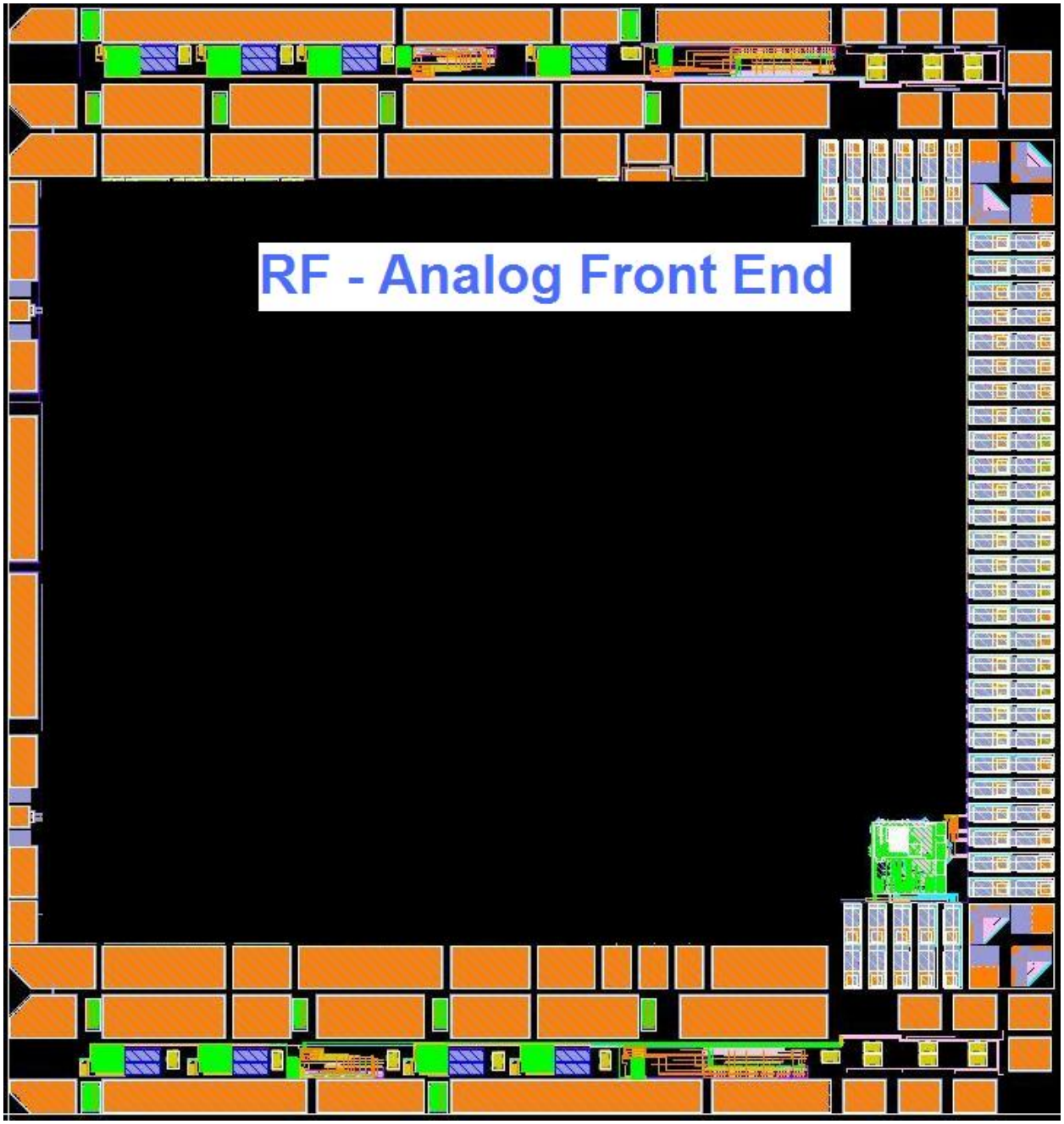


Figure 120: I/O ring of the Parallel Interface Section seen to the right of the CMOS RF Radio Chip

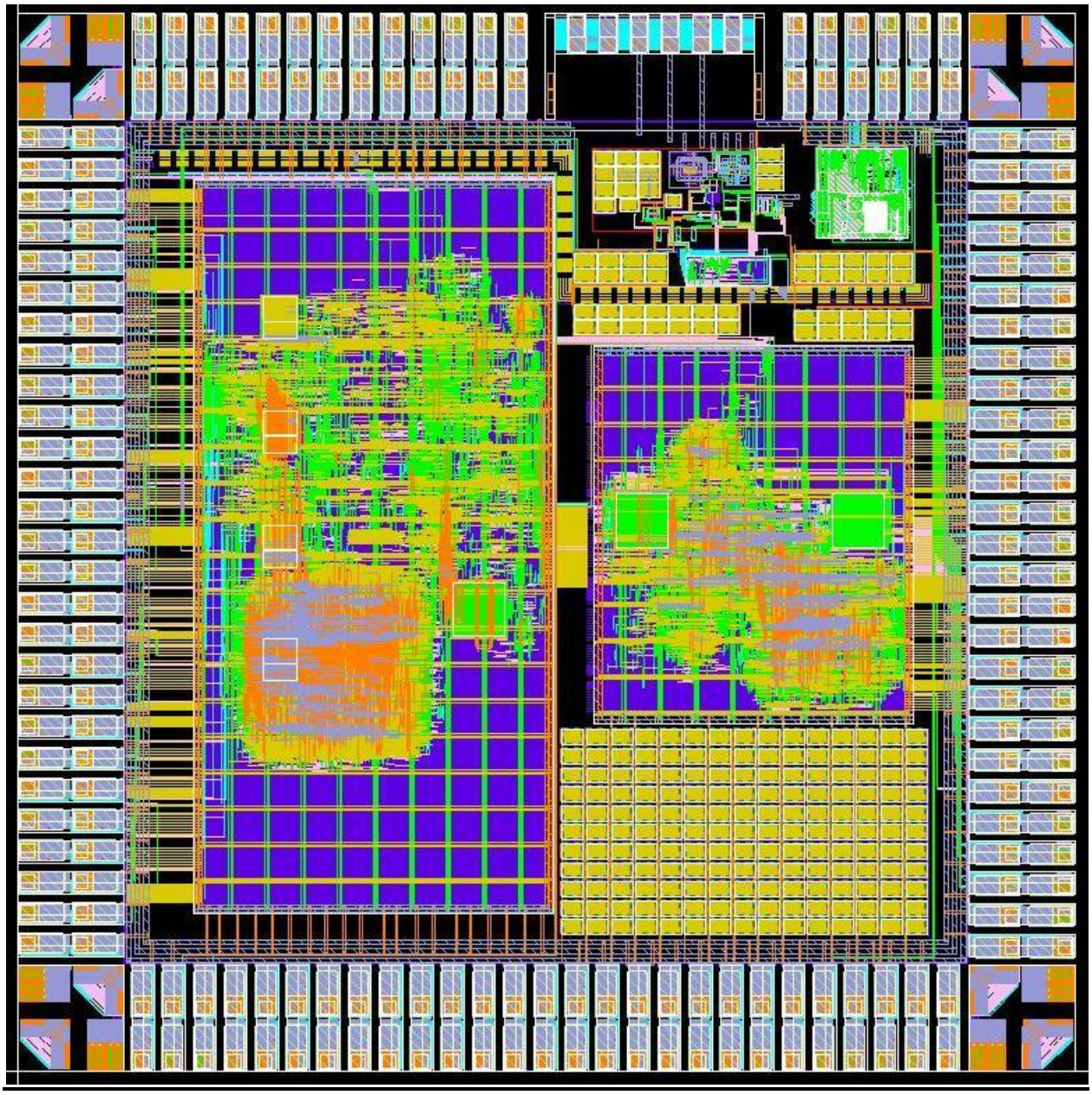


Figure 121: I/O Ring on the Baseband chip

Figure 122: Die photograph of the Parallel I/O

7 CONCLUSION

In this work, a complete digital communication block set for control and tuning of Digital CMOS Radio has been presented. A novel communication protocol was developed, which combined the best of SPI and I2C protocols. The implementation was done using the ASIC development flow outlined in Chapter 2 and was tested to be functioning at clock speeds of over 750MHz. Several tuning, self-healing and calibration algorithms were designed for the Digital Radio single-chip solution, all using the same reference flow and Standard Cell Library from various vendors. A list of claims and innovations is presented next.

1. A design technique for an on-chip high-speed serial communication for a system-on-chip has been invented with special focus on lowering the power consumption.

2. The serial interface has been optimized for achieving extremely high data transfer speeds. It has been tested at 750MHz on the 90nm technology node. The design is scalable and is expected to achieve data rates in the gigahertz regime at 65nm and 45nm.

3. The architecture is scalable to accommodate as many slave devices as needed and is realizable in the vanilla CMOS process. Other serial protocols need special hardware for interface, like I2C needs open drain output drivers with data bus polling capability.

4. A unique built-in self-resetting technique has been implemented to eliminate the need for a separate ‘reset’ pin and save up on valuable real estate.

5. The ‘Common Mode’ based broadcast technique enables all the slaves to change states as needed thus allowing a significant speedup in the throughput. This is indispensible for timing sensitive state changes in which all the modules have to simultaneously reach a particular state with a predefined time period.

6. The individual slaves are configured by writing the desired data pattern into them. A special ‘Short Write’ mode allows the master device to very quickly change the settings

on the slave devices. This is essential in Gigabits per second RF transceiver based applications.

7. Most serial protocols spend significant time in handshaking. Our system eliminates these handshaking signals (like acknowledge signal) and leverages the overall RF system feedback loop to ensure correct data transmission.

8. A new module can be connected to the address based serial communication system by simply connecting to the SPI bus. This “plug-in” feature eases the interfacing burden on individual RF block designers.

9. Individual SS lines avoided; saves up a lot of silicon real-estate by avoiding extra routing and pads.

10. Since most SoC control applications will be with a single master, all overhead associated with multi-master contention resolution is eliminated.

The serial communication interface, high speed digital I/Os, Baseband chip along with the RF Analog Front end, form the core of the digital CMOS transceiver. This will pave the way for a next-generation connectivity solution which promises multi-gigabit transfer rate for real-time video streaming at low cost and in a small form-factor package.

REFERENCES

[1] http://wireless.fcc.gov/outreach/2004broadbandforum/comments/YDI_benefits60GHz.pdf (Accessed April 1st, 2010)

[2] Pinel S., Iyer G., et. al, "60GHz Single-Chip CMOS Digital Radios and Phased Array Solutions for Gaming and Connectivity," *Journal on Selected Areas in Communications,* 2008

[3] http://www.ecma-international.org/publications/standards/Ecma-369.htm (Accessed April 1st, 2010)

[4] Laskar, J. Pinel, S. Dawn, D. Sarkar, S. Sen, P. Perumana, B. Yeh, D. Barale, F., "60GHz entertainment connectivity solution," ICUWB, *IEEE International Conference on Ultra-Wideband*, 2009, pp 17-21, Sept 2009

[5] http://en.wikipedia.org/wiki/Physical_Layer (Accessed April 1st, 2010)

[6] http://en.wikipedia.org/wiki/Media_Access_Control (Accessed April 1st, 2010)

[7] Zhu Dongmei ; Fu Dongbing ; Shi Jiangang ; Li Kaicheng ; Nat. Labs. of Analog Integrated Circuits, Sichuan Inst. of Solid-state Circuits, Chongqing, China, "Digital static calibration technology used for 400MSPS, 16-bit DAC," *IEEE International Conference of Electron Devices and Solid-State Circuits*, pp 91-94, Jan 2009

[8] Ping-Ying Wang ; Zhan, J.-H.C. ; Hsiang-Hui Chang ; Chang, H.-M.S. ; RF Div., MediaTek Inc., Hsinchu, Taiwan, "A Digital Intensive Fractional-N PLL and All-Digital Self-Calibration Schemes", *IEEE Journal of Solid-State circuits,* Vol. 44, pp 2182 – 2192, August 2009

[9] Yuhua Cheng ; Conexant Syst., Newport Beach, CA, "The influence and modeling of process variation and device mismatch for analog/rf circuit design," *Proceedings of the Fourth IEEE International Caracas Conference on Devices, Circuits and Systems*, pp D046-1 - D046-8, August 2002

[10] www.national.com/appinfo/lvds/files/lvds_ch1.pdf (Accessed April 1st, 2010)

[11] www.jedec.org/download/search/jesd8-13.pd (Accessed April 1st, 2010)

[12] J. Laskar, S. Pinel, D. Dawn, S. Sarkar, B. Perumana and P. Sen, "The Next Wireless Wave is a Millimeter Wave", *Microwave Journal*, pp22-36, August 2007

[13] Jan Sevenhans, Frank Op't Eynde, and Peter Reussens "The Silicon Radio Decade," *IEEE Transactions on Microwave Theory and Techniques.* Vol. 50, No. 1, January 2002

[14] Asad A. Abidi, "RF CMOS Comes of Age," *IEEE Journal of Solid-State Circuits.*Vol. 39, No. 4, April 2004

[15] Peter Smulders, "Exploiting the 60 GHz Band for Local Wireless Multimedia Access: Prospects and Future Directions," *IEEE Communications Magazine*, January 2002

[16] http://www.freescale.com/files/microcontrollers/doc/ref_manual/S12SPIV3.pdf (Accessed April 1st, 2010)

[17] http://en.wikipedia.org/wiki/OSI_model (Accessed April 1st, 2010)

[18] Dickson, D. ; Jett, P., Time Domain Corp., Huntsville, AL, "An application specific integrated circuit implementation of a multiple correlator for UWB radio applications," *IEEE Military Communications Conference Proceedings*, Vol. 2, pp 1207-1210, August 2002

[19] http://www.tsmc.com/english/c_services/c01_design/c0105_reference.htm (Accessed April 1st, 2010)

[20] IEEE Standard for Verilog Hardware Description Language, 1364-2005

[21] http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Documents/dc_graphical_ds.pdf (Accessed April 1st, 2010)

[22] http://www.cadence.com/rl/Resources/datasheets/encounter_rtlcompiler.pdf (Accessed April 1st, 2010)

[23] http://www.synopsys.com/Tools/Implementation/SignOff/Documents/primetime_suite_ds.pdf (Accessed April 1st, 2010)

[24] http://portal.model.com/modelsim/resources/references/modelsim_pe_ref.pdf (Accessed April 1st, 2010)

[25] Ashenden, P.J., VHDL standards, Design & Test of Computers, *IEEE*, pp 122 – 123, Oct 2001

[26] http://www.synopsys.com/community/interoperability/pages/tapinsdc.aspx (Accessed April 1st, 2010)

[27] http://www.vhdl.org/sdf/sdf_3.0.pdf (Accessed April 1st, 2010)

[28] http://www.alteraforum.com/forum/showthread.php?s=4dc2e1bb47eb37f77a10c9f7ddbf3891&t=1119 (Accessed April 1st, 2010)

[29] Bidermann, W. ; Jorgensen, J. ; Digital Equipment Corp., "Issues In High-speed Clocking," *IEEE International Solid-State Circuits Conference*, pp 84 – 85, Feb 1991

[30] http://www.cadence.com/products/di/edi_system/pages/default.aspx (Accessed April 1st, 2010)

[31] http://www.cadence.com/rl/Resources/datasheets/soc_encounter_ds.pdf (Accessed April 1st, 2010)

[32] Guirong Wu ; Song Jia ; Yuan Wang ; Ganggang Zhang ; Key Lab. of Microelectron. Devices & Circuits, Peking Univ., Beijing, China, "An efficient clock tree synthesis method in physical design," *IEEE International Conference of Electron Devices and Solid-State Circuits*, pp 190-193, Jan 2009

[33] Lide Zhang ; Bai, L.S. ; Dick, R.P. ; Li Shang ; Joseph, R. ; EECS Dept., Northwestern Univ., Evanston, IL, USA, "Process variation characterization of chip-level multiprocessors," *Design Automation Conference*, 2009, pp 694-697

[34] http://www.calypto.com/whitepapers.php#poweradviser (Accessed April 1st, 2010)

[35] http://standards.ieee.org/reading/ieee/std_public/description/testtech/1149.1-1990_desc.html (Accessed April 1st, 2010)

[36] www.nxp.com/acrobat_download2/literature/9398/39340011.pdf (Accessed April 1st, 2010)

[37] datasheets.maxim-ic.com/en/ds/DS4550.pdf (Accessed April 1st, 2010)

[38] http://www.cadence.com/rl/Resources/datasheets/virtuoso_ade_fam_ds.pdf (Accessed April 1st, 2010)

[39] http://www.byteparadigm.com/product-gp-22050-14.html (Accessed April 1st, 2010)

[40] http://www.cypress.com/ - CY3684 EZ-USB FX2LP Development Kit

[41] http://www.usb.org/ (Accessed April 1st, 2010)

[42] http://www.1394ta.org/ (Accessed April 1st, 2010)

[43] http://www.st.com/stonline/products/technologies/soc/90plat.htm (Accessed April 1st, 2010)

[44] http://www.tsmc.com/download/english/a05_literature/90nm_Brochure.pdf (Accessed April 1st, 2010)

[45] https://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/9BF3EC138A82223B8725708B004C6471/$file/ASIC%20Brochure_June0407_final.pdf (Accessed April 1st, 2010)

[46] http://www.samsung.com/global/business/semiconductor/products/asic/Products_90nmTechnology.html (Accessed April 1st, 2010)

[47] Duvvury, C. ; Rountree, R.N. ; Adams, O. ; Texas Instruments. Inc., Dallas, TX, "Internal chip ESD phenomena beyond the protection circuit," *IEEE Transactions on Electron Devices*, Vol. 35, Issue 12, pp 2133-2139, Dec 1988

[48] Broddeck, T. ; Bauch, H. ; Guggenmos, X. ; Wagner, R. ; SIEMENS AG, "Reproducibility of field failures by ESD models - comparison of HBM, socketed CDM and non-socketed CDM," *Proceedings of the 7th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis*, 1996, pp 1719-1722

[49] Duvvury, C. ; Rountree, R.N. ; Fong, Y. ; McPhee, R.A. ; Texas Instruments Inc, "ESD Phenomena and Protection Issues in CMOS Output Buffers," *25th Annual Reliability Physics Symposium*, 1987, pp 174-180

[50] http://www.jedec.org/sites/default/files/docs/22A115B_0.pdf (Accessed April 1st, 2010)

[51] Ming-Dou Ker ; Jeng-Jie Peng ; Hsin-Chin Jiang ; Nanoelectronics & Gigascale Syst. Lab., Nat. Chiao-Tung Univ., "Failure analysis of ESD damage in a high-voltage driver IC and the effective ESD protection solution [CMOS]," *Proceedings of the 9th International Symposium on the Physical and Failure Analysis of Integrated Circuits*, 2002, pp 84-89

[52] Dan Clein, "CMOS IC Layout: Concepts, Methodologies, and Tools" (*Newnes*)

[53] http://www.jedec.org/download/search/JESD8-13.pdf (Accessed April 1st, 2010)

[54] Steven H. Voldman, "ESD: Circuits and Devices," *John Wiley and Sons*, 2006

[55] Sanjay Dabral, Timothy Maloney, "Basic ESD and I/O Design," *John Wiley and Sons*,1998

[56] http://www.esda.org/dd_course1.html (*User Training at LSI Logic, India*) (Accessed April 1st, 2010)