Integration of a Digital Built-in Self-Test for On-Chip Memories

XIAO LUO
MASOUD NOURIPAYAM
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Integration of a Digital Built-in Self-Test for On-Chip Memories

Xiao Luo
soc15xlu@student.lu.se
Masoud Nouripayam
soc14mno@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Joachim Rodrigues

Examiner: Pietro Andreani

December 12, 2017

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

The ability of testing on-chip circuitry is extremely essential to ASIC implementations today. However, providing functional tests and verification for on-chip (embedded) memories always poses a huge number of challenges to the designer.

Therefore, a co-existing automated built-in self-test block with the Design Under Test (DUT) seems crucial to provide comprehensive, efficient and robust testing features. The target DUT of this thesis project is the state-of-the-arts Ultra Low Power (ULP) dual-port SRAMs designed in ASIC group of EIT department at Lund University.

This thesis starts from system RTL modeling and verification from an earlier project, and then goes through ASIC design phase in 28 nm FD-SOI technology from ST-Microelectronics. All scripts during the ASIC design phase are developed in TCL.

This design is implemented with multiple power domains (using CPF approach and introducing level-shifters at crossing-points between domains) and multiple clock sources in order to make it possible to perform various measurements with a high reliability on different flavours of a dual-port SRAM.

Acknowledgement

This work would not have been possible without the help and support from many around us.

First and foremost, we should express our sincere gratitude to our great supervisor, professor Joachim Rodrigues for all his support, trust and guidance through the whole journey of our master studies and in particular thesis project.

We are thankful to Babak Mohammadi, for his great and friendly support and guidance during this project. We are also thankful to Oskar Andersson for his tremendous support.

Another thanks to our friends and Ph.D students in ASIC group of EIT department who have supported us for this thesis.

Last but not least, we are thankful to our families for their unconditional love an support.

Xiao Luo and Masoud Nouripayam
Lund, December 2017

Table of Contents

List of Figures

List of Tables

Acronyms

AISC Application-Specific Integrated Circuit.

BIST Built-in Self Test.

CPF Common Power Format.

CS Chip Select.

DFT Design for Testability.

DR Data Registers.

DRC Design Rule Checking.

DUT Device Under Test.

EIT Faculty of Electrical and Information Technology.

ERC Electrical Rule Checks.

FD-SOI Fully Depleted Silicon On Insulator.

GDSII Geometric Data Stream.

HL High-Low.

IC Integrated Circuit.

IEEE Institute of Electrical and Electronics Engineers.

IP Intellectual Property.

IR Instruction Registers.

JTAG Joint Test Action Group.

LF Loop Filter.

LFSR Linear-Feedback Shift Register.

LH Low-High.

LVS Layout Versus Schematic.

MSV Multi-Supply Voltage.

PD Phase Detector.

PG Pattern Generator.

PnR Place and Route.

QoR Quality of Result.

RC Resistance-Capacitance.

RTL Register-Transfer Level.

SE Signature Evaluator.

SRAM Static Random Access Memory.

STM ST-Microelectronics.

TAP Test Access Port.

TCL Tool Command Language.

TT Typical-Typical Corner.

ULP Ultra Low Power.

VCO Voltage Controlled Oscillator.

WE Write Enable.

XOR exclusive OR Gate.

Chapter 1

Introduction

A big challenge in today's Application-Specific Integrated Circuit (AISC) designs is the efficient test and verification. This challenge usually is bigger when it comes to Static Random Access Memory (SRAM). Therefore, to address the daunting challenges of memory verification, it is required to develop a test circuitry which:

- performs automatic tests,
- supports different modes and configurations for a comprehensive verification,
- is not a bottleneck for devices under test

Reviewing the list above brings a Built-in Self Test (BIST) circuitry to the mind which can be integrated with Joint Test Action Group (JTAG) standard to compensate the necessity of more dedicated test pins at the physical I/O interface with a few pins.

1.1 Purpose and Scope

This thesis takes a step forward in developing a prototype of an automated internal self-test circuitry for performing reliable tests and measurements on SRAMs, in particular, the dual-port Ultra Low Power (ULP) SRAM designed in ASIC group of EIT department is the target of this project. Fulfilling the objectives of this thesis requires to go through the whole process of ASIC design, meaning that it ranges from Register-Transfer Level (RTL) modelling and verification, RTL synthesis, Place and Route (PnR) and finally Geometric Data Stream (GDSII) preparation and validation for chip fabrication. Moreover, during the project it is also taken into consideration to develop the ASIC flow scripts of the system design in such a way to make them reusable as a foundation for future works and make them independent of the technology. In this thesis, a RTL implementation has been developed, optimized and later verified (be able to test dual-port SRAM operations). In the ASIC implementation steps also, the prior scripts are used, then updated to new requirements of the project as well as been adapted to new design flow and new design tool.

1.2 Thesis Outline

The chapters following the introduction are about to briefly expose the important pois of this thesis journey. "RTL Design and Modeling" is basically the front-end and algorithmic development of the system. Then, the "ASIC Implementation" provides the reader with the important information and results regarding the back-end phase of system design.

1.3 Software Tools

To be able to implement this thesis from RTL development to ASIC implementation, the following tools have been used:

- **QuestaSim**, from Mentor Graphics, for RTL model development, simulation and verification
- **MATLAB** for implementing the functional model of the system and generation of test patterns(stimuli file).
- **Design Vision**, from Synopsys, for performing synthesis flow
- **Innovus Implementation System**, from Cadence, for performing the Place & Route flow
- **Virtuoso**, from Cadence, for importing the final GDSII (layout) and netlists (schematic) and make them ready for GDSII exported to chip fabrication
- **Calibre**, from Mentor Graphics, for performing drc and lvs on the layout view of the system
- **TCL** is the language used for scripting the whole flow

1.4 Used Technology and IP blocks

This design is implemented in *28 nm Fully Depleted Silicon On Insulator (FD-SOI)* technology from ST-Microelectronics(STM). Moreover, the Intellectual Property (IP) used in this project are listed below:

- 2048 words x 32 bits synchronous dual-port high performance SRAM from STM (verilog behavioral model is used)
- Charge Pump PLL with maximum VCO frequency 4000 MHz and Output frequency of $(31.25MHz - 2000MHz)$ from STM (verilog behavioral model is used)
- Dual-port SRAM IP block from EIT department, Lund University. Reference [12] has investigated the performance of differe configurations of this SRAM IP block.

It is worth mentioning that initially 8 SRAM macros from the STM were used to develop the ASIC flow. However, the flow is developed in a way that later by minor modifications, 6 STM SRAMs can be replaced by 3 difference flavors[1] (2 macros per flavor) of EIT SRAMs.

[1]SRAM with different bit-cells area and design rule check (standard and pushed)

Chapter 2

RTL Design and Modeling

This system is developed as a test prototype to show the performance of the SRAM. Therefore, a BIST automatic test block is developed and integrated into the system due to the cumbersome process of testing a memory. [3]

To interface the BIST and the memory to outside world, JTAG is used as a known standardized access port which is basically used with test logic owing to its low overhead and minimized physical interface[4]. Figure 2.1 shows the overall relation among different parts of the system. Besides, a brief explanation about each part and their consisting blocks will be reviewed in coming sections.

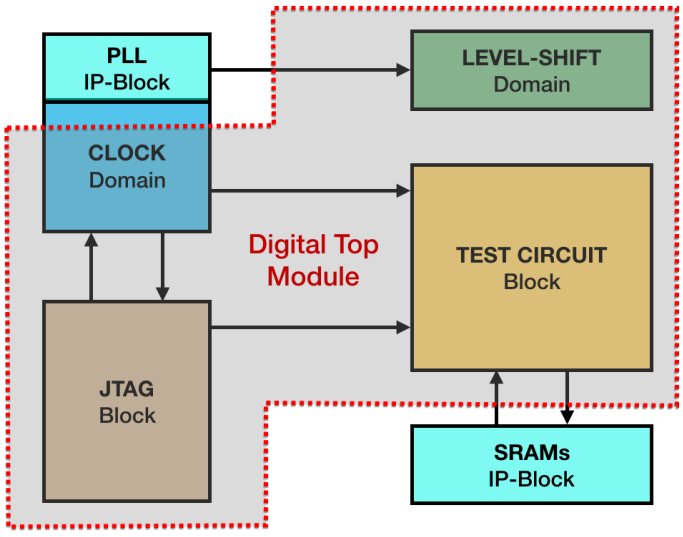


Figure 2.1: An overview of major blocks of the system

2.1 Test Circuit Block

Test circuit block is the main block that controls and issues the test signals to be propagated elsewhere in the system. The Test circuit block mainly consists of the BIST logic and other peripheral circuits that secure proper functionality of the BIST. *Test Controller* is the module which wraps all data (configuration) translators which in turn directs the system to select among different test modes (Scan Memory, Scan Internal Storage, BIST) and issuing control signals. Figure 2.2 shows the overall view of this block.

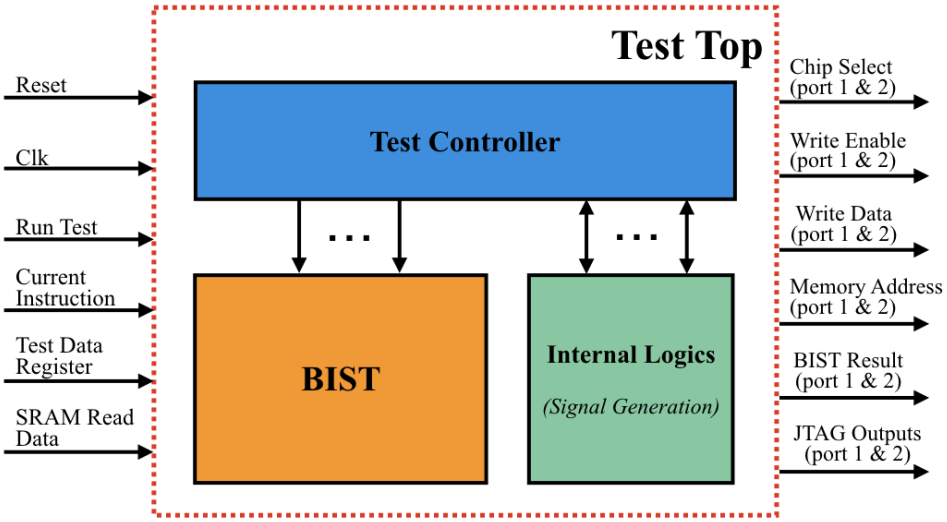


Figure 2.2: An overview of *Test* block

2.1.1 BIST Block

BIST is a widely used method of Design for Testability (DFT). BIST basically requires three kinds of sub-modules cooperating with the Device Under Test (DUT):

BIST Controller (simply a FSM which can be centralized or distributed in different modules) to activate *Pattern Generator (PG)* and control the *Signature Evaluator (SE)*.

Pattern Generator creates a sequence of stimuli to DUT; SE compacts the real-time DUT test response into a signature and compares this signature with the expected one. The output of this comparison indicates whether DUT behaves as expected or not.

BIST brings considerable benefit to the Test Circuit, such as a lower dependency on external test environment, which reduces the cost of test significantly and a better fault coverage can be achieved as the test circuitry is incorporated inside chips. Moreover, BIST grants DUT the capability of performing the self-test, hence, test process achieves a better efficiency and DUT can be tested at its own full speed rather than at the speed clamped by Test System.[6]

In this design, BIST has its controller (FSM) distributed among its internal modules, hence, a subtle communication between them supports the Test Circuit functionality.

Since it provides different testing features for both ports of a dual-port SRAM, there is a duplicate of each consisting module (i.e. Comparator_Port1 and Comparator_Port2). BIST internal modules and their respective relations are shown in Figure 2.3. The following subsections explain about how key modules of BIST work.

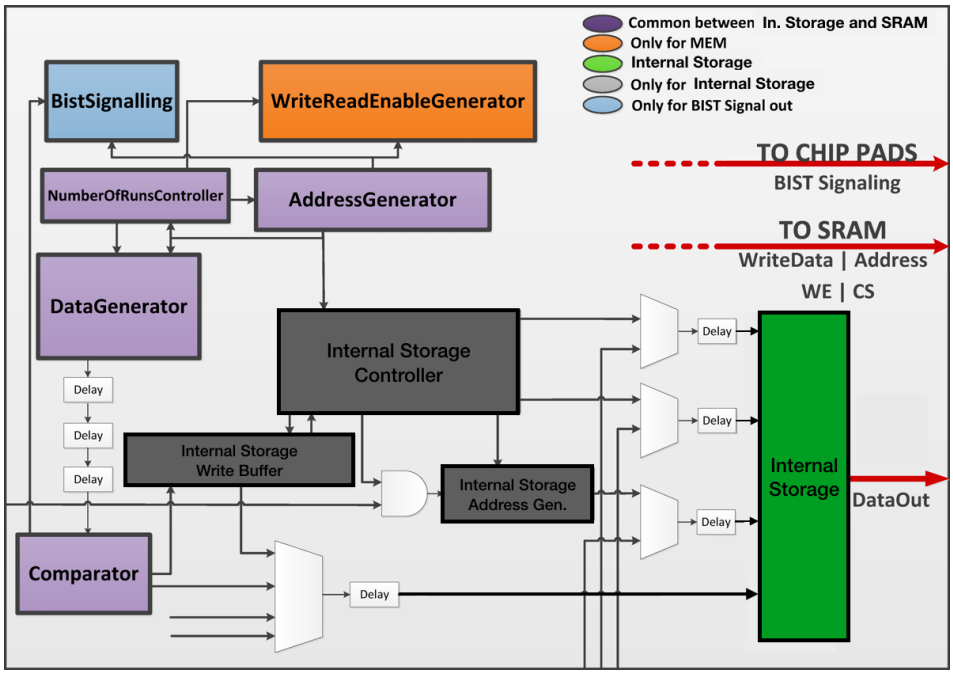


Figure 2.3: A detail view of BIST consisting modules and their interactions

Address Generator

This module provides the addresses (locations) of the SRAM to be written on or read from. In fact, *Address Generator* is the engine which generates addresses in a specified range with sequential (or reverse sequential), Odd(or Even), hop by X addresses and LFSR pseudo-random fashions. Address space length has a logarithmic relation with the number of existing words in the memory. Since each

memory bank in the design has $64Kb$ (2048 words x 32 bits/word), the address space length would become $log_2(2048) = 11bits$ per address.

For empowering the *Address Generator* engine to produce address in the specified range in one of the above-mentioned methods, the following configuration data are required to be received from JTAG:

- Read-Write mode
- Start address
- End address
- Increment (or decrement) value
- Number of consecutive Read-Write

The configuration data helps the engine to generate address of a specified range in addition to assist other modules to define their current state (read or write state) and prepare the system for the next state. Moreover, to produce random addresses in the specified range in the test, a divider is needed to calculate the modulus according to:

$$(Address\ Range) = (Biggest\ Address) - (Smallest\ Address) + 1$$

$$(Random\ Address) = (LFSR^1\ Random\ Address) mod (Address\ Range)$$

and due to extra delays introduced by divider, the random address generation is supposed to be an extra feature accompanied with different methods of balancing the delays.

Data Generator

This module, based on the initial configurations coming from JTAG, generates automatically the following data patterns to be written on the memory :

- Checkerboard (32b'10101010101...)
- Reverse checkerboard (32b'01010101010...)
- All '1'
- All '0'
- Sequential sweep in a range with initial increment (or decrement) value
- LFSR random data

In addition, this module requires two control signals ("address generation is finished" and "switch between Read-Write") from *Address Generator* to control its operations and enable/disable the generator engine.

[1] Linear-Feedback Shift Register (LFSR)

Read-Write Enable Generator

This module by receiving data from other blocks in the system hierarchy, generates and controls the enable signals being sent to SRAM, namely, *Chip Select (CS)* (activates the memory bank) and *Write Enable (WE)* (decides which operation (read or write) is to be done). The SRAM *CS* and *WE* are both active low, which implies that when *CS* is clamped to '0' the memory is active for any operations. During the period of *CS* being '0', if *WE* becomes '0' the memory is ready for writing data and if *WE* becomes '1' the memory is ready to be read from. Figure 2.4 shows how SRAM memory reacts to different values of *CS* and *WE*. Besides, Figure 2.5 presents the relationship between the memory bank to be selected for an operation and the coming *CS* from JTAG.

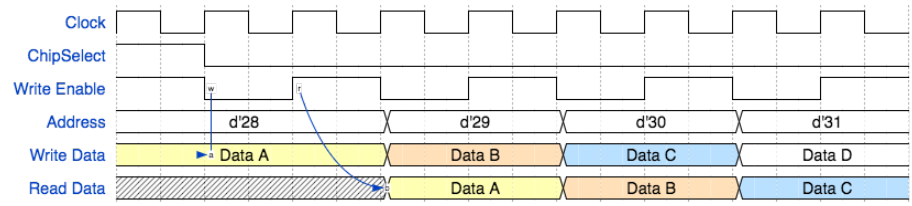


Figure 2.4: SRAM operations with CS and WE signals

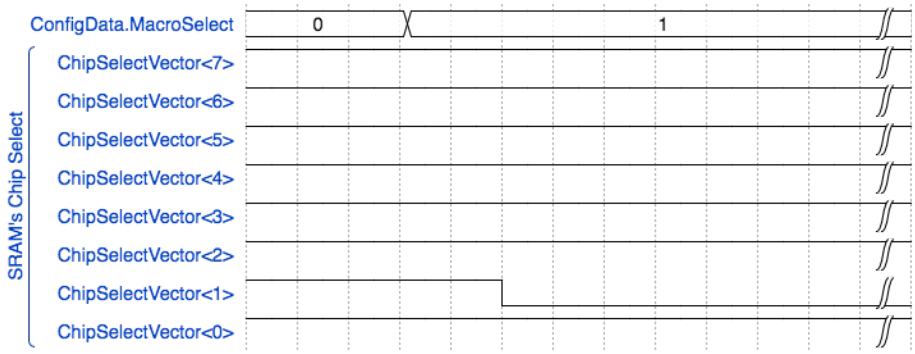


Figure 2.5: Selecting a specific SRAM bank with CS signal

Multiple-Run Controller

This module controls the number of times a specific test is to be repeated over and over (especially for evaluating the average power consumption) by receiving the configuration data from JTAG as well as *Address Generator*'s current state signals. By tracking an internal test-run counter, this controller notifies other modules in the BIST to wind up their operations whenever it enters the *Last Run*.

Comparator

This module performs the fault discovery process of SRAM by a comparison between the generated data from *Data Generator* (the expected data to be written on memory) and the read data from SRAM. If the comparison result shows that two inputs are identical, it passes a '1' out. Moreover, this module, using an exclusive OR Gate (XOR), defines which bits on the memory were failed either during write or read time. If a read bit from the memory is the same as its corresponding one in the data coming from *Data Generator*, the result of XOR is '0'. The *Pass-Fail* and *Failed Positions* output signals from this module are both consumed in the *Internal Storage Write Buffer* to track fault discovery process.

Internal Storage

Internal Storage is basically a block of Standard Cell Memory (SCM) placed in Test Circuit to verify SRAM operations by different store-modes defined by initial test configuration data.

Although having a bigger volume of *Internal Storage* gives the flexibility of verifying a bigger chunk of the SRAM, the decision about the proper size of this module is crucial in order not to put any extra load on the system as well as the complexity in ASIC implementation phases. The size of this module in the current implementation is defined experimentally to be 2 Kb or 64 Words.

Internal Storage Address Generator

This engine generates addresses for *Internal Storage* by receiving a pack of configuration data from JTAG and higher modules in system hierarchy in the address space of 6-bit length (the *Internal Storage* can store at maximum 64 Words and address space length is $log_2(64) = 6$).

Internal Storage Write Buffer

This module in BIST plays a big role in keeping track of the correct functionality of the memory. The configuration data from JTAG contains information about how to treat the read data from SRAM. In fact, the *Internal Storage Write Buffer* is a buffer with 32-bit length of which each bit corresponds to one full word on SRAM. For instance, if the write (or read) operation is done correctly, (*Pass-Fail* equaling to '1' is placed as the bit in its corresponding position in this module.

Internal Storage Controller

This module controls when, which type of data should be written on the *Internal Storage*. It means that the controller manages when an enable signal should be issued to store a data on *Internal Storage* based on initial configuration data. The configuration can be either of the following keywords:

- Correct Bits: set the flag that the *Failed Positions* coming from comparator is to be written on *Internal Storage*

- Correct Words: set the flag that a 32-bit full vector from *Internal Storage Write Buffer* (defines which words are passed/failed) is to be written on *Internal Storage*.

- Actual Data: set the flag that the actual data read from the SRAM is to be written on *Internal Storage*

2.2 JTAG Implementation

IEEE 1149.1[1] is a standard that gives designer the possibility of performing extensive debugging and diagnostic test on Integrated Circuit (IC) through a small number of dedicated test pins.[2]. The essence of JTAG is the boundary-scan, meaning that the stimuli to JTAG is serially scanned into and then out of the design. The circuitry of JTAG basically consists of Test Data Registers (DR), Instruction Registers (IR), Test Access Port (TAP) and TAP controller. The block diagram is presented in Figure 2.6[5].

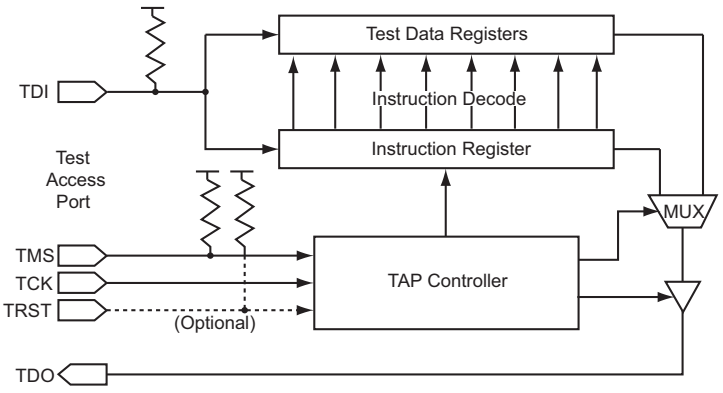


Figure 2.6: The block diagram of JTAG

This section briefly describes the implementation of this module.

Following the standard, this module operates by using a 16-state FSM and its *TCK* is clocked by external clock source. The module loads the necessary data from *TDI* pin to define which instruction is about to be processed and then creates the *Current Instruction* vector for other modules use. Directly after capturing the instruction, *TDI* pin feeds the test data serially to JTAG which then is parallelized to create the *Test Data vector* used by other modules.

Current Instruction is kept into a 5-bit register -which is referred to by all internal modules of the Test Circuit through the whole period of each test- until a new test configuration arrives.

[2]TCK, TMS, TDI, TDO are the four mandatory pins standardized in IEEE 1149.1, more pins may be added according to interest.

Run Test is a single-bit signal which tells other modules whether a test is ready to start/continue. Following the TAP controller, *Run Test* becomes '1' after *Update_IR*, *Update_DR* or *TestLogicReset* state.

Parallel Output contains all necessary information for a test configuration required by different blocks of the system. Similar to *Current Instruction*, after being fully captured, it is kept in a register and propagated through the whole system for defining the test conditions reference point. The JTAG standard gives designers the flexibility to define the private instructions for required system processes. Table 2.1 lists all used private instructions in this system. *TDO*, the "serial output

Table 2.1: JTAG private instructions

Instruction	Binary	Description
INSTR_CAPTURE	11010	Capture new instruction
SCAN_MEMORY	10000	Scan mode for SRAM
SCAN_IN_STORAGE	01000	Scan mode for internal storage
PERFORM_BIST	00011	Process BIST mode for SRAM
CONFIG_BIST	00111	Config BIST mode
CONFIG_PLL	11000	Config PLL mode
SEL_CLOCK_SOURCE	01100	Enable a clock source
CLOCK_CONFIG	00110	Select clock configuration
SRAM_CONFIG_BITS	01010	Config SRAM (from EIT)

for test instruction and data for the test logic"[5],provides simpler troubleshooting and control process by verifying the input configuration data at the end of each test mode, in addition to off-chip access to *Internal Storage* for SRAM verification.

Parallel Input returning to JTAG from Test block and feeding TDO pin can take the following values:

- Clock control information comes as an output from the *Clock Core* domain when the current instruction is CLOCK_CONFIG or CONFIG_PLL
- Configuration data for dual-port SRAM IP from EIT department when the current instruction is SRAM_CONFIG_BITS
- Test configuration data from Test block when *Current Instruction* is none of the above.

2.3 Clock Core Domain

It is the key module in clocking the whole system which integrates multiple clock sources in the design and decides which of them should be used for a specific test instance. Figure 2.7 shows the overall view of this block that contains:

- External clock source (coming externally from a chip pad)

- PLL IP block (from ST-Microelectronics)
- Multiple divisions of ring oscillator (RingOsc, RingOsc/2, RingOsc/4, RingOsc/8) shown in Figure 2.8a and 2.8b

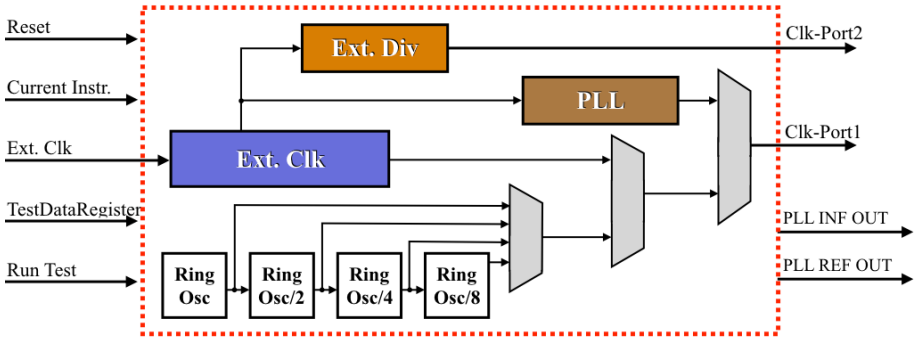


Figure 2.7: An overview of *Clock Core* domain

PLL is a widely used high frequency clock source due to its ability of controlling the frequency in a highly accurate way with less phase variation. It engages its own digital and analog modules and consists of Voltage Controlled Oscillator (VCO), Phase Detector (PD), Loop Filter (LF) and one or more frequency dividers. PLL can be used in two different modes:

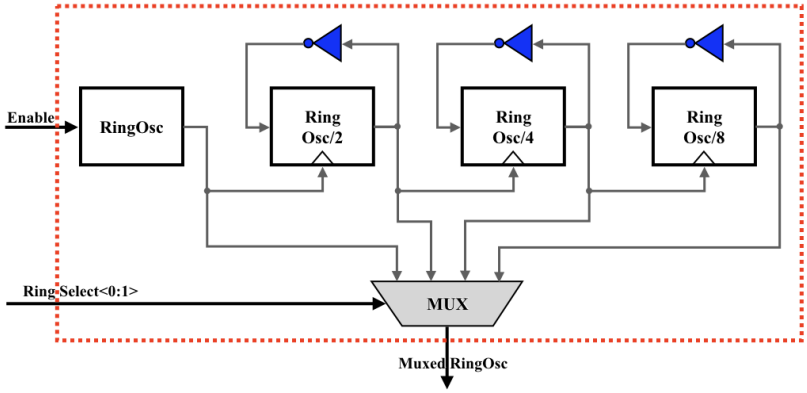
- *Fine Locked*: PLL doesn't send out its output clock until the desired frequency is achieved.
- *Coarse Locked*: once PLL frequency enters a certain range of desired frequency, it sends the output clock, while keeps refining the output frequency.

This system is planned to work mainly with PLL as the most accurate on-chip frequency generator. However, in case any failure occurs in PLL operations, *Ring Oscillator* along with multiple frequency dividers are in-placed as a backup source. Additionally, the external clock sources play an important role in debugging and accessing the internal circuitry from outside.

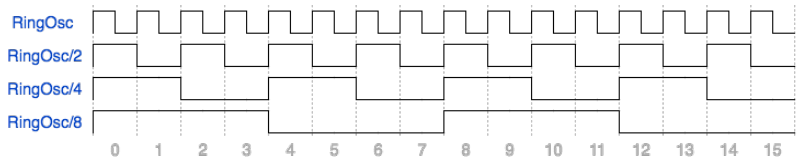
2.4 Voltage Level-Shifting and Buffering

This design is about to test multiple SRAM configurations. Since the SRAMs are supposed to be operational in near sub-threshold region, while the output signals are truly prone to noise that in turn may cause a flip in their information, to overcome issues which are raised because of weak signals from/to SRAM domains to/from *Digital Core* domain, level-shifter cells are placed at crossing points of domains with different power supplies.

Moreover, a domain is defined to place strong buffers so as to increase the drive strength of signals to output pads. This domain is driven by the maximum



(a) The diagram of the Ring Oscillators



(b) Waveforms corresponding to RingOsc and dividers

Figure 2.8: A detailed view of ring oscillators

available supply voltage in the utilized technology. In addition, Innovus (PNR tool) will later map them into the proper level-shifter cells with the configuration to maintain the balance for signal rise/fall time. It should be noted that in a general perspective, level-shifters are buffers even though, they are specialized to leveling up/down the input voltage with some special considerations.

2.5 Test Modes Implementation

The RTL simulation and verification is an important stage in system design. Thus, the test vector (stimuli) generation based on MATLAB functional model is a great step towards the system verification especially due to the convenience it brings to system modeling.

This testing system includes two important test modes, SCAN mode and BIST mode. In MATLAB model, several functions have been developed for creating a test vector for each mode. The coming parts describe how a vector for each mode is created.

Test modes commonalities are global parameters used in common in MATLAB functions and basically emulates a special behaviour of the system. As it was mentioned before, all JTAG instructions have a special binary code, and in MATLAB the binary equivalence of each is written into test vectors.

BIST is the mode to perform automatic SRAM operations test. The architecture of BIST in this design facilitates both single-port (read and/or write using one port at each time) and dual-port operations (read and/or write simultaneously using both ports) for testing SRAMs. The test vector in this mode is 122 bits and its information is described in Table 2.2:

Table 2.2: Description of 122-bit BIST Test Vector

Sub-sections in Test Vector	DESCRIPTION
Data Generation mode	3 bits for pre-defined modes
Read-Write mode	2 bits for pre-defined modes
Internal Storage Write mode	2 bits for pre-defined modes
Word Length	32 bits wordline for each port
Data Increment	4 bits
SRAM Address Length	11 bits for each port ($log2(2048)$)
SRAM Address Increment	4 bits
BIST Read-Write Count	4 bits for number of consecutive reads/writes
SRAM Port Selector	1 bit for selecting SRAM port
Internal Storage Address	6 bits ($log2(64)$)
Internal Storage Address Increment	4 bits
Number of Repetitive Runs	3 bits for controling multiple-run
SRAM Bank Selector	3 bits for selecting an SRAM bank out of 8 available ones

Scan mode is also the test mode for forcing the memory or *Internal Storage* to write a specific data and read that in order to check their functionalities. If the system is supposed to perform a SCAN test, the instruction coming from JTAG test vector activates a selector inside the *TestController* in order to disable the BIST mode and to select SCAN test on:

- *Internal Storage*, if the instruction is `SCAN_IN_STORAGE`
- SRAM, if the instruction is `SCAN_MEMORY`

To make it clear how test vectors are generated in MATLAB, an example of configuring BIST's internal modules for testing SRAM with single-port mode is described in following steps:

1. Pre-allocating a long vector to place all necessary bits in

2. Invoking *applyJTAGreset* function along with proper signalling of *TMS* in order to reset JTAG to *Run_ Test_ IDLE* state
3. Invoking *SetBistConfigMode* function to create an instruction vector ("00111") in order to perform the BIST configuration (`CONFIG_BIST` instruction)
4. Defining the variables which are the configuration data for internal BIST modules
5. Invoking *PerformBistConfiguration* function which places all data into the test vector
6. Invoking *SetBistMode* function to create an instruction vector ("00011") in order to perform the BIST test (`PERFORM_BIST` instruction)
7. Exporting vectors into the stimuli file to be used in Modelsim test-bench

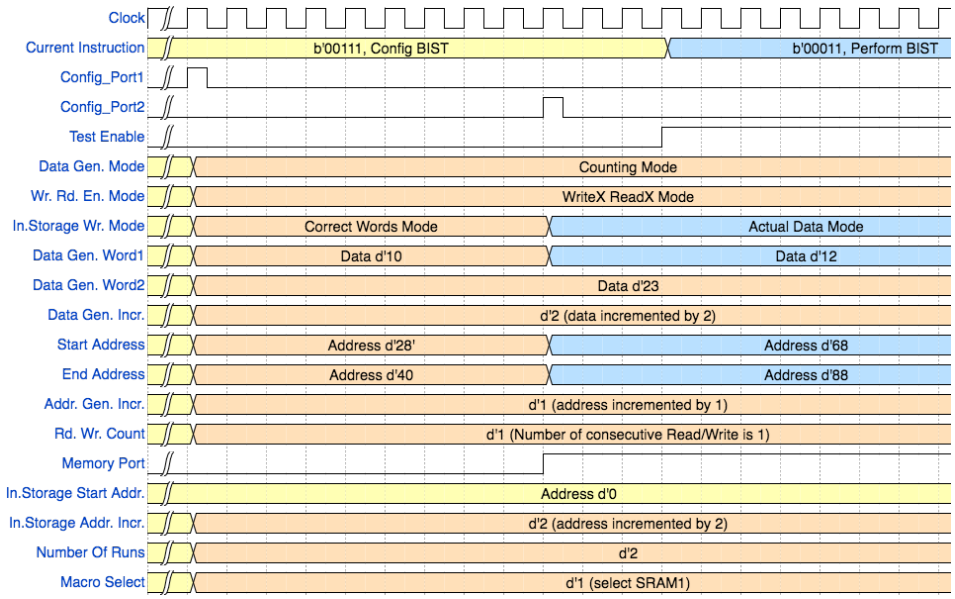
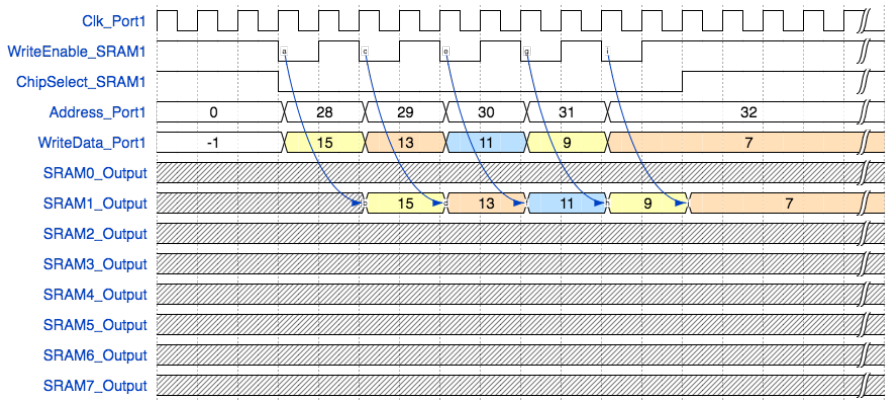
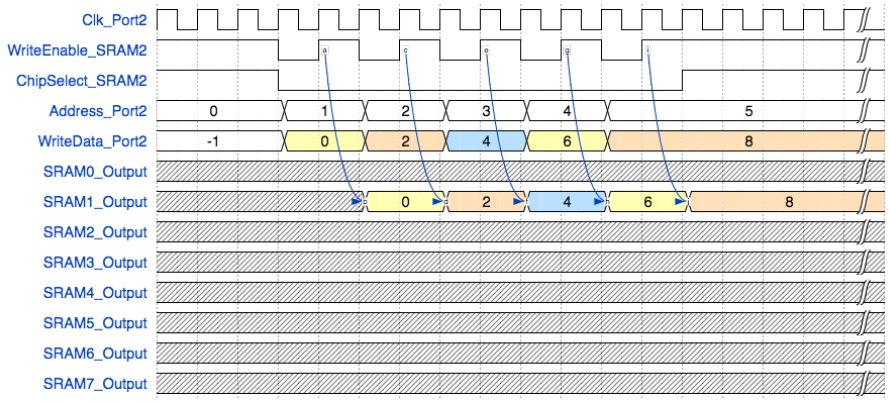


Figure 2.9: Dual-port SRAM configuration for simultaneous test run

To create a test vector for testing two ports of SRAM simultaneously, the step 4&5 are done twice (once for port1 and once for port2). Then the step 6&7 should be processed. Figure 2.9 shows the wave forms related to configuring both SRAM ports and running the BIST test on both simultaneously. Moreover, Figure 2.10 presents the result of the cooperation among all BIST modules to perform writing and reading operations on both ports of one memory bank by generating/updating data, address and control signals automatically.



(a) Operations at port1



(b) Operations at port2

Figure 2.10: Dual-port BIST operations, port1 and port2 of one SRAM bank

Moreover, to select among different existing clock sources in the system the following instructions should be invoked:

- `SEL_CLOCK_SOURCE` instruction to select among different clock sources
- `CLOCK_CONFIG` instruction to select among different modes of each source type (for instance the one among 4 available Ring Oscillator divisions)

Chapter 3
ASIC Implementation

3.1 Synthesis Flow

Synthesis is an important step towards delivering GDSII format for ASIC chip fabrication. In fact, "synthesis flow is an automated process which through defining different design constraints[1], an optimal way is sought for RTL description optimization and mapping to logic gates."[7] The following explains how the synthesis flow for this design is done and present the results.
This design is synthesized with Design Vision from Synopsys. Since the target technology is 28 nm FD-SOI from STM, all the netlists, timing constraint files (sdc, sdf) are generated by using this technology's libraries with different supply voltages in addition to using different height for standard cells in this technology. The reason to use standard cells of different height is that the level-shifter cells to be placed in *Output Buffer* domain are only available in *SC_ 8* libraries and for the rest of modules and domains the *SC_ 12* can be used.

As mentioned before, this design is a multi-supply voltage, multi-clock system which targets to test on-chip memory performance. Additionally, this design combines different digital and analog parts together. Therefore, to explore the maximum achievable frequency for this design, synthesis is performed once for the full design and once for the digital circuitry as a macro-block basically for the two reasons:

- Digital part is responsible for delivering signals and clocks to SRAM memory and therefore, it is crucial to see if digital part is truly capable of doing so.
- Since SRAM banks as IPs have their own timing and frequency constraints, the target is to provide the required frequency to memory and test its functionality.

Refering to Figure 2.1, JTAG and Test blocks are combined to form the *Digital Core* to be sent through the flow.

Furthermore, Figure 3.1 derived from the result of *reference_ report* of the synthesis tool, shows the area distribution over different modules in BIST, which certainly gives a better picture of Digital Core to readers.

[1] such as area, clock period, leakage power or power budget, etc.

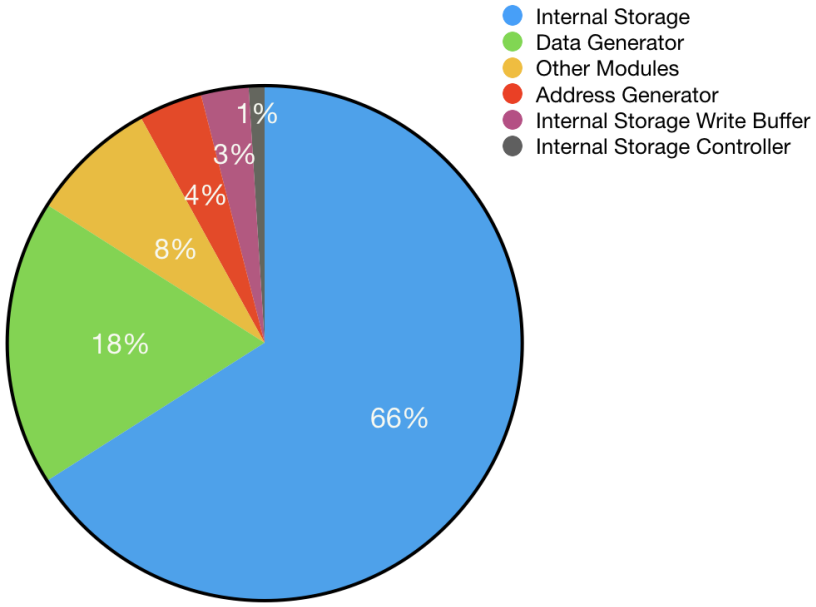


Figure 3.1: Area distribution among different constituents of BIST

3.1.1 Frequency Exploration

The switching speed (of signal delivery) of gates and transistors is higher at a higher supply voltage. Therefore, this design is tested with lowest available voltage in utilized technology in order to get a better view on how fast *Digital Core* of the system can run. Moreover, when defining system delays, instead of using "*set_clock_latency*" command to create *ideal clocking* (means clock networks have a specified latency or zero latency by default), "*set_propagated_clock*" command is used in which delays are propagated through the clock network to determine latency at register clock pins, including the delay resulting from parasitic capacitance and resistance.[8] Finally, it is worth mentioning that to find the optimum clock period (frequency), the synthesis process has been done in a loop over the range. The range itself is defined by testing a few values to get a sense of required timing in order not to violate any timing path.

The loop tests over a range of possible clock periods to give the optimum one and discovers that the maximum achievable frequency at $800mV$ as supply voltage with no body-biasing is:

- $2.32GHz$ (clock period is $430ps$) in Typical-Typical Corner (TT).

However, since the mentioned frequency in each process corner is exactly on the verge of meeting and violating the timing, this frequency is not safe to use since a slight deviation in the signal flow through the design may end up in a timing violation.

Referring to *Quality of Result (QoR)* report from synthesis tool, the critical path length[2] under the same voltage is:

- $410ps$ in Typical-Typical Corner at $2.32GHz$

3.1.2 Timing Constraints in the Full Design

There are usually two approaches in order to impose timing constraints to a design:

- The timing constraints reflected in *SDC* generation in synthesis flow
- The timing constraints in PNR flow and in clock tree synthesis step

However, since this design uses multiple clock sources, having the timing constraints in the synthesis process helps the whole process of falling in an improper path timing. Referring to Figure2.7, it is shown that the final clock path of the whole design is the output of a 3-multiplexer chain which according to the JTAG instruction, one of all different clock configurations is propagated through the whole design. Therefore, it is necessary to restrict the clock paths to get influenced by each other. Considering two following points, it gives the conclusion that output of a clock multiplexer creates separate clock skew groups with its respective incoming clocks:

- It is necessary to put different clock paths in mutually exclusive groups to limit their mutual influence almost to zero
- Output of each multiplexer behaves like a generated clock with respect to its different incoming clocks

Figure 3.2 shows the above concept. It should be noted that usually the tool can recognize the master clocks and probably some simple clock generation. However, for a more complex clock configuration, designers should define the constraints for the tool. Moreover, since the place and route tool uses the generated sdc timing constraint file, the clock groups and definitions are reflected all along to the end of routing process.

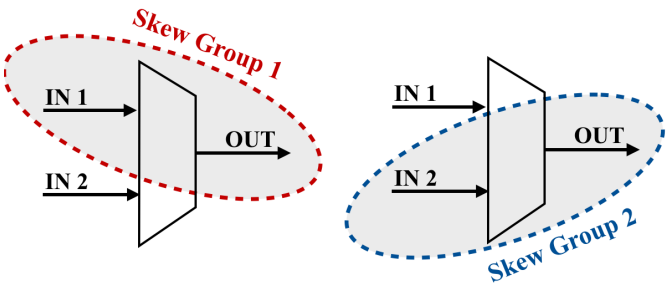


Figure 3.2: Separate clock skew groups in a clock multiplexer

[2]Critical path length is the total delay of the critical path, without considering clock network delay, input delay, or output delay. In this report generated by *report_timing*, it is the cumulative delay from the beginning to the end of the data path.[9]

The following lines of code show how the mutually exclusive groups should be implemented in the tool using Tool Command Language (TCL):

create_ generated_ clock -name mux_ clk1 -combinational mux/IN1 -soure mux/IN1
create_ generated_ clock -name mux_ clk2 -combinational mux/IN2 -soure mux/IN2
set_ clock_ groups -physically_ exclusive -group mux_ clk1 -groupmux_ clk2

3.2 Place and Route

Place and Route (PnR) is the step to perform physical implementation and routing of the created netlists from synthesis step in ASIC implementation flow. In fact the complex back-end flow can generally be divided into five main stages[10]:

- Floor Plan: the following steps in an iterative order are processed:
  - Chip dimension and aspect ratio.
  - Quantity and order of pads at each side.
  - Controlling the blocks, on-chip memories, data-path using partitioning
  - Placement of IP-blocks or macros
  - Global nets planning to reduce the effect of IR drop and clock skew.
  - Pin planning to minimize the unnecessary interconnections among blocks.
- Timing-driven placement: Place standard cells, macros, IP-blocks into specified places; Optimize the area occupied by signal routing; Make the placement meet timing constraints
- Connecting pins of placed standard cells and macros by running global and detailed routing. The global routing refers to partitioning the placed components into sub-regions which are assigned by topological paths of nets. The detailed routing routes the net inside each sub-region.
- Layout Completion:
  - Attach N-well polygons to reduce DRC violations
  - Insert decoupling capacitors for noise reduction
  - Fill floating metals for evenly distributed metals
  - Modify long wires or adding diodes to eliminate antenna violations
- Layout Verification: Design Rule Checking (DRC); Layout Versus Schematic (LVS); Electrical Rule Checks (ERC); Resistance-Capacitance (RC) extraction; Post-PnR simulation.

The coming sections summarize the important points and figures of PnR steps of this built-in self test system.

3.2.1 Chip Pads Planning

Figure 3.3 shows the pad placement in the design and the following list gives a better perspective of different types of pads used in this design.

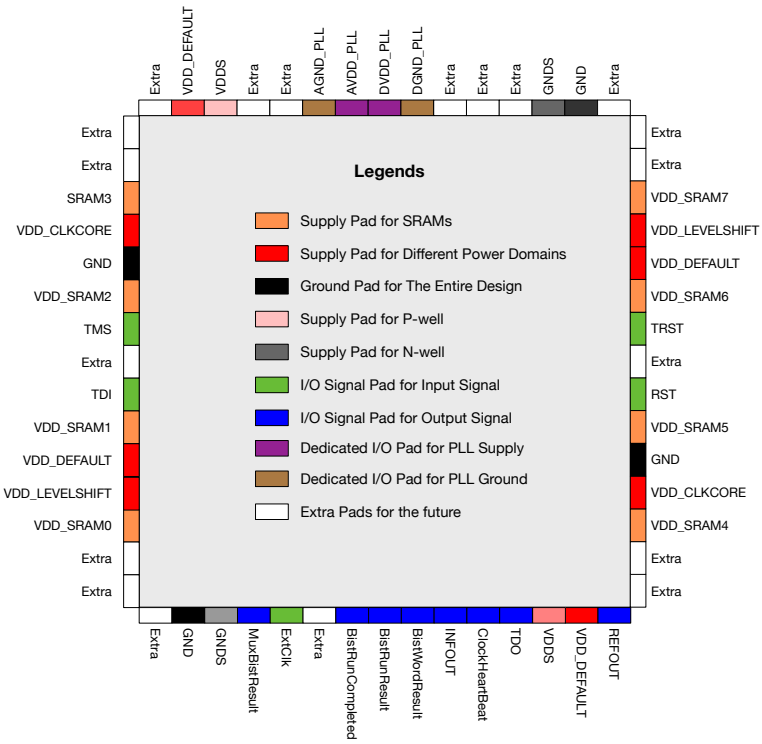


Figure 3.3: Diagram of pads planning

- **Memory supplies**: As displayed in Figure 3.3, each memory bank is assigned by an individual power supply (marked as *orange*) in order to operate different memories with different voltages simultaneously. And consequently, the power measurement to each memory will not be interfered by other memories.
- **Macro supplies**: Each of (*Clock Core* domain, *Output Buffer* domain and *Digital Core* domain[3]) is powered by a pair of supply pads (marked as *red*). In addition, two supply pads in each pair are placed face to face in the I/O ring for an even power distribution.
- **Body bias supplies**: Two pads (marked as *pink*) serve the P-well and two pads (marked as *gray*) serve the N-well. They are placed at opposite edges of the chip.
- **Ground**: Four ground pads (marked as *black*) are located at each edge of the chip to evenly distribute the ground connections.

- **Extra pads**: The extra pads (marked as *white*) are inserted in the I/O ring as the reserved place for feature expansions in the near future.
- **PLL supplies**: Two dedicated power supply pads (marked as *purple*) are assigned to the PLL, one is for digital sub-modules and one is for analog.
- **PLL ground**: Another two dedicated ground pads (marked as *brown*) are connected to PLL. Similar to PLL power supplies, two ground pads are given to digital respective analog sub-modules. Furthermore, the type of these 2 pads is supposed to be *Signal I/O pad* as they should not be shared with the I/O ring according to PLL's integration guideline.
- **Input/Output Signal pads**: The input signal pads of the system (marked as *green*) feed the stimuli to the chip. And the output signal pads (marked as *blue*) deliver the output signal towards the outside world.

3.2.2 Multi-Power Domain Approach using CPF

In the chip-engineering, the strategy of efficient power management is certainly an important part during the design. This strategy particularly comes to perspective when a multi-power domain approach is needed during implementation based on specifications. The Multi-Supply Voltage (MSV) technique (meaning that different blocks operate at different voltages to reduce power consumption) requires level shifters on signals that go from one voltage level to another. Without level shifters, signals that cross voltage levels will not be sampled correctly.[11]
In addition to above introduction, this design needs a MSV approach since it is targeting testing SRAM memories with different configurations. Thus, such a MSV approach can play a big role in finding different parameters of each SRAM configuration during measurement. Implementing a low-power design requires a proper declaration of power architecture and this is the point the CPF comes into the design in order to specify power-saving techniques. It should be noted that apart from all kinds of different advantages, one can get from the implementation of Common Power Format (CPF), this design can be benefited from a CPF power management approach basically "by providing functional modeling of low-power constructs, minimizing the need for manual intervention, and using a robust verification to eliminate silicon failure risks that stem from functional and structural flaws." [11]

CPF implementation for this design requires considerations of following points:

- Defining library sets to put necessary supply voltages library files
- Specifying different power domains by assigning modules, blocks and IPs to their corresponding domains for a reliable power management flow, as well as reliable verification and measurement result in the future.

[3] *Digital Core* domain contains the operational circuit, which is the entire circuitry excluding memory banks, PLL block and the cells belong to *Output Buffer* and *Clock Core* domains.

- Introducing level-shifter cells based on their placed-in (and outside block) along with the corresponding voltage supplies to these blocks (where the level-shifters are placed in the design are based on the clear specification in the CPF, the definition of the level shifter rules as well as using the proper technology library for the intended High-Low (HL) or Low-High (LH) transitions).
- Specifying necessary power, ground and bias nets and assigning these nets with their corresponding domains.
- Some IPs inherently take two primary power nets (The internal analog & digital sub-modules require different power net in terms of the voltage), so it is necessary to define the domain primary power net based on the level-shifter requirements and the internal cells operating supply voltage)

Finally to put the CPF implementation into effect in the early stage of the PnR step, after initializing the design, the CPF script file should be loaded and committed by PnR tools. And then all other steps will be processed with honoring the power management implementation by refering the CPF script file.
The Following lines of code is an exmple of how the CPF should be implemented:

3.2.3 Floor Plan and Power Plan

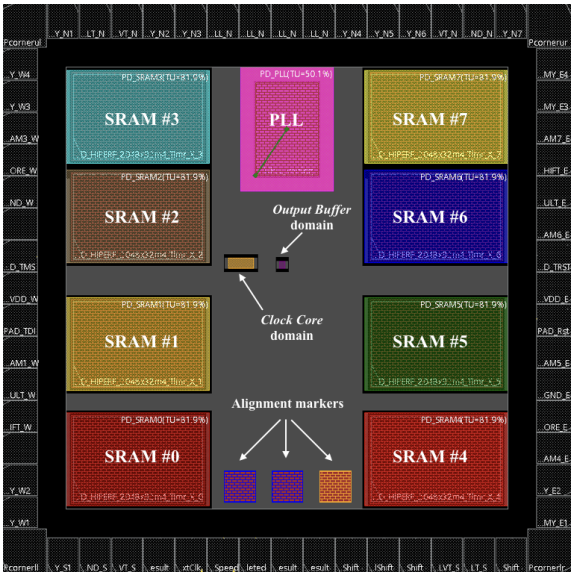


Figure 3.4: Floorplan with multi-power domains

As shown in Figure 3.4, eleven power domains indicated by different colours are planned in the layout (excluding three squares at the middle bottom), where the quantity of memory banks are set as eight because each of them will serve different

configuration or will be of different flavour. In addition, memories are assigned by individual power domains for a precise power measurement without being interfered with other banks, as well as for operating them at different voltage levels simultaneously; *Clock Core* domain and *Output Buffer* domain are both placed at middle of the chip so that an evenly distributed signal routing is able to be assured; PLL is located at the middle top. As PLL is sensitive to noise, it requires some special treatment in SoC placement: With the jitter taken into account, PLL supplies and ground for both analog and digital sub-modules should be connected with dedicated I/O pads, and theses pads are supposed to be adjacent to each other for cancellation of the noise present on supplies and ground (They are not supposed to be shared with the I/O ring). Also, other circuits should be as far away from the PLL as possible because the substrate noise increases the jitter at PLL output; No signal or supply/ground routing over the PLL, etc. Besides, the three squares placed at the middle bottom are the reserved areas for *emetrology* insertion, electrical-metrology, meaning these areas will be inserted by markers to be used for alignment during fabrication.

Refer to Figure 3.5, power rings and power stripes present that eight memory banks, along with *Clock Core* domain and *Output Buffer* domain, have their own individual supply rings (uncorreated power net and independent power pad). In the meanwhile, the ground and body bias of these blocks share the same net. PLL is connected to its dedicated supply and ground nets that are fed by PLL pads directly. Two supply nets are given to PLL, digital respective analog sub-modules. The same as for two ground nets. Furthermore, the *Digital Core* domain is the unoccupied area in the power planning (gray zones).

Table 3.1 lists all supply and ground nets. Except of the PLL, the remaining circuitry is using shared ground and body bias.

Table 3.1: Power and ground nets in PNR

Memomry banks	VDD_SRAM0 , VDD_SRAM1 , VDD_SRAM2 , VDD_SRAM3 VDD_SRAM4 , VDD_SRAM5 , VDD_SRAM6 , VDD_SRAM7
Digital Core domain	VDD_VDD_DIGITAL_CORE
Clock Core domain	VDD_CLOCK_CORE
Output Buffer domain	VDD_OUTPUT_BUFFER
PLL supply	VDD_PLL_ANALOG , VDD_PLL_DIGITAL
PLL ground	GND_PLL_ANALOG , GND_PLL_DIGITAL
Ground	GND
Body bias	GNDS , VDDS

Besides, there are a few more points noteworthy in Figure 3.5:

- Higher metal layers are selected for power rings and power stripes in order to reduce the effect of IR drops more effectively, because the higher metal layer is less resistive than the lower one.

- It is worthy of mention that all pins (signals ,supplies and ground) of the memory bank are only located at north and south, and vertical supply/ground stripes have already been implemented alternatively inside the IP-block. Therefore, in order not to squeeze all power and signal routing at top and down side, The internal power strips of memory banks are all connected to their outer rings through horizontal stripes. Consequently, for each bank, the power routing takes place at west and east, while signal routing does at north and south.

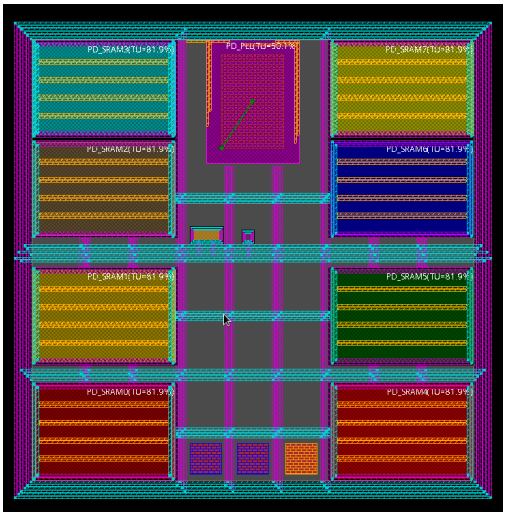


Figure 3.5: Layout of power planning

3.2.4 Special Routing

In Innovus, **sRoute** is the main engine for routing power nets. The routing is often straightforward, however, according to floor plan and power plan, sometimes this process becomes complex as it is not processes as designer's expectation/ For instance, the *Clock Core* domain and *Output Buffer* domain are both located inside the *Digital Core* domain, *sRoute* extends these two domains' power rails all over the *Digital Core* domain, and considering that the two domains must be surrounded by core-row gaps for domain-domain isolation, the solution to this over extention issue is listed as below:

- In floor-plan, process *cutRow* followed by *createRow* to move the core-row gaps outside the two domains.
- Run *sRoute-corePin* for the two domains.
- Create routing blockage over *Clock Core* and *Output Buffer* domains in order not to overlap the two domains with the extended *corePin* stripes from *Digital Core* domain.

- Run *sRoute-CorePin* for *Digital Core* domain.
- Remove the routing blockages over the two domains and go for signal routing.

3.2.5 Design Placement

Figure 3.6 shows the standard cell placement in the *Digital Core* domain which is the target domain for operational circuit. As it is shown clearly in the layout, the tool is restricted from placing standard cells beneath the power rails in order to avoid the probable blocks in front of signal routing. At the end of this chapter, the final layout after the entire PnR flow is displayed in Figure 3.7.

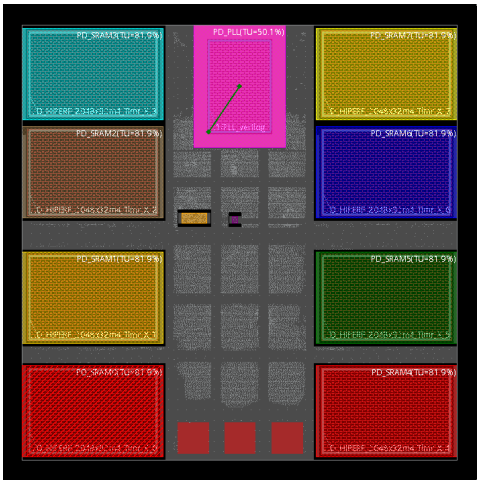


Figure 3.6: Standard cell placement in *Digital Core* domain

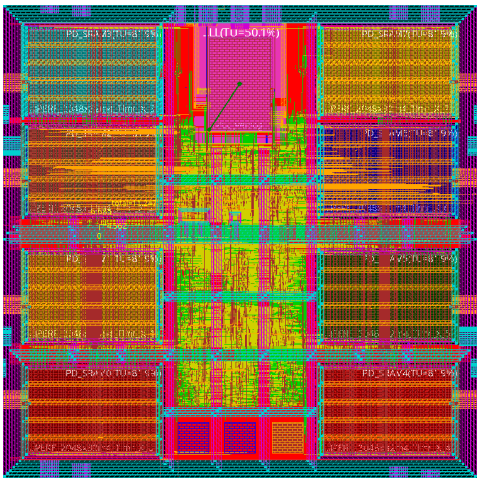


Figure 3.7: Overview of the final layout in PNR phase

Chapter 4
Conclusion

This thesis focuses on both front-end and back-end development of an automated self-test circuitry that is responsible to take care of SRAM memories during functionality and performance measurements to find all different process and operation corners.

From the above definition of the project, it comes up that the importance here is twofold: operating different memory tests internally and in a more automized fashion.

This project started with an algorithmic development and functional verification of the system based on a prior design and went far in detail in ASIC implementation stages (synthesis, place & route). However, the back-end flow of ASIC implementation is an experience-oriented and complex task which aggressively demands time.

There were several features available in this design (such as multiple clock sources, level-shifter implementation, multiple power domains, etc.) which in turn posed new challenges needed to be solved. The developed framework and flow nonetheless, is reusable and possible to be used as a foundation for future works.

Moreover, this document presented a behavioral description of logic implementation and finally in ASIC implementation chapter the important steps and results of the process were explained.

Chapter 5
Future Work

This thesis accomplishes the defined objectives which were to approach a fully functional prototype of a built-in automated test circuitry for dual-port SRAM. However, to go beyond the scope of this project for performing more optimization, some important points are noted here to be pursued for a future work.

- Referring to researchers in EIT ASIC department, the test circuitry often becomes the main problem in high-speed measurements in a way that DUT could not be pushed to be tested on the maximum speed at which it could operate. Therefore, improving the speed and performance further stays on the top of priority list for optimization.
- Improving the core utilization by squeezing the Digital Core into a compact macro to be able to add it as a built-in self test block to each memory.
- Test circuitry can be optimized to become more self-contained, in such a way that it can behave automatically in:
  - Memory initialization and reset
  - Internal self-test engine to find the worst corners of the memory and adjust the operating voltage with respect to that
  - Improving the currently existing blocks in order to add more test features to the memory (such as jumping to random address to test memory to the full extent, self-generating stimuli sequences instead of reading ones generated from external generator.)
- The test is limited by the internal storage volume. With bigger internal storage, a bigger volume of the SRAM can be tested in one run. However, increasing internal storage will impose bigger area cost and more difficult signal routing in PNR phase. Therefore, a good practice could be creating a bigger internal storage as a macro with controlled placement approach.
- Clock tuning for different ports of the SRAM, in such a way that successive reading and writing operations on a specific address would not be a source of memory content corruption

References

[1] IEEE Computer Society. *IEEE Standard for Test Access Port and Boundary-Scan Architecture*. May 2003.

[2] Rob Oshana. *Introduction to JTAG*,
https://goo.gl/q7B8iH, Visiting 2017 Oct.

[3] Built-in Self Test (BIST),
http://eesemi.com/bist.htm, Visiting 2017 Oct.

[4] JTAG Test Overview,
https://www.corelis.com/education/JTAG-Test-Overview.htm

[5] Microsemi. *IEEE Standard 1149.1 (JTAG) in the SX/ RTSX/ SXA/ eX/ RT54SX-S Families | Application Note AC160*. Microsemi, 2012.

[6] Peter Marwedel, *Embedded System Design - Embedded Systems Foundations of Cyber-Physical Systems, 2nd Edition*. Springer, 2011.

[7] Jie-Hong Roland Jiang, Srinivas Devadas. *Electronic design automation: synthesis, verification, and test*. Morgan Kaufmann, 2008.

[8] Synopsys Solvnet Article 033678,
https://solvnet.synopsys.com, Visiting 2017 Oct.

[9] Synopsys IC Compiler Manual, IC Compiler Tool Commands.
https://solvnet.synopsys.com, Visiting 2017 Nov.

[10] Patrick Lee. *Introduction to Place and Route Design in VLSIs*. Lulu Press, 2006.

[11] Power Forward. *A Practical Guide to Low-Power Design User Experience with CPF*. Power Forward, 2006.

[12] Axel Andersson, John Gustavsson. *Design of a Memory Compiler*. Department of Electrical and Information Technology, Lund University, 2016.

[13] Oskar Andersson. *Ultra-low Voltage Embedded Memories - Design Aspects and a Biomedical Use-case*. Department of Electrical and Information Technology, Lund University, 2016.

[14] Babak Mohammadi. *Ultra-low Power Design Approaches in Memories and Assist Techniques*. Department of Electrical and Information Technology, Lund University, 2017.

[15] Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic, *Digital Integrated Circuits - A Design Perspective, 2nd Edition*. PH, 2003.

Printed by Tryckeriet i E-huset, Lund 2017

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2017-612

http://www.eit.lth.se