A Fault-Tolerant Strategy for Embedded-Memory SoC OFDM Receivers

by

Vadim Smolyakov

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

A FAULT-TOLERANT STRATEGY FOR EMBEDDED-MEMORY SOC OFDM RECEIVERS

Vadim Smolyakov

Master of Applied Science, 2011

Graduate Department of Electrical and Computer Engineering

University of Toronto

Abstract

The International Technology Roadmap for Semiconductors projects that embedded memories will occupy increasing System-on-Chip area. The growing density of integration increases the likelihood of fabrication faults. The proposed memory repair strategy employs forward error correction at the system level and mitigates the impact of memory faults through permutation of high sensitivity regions. The effectiveness of the proposed repair technique is demonstrated on a 19.4-Mbit de-interleaver SRAM memory of an ISDB-T digital baseband OFDM receiver in 65-nm CMOS. The proposed technique introduces a single multiplexer delay overhead and a configurable area overhead of $\lceil M/i \rceil$ bits, where M is the number of memory rows and i is an integer from 1 to M, inclusive. The proposed strategy achieves a measured 0.15 dB gain improvement at a 2×10^{-4} Quasi-Error-Free (QEF) BER in the presence of memory faults for an AWGN channel.

Acknowledgments

I would like to thank my advisor Professor Glenn Gulak for his exceptional dedication and effectiveness as a researcher, a teacher and a mentor, remarkable resourcefulness, boundless enthusiasm and outstanding work ethic. I'm grateful for the opportunity to meet and work with such great Professor. Thank you for setting a superb example, stretching my boundaries through challenging assignments and for providing numerous growth and self-development opportunities.

I would like to acknowledge the members of my MASc committee: Prof. Raviraj Adve, Prof. Paul Chow, Prof. Aleksandar Prodic, and Prof. Glenn Gulak for their time and valuable expert comments.

I would like to thank Curtis Ling and Tim Gallagher at MaxLinear Inc for the opportunity to realize the subject of this thesis, stimulating review meetings, engineering guidance and unparalleled dedication to the company. I would like to express thanks to the Digital Systems team: Anand Anandakumar, Josephine Chu, Lynn Greiner, Seung-Chul Hong, Hyungjin Kim, Liping Li, James Qiu, Sridhar Ramesh, Ioannis Spyropoulos, Pawan Taluja and Mingrui Zhu in alphabetical order. I appreciate the guidance of Reza Rad and Nitin Nigam in DFT architecture and advice of Arun Kedambadi and Jining Duan in all ASIC related matters.

I would like to thank Jeet Narsinghani and Jaro Pristupa for their expert support with EDA tools and SoC test methodology throughout my graduate studies.

I would like to acknowledge my fellow graduate students. In particular, Glenn's group: Mario Milicevic, Saeed Moradi, Nassim Nikkhoo, Dimpesh Patel, Mayukh Roy, Mahdi Shabany, Ameer Youssef, Meysam Zarghamm and Electronics Group (BA-5000 and BA-5158) at large: Karim Abdelhalim, Behrooz Abiri, Saber Amini, Kevin Banovic, Mike Bichan, Yunzhi (Rocky) Dong, David Halupka, Derek Ho, Safeen Huda, Hamed Jafari, Sadegh Jalali, Bert Leesti, Kostas Pagiamtzis, Ali Reza, Alain

Rousson, Shayan Shahramian, William Song, Colin Tse, Oleskiy Tyshchenko, Kentaro Yamomoto, Hemesh Yasotharan, and Andy Zhang in alphabetical order.

Finally, I would like to thank my parents, Sergey and Valeriya, for their continuous encouragement and support.

Contents

List of Figures **vii**

List of Tables **x**

1 Introduction **1**

1.1 Motivation 1

1.2 Objectives 3

1.3 Thesis Outline 3

2 OFDM Receiver Overview **5**

2.1 Introduction 5

2.2 System Overview 5

2.3 Stream Modules 8

2.4 Block Modules 9

2.5 FEC Modules 10

2.5.1 Inner Code 11

2.5.2 Outer Code 14

2.6 Frequency Time Deinterleaver 16

2.6.1 Overview 16

2.6.2 Frequency Deinterleaver 19

2.6.3 Time Deinterleaver 21

2.6.4 Deinterleaver Architecture 23

2.7 Summary 26

3 Fault-Tolerant Techniques **27**

3.1 Introduction 27

3.2 Yield Model 27

3.3 Fault Model 37

3.4 Repair Strategies 43

3.5 Proposed Repair Technique 47
3.6 Simulation Results 49
3.7 Summary 55

4 VLSI Implementation **57**
4.1 Introduction 57
4.2 DFT Architecture 57
4.3 Memory Repair Architecture 60
4.4 Memory Repair Verification 66
4.5 Measurement Results 68
4.6 Summary 73

5 Conclusion **74**
5.1 Summary of Contributions 74
5.2 Future Directions 75

Appendices **77**

A ISDB-T Standard **77**
A.1 ISDB-T Overview 77
A.2 OFDM Transmission Parameters 77
A.3 Channel Models 80

B Interleaver Pseudocode **82**

C SRAM Embedded Memory **84**
C.1 The 6-T SRAM Cell 84

References **88**

List of Figures

1.1 Embedded Memory Increase as a Percentage of SoC Area [1]. 1
1.2 Variability Induced Failure Rate vs Technology Node [2]. 2

2.1 Generic OFDM Receiver SoC Architecture with Digital Demodulator Highlighted. 6
2.2 Digital Demodulator Architecture with Frequency Time Deinterleaver Highlighted. 6
2.3 An Area Accurate ISDB-T Receiver SoC Floorplan with Frequency Time Deinterleaver Highlighted. 7
2.4 OFDM Stream Modules. 8
2.5 OFDM Block Modules. 9
2.6 OFDM FEC Blocks. 11
2.7 Viterbi Decoding of a Rate-1/2 Convolutional Code with Constraint Length $m = 7$, Generated by $G1 = 171_{oct}$, $G2 = 133_{oct}$. 12
2.8 A Viterbi Decoder Architecture. 13
2.9 A Reed-Solomon Decoder. 14
2.10 Concatenated Code Configurations. 15
2.11 Frequency Time De-interleaver Overview. 16
2.12 Convolutional Interleaver (Tx) - Deinterleaver (Rx) Pair. 17
2.13 OFDM Frame Structure [3]. 18
2.14 Frequency Deinterleaver Block Diagram. 19
2.15 Carrier Randomization for an OFDM Symbol $S_{(i,\ j,\ k)}$ in Mode 2. . . 20
2.16 Inter-segment Frequency De-Interleaver. 21
2.17 Time Deinterleaver Block Diagram. 22
2.18 A Conceptual FTDI SRAM Architecture. 24
2.19 FTDI Type I SRAM Tile Architecture. 24
2.20 FTDI Type I SRAM Memory $1K \times (16 \times 40)$ Architecture. 25

3.1 Fabrication Defects in a CMOS Process. 28

3.2 Plot of $p_X(k)$ vs k for Poisson and Negative Binomial Distributions with $\lambda = 0.0546$. 34
3.3 Plot of $p_X(k)$ vs k for Poisson and Negative Binomial Distributions with $\lambda = 42$. 34
3.4 Yield Plot. 36
3.5 Generic Memory Architecture. 38
3.6 6-T SRAM Memory Cell. 39
3.7 Example SRAM Fault Types. 40
3.8 A Faulty Memory Array with M=8, N=10, SR=2 and SC=3. 45
3.9 A Bipartite Graph G of the Faulty Memory Array in Figure 3.8. . . 45
3.10 A Maximum Matching Flow Graph M of the Faulty Memory Array in Figure 3.8. 46
3.11 Sensitivity Coefficient for the FTDI Embedded Memory. Memory columns store {I[14:9], Q[8:3], CN[2:0]} associated with a single OFDM symbol. Memory rows store OFDM symbols $S_{(i\ ,j\ ,k)}$. 47
3.12 Simulated BER Plot (Mode 3, Layer B: 64-QAM, R=3/4, N_R=1, AWGN, 2000 OFDM symbols, FEC with 0 and 2 Iterations). 49
3.13 Simulated BER Plot (Mode 3, Layer B: 64-QAM, R=3/4, N_R=1, AWGN, 2000 OFDM symbols). 52
3.14 Simulated BER Plot (Mode 3, Layer B: 64-QAM, R=3/4, N_R=1, AWGN, 2000 OFDM symbols, 0 iterations). 53
3.15 Simulated BER Plot (Mode 3, Layer B: 64-QAM, R=3/4, N_R=1, AWGN, 2000 OFDM symbols). 54

4.1 SoC Level DFT Architecture with JTAG IEEE 1149.1 Interface [4]. . 58
4.2 SMS (STAR Memory System) Block Level Architecture with SECT IEEE P1500 Interface [4]. 58
4.3 STAR Processor and Memory Wrapper Architecture [4]. 59
4.4 Proposed Memory Repair Architecture. 61
4.5 Configurable Row Address Fault Register Architecture. 62
4.6 Memory Repair Enable Architecture for an N-bit Word with Two Sensitivity Regions. 62
4.7 Interleave (ITL) Block Architecture for 8-bit I, 8-bit Q and 4-bit CN. 64
4.8 Memory Repair Verification Platform. 66
4.9 Memory Repair FPGA Prototype. 67

4.10 Measured Results (FPGA). 69
4.11 Measured Results (FPGA): BER vs Faulty LSB+i Data for an AWGN Channel. 70
4.12 Measured Results (FPGA): QEF BER vs MSB-i Faults for TU-6 and TU-12 Fading Channels. 71

C.1 6-T SRAM Read Stability and Write Ability. 84
C.2 6-T SRAM Static Noise Margin. 85
C.3 6-T SRAM 2×2 Memory Array Layout. 86

List of Tables

2.1 Puncture Pattern for a Rate-1/2 Convolutional Code with $m = 7$, $G_1 = 171_{oct}$, $G_2 = 133_{oct}$. 13

2.2 Interleaver Characteristics. 18

2.3 ISDB-T Mode 3: Typical Transmission Parameters. 21

3.1 Expected Number of Faults (λ) in the Five Largest Embedded Memory Blocks. 33

3.2 MBIST Test Patterns [5], [6]. 42

3.3 Simulation Parameters. 50

4.1 ASIC Synthesis Results of the Proposed Memory Repair in 65 nm CMOS for the Worst-Case PVT: SS, 1.08V, 125°C. Δ_i: Repair Overhead When the Faulty Row Address is Recorded for Every i Memory Rows. 65

4.2 SRAM Memory Repair Performance Comparison. *Based on $i = 4$ in Table 4.1. 65

4.3 FPGA Post Place and Route Resource Summary. 67

A.1 ISDB-T: OFDM Segment Parameters [3]. 78

A.2 ISDB-T: OFDM Transmission Signal Parameters [3]. 79

A.3 Channel Models. 80

A.4 Channel Models. 81

List of Symbols

OFDM Receiver:

y	Complex received symbol vector
H	Complex MIMO channel matrix
$\widehat{H}$	Estimate of the complex MIMO channel matrix
s	Complex transmitted symbol vector
$\hat{s}$	Estimate of the complex transmitted symbol vector
v	Complex noise vector
σ^2	Noise variance
k	Message length
n	Codeword length
R	Code rate
t	Number of correctable errors
α	Primitive element of a finite field
N_T	Number of transmit antennas
N_R	Number of receive antennas
N_{FFT}	N-point Fast Fourier Transform
N_{iter}	Number of FEC decoding iterations
F_{offset}	Frequency offset
F_{drift}	Frequency drift
T_{offset}	Sampling offset

Deinterleaver:

π	Interleaver
π^{-1}	Deinterleaver
m	Deinterleaver degree
I	Deinterleaver length
b	Burst error length
d	Minimum deinterleaver delay
$S_{(i,\ j,\ k)}$	OFDM symbol number i, carrier j, segment k

Yield Model:

Y	Overall yield
Y_{gross}	Gross yield
Y_{rnd}	Random fault yield
Y_{red}	Redundancy yield
i	Fabrication process step
m	Total number of process steps
j	Fault type
k	Total number of faults
D_i	Fault density
A_i	Critical area
λ_i	Expected number of faults
α_i	Clustering coefficient
l	Chip index
L	Number of chips on a wafer
λ_{wafer}	Expected number of faults per wafer
N_B	Number of blocks on a chip
N_R	Number of redundant blocks on a chip
λ_{red}	Expected number of faults with redundancy

Memory Repair:

M	Memory rows
N	Memory columns
SR	Spare rows
SC	Spare columns
N_{SA}	Number of stuck-at faults
N_{SO}	Number of stuck-open faults
N_{DR}	Number of data retention faults
N_{TF}	Number of transition faults
N_{CF}	Number of coupling faults
ζ	Sensitivity Coefficient

List of Acronyms

AGC	Automatic Gain Control
ASIC	Application-Specific Integrated Circuit
ATE	Automated Test Equipment
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BIRA	Built-In Redundancy Analysis
BIST	Built-In Self-Test
BL	Bit Line
CF	Coupling Fault
CMOS	Complementary Metal Oxide Semiconductor
CORDIC	Coordinate Rotation Digital Computer
CN	Carrier-to-Noise Ratio
CP	Continual Pilot
CPU	Central Processing Unit
CSI	Channel State Information
DRAM	Dynamic Random Access Memory
DRF	Data Retention Fault
DFT	Design for Test
DPN	De-puncturer

ECC	Error Correction Code
EXIT	Extrinsic Information Transfer Chart
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
FTDI	Frequency Time De-Interleaver
GF	Galois Field
GI	Guard Interval
ICI	Inter-Carrier Interference
IF	Intermediate Frequency
ISDB-T	Integrated Services Digital Broadcasting - Terrestrial
JTAG	Joint Test Action Group IEEE 1149.1
LFSR	Linear Feedback Shift Register
LLR	Log Likelihood Ratio
LSB	Least Significant Bit
LUT	Look Up Table
MBIST	Memory Built-In Self-Test
MIMO	Multiple-Input Multiple-Output
MISR	Multiple Input Signature Register
MSB	Most Significant Bit
OFDM	Orthogonal Frequency Division Multiplexing
PER	Packet Error Rate
PMF	Probability Mass Function

PVT Process Voltage Temperature

QAM Quadrature Amplitude Modulation

QEF Quasi Error Free

QPSK Quadrature Phase Shift Keying

RAM Random Access Memory

RF Radio Frequency

RS Reed Solomon

RSSI Received Signal Strength Indicator

SAF Stuck At Fault

SECT Standard for Embedded Core Test IEEE P1500

SNM Static Noise Margin

SoC System on Chip

SOF Stuck Open Fault

SOVA Soft Output Viterbi Algorithm

SISO Soft-Input Soft-Output

SMS STAR Memory System

SP Scattered Pilot

SRAM Static Random Access Memory

STAR Self Test And Repair

TF Transition Fault

VA Viterbi Algorithm

VLSI Very Large Scale Integration

WL Word Line

1 Introduction

1.1 Motivation

The International Technology Roadmap for Semiconductors (ITRS) projects that embedded memories will occupy an increasing percentage of a System-on-Chip (SoC) area [1], [7] as shown in Figure 1.1. As a result, the overall yield of an SoC is becoming increasingly dependent on memory yield. The high density of integration, aggressive design rules and small transistor geometries make embedded memories particularly susceptible to manufacturing defects.

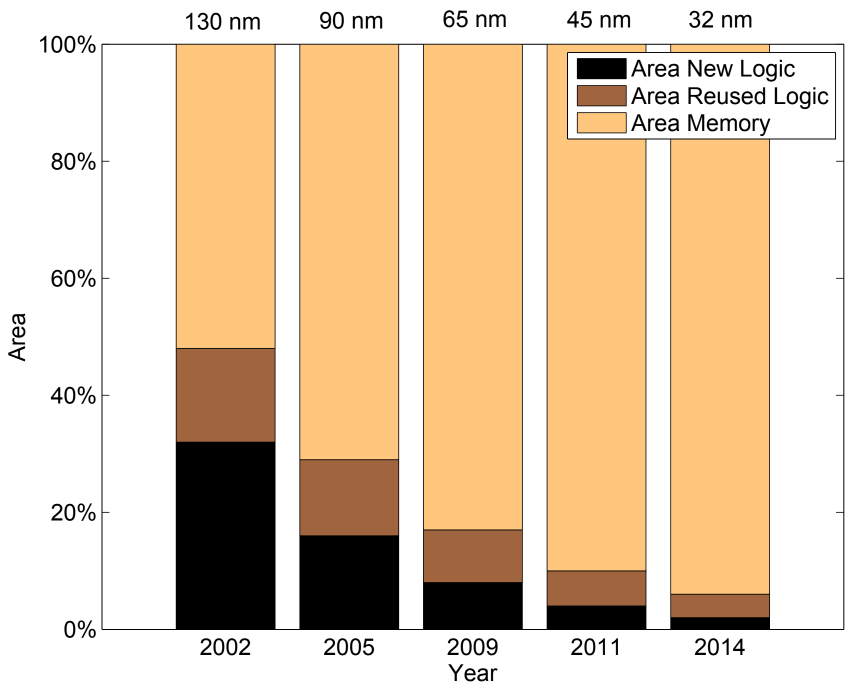


Figure 1.1: Embedded Memory Increase as a Percentage of SoC Area [1].

Manufacturing process variations also dramatically reduce reliability and yield of

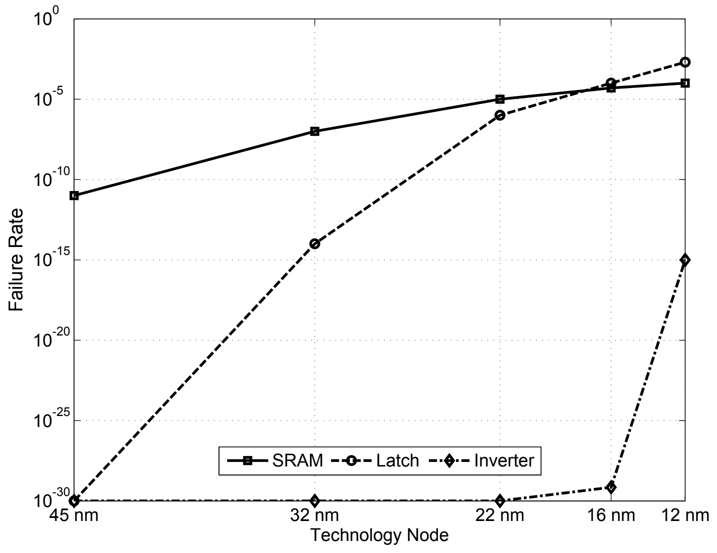


Figure 1.2: Variability Induced Failure Rate vs Technology Node [2].

fabricated SoCs as shown in Figure 1.2[1]. Hence, demand will increase for circuits that consume large die areas, but are highly adaptable to internal failures. Such designs can help control costs of design verification, manufacturing and testing [2].

Repair strategies that utilize redundant resources such as spare rows and columns to repair faulty memory cells introduce area overhead and contribute to the cost of the SoC. However, not all memory cells contribute equally to system-level performance. For example, in baseband signal processing a faulty LSB bit when compared to an MSB error leads to a smaller performance degradation as measured by system performance parameters such as the Bit Error Rate (BER). Similarly, memories that store data prior to filtering and error correction operations exhibit higher error tolerance. This variation in sensitivity to bit errors can be exploited to minimize the impact of errors, whereby memory regions with high sensitivity to error are permuted with regions of low error sensitivity without resorting to redundant rows and columns.

In many statistical signal processing applications such as digital communications and video processing a certain number of errors can be tolerated without a notice-

[1] Failure rates were obtained by simulating the three canonical circuits (i.e., an SRAM cell, a latch, and an inverter) under the influence of manufacturing process variability. The respective criteria for failure: for an SRAM cell, the failure to read, write or standby; for a latch, set-up or hold failure; and for inverter, a failure to invert the output [2].

able degradation in performance or user experience of the device. As a result, in memory-intensive algorithms considerable area savings can be achieved by mitigating the impact of errors without employing redundant resources and instead re-mapping faulty memory cells containing high value content with working memory cells containing low value content. Thus, a *fault sensitivity coefficient* can be assigned for each memory cell based on a system performance metric such as BER and exploited to reduce implementation area and hence cost.

This thesis presents a memory repair strategy that interfaces with a Built-In Self-Test (BIST) memory wrapper to detect and analyze the location of errors and minimizes their impact by permuting memory regions of high and low error sensitivity, while minimizing area and timing overhead.

1.2 Objectives

The main objective of this thesis is to develop a cost-effective memory repair solution for an SoC-based OFDM receiver containing embedded memory. A successful repair technique should introduce minimal area, speed, and power overhead and provide acceptable error tolerance as measured by the Bit Error Rate (BER) and the Packet Error Rate (PER) at the Quasi-Error-Free (QEF) operating point for an AWGN channel as well as TU-6 and TU-12 fading channels as described in Appendix A. The objectives of this thesis are to develop a memory repair strategy to:

- Increase SoC Yield
- Lower Memory Repair Cost by Minimizing Logic and Area Overhead
- Improve Performance as Measured by BER, Clock Rate, and Power Dissipation.

Without loss of generality, the operation and performance improvement of the memory repair strategy is illustrated with an ISDB-T OFDM receiver due to its large de-interleaver memory requirements as outlined in the ISDB-T standard [3].

1.3 Thesis Outline

The thesis is organized into *five* chapters. *Chapter 1* provides a motivation for the topic, states thesis objectives and presents the thesis outline. *Chapter 2* describes the embedded-memory OFDM receiver partitioned into stream, block, and Forward Error Correction (FEC) modules. The de-interleaver embedded memory architecture as well as the yield and fault models are described. *Chapter 3* reviews fault-tolerant memory repair strategies including the proposed repair technique. Simulation results of the proposed technique are presented. *Chapter 4* details a Very Large Scale Integration (VLSI) implementation of the proposed repair technique including the repair architecture, simulation results, FPGA prototype and measurement results. *Chapter 5* concludes the thesis with a summary of contributions and future directions. *Appendix A* provides important design parameters for the ISDB-T standard including OFDM transmission parameters and fading channel models. *Appendix B* contains pseudocode describing the operation of various interleavers and their corresponding de-interleavers. *Appendix C* reviews the operation of the 6-T SRAM memory cell.

2 OFDM Receiver Overview

The wireless telegraph is not difficult to understand. The ordinary telegraph is like a very long cat. You pull the tail in New York, and it meows in Los Angeles. The wireless is the same, only without the cat.

Albert Einstein

2.1 Introduction

Orthogonal Frequency Division Multiplexing (OFDM) receivers use multiple carriers to transmit data. OFDM transmission schemes are ubiquitous in wireline (DVB-C, MoCA, G.hn) as well as wireless (DVB-T/H, LTE, ISDB-T) standards. The design differences across OFDM receivers supporting different standards can be abstracted and grouped into *stream*, *block*, and *Forward Error Correction (FEC)* modules. According to the International Technology Roadmap for Semiconductors (ITRS), the amount of SoC embedded memory used in each of the OFDM receiver modules is expected to rise [7] and so is the number of manufacturing errors due to technology scaling and high density of embedded memory. Thus, to address the problem of increasing fault density, a generic model of an embedded-memory OFDM receiver is developed. Without loss of generality, the proposed memory repair strategy is illustrated on an ISDB-T OFDM receiver due to large memory requirements for frequency time de-interleaving (FTDI) as dictated by the ISDB-T standard [3].

2.2 System Overview

A generic OFDM receiver architecture is presented in Figure 2.1. The receiver can be partitioned into RF tuner, CPU and digital DEMOD sub-systems. The RF tuner

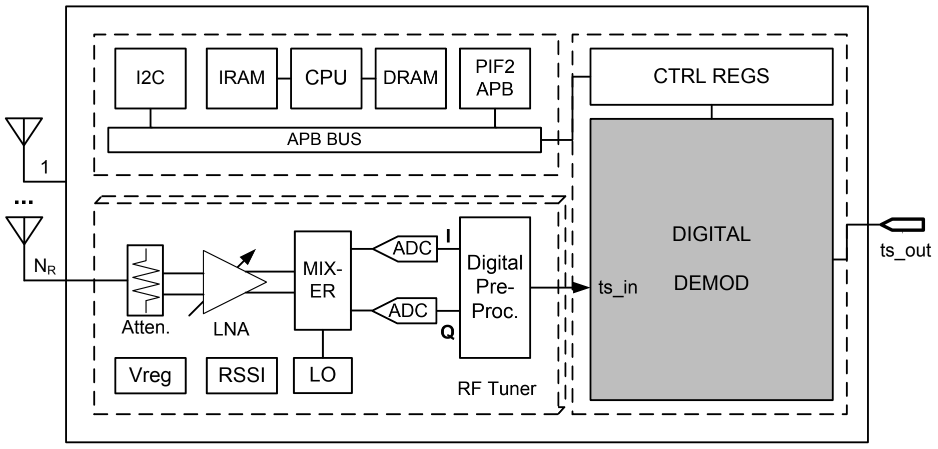


Figure 2.1: Generic OFDM Receiver SoC Architecture with Digital Demodulator Highlighted.

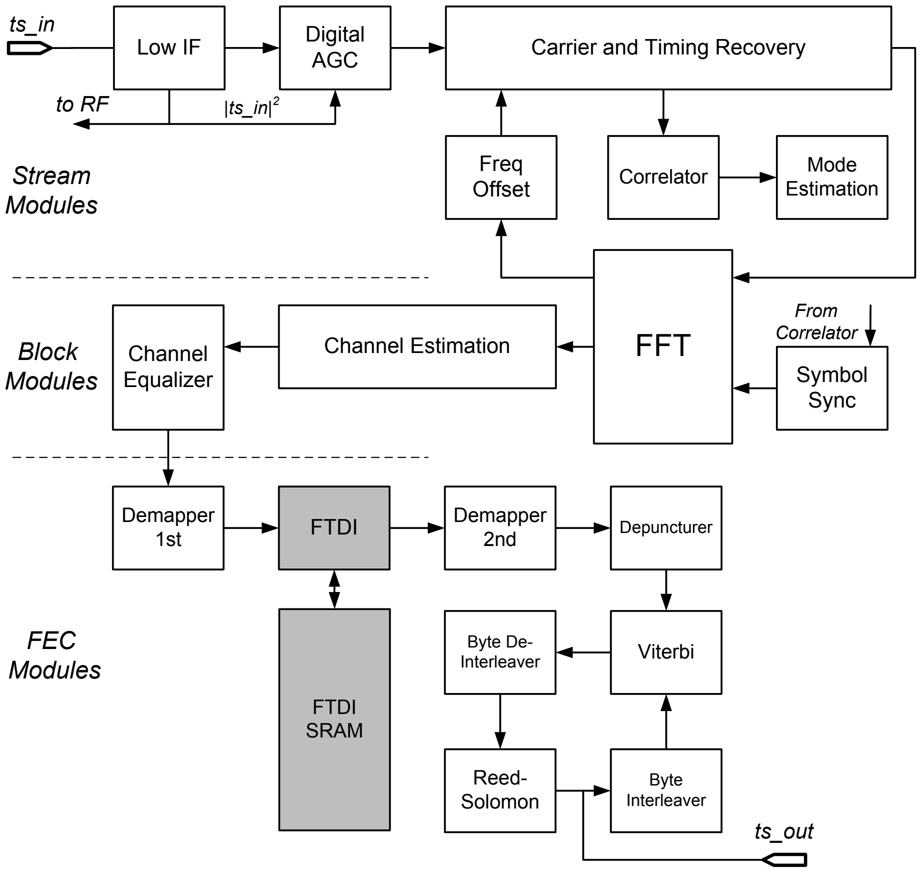


Figure 2.2: Digital Demodulator Architecture with Frequency Time Deinterleaver Highlighted.

sub-system down-converts the RF input to digital baseband using up to N_R receive antennas. The CPU sub-system controls the operation of the demodulator through the firmware and register interface. The digital DEMOD sub-system processes the digitized baseband input stream.

The focus of this work is on the digital demodulator detailed in Figure 2.2. An OFDM demodulator consists of *three* main processing modules: *stream*, *block*, and *FEC* modules. The stream modules perform synchronization and mode estimation functions. The block modules compute the FFT, and carry out channel estimation and equalization. The FEC modules perform frequency and time de-interleaving in addition to error correction illustrated by soft-output Viterbi and Reed-Solomon decoders ultimately producing the output transport stream *ts_out*.

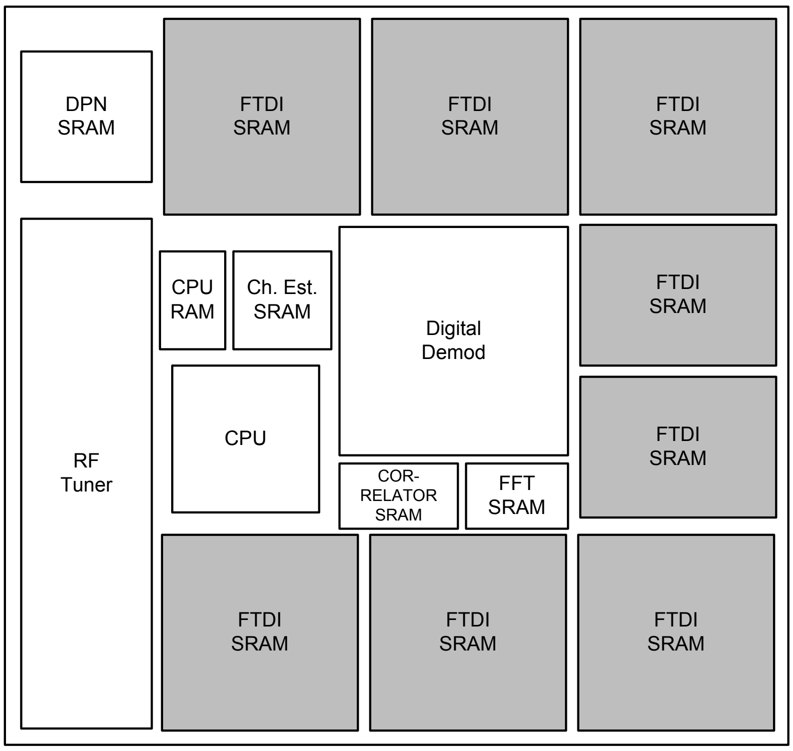


Figure 2.3: An Area Accurate ISDB-T Receiver SoC Floorplan with Frequency Time Deinterleaver Highlighted.

Frequency and time de-interleaver (FTDI) operations require large embedded memory resources. As a result, the frequency-time deinterleaver is the single largest embedded memory on chip as can be seen from an area accurate floorplan in Figure 2.3. The de-interleaver memory occupies 62% of the total SoC core area. Therefore, the de-interleaver memory is the single largest area contributor, followed by the depuncturer (DPN) at 4%, the channel estimator at 1.9%, the FFT at 1.5% and the correlator at 0.5% of embedded SRAM area expressed as a percentage of the total core area. Furthermore, due to the high density of embedded SRAM, the probability of SRAM errors per unit area caused by manufacturing defects is higher than digital logic further magnifying its importance. Thus, area efficient memory repair strategies must be developed to address the higher probability of SRAM bit errors. We briefly review the operation of stream, block and FEC modules and comment on their respective memory demands and opportunities.

2.3 Stream Modules

A block diagram of the stream modules is shown in Figure 2.4.

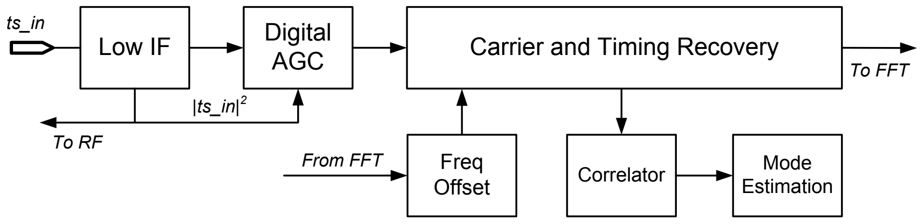


Figure 2.4: OFDM Stream Modules.

In the *Low IF* block, the DC-offset of the transport stream (TS) input is removed to avoid front-end circuit saturation and to allow for an accurate spectrum estimate of the low frequency components. The signal is further filtered using a cascade of IIR filters and the signal power is computed and forwarded to the digital Automatic Gain Control (AGC) module. The *digital AGC* scales I/Q according to the power computation in the Low IF block to reduce dynamic range through amplification of signals with low input-power or attenuation of high input-power signals.

The *carrier and timing recovery* module applies frequency offset correction to the input I/Q data via the COordinate Rotation DIgital Computer (CORDIC) algorithm [8], [9] using a rotation angle supplied by the frequency offset module in which the integer and the fractional carrier frequency estimates are combined into a single control parameter. Next, temporal Inter-Carrier Interference (ICI) cancelation is performed and the rotated I/Q data is re-sampled to the FFT sampling frequency. The *correlator* is used for mode and channel estimation. The transmission mode and the guard interval (GI) are estimated by the maximum value of cross-correlation $\phi_{a_i x}$ between the received signal x and a stored set of vectors a_i corresponding to i different modes and GI lengths:

$$\phi_{a_i x}[n] = \sum_{k=0}^{N-1} a_i[k]x[n+k] \tag{2.1}$$

Embedded SRAM memory is used in the implementation of delay buffers for GI correlation, ICI cancelation, as well as Look-Up Tables (LUTs) for CORDIC rotation angles. The combined memory area for stream blocks constitutes 1.7% of the SoC core area. Due to the small memory size, the expected number of memory faults is lower, while memory repair poses a larger percentage of area overhead. Thus, the benefits of memory repair may not justify the implementation costs, and therefore memory instances of small size are typically unrepairable, although memory built-in self-test (MBIST) is used to test for memory faults.

2.4 Block Modules

The block modules convert the processed I/Q data into the frequency domain, collect channel information and perform channel equalization as shown in Figure 2.5. The

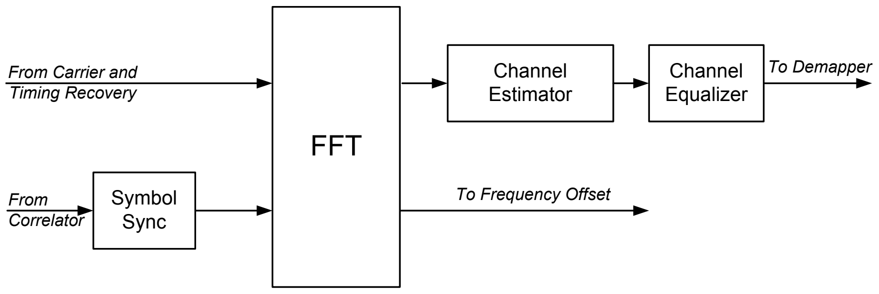


Figure 2.5: OFDM Block Modules.

symbol sync block issues an FFT start signal based on the correlation of the cyclic prefix with the received symbol. An 8K FFT computes the frequency spectrum of the OFDM frame data. The *frequency offset* module estimates frequency offset larger than the carrier interval according to pilot tones embedded in the OFDM frame [3].

The *channel estimator* constructs a channel delay profile by interpolating the amplitudes of the pilot signals. For multiple antenna configurations, the weight factors used in maximal-ratio combining are also computed from noise power estimates of the delay profile:

$$Y = \frac{\sum_{i=0}^{N_R} \frac{\widehat{H}^{i*}}{\sigma_i} \frac{\widehat{Y}^{i}}{\sigma_i}}{\sum_{i=0}^{N_R} \frac{|\widehat{H}^{(i)}|^2}{\sigma_i^2}} \tag{2.2}$$

where $\widehat{H}$ is the channel estimate, N_R is the number of receive antenna, and σ_i is the standard deviation of the noise. The *channel equalizer* uses the channel estimate $\widehat{H}$ for channel compensation:

$$\hat{s} = \frac{y}{\widehat{H}} = \frac{y \times \widehat{H}^*}{|\widehat{H}|^2} \tag{2.3}$$

Embedded memory is used in the implementation of adaptive channel estimation and computation of the 8K FFT. The memory size depends on the particular architecture used to implement the block modules. For example, single and multi-path Delay Commutator (DC) and Delay Feedback (DF) FFT architectures [10], [11] offer different design trade-offs in terms of memory storage of twiddle multipliers, the number of additions and multiplications, butterfly and multiplier utilization, and complexity of the control logic. In particular, the use of mixed and higher radix butterfly architectures such as R2^2SDF [12] can lower computational complexity and provide optimum performance and memory storage well suited for VLSI implementation. The embedded memory used to implement the block modules represents 6.1% of the SoC core area.

2.5 FEC Modules

The FEC modules carry out forward error correction as shown in Figure 2.6. The FEC modules include a *frequency and time de-interleaver (FTDI)*, a *de-puncturer (DPN)*, a *Viterbi decoder* for the inner convolutional code and a *Reed-Solomon (RS) decoder* for the outer Reed-Solomon code.

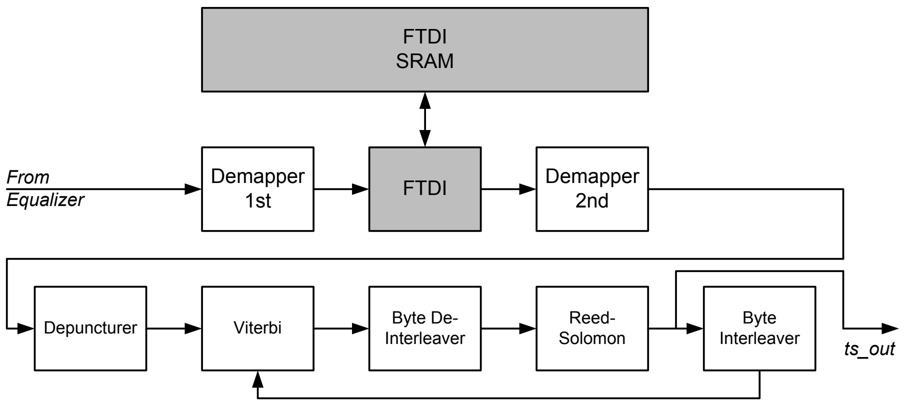


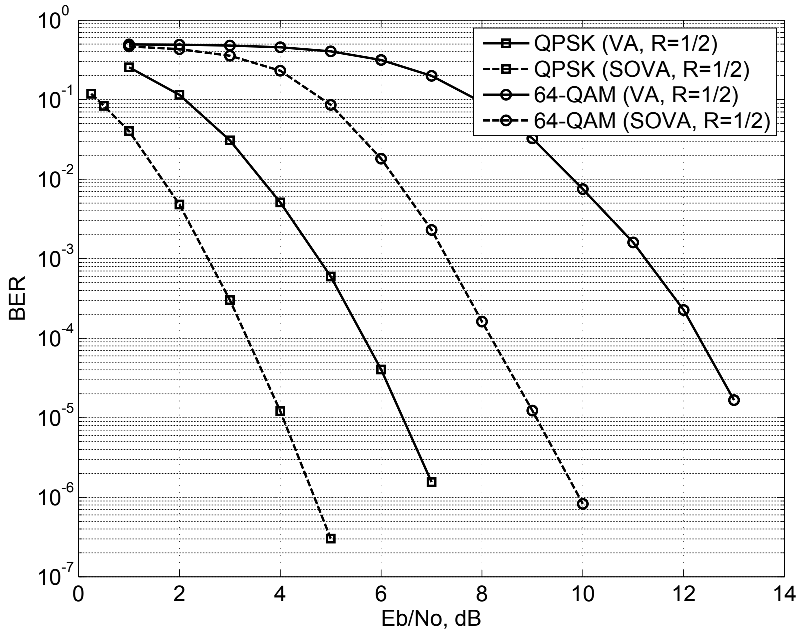
Figure 2.6: OFDM FEC Blocks.

2.5.1 Inner Code

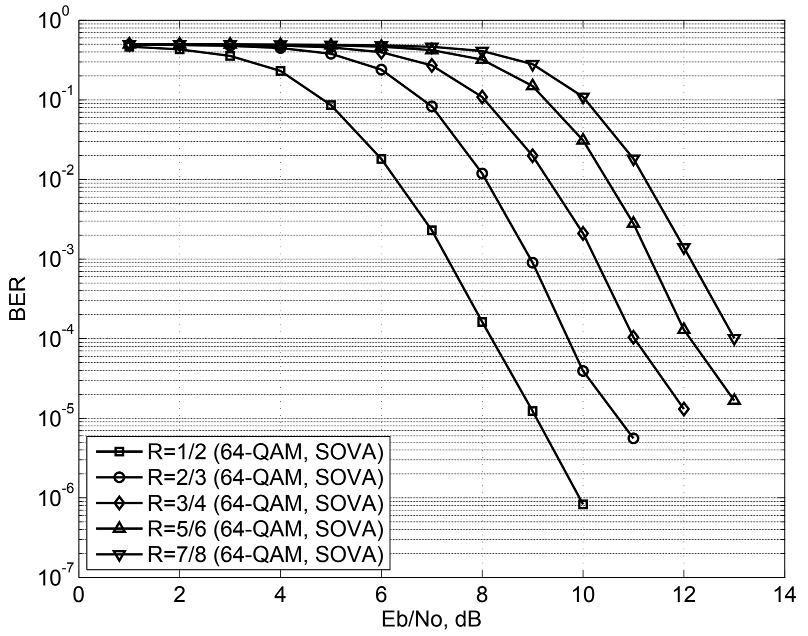
A punctured convolutional code with a constraint length of $m = 7$ and a coding rate of 1/2 is used as the inner code with generator polynomials $G_1 = 171_{oct}$ and $G_2 = 133_{oct}$ [3]. The Viterbi Algorithm (VA) [13] is used for maximum likelihood sequence decoding of convolutional codes. The 64-state Viterbi can be visualized by means of a trellis diagram. The Viterbi algorithm operates on a trellis by computing a metric for each branch in a trellis such as the Euclidean distance between the received and the ideal constellation points for an AWGN channel. Therefore, a path in a trellis with the minimum accumulated path metric represents the most likely transmitted sequence. By selecting the common ancestor on the minimum distance survivor path from each state in the trellis, the Viterbi algorithm performs maximum likelihood sequence detection. Moreover, the difference in branch metrics between the minimum distance path and the next shortest path can be used to calculate a reliability value for each decoded bit. The Log-Likelihood Ratio (LLR) is used to quantify reliability and improve decoding performance in the Soft Output Viterbi Algorithm (SOVA) [14]. The LLR of bit x_k given the received vector y is defined as:

$$\mathrm{LLR}(x_k|y) = \ln \frac{P[x_k = 1|y]}{P[x_k = 0|y]} \tag{2.4}$$

Figure 2.7(a) shows the decoding performance of hard and soft output Viterbi algorithms for a rate-1/2 convolutional code generated by $G_1 = 171_{oct}$ and $G_2 = 133_{oct}$. Figure 2.7(b) shows the decoding performance of SOVA for variable code rates created using the puncture pattern in Table 2.1.



(a) BER Plot of the Hard and Soft Decision Viterbi Decoders with Code Rate R=1/2 for an AWGN Channel.



(b) BER Plot of the Soft Decision Viterbi Decoder with Variable Code Rates for an AWGN channel.

Figure 2.7: Viterbi Decoding of a Rate-1/2 Convolutional Code with Constraint Length $m = 7$, Generated by $G1 = 171_{oct}$, $G2 = 133_{oct}$.

Code Rate	Puncture Pattern	Transmitted Sequence
1/2	X : 1 Y : 1	X1, Y1
2/3	X : 1 0 Y : 1 1	X1, Y1, Y2
3/4	X : 1 0 1 Y : 1 1 0	X1, Y1, Y2, X3
5/6	X : 1 0 1 0 1 Y : 1 1 0 1 0	X1, Y1, Y2, X3, Y4, X5
7/8	X : 1 0 0 0 1 0 1 Y : 1 1 1 1 0 1 0	X1, Y1, Y2, Y3, Y4, X5, Y6, X7

Table 2.1: Puncture Pattern for a Rate-1/2 Convolutional Code with $m = 7$, $G_1 = 171_{oct}$, $G_2 = 133_{oct}$.

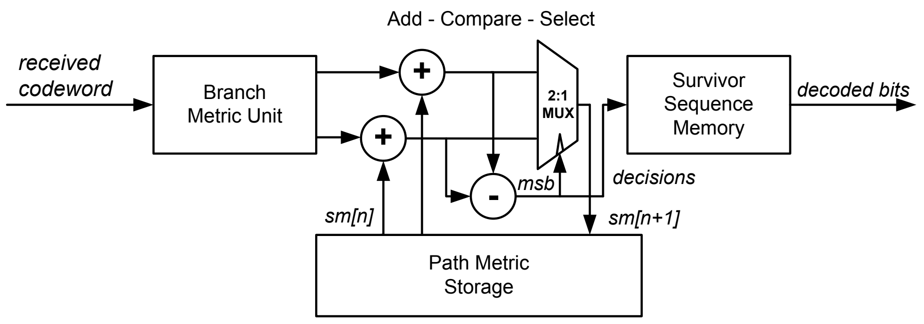


Figure 2.8: A Viterbi Decoder Architecture.

There are three main components to the Viterbi algorithm: branch metric computation, add-compare-select and traceback survivor sequence decoding as shown in Figure 2.8. The branch metric computation unit calculates the distance between the received signal and the ideal expected signals. The add-compare-select unit accumulates the path metric. The traceback unit or survivor path decode keeps track of the minimum distance path through the trellis. While branch metric computation and add-compare-select units carry out arithmetic operations, the survivor path decode is the most memory intensive block in the Viterbi algorithm. Typically, path histories of length $10 \times m$ are maintained and for the 64-state SOVA with 10-bit LLR a traceback memory of $(10 \times 7) \times 64 \times 10 = 44,800$ *bits* is needed.

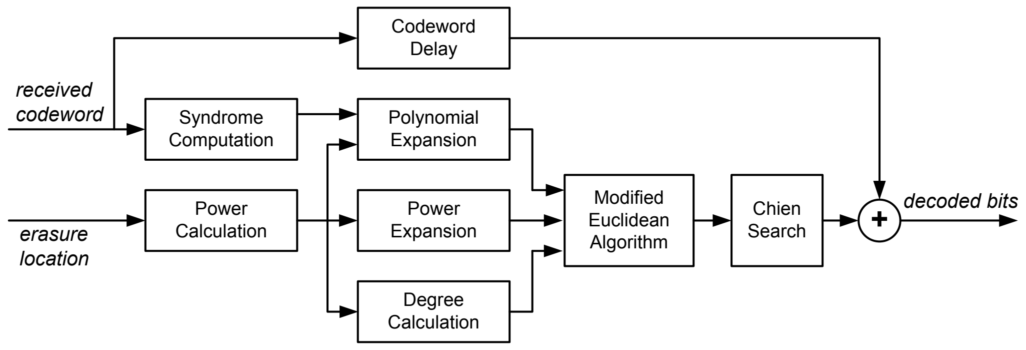


Figure 2.9: A Reed-Solomon Decoder.

2.5.2 Outer Code

A shortened Reed-Solomon RS(204, 188) code is used as an outer code by padding a RS (255, 239) code with 51 bytes of zeros. Reed-Solomon codes consist of non-binary symbols and therefore the correction of a single symbol results in the correction of multiple bits. Thus, Reed-Solomon codes are well suited for burst error correction. Reed-Solomon codes are linear block codes of length q-1 over the finite field GF(q) and are a subclass of q-ary BCH codes [15]. For a t-error correcting RS code with symbols from GF(q), the code parameters are:

- Block Length: $n = q - 1$
- Parity Symbols: $n - k = 2t$
- Dimension: $k = q - 1 - 2t$
- Minimum Distance: $d_{min} = 2t + 1$

The decoders of Reed-Solomon codes operate on syndromes derived from evaluation of the received word represented as a polynomial over GF(q) as shown in Figure 2.9. From the syndromes, an equation is derived, the solution to which gives an error locator and error magnitude polynomials. The Berlekamp-Massey [16] and Euclidean [17] algorithms are two efficient ways of solving the syndrom equation to determine the error location and error magnitude polynomials. The actual error locations are determined by finding the roots of the error-locator polynomial. A Chien search [18] provides a fast algorithm for determining the roots of a polynomial over a finite field. The error magnitudes can be evaluated using the Forney algorithm. Finally, error correction is performed by adding the error magnitudes to the received codeword.

The primitive polynomial used to define GF(2^8) is:

$$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \tag{2.5}$$

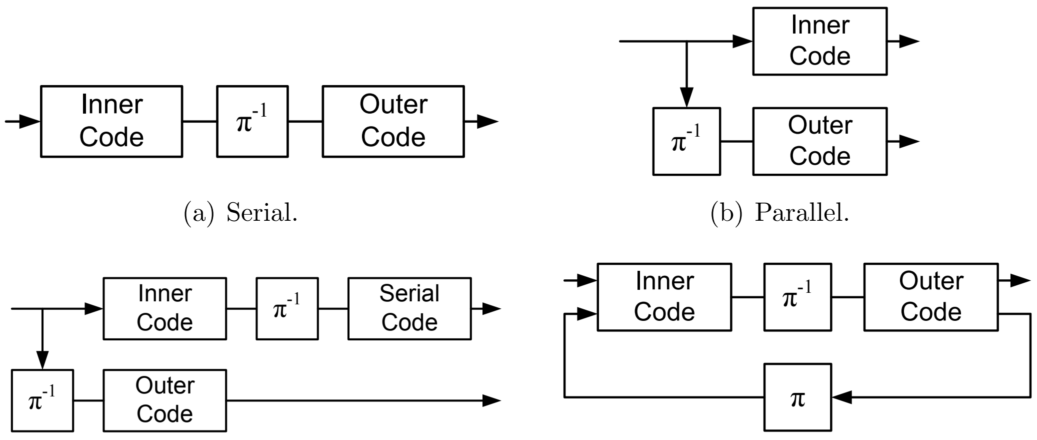
The shortened RS (204, 188) is generated by the following polynomial:

$$g(x) = (x - \alpha^0)(x - \alpha^1)(x - \alpha^2)\ldots(x - \alpha^{15}), \tag{2.6}$$

where $\alpha = 02_{HEX}$ [3]. The RS decoder is capable of correcting 8 consecutive bytes in error:

$$t = \lfloor\frac{n-k}{2}\rfloor = \lfloor\frac{204-188}{2}\rfloor = 8\ bytes \tag{2.7}$$

The implementation of the RS decoder in Figure 2.9 for an $(n,\ k)$ block code requires $(13n - 13k)$ registers, $(11n - 11k - 4)$ GF adders, $(n - k + 1)$ constant GF multipliers, $8 \times (n - k) - 1$ GF multipliers, in addition to memory required to store n input symbols: $256 \times 20\ bits = 5,120\ bits$.



(a) Serial. (b) Parallel.

(c) Hybrid. (d) Iterative.

Figure 2.10: Concatenated Code Configurations.

A concatenation of inner and outer codes can achieve lower decoding complexity than a single code of comparable performance. A concatenated code is constructed by combining inner and outer codes separated by an interleaver in a serial, parallel, hybrid or iterative configuration as shown in Figures 2.10(a)-2.10(d). An iterative, turbo-like [19] configuration is used in the generic OFDM receiver model in Figure 2.2 in which the extrinsic information is exchanged bewteen two Soft-Input Soft-Output

(SISO) decoders. A useful tool in the study and design of iterative decoding is the Extrinsic Information Transfer (EXIT) chart introduced in [20].

The bulk of embedded memory is used in the implementation of the Forward Error Correction modules. In particular, embedded memory is used in the implementation of the frequency and time de-interleaver followed by the de-puncturer, Viterbi survivor sequence memory [21], [22], convolutional byte interleavers and the implementation of RS decoding algorithms such as Berlekamp-Massey [16], Euclidean [17] or Koetter-Vardy [23], [24] algorithms. The embedded memory used to implement the FEC modules occupies 68% of the SoC core area, while the frequency-time de-interleaver memory occupies 62% of the SoC core area. Thus, the frequency-time de-interleaver presents the greatest opportunity for developing efficient memory repair strategies.

2.6 Frequency Time Deinterleaver

2.6.1 Overview

A high level block diagram of the frequency time deinterleaver (FTDI) is shown in Figure 2.11. The de-interleaver performs the reverse of the interleaver sequence of operations as detailed in the ISDB-T standard [3]. An interleaver changes the or-

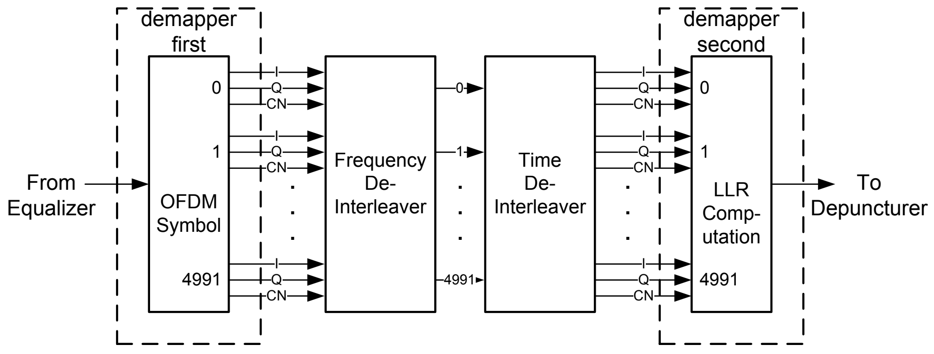


Figure 2.11: Frequency Time De-interleaver Overview.

der of symbols before transmission to convert long burst errors into shorter burst or random errors that can be more easily corrected by the error correction logic [25], [26].

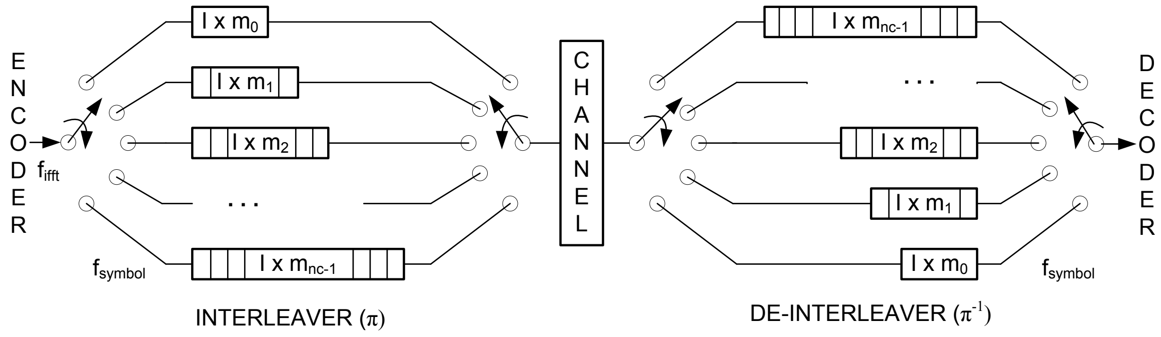


Figure 2.12: Convolutional Interleaver (Tx) - Deinterleaver (Rx) Pair.

Interleavers are characterized by an encoding delay and storage capacity and can take on a *convolutional* or a *block* form [25]. A *block interleaver* of degree m formats the input symbol vector of length $m \times n$ into a rectangular array of m rows and n columns such that a consecutive pair of symbols at the input appears m symbols apart at the output. The rectangular array is filled row-by-row and the interleaver output is read out column-by-column. As a result, an $(n,\ k)$ block code that can handle burst errors of length $b < \lfloor \frac{1}{2}(n-k) \rfloor$ when combined with an interleaver of degree m creates an interleaved $(mn,\ mk)$ block code that can handle bursts of length $m \times b$ [27]. A *convolutional interleaver* of degree m consists of m shift registers with the i^{th} register having a storage capacity of $(i-1) \times d$, for a fixed integer d and $i = 1,\ 2,\ 3, \ldots, m$. Each new input symbol is written to a new shift register, while the oldest symbol in the shift register is shifted to the output. Convolutional interleavers reduce the required storage space to approximately half of block interleavers but require a higher degree of clock synchronization. The synchronization period can be reduced with the use of helical interleavers [28]. Appendix B provides pseudocode describing the operation of the various interleavers and their corresponding de-interleavers. Table 2.2 summarizes the delays and storage capacities for the three types of interleavers. Convolutional interleaving is used to reduce the total transmission and reception time as well as decrease the required memory storage.

Figure 2.13 introduces an OFDM frame structure used in the ISDB-T standard [3]. The ISDB-T features three transmission modes (1, 2, and 3) over a 5.575 MHz channel bandwidth divided into 13 OFDM segments with different transmission parameters for each mode as summarized in Appendix A. Internally, the data is divided into

Type	Storage Capacity	Delay
Block	$m \times n$	$m \times (n-1)$
Convolutional	$(m/2) \times (m-1) \times d$	$m \times (m-1) \times d$
Helical	$m \times n$	$m \times n\lceil (m+1)/n \rceil$

Table 2.2: Interleaver Characteristics.

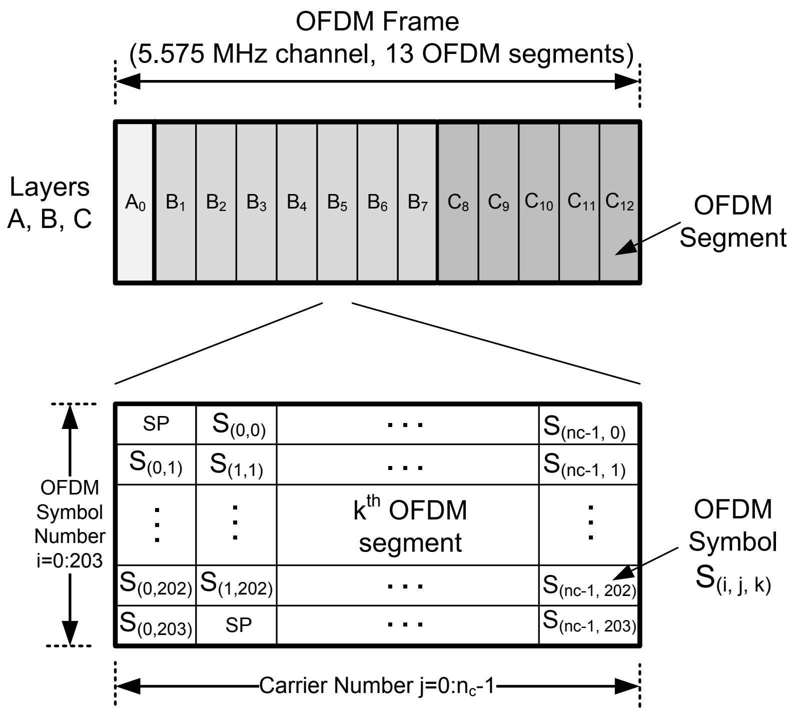

OFDM Frame
(5.575 MHz channel, 13 OFDM segments)
Layers A, B, C
A_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7 C_8 C_9 C_{10} C_{11} C_{12}
OFDM Segment
SP
$S_{(0,0)}$
$S_{(nc-1, 0)}$
$S_{(0,1)}$
$S_{(1,1)}$
$S_{(nc-1, 1)}$
kth OFDM segment
$S_{(0,202)}$
$S_{(1,202)}$
$S_{(nc-1, 202)}$
$S_{(0,203)}$
SP
$S_{(nc-1, 203)}$
OFDM Symbol Number i=0:203
OFDM Symbol $S_{(i, j, k)}$
Carrier Number j=0:n_c-1

Figure 2.13: OFDM Frame Structure [3].

three hierarchical layers (A, B, and C) that can have different information bit rates to accommodate video, audio, and data transmission [29]. Each OFDM segment consists of $S(i,\ j,\ k)$ OFDM symbols identified by the symbol number i ($\{i \in \mathbb{Z} \mid i \in [0,\ 203]\}$), the carrier number j ($\{j \in \mathbb{Z} \mid j \in [0,\ n_c - 1]\}$), where n_c is the number of carriers equal to 96, 192, and 384 for Modes 1, 2, and 3, respectively; and a segment number k ($\{k \in \mathbb{Z} \mid k \in [0,\ 12]\}$).

2.6.2 Frequency Deinterleaver

A block diagram of the frequency deinterleaver is shown in Figure 2.14. The de-mux block routes OFDM symbols based on the segment number into partial-reception[1], differential modulation (DQPSK) and coherent modulation (QPSK, 16-QAM, 64-QAM) streams.

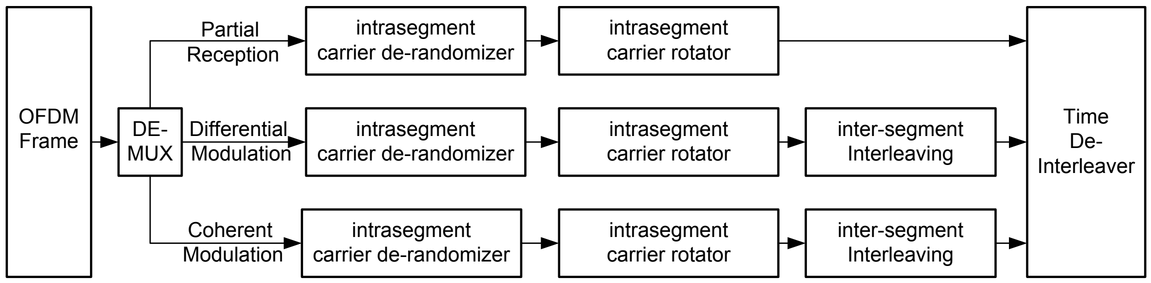


Figure 2.14: Frequency Deinterleaver Block Diagram.

The *intra-segment* frequency de-interleaving consists of a carrier de-randomizer followed by a carrier rotator. The carrier de-randomizer modifies the carrier number j, while the carrier rotator changes the symbol number i of an OFDM symbol $S_{(i,\ j,\ k)}$ within an OFDM segment k. Both are intended to eliminate periodicity in carrier arrangement in order to prevent burst errors of a specific segment's carriers, which may occur if a reciprocal of the carrier arrangement period matches the frequency-selective fading of the channel. The *inter-segment* de-interleaving modifies the segment number k of an $S_{(i,\ j,\ k)}$ OFDM symbol. Inter-segment de-interleaving is conducted across segment and layer boundaries to maximize the benefit of frequency interleaving.

[1] Reception of only one OFDM segment at the center of a group of segments for low SNR.

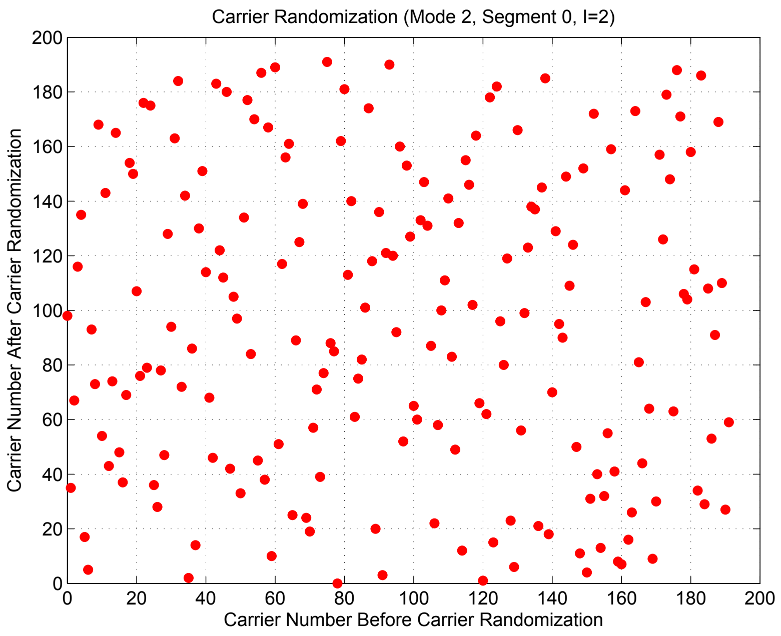


Figure 2.15: Carrier Randomization for an OFDM Symbol $S_{(i,\ j,\ k)}$ in Mode 2.

More specifically the permutation functions of the three blocks are given by:

The *carrier de-randomizer* changes the carrier number j according to randomization tables defined in the standard [3]. Figure 2.15 shows a sample mapping of the k^{th} OFDM segment for transmission Mode 2.

The *carrier rotator* permutes the symbol number i within an OFDM segment k as follows:

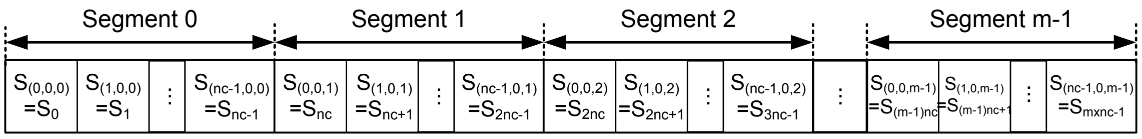
$$S_{(i,\ j,\ k)} \Rightarrow S_{(k+i\ mod\ n_c,\ j,\ k)} \tag{2.8}$$

where $S_{(i,\ j,\ k)}$ is an OFDM symbol number i ($\{i \in \mathbb{Z} \mid i \in [0,\ 203]\}$), transmitted by carrier j ($\{j \in \mathbb{Z} \mid j \in [0,\ n_c - 1]\}$), with the carrier number n_c equal to 96, 192, and 384 for Modes 1, 2, and 3, respectively; in the OFDM segment k ($\{k \in \mathbb{Z} \mid k \in [0,\ 12]\}$).

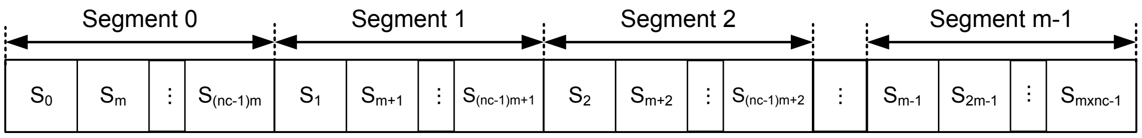
The *inter-segment* frequency de-interleaver permutes the symbol number i between the OFDM segments as follows:

$$S_{(i,\ j,\ k)} \Rightarrow S_{(k\ mod\ m,\ j,\ i\ mod\ n_c)} \tag{2.9}$$

where m is the degree of the inter-segment de-interleaver. Figures 2.16(a) and 2.16(b) show the input data stream of $m \times n_c$ OFDM symbols before and after the inter-segment frequency de-interleaver.



(a) Arrangement of OFDM Symbols $S_{(i,\ j,\ k)}$ Before De-Interleaving.



(b) Arrangement of OFDM Symbols $S_{(i,\ j,\ k)}$ After De-Interleaving.

Figure 2.16: Inter-segment Frequency De-Interleaver.

2.6.3 Time Deinterleaver

The data sub-carriers are deinterleaved in time between the 13 OFDM segments as well as within each segment. *Inter-segment* time deinterleaving is performed by distributing the data across OFDM segments as shown in Figure 2.17. *Intra-segment* time deinterleaving is carried out via convolutional deinterleavers introduced in Figure 2.12.

Layer Name	Number of Segments	Modulation Scheme	Code Rate, R	Deinterleaver Length, I
A	1	QPSK	2/3	4
B	12	64-QAM	3/4	2
C	0	16-QAM	3/4	1

Table 2.3: ISDB-T Mode 3: Typical Transmission Parameters.

Each OFDM segment contains n_c data sub-carriers for a maximum of $13 \times 384 = 4992$. The length of the delay buffers used for intra-segment interleaving is given by the interleaver length I multiplied by the delay function m_i, where

$$m_i = (i \times 5)\ mod\ 96 \tag{2.10}$$

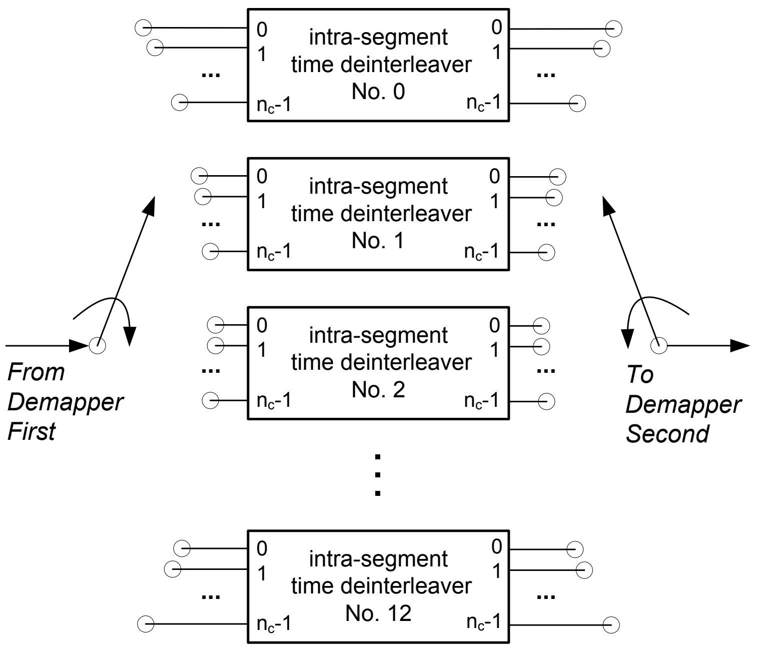


Figure 2.17: Time Deinterleaver Block Diagram.

and i ($\{i \in \mathbb{Z} \mid i \in [0,\ n_c - 1]\}$) with n_c equal to 96, 182, and 384 for Modes 1, 2, and 3, respectively. For example, in Mode 3 with typical transmission parameters for layer B according to Table 2.3, the worst-case time deinterleaver delay is

$$\max_i[I \times m_i] = \max_i[2 \times (i \times 5)\ mod\ 96] = 190\ symbols \tag{2.11}$$

Based on Figure 2.13, the number of information bits in one OFDM frame for typical transmission parameters in Table 2.3 is:

$$203 \times 384\ \frac{sym.}{seg.} \times (1\ seg. \times \frac{2\ bits}{QPSK\ sym.} + 12\ seg. \times \frac{6\ bits}{64QAM\ sym.}) = 5,768,448\ bits$$

Assuming a 20-bit de-interleaver word with 8-bit I, 8-bit Q and 4-bit CN, the de-interleaver storage capacity required for one OFDM frame is:

$$13\ seg. \times I \times \sum_{i=0}^{n_c-1} (i \times\ 5\ mod\ 96) \times 20\ \frac{bits}{seg.} = 18,969,600\ bits.$$

2.6.4 Deinterleaver Architecture

Frequency and time de-interleaving is implemented by address generation logic for one OFDM frame stored in the deinterleaver SRAM memory. To carry out frequency de-interleaving, a combination of look-up tables and computational logic are used to implement the carrier de-randomizer followed by the carrier rotator equation (2.9):

$$S_{(i,\ j,\ k)} \Rightarrow S_{(k+i\ mod\ n_c,\ j,\ k)}$$

Inter-segment frequency de-interleaving is accomplished by rotating i and k indices and implementing the modulo operation [30]. Time de-interleaving is carried out according to Figure 2.17. The sequence of frequency and time de-interleaving operations is controlled by a finite state machine.

To support multiple transmission modes, the de-interleaver requires approximately 19.4-Mbits of memory. In addition, the high density of integration of SRAM memory increases the likelihood of fabrication errors. On the other hand, the de-interleaver operations precede forward error correction, and therefore a higher degree of fault tolerance is expected because bit errors introduced by faulty memory cells can be corrected by the downstream error correction modules. To summarize, the following two observations can be made about the de-interleaver memory:

- Large capacity memories occupy large area and as a result have a higher probability of bit errors.
- Memories that store data prior to filtering and forward error correction have higher error tolerance.

The de-interleaver memory is partitioned into two rows of four RAM tiles each as shown in Figure 2.18. Type I RAM tiles consist of four 16K×40 memory instances. Type II RAM tiles contain two instances of 16K×40 and one instance of 12.5K×40. The total capacity of the de-interleaver memory illustrated in Figure 2.18 is therefore $28 \times 16K \times 40 + 2 \times 12.5K \times 40 = 18,350,080 + 1,024,000 = 19,374,080$ bits.

The architecture of the Type I RAM tile is shown in Figure 2.19. The four memory instances are folded internally into 1K rows and 16×40-bit wide columns as shown

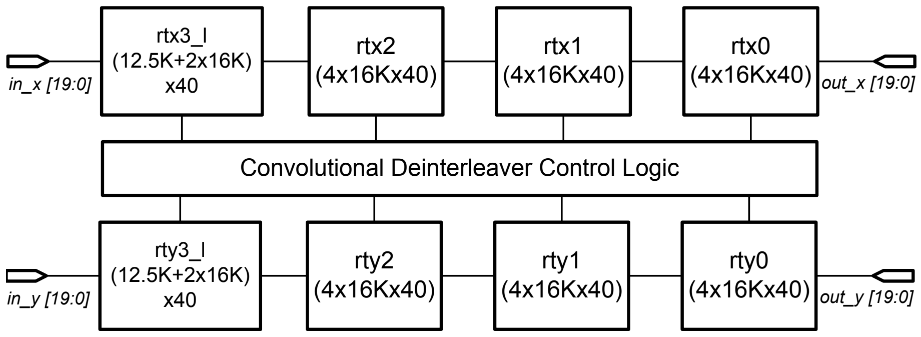


Figure 2.18: A Conceptual FTDI SRAM Architecture.

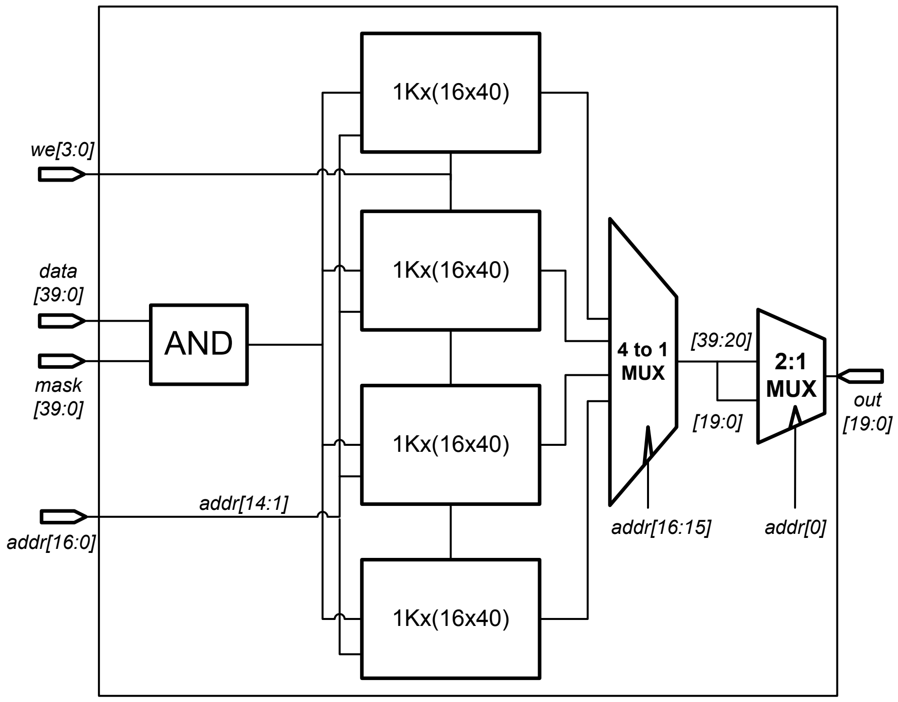


Figure 2.19: FTDI Type I SRAM Tile Architecture.

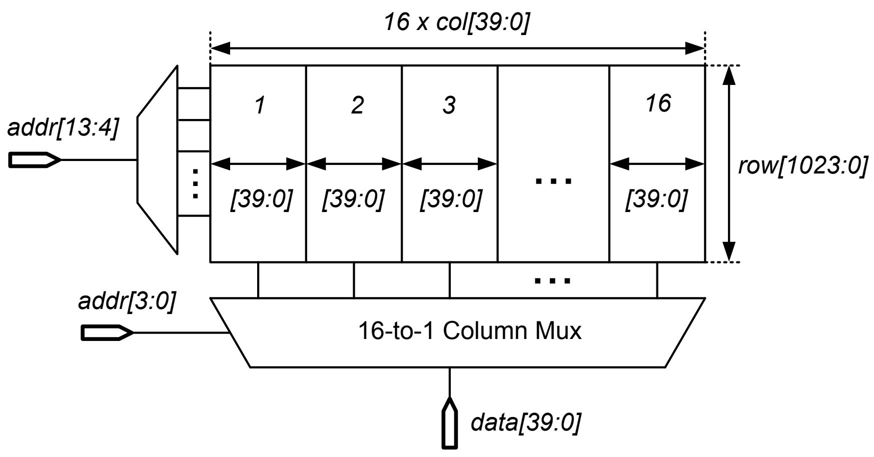


Figure 2.20: FTDI Type I SRAM Memory $1K \times (16 \times 40)$ Architecture.

in Figure 2.20.

The de-interleaver memory instances are implemented as single port SRAM memories with 1 clock cycle latency. At every clock cycle, only one memory instance for read and one for write are enabled to reduce power consumption [31]. Typical values are provided in Example 2.1 below:

Example 2.1: For a $1K \times (16 \times 40)$ instance, the power dissipation is 28.42 $\mu W/MHz$ for read and 33.93 $\mu W/MHz$ for write for TT/1.2V/25°C. In the worst-case of 4992 data sub-carriers in Mode 3, the effective symbol length is 1008 μs. Therefore, there are 4992 read and 4992 write operations with a symbol rate of approximately 1 kHz. The dynamic power for read and write operations can be computed as follows:

$$P_{d,\ read} = 28.42 \frac{\mu W}{MHz} \times \frac{4992}{1008\ \mu s} = 140.7\ \mu W \tag{2.12}$$

$$P_{d,\ write} = 33.98 \frac{\mu W}{MHz} \times \frac{4992}{1008\ \mu s} = 168.3\ \mu W \tag{2.13}$$

Therefore, the total dynamic power is $P_d = 140.7\ \mu W + 168.3\ \mu W \approx 0.3\ mW$ at TT/1.2V/25°C.

2.7 Summary

The ISDB-T receiver consists of three main processing modules: stream, block and FEC modules. The stream modules perform synchronization and mode estimation functions. The block modules compute the FFT, construct channel delay profile and perform channel equalization. The FEC modules carry out frequency and time de-interleaving as well as iterative soft-output Viterbi and Reed-Solomon decoding. The embedded memory used to implement the frequency and time de-interleaver occupies 62% of the total SoC core area and thus presents the greatest opportunity for developing efficient memory repair strategies.

The frequency-time de-interleaver imposes large memory requirements in order to support de-interleaving operations for all transmission modes. The 19.4 Mbit de-interleaver memory is the single largest embedded memory on the SoC. Implemented as high density SRAM, the de-interleaver memory block is susceptible to fabrication faults and therefore area efficient memory repair strategies must be considered. Despite its large size, which increases the likelihood of SRAM faults, the de-interleaver memory is located before the forward error correction modules. Thus, greater error tolerance to fabrication faults is expected due to iterative error correction of the soft-output Viterbi and Reed-Solomon decoders. Additional fault tolerance is offered by the proposed repair technique.

To develop efficient memory repair techniques, Chapter 3 provides an introduction to yield and fault models, presents a number of repair strategies as well as simulation results of the proposed repair technique.

3 Fault-Tolerant Techniques

The greatest of all faults is to be conscious of none.

THOMAS CARLYLE

3.1 Introduction

Fault-tolerant memory repair techniques can be classified according to their use of redundant resources. Repair techniques with redundancy have an upfront cost of additional area and require an optimum allocation of the limited number of spare resources. Repair techniques without redundancy, to be a viable alternative, must introduce lower area overhead and comparable repair performance. Repair techniques without redundancy include syndrome error detection, correction and storage of error information [32], de-clustering of memory failures [33], and permuting memory cells so as to change uncorrectable memory words into ones that can be corrected [34].

In order to develop effective fault-tolerant repair techniques it is important to understand the nature of memory faults and to quantify their effect on yield. This chapter develops the theoretical background for yield and fault models based on prior work, presents a number of repair strategies as well as simulation results of the proposed repair technique.

3.2 Yield Model

Yield can be defined as a fraction of manufactured chips that pass a set of stringent production tests. Yield can be divided into *two* classes: *gross yield* and *random fault yield.*

The *gross yield* refers to gross defects that cause parts of a wafer or the entire wafer to have non-functional chips. Examples of gross defects that occur during m process steps required to fabricated the chip include errors in the fabrication process such as over or under exposure of photoresist, over or under etching, mask misalignment in the process of photolithography; incorrect doping profile, temperature or time settings during ion implantation and high device parameter variation. Early warning systems can be employed to prevent manufacturing products with gross defects such as defect monitors and test circuits fabricated on-chip or in kerfs between the dies [35]. The gross yield can be modeled as:

$$Y_{gross} = \prod_{i=1}^{m} Y_{0i} \tag{3.1}$$

where $\{Y_{0i} \in \mathbb{R} \mid Y_{0i} \in [0,\ 1]\}$ represents the impact of gross defects in the process step i on the gross yield Y_{gross}.

The *random fault yield* is based on statistical models of random factors that affect chip yield. These include gate oxide pin holes, particle contamination, overlay faults, process-induced shorts and opens, layer thickness and critical dimension variations, in addition to fault clustering induced by dust particles at exposed edges of the silicon wafer and other random effects as shown in Figure 3.1. Random faults can be modeled in terms of the *average* number of faults λ_j of type j with a particular *statistical distribution* [35].

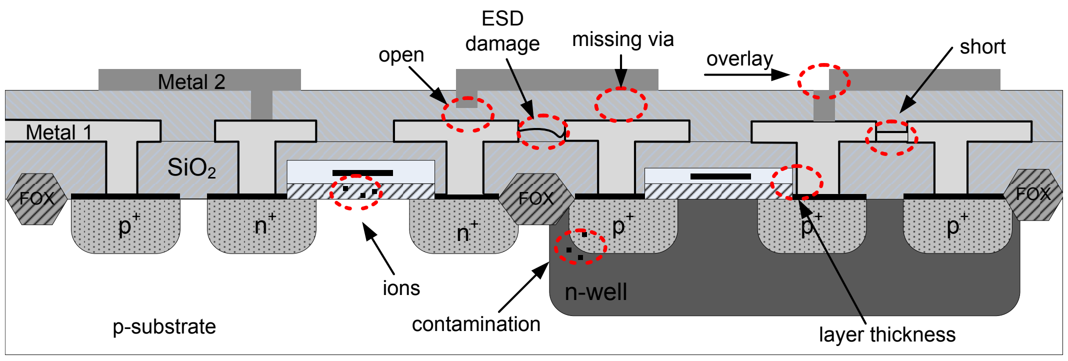


Figure 3.1: Fabrication Defects in a CMOS Process.

The average number of faults λ_j can be approximated by a fault density D_j over a critical area A_j:

$$\lambda_j = A_j \times D_j \tag{3.2}$$

where the critical area A_j is defined as the area with a non-zero fault density D_j. The fault densities are based on empirical analysis of a large number of fabricated ICs. The fault densities are typically reported by the foundry and used internally for process improvement. Equation (3.2) can be generalized as a function of space and time over a critical volume:

$$\lambda_j(t) = \int_{V_j} V_j(x, y, z)\ D_j(x, y, z, t)\ dV \tag{3.3}$$

provided that accurate fault models based on exhaustive pathological data are available. The fault densities vary across fault types as well as technology nodes and in general improve over time as the process matures.

To model the *statistical distribution* of faults during the manufacturing process, consider the probability generating function $G(s; t)$:

$$G(s; t) = \sum_{x=0}^{\infty} p(x, t)\ s^x \tag{3.4}$$

where s is a dummy real variable. Thus, the distribution $p(x, t)$ of x faults on a chip at time t can be expressed as:

$$p(x, t) = \frac{1}{x!} \frac{\partial^x G(s; t)}{\partial s^x}\bigg|_{s=0} \tag{3.5}$$

The probability generating function (3.4) can be shown [35] to satisfy the differential equation:

$$\frac{\partial}{\partial t} G(s; t) = (s - 1) \sum_{x=0}^{\infty} p(x, t) f(x, t) s^x \tag{3.6}$$

where $f(x, t)\ dt$ is defined as a probability that a fault will occur during the interval $[t,\ t + dt]$ on a chip that already contains x faults. If we assume that the probability of a fault occurring on a chip is independent of the number of existing faults, i.e.

$f(x,t) = f(t)$, then (3.6) can be re-written as:

$$\frac{\partial G(s;t)}{\partial t} = (s-1)f(t)G(s;t) \tag{3.7}$$

The first order differential equation (3.7) can be solved for $G(s;\ t)$:

$$G(s;t) = e^{\lambda(s-1)} \tag{3.8}$$

where

$$\lambda = \int_0^t f(\tau)d\tau \tag{3.9}$$

Equation (3.8) is a generator function for a Poisson probability distribution $p_X(k)$ with $E[X] = \lambda$. Recall that the probability mass function (PMF) for a Poisson random variable is:

$$p_X(k) = e^{-\lambda}\ \frac{\lambda^k}{k!}, \quad k = 0, 1, 2, ... \tag{3.10}$$

A Poisson distribution arises in applications when random events occur in a certain period of time or in a certain region in space. In this case, the space region is the critical memory area affected by k fabrication faults.

The random fault yield Y_{rnd} is defined as the probability of having zero faults on a chip. If we assume that $f(x,t) = f(t)$, Y_{rnd} can be expressed as:

$$Y_{rnd} = p(0,t) = G(0;t) = e^{-\lambda} \tag{3.11}$$

It can be advantageous to break up the integral in (3.9) into m individual process steps, where the average number of faults collected during the process step i is:

$$\lambda_i = \int_{t_{i-1}}^{t_i} f(\tau)\ d\tau \tag{3.12}$$

Then, the random fault yield Y_{rnd} in (3.11) can be expressed as:

$$Y_{rnd} = \exp(-\sum_{i=1}^{m}\ \lambda_i) = \prod_{i=1}^{m}\ e^{-\lambda_i} = \prod_{i=1}^{m}\ Y_{rnd,\ i} \tag{3.13}$$

where $Y_{rnd,\ i}$ is the random fault yield corresponding to the process step i. Given a large number of process steps m and a small probability of finding k random faults

$p_{X_i}(k)$ in the process step i, the Poisson random variable can be used to approximate a binomial distribution, provided $\lambda_i = m \times p_{X_i}(k)$ is constant.

However, experimental data shows that the mean and the variance of the fault distribution in the process step i do not match λ_i, as predicted by the Poisson model [36]. This is caused by a non-constant distribution of the average number of faults across the wafer, i.e. the value λ_i varies from chip to chip. Therefore, a mixed Poisson distribution is used with gamma distribution as a compounder or a mixing function: $Y_{rnd} \sim Pois(\Lambda)$ and $\Lambda \sim Gamma(\alpha,\ \lambda/\alpha)$. The result is a generalized negative binomial distribution:

$$p_{X_i}(k) = \frac{\Gamma(\alpha_i + k)}{k!\Gamma(\alpha_i)} \frac{(\lambda_i/\alpha_i)^k}{(1+\lambda_i/\alpha_i)^{k+\alpha_i}} \tag{3.14}$$

with $E[X_i] = \lambda_i$ and $VAR[X_i] = \lambda_i(1 + \lambda_i/\alpha_i)$, where the variation in λ_i is modeled by a clustering parameter α_i. The corresponding random yield based on the negative binomial distribution in (3.14):

$$Y_{rnd,\ i} = p_{X_i}(0) = (1 + \frac{\lambda_i}{\alpha_i})^{-\alpha_i} \tag{3.15}$$

Low values of α_i are needed to model a high degree of fault clustering. As α_i approaches infinity, (3.15) will approach $e^{-\lambda_i}$, which is the component yield of the random fault model in (3.11).

Assuming that in each process step i, random faults are independent and identically distributed (iid), Y_{rnd} can be expressed as:

$$Y_{rnd} = \prod_{i=1}^{m} p_{X_i}(0) = \prod_{i=1}^{m} (1 + \frac{\lambda_i}{\alpha_i})^{-\alpha_i} \tag{3.16}$$

Combining the gross yield in (3.1) with the random fault yield in (3.16), the total chip yield is given by:

$$Y = \prod_{i=1}^{m} Y_{0i} \prod_{i=1}^{m} (1 + \frac{\lambda_i}{\alpha_i})^{-\alpha_i} \tag{3.17}$$

Embedded memories have a much denser layout in comparison to core logic and therefore higher fault densities. The fault density of embedded memories can be expressed as a multiple c_j of logic fault density D_j, where c_j may typically vary from 2 to 4. Furthermore, it can be assumed that memory yield is directly related to chip yield, i.e. an unrepairable memory will result in the entire chip being discarded during production testing. As a result, the following random fault yield model based on a Poisson distribution is used by the target foundry:

$$Y_{rnd} = \prod_{i=1}^{m} Y_{rnd,\ i} = \prod_{i=1}^{m} (1 + A_i D_i (1 + c_i R_i))^{-1} \tag{3.18}$$

where A_i is the critical area in $[mm^2]$, D_i is the fault density per layer $[1/mm^2/layer]$, R_i is the ratio of critical memory area to core area, c_i is a parameter accounting for higher density (lower yield) of an embedded SRAM memory.

Example 3.1: In the yield equation (3.18), as the fault density approaches zero, the overall yield tends to 100%. In the limiting case of $D_i = 0$:

$$Y_{rnd} = \prod_{i=1}^{m} Y_{rnd,\ i} = \prod_{i=1}^{m} (1)^{-1} = 1 \tag{3.19}$$

Furthermore, if the chip does not contain embedded memory, the density factor $c_i = 0$ and the yield equation (3.18) reduces to:

$$Y_{rnd} = \prod_{i=1}^{m} Y_{rnd,\ i} = \prod_{i=1}^{m} (1 + A_i D_i)^{-1} \tag{3.20}$$

Hence, the factor $(1 + c_i R_i)$ accounts for a higher fault density (lower yield) of ICs with embedded memory.

Example 3.2: Based on a fault analysis of a statistically significant number of SRAM memory arrays in 65-nm CMOS technology, an estimated SRAM fault density was found to be $1.0075 \times 10^{-4}\ mm^{-2}$. If the frequency-time de-interleaver (FTDI) SRAM memory is implemented in the same process, we can find the expected number of faults in the FTDI memory. Assuming an equal fault density $D_j = 1.0075 \times 10^{-4}\ mm^{-2}$ for each of the $m = 21$ memory mask layers, a critical area $A_{j,\ FTDI}$, the total chip area A_j, the ratio of the critical memory area to core area R_j and a density factor $c_j = 2$, the expected number of faults per memory mask layer i is:

$$\begin{aligned}\lambda_i &= A_j D_j(c_j R_j) = A_j D_j(2 \times R_j) = A_j D_j(2 \times A_{j,\ FTDI}/A_j) \\ &= 2 \times D_j A_{j,\ FTDI} = 2 \times D_j(28 \times A_{16K\times 40} + 2 \times A_{12.5K\times 40}) \\ &= 2 \times (1.0075 \times 10^{-4}\ mm^{-2}) \times (12.1749\ mm^2 + 0.7200\ mm^2) \\ &= 0.0026\end{aligned}$$

$$\lambda = \sum_{i=1}^{m=21} \lambda_i = 0.0546$$

Therefore, if we assume a uniform fault density of $1.0075 \times 10^{-4}\ mm^{-2}$ for all 21 memory mask layers with the critical area of each layer equal to the dimensions of the memory, then $\lceil 5.46 \rceil = 6$ out of every 100 chips are expected to have a single bit error in the FTDI memory.

Using the approach in Example 3.2, the expected number of faults λ was calculated for every embedded memory in the OFDM receiver. The top five memories ranked by area are listed in Table 3.1.

Functional Block	Instance Size	Number of Instances	Mbits	Area, mm^2	Expected Number of Faults λ
FTDI	16384×40	28	18350080	12.1749	0.0515181
FTDI	12800×40	2	1024000	0.7200	0.0030468
DPN	6640×60	4	2124800	0.7970	0.0067447
FFT	8192×26	2	1703936	0.3097	0.0052426
CSI	16384×26	1	1703936	0.2932	0.0049626
ICI	8192×20	1	655360	0.1228	0.0020786

Table 3.1: Expected Number of Faults (λ) in the Five Largest Embedded Memory Blocks.

The probability of finding k memory faults on a chip can be computed from the Poisson distribution in (3.10) or the Negative Binomial distribution in (3.14) as shown in Figure 3.2 with $\lambda = 0.0546$ corresponding to the average number of faults in the de-interleaver memory as calculated in Example 3.2.

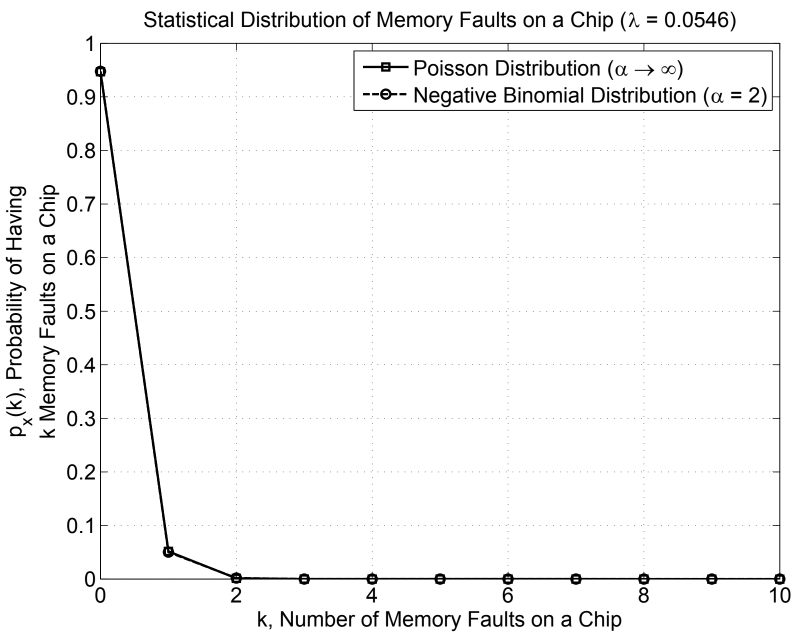


Figure 3.2: Plot of $p_X(k)$ vs k for Poisson and Negative Binomial Distributions with $\lambda = 0.0546$.

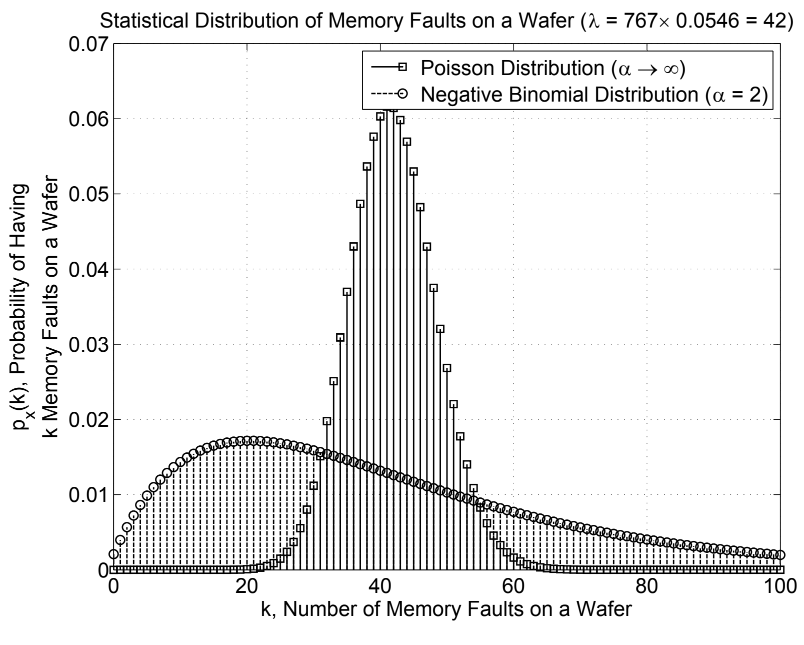


Figure 3.3: Plot of $p_X(k)$ vs k for Poisson and Negative Binomial Distributions with $\lambda = 42$.

The probability of finding k memory faults on a chip can be extended to a silicon wafer. Assuming L chips per wafer with each chip l having an iid random fault distribution $X_l \sim Pois(\lambda_l)$, the expected number of faults per wafer is the expected value of the sum of L Poisson random variables:

$$\lambda_{wafer} = E\ [\ \sum_{l=1}^{L} X_l\ ] = E\ [\ Pois(\sum_{l=1}^{L} \lambda_l)\ ] = L \times \lambda_l \tag{3.21}$$

Example 3.3: For a $250 - mm$ wafer and an estimated die area of $8\ mm \times\ 8\ mm$, the number of chips per wafer L is approximately equal to:

$$L \approx \frac{\pi(250\ mm/2)^2}{8\ mm \times 8\ mm} \approx 767 \tag{3.22}$$

Therefore, the expected number of faults per wafer is:

$$\lambda_{wafer} = E\ [\ \sum_{l=1}^{L=767} X_l\ ] = E\ [\ Pois(\lambda_l \times L)\ ] \approx 42. \tag{3.23}$$

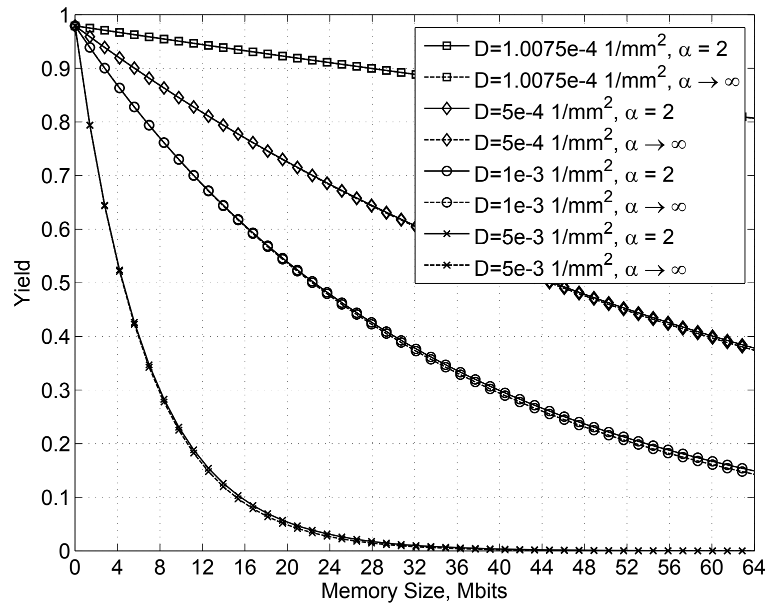
The PMF of $Pois(\lambda_l \times L)$ is plotted in Figure 3.3.

Poisson, binomial and negative binomial distributions are fundamental to modeling the random fault yield. The generalized negative binomial yield model in (3.16):

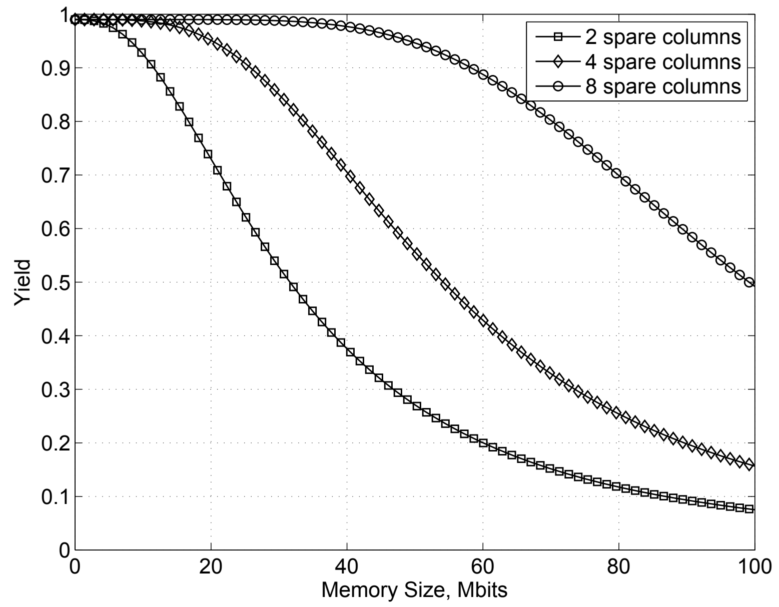
$$Y = \prod_{i=1}^{m} Y_{0i} \prod_{i=1}^{m} (1 + \frac{\lambda_i}{\alpha_i})^{-\alpha_i}$$

reduces to the Poisson model when $\alpha_i \to \infty$ and to the binomial model when $\alpha_i = -N$. Figure 3.4(a) shows a yield vs. de-interleaver memory size plot based on (3.16) with $Y_{0i} = 0.999\ \forall i$, $D_j = 1.0075 \times 10^{-4}\ mm^{-2}\ \forall j$, clustering coefficient $\alpha_i = \{2, \infty\}\ \forall i$, and $m = 21$ memory mask layers.

Yield improvement can be achieved by including redundancy such as spare memory rows and columns at the cost of increased chip area and therefore an increase in cost per chip and a potential decrease in the number of chips per wafer. A chip yield with redundancy Y_{red} consisting of N_B identical blocks and N_R redundant blocks is the



(a) Yield as a Function of Memory Size and Fault Density: Negative Binomial Model, $Y_{oi} = 0.999\ \forall i$, Fault Density $D_j\ \forall j$, Clustering Coefficient $\alpha_i = \{2,\ \infty\}\ \forall i$, $m = 21$.



(b) Yield as a Function of Memory Size and Redundancy: Negative Binomial Model, $Y_{oi} = 0.99\ \forall i$, Fault Density $D_j = 1.0075 \times 10^{-4} mm^{-2}\ \forall j$, Clustering Coefficient $\alpha_i = 2\ \forall i$.

Figure 3.4: Yield Plot.

probability that at least $N_B - N_R$ blocks are fault free [37]:

$$Y_{red} = \sum_{k=N_B-N_R}^{N_B} p_k = \sum_{k=N_B-N_R}^{N_B} \binom{N_B}{k} (e^{-\lambda_i})^k (1 - e^{-\lambda_i})^{N_B-k} \tag{3.24}$$

where a Poisson model of independent and identically distributed faults is assumed. To account for fault clustering, the expression in (3.24) can be inverted to find an equivalent λ_{red}, that can then be substituted into the negative binomial model in (3.16). Figure 3.4(b) shows an improvement in yield due to redundancy with $N_B = 16 \times 40 = 640$, $N_R = \{2,\ 4,\ 8\}$, $Y_{0i} = 0.99\ \forall i$, $D_j = 1.0075 \times 10^{-4}\ mm^{-2}\ \forall j$ and a clustering coefficient $\alpha_i = 2\ \forall i$.

An expression for yield probability with redundancy is further developed in [37] and the general case requires a model that specifies fault distribution for any sub-area of the chip as well as the correlation of faults between the sub-areas. When a closed form yield expression cannot be found for complex redundancy schemes with clustered faults, Monte Carlo simulation may be used, in which faults are introduced on a wafer according to a statistical fault distribution and the percentage of functional chips is calculated. Alternatively, a simpler yield model can be derived based on a Poisson distribution and adjusted to take clustering into account using an appropriate mixing function.

3.3 Fault Model

To develop an accurate fault model, it is important to consider both top-level and cell-level views of the embedded memory. Figure 3.5 shows a conceptual top-level architecture of a $2^X \times 2^Y$ memory. It consists of three main functional blocks: *the memory array*, *the address decoder*, and *the read / write circuits.*

The memory array matrix has $M = 2^X$ rows and $N = 2^Y$ columns with a total capacity of 2^{X+Y} bits. A memory cell is accessed by activating one of the 2^X rows or word lines (WLs) and communicating the data via the 2^Y columns or bit lines (BLs). *The row address decoder* selects one of the 2^X word lines according to the X-bit row address by raising the line voltage. During a *read operation*, the information stored

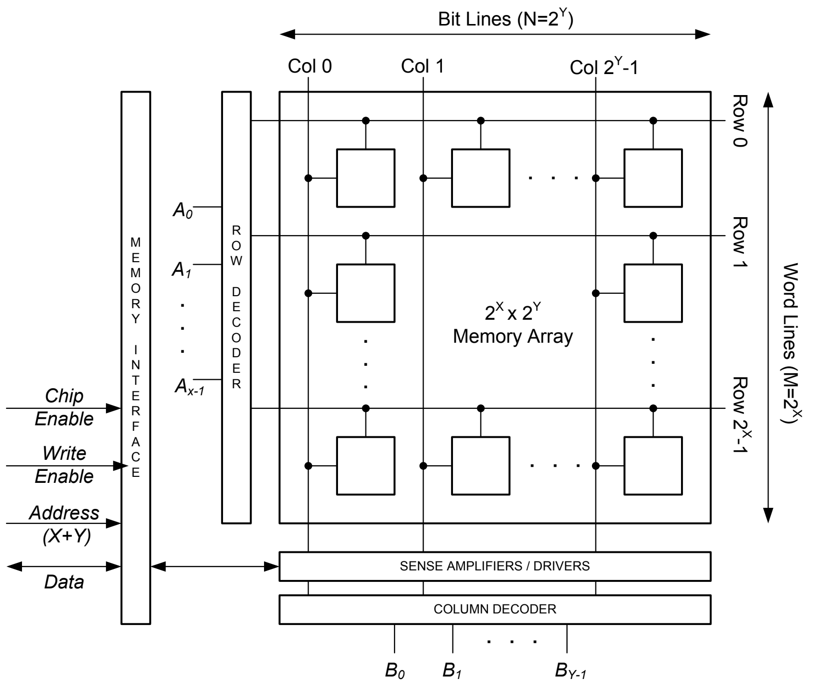


Figure 3.5: Generic Memory Architecture.

in memory cells that are connected to the selected word line is transferred to the respective bit lines. A small voltage difference characteristic of a logic “1” is amplified by the sense amplifier to a full-swing digital signal. *The column decoder* then selects the bit of the particular column whose Y-bit address is applied at its input. During a *write operation* the input bit to be stored is applied to the data line. The target cell is selected by a combination of its row and column addresses, while the sense amplifier acts as a *driver* forcing the contents of the selected cell to the input data. The time it takes to complete a memory operation is referred to as the *memory access time.*

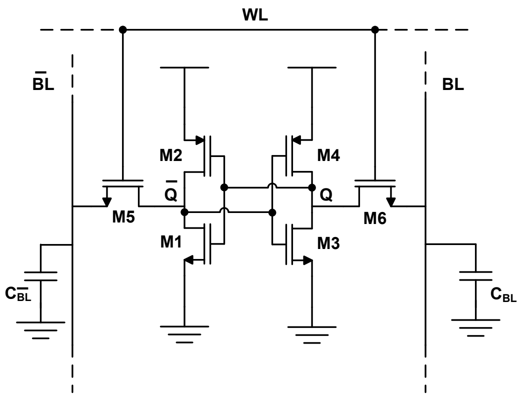
The memory access time increases with memory size due to higher parasitic capacitance added to WLs and BLs by the additional SRAM cells. The bit line delay can be modeled as a distributed RC ladder network consisting of $K = 2^X$ stages with a total Elmore [38] delay τ_d:

$$\tau_d = \sum_{j=1}^{K} C_j \sum_{k=1}^{j} R_k \qquad (3.25)$$

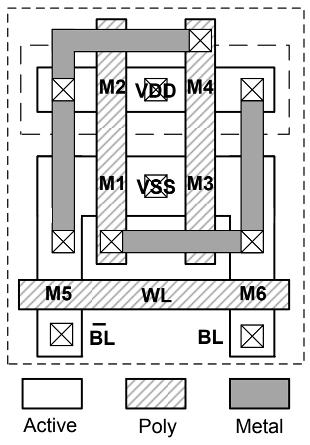
where $C_j = C_{gs} + C_{gb}$ is the parasitic capacitance contributed by a single SRAM cell and $R_k = R_{\square} \times l_k$ is the resistance of the metal wire of a single stage. If we further assume a $uniform$ distribution of R and C for each stage, then the total delay in (3.25) becomes:

$$\tau_d \;=\; \sum_{j=1}^{K}(\frac{C_{total}}{K})\sum_{k=1}^{j}(\frac{R_{total}}{K}) = (\frac{C_{total}}{K})(\frac{R_{total}}{K})\sum_{j=1}^{K}\sum_{k=1}^{j}1 = (\frac{C_{total}}{K})(\frac{R_{total}}{K})\frac{K(K+1)}{2}$$

Note that as the number of stages increases, $\lim_{K\to\infty}\tau_d = \frac{1}{2}R_{total}C_{total}$. To accelerate memory access, the array can be divided into blocks with shorter WLs and BLs and with a subset of address bits acting as block select signals as done in Figure 2.18.



(a) 6-T SRAM Cell Schematics.



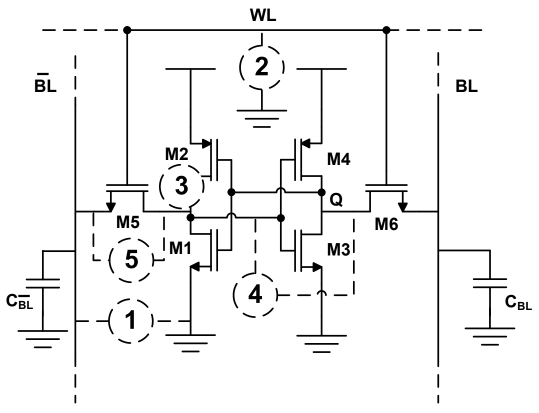
(b) 6-T SRAM Cell Layout.

Figure 3.6: 6-T SRAM Memory Cell.

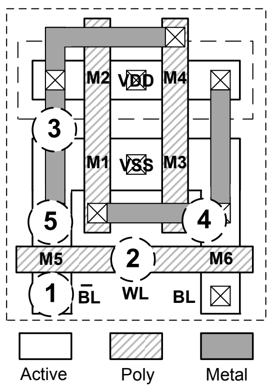
A 6-T SRAM cell as shown in Figure 3.6(a) consists of two cross-coupled inverters and two access transistors. The cross-coupled inverters regenerate the data through positive feedback, while access transistors provide bi-directional current flow between the bit lines and the memory cell, once activated by the word line. Figure 3.6(b) shows a sample layout of the 6-T SRAM cell in a standard CMOS process. For a review of the 6-T SRAM cell operation please refer to Appendix C.

The impact of memory faults is different for each of the three functional blocks: *the memory array*, *the address decoder*, and *the read / write circuits*. For example, a faulty address decoder bit can can cause a number of memory row cells to become inaccessible. Similarly, a faulty sense amplifier may fail to output the contents of a memory column cell. Thus, to unite different functional units into a single fault model, *faults in the address decoder and the read / write circuits can be modeled as the corresponding single or multi-bit faults in the memory array.* The memory array faults can be grouped into one of the following categories [39]:

1. *Stuck at Faults (SAF)*: A memory cell value is stuck-at-zero (s-a-0) or stuck-at-one (s-a-1) and the contents of the cell cannot be altered.
2. *Stuck Open Faults (SOF)*: A memory cell is stuck open and the contents of the cell cannot be accessed.
3. *Data Retention Faults (DRF)*: A memory cell fails to retain its value after a certain period of time.
4. *Transition Faults (TF)*: A memory cell fails in at least one $0 \rightarrow 1$ or $1 \rightarrow 0$ transitions.
5. *Coupling Faults (CF)*: A state, an operation or a transition due to a write to one memory cell (coupling cell) affects the value of another memory cell (coupled cell).



(a) 6-T SRAM Cell Schematics.



(b) 6-T SRAM Cell Layout.

Figure 3.7: Example SRAM Fault Types.

Figures 3.7(a) and 3.7(b) illustrate the different fault types described above. First, the bit line $\overline{BL}$ connected to ground will cause the cell to become *stuck-at* with a constant value of $Q = 1$. This will also introduce a *coupling fault* in the vertical direction for all cells sharing the faulty $\overline{BL}$. Second, a word line shorted to ground will cause the cell to become *stuck-open*. The access transistors M5 and M6 will be in the cut-off region and will prevent data transmission through the bit lines. Similarly, a *coupling fault* is introduced in the horizontal direction for all cells sharing the faulty WL. Third, if the drain of the pull-up transistor is open, the re-generative feedback loop becomes an open loop and causes a *data retention fault*. Fourth, if a polysilicon gate is drawn such that it extends over a diffusion region of an access transistor, this can lead to an additional access transistor controlled by a cell node. This causes a *transition fault* when the gate of M3 extends over the diffusion region of M6, the cell can fail a $Q_{0\rightarrow1}$ transition, since the created access transistor will counteract the access transistor M6. Fifth, a cell node connected to the bit line ($\overline{BL}$) introduces a *state coupling* fault. If the cell node $V_{\overline{Q}} = 0V$, other cells along the same bit line will act as if they are *stuck-at* a certain value. If $V_{\overline{Q}} = V_{DD}$, the cells will function normally during a read.

Stuck-at faults (SAF) account for more than 50% of memory array faults [39]. Therefore, a stuck-at model can be used as a first order approximation to a faulty memory array as illustrated in the following example.

Example 3.4: Given a C model of an OFDM receiver, the impact of frequency-time de-interleaver memory faults can be analyzed by modifying the contents of the FTDI memory array and observing the impact on overall BER. The worst-case buffer delay used in an intra-segment interleaver is equal to $\max_i \; [\, I \times m_i \,] = 190$ OFDM symbols as derived in equation (2.11). Thus, as a first-order, worst-case, approximation to fault-distribution in the physical memory, we can modify the contents of the C model memory array over a fault window of 190 OFDM symbols in duration with a uniform distribution of N stuck-at-faults, alternating between s-a-0 and s-a-1:

```
if (bit_err_cnt % 2) //stuck-at-zero (s-a-0) fault
  faulty_data = (rxqam_t)* address & 0xFFFFFFFE; //LSB s-a-0
else                  // stuck-at-one (s-a-1) fault
  faulty_data = (rxqam_t)* address | 0x00000001; //LSB s-a-1
```

Therefore, by stepping through each bit in a memory word and forcing the bits to

Test Name	$O(N)$	Description	Faults Covered
MATS	4N	$\{\Updownarrow (w0, r0, w1, r1)\};$	SAF, SOF
MATS+	5N	$\{\Updownarrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0)\}$	SAF, DRF, AD
MATS++	6N	$\{\Updownarrow (w0); \Uparrow (r0, w1);$ $\Downarrow (r1, w0, r0)\}$	SAF, SOF, DRF TF, AD
March C-	10N	$\{\Updownarrow (w0); \Uparrow (r0, w1); \Uparrow (r1, w0);$ $\Downarrow (r0, w1); \Downarrow (r1, w0); \Updownarrow (r0)\}$	SAF, DRF, TF, AD
March B	17N	$\{\Updownarrow (w0); \Uparrow (r0, w1, r1, w0, r1); \Uparrow (r1, w0, w1);$ $\Downarrow (r1, w0, w1, w0); \Downarrow (r0, w1, w0)\}$	SAF, DRF, TF, AD

Table 3.2: MBIST Test Patterns [5], [6].

s-a-0 or s-a-1 faults one-by-one, we can measure the impact of individual bit faults on a system performance metric such as BER. This is explored further in Section 3.5.

March Test

An error map showing faulty cell locations for memory repair can be obtained via a memory Built-In Self-Test (MBIST). MBIST is a Design-for-Test (DFT) technique in which test generation and test application is accomplished through built-in hardware [40]. The MBIST hardware architecture supports a parameterizable series of *march tests* in which the address pointer *marches* through the address space writing $(w0, w1)$ and reading $(r0, r1)$ bit patterns and comparing the read-out data with the expected result. Table 3.2 summarizes several important march algorithms. For instance, MATS+ is defined as $\{\Updownarrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0)\}$, where $\Updownarrow$, $\Uparrow$ and $\Downarrow$ indicate any, up and down address order directions, respectively.

Test patterns can be generated by a Linear Feedback Shift Register (LFSR), compressed by a Multiple Input Signature Register (MISR) and communicated to individual memory modules via a BIST processor. March tests are executed in parallel for each memory instance, the results are communicated back to the processor and can be accessed externally via the JTAG interface [41].

3.4 Repair Strategies

Fault-tolerant memory repair techniques can be grouped into hard, soft, combination and accumulative repair strategies [7] based on how repair information is acquired and retrieved.

Hard Repair

The repair information is stored in a one-time programmable, non-volatile eFuse and retrieved during chip boot-up. The repair signature is compressed and recorded permanently in the eFuse through the process of electromigration of the silicide layer induced by an application of higher-than-nominal voltage. The hard repair technique introduces a minor latency overhead in order to load the eFuse information during the power on reset. For example, an estimated latency required for hard repair based on a parallel test of $1K \times (16 \times 40)$ memories is: $2300\ T_{CK} \times 313\ \frac{ns}{T_{CK}} = 0.72\ ms$, where T_{CK} is the test clock period that may be further decreased to curtail the latency requirement. The primary advantage of hard repair is the ability to detect and repair memory faults revealed under extreme operating conditions generated by the tester. The Automated Test Equipment (ATE) is capable of uncovering faults that would not be found during normal operation conditions through aggressive voltage and temperature scaling. The eFuse module is typically limited by a maximum number of available eFuses [4] and its area scales better with the process compared to laser fuses.

Soft Repair

The soft repair information is generated each time the chip is powered on. In contrast to hard repair, a soft repair signature is volatile, and therefore soft repair keeps track of embedded memory problems as they arise during the operation of the device. As a result, the differentiating advantage of soft repair is the ability to repair memory faults that arise in the field throughout the life-time of a chip. The repair information is generated based on MBIST march tests. The latency introduced by soft repair for the parallel test of $1K \times (16 \times 40)$ memories is comparable to the hard repair technique: $1670\ T_{CK} \times 313\ \frac{ns}{T_{CK}} = 0.52\ ms$.

Combinational Repair

A combinational repair strategy combines the advantages of both hard and soft repair at the expense of increased latency. Hard repair covers memory faults unmasked by PVT corner cases at production time, while soft repair complements by repairing current faults acquired throughout the chip's life-time. The combined latency for the parallel test of $1K \times (16 \times 40)$ memories is $3970\ T_{CK} \times 313\ \frac{ns}{T_{CK}} = 1.24\ ms$, which may further be decreased by reducing T_{CK} period.

Cumulative Repair

A cumulative repair strategy accumulates repair information in a *re-programmable*, non-volatile eFuse module. Consequently, soft repair information generated in the field is added to and stored with the hard repair information generated at production time. The repair signatures stored in the re-programmable eFuse box are non-volatile but limited in terms of the maximum number of eFuses available.

Redundancy Allocation

To reduce repair overhead for memories with redundant rows and columns, an optimum allocation strategy can help conserve redundant resources while maximizing the benefit of a particular allocation of spare rows and columns. The problem of finding the minimum number of spares required to cover the maximum number of bit errors can be modeled as a vertex covering problem of a graph [42].

An $M \times N$ memory array is represented as a bi-partite or a two-colorable graph $G = (V, E)$ with two sets of nodes R and C in V, corresponding to faulty memory rows and columns, respectively. The two nodes are connected by an edge whenever there is a bit error in the corresponding row and column. For example, a rectangular memory array with M=8 rows and N=10 columns and the corresponding bipartite graph are shown in Figures 3.8 and 3.9. In the example of Figure 3.8, the number of Spare Rows (SR) is 2 and the number of Spare Columns (SC) is 3.

Because the redundancy allocation problem is modeled as a graph, graph-theoretic methods can be applied to find the minimum number of spare rows and columns required to repair the memory. In particular, we are interested in finding a minimum

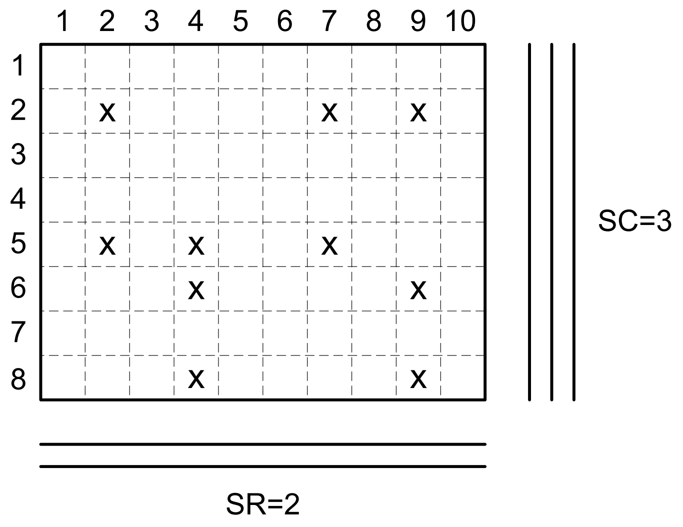


Figure 3.8: A Faulty Memory Array with M=8, N=10, SR=2 and SC=3.

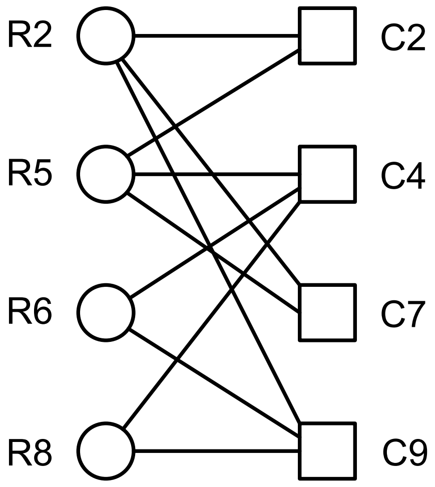


Figure 3.9: A Bipartite Graph G of the Faulty Memory Array in Figure 3.8.

vertex cover (lowest number of redundant resources) required to cover the maximum number of edges (highest number of bit errors).

For a bipartite graph, the minimum number of vertices that cover all the edges is equal to the number of edges in any maximum matching of the graph [43]. A matching M of a graph G is a subset of the edges with the property that no two edges of M share the same node. M is a maximum matching if G has no matching M' with $|M'| > |M|$, where $| \cdot |$ is the number of elements in a set.

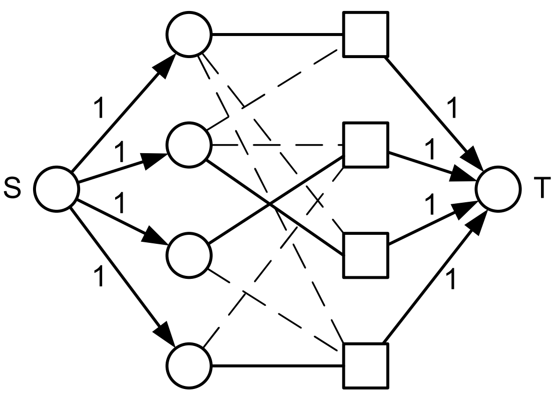


Figure 3.10: A Maximum Matching Flow Graph M of the Faulty Memory Array in Figure 3.8.

The largest bipartite matching M can be found using a network flow as shown in Figure 3.10. A source node S is connected to every vertex in R by an edge of weight 1. Similarly, a sink node T is connected to every vertex in C by an edge of weight 1. Each edge in the bipartite graph G is assigned a weight of 1. The maximum possible flow from S to T defines the largest matching in G. Finally, the number of edges in a maximum matching M is equal to the minimum number of vertices that cover all the edges or equivalently the minimum number of spares required to repair the memory. Therefore, in the example of Figure 3.8, only *four* spares are required to fully repair the memory, e.g. SR={2, 5}, and SC={4, 9}.

3.5 Proposed Repair Technique

The proposed repair technique is a fault-tolerant scheme aimed at improving the yield of an embedded-memory OFDM receiver without the use of redundancy. The proposed repair technique assigns a *sensitivity coefficient* for each memory cell based on the impact of cell fault on a system performance metric such as the Bit Error Rate (BER). Thus, in the simplest form of the technique, each addressable word in the memory array is divided into fields or blocks of high and low sensitivity to memory cell failure. *To minimize the impact of memory faults on system performance, the data block is permuted such that bits with higher fault sensitivity coefficient are assigned fault-free memory locations while bits with lower fault sensitivity coefficient are assigned faulty memory locations.* Figure 3.11 shows a section of the de-interleaver memory used to store soft I and Q data along with the carrier-to-noise (CN) ratio.

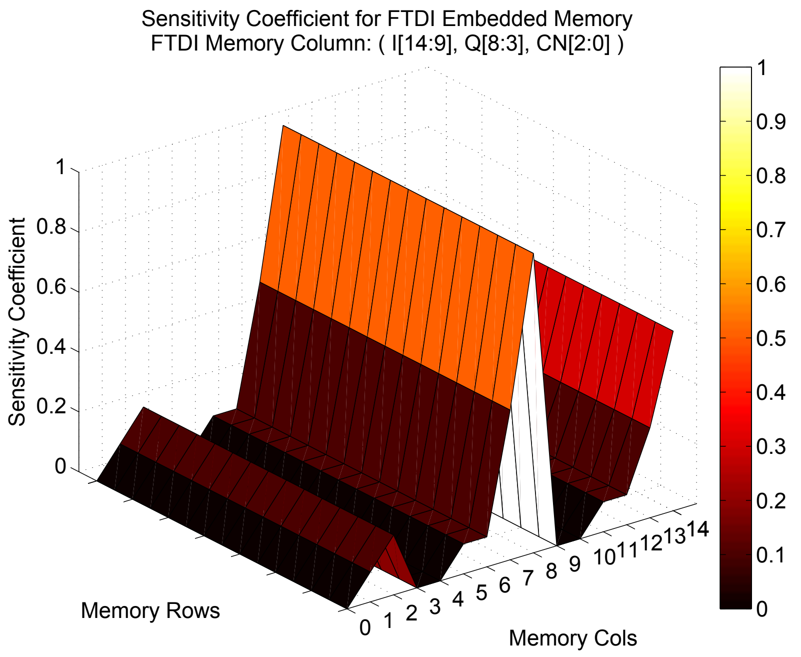


Figure 3.11: Sensitivity Coefficient for the FTDI Embedded Memory. Memory columns store {I[14:9], Q[8:3], CN[2:0]} associated with a single OFDM symbol. Memory rows store OFDM symbols $S_{(i\ ,j\ ,k)}$.

A sensitivity coefficient ζ is assigned for each bit as a difference in BER caused by the stuck-at memory cell compared to the nominal or fault-free cell, normalized to 1:

$$\zeta = \frac{1}{C}\left(BER_{SA} - BER_{FF}\right) \tag{3.26}$$

where, the subscripts *SA* and *FF* represent stuck-at and fault-free cases, respectively, and C is a normalization constant. Figure 3.11 shows a colorbar with high error sensitivity bits in light red and low error sensitivity bits in dark red. As expected, the data bits of I, Q, and CN in Figure 3.11 that are closest to the MSB have a higher sensitivity coefficient compared to bits that are closest to the LSB. Thus, the impact of memory faults on system performance can be minimized if sensitive-to-error (MSB) data is permuted with less sensitive (LSB) data when the MSB memory region contains faulty memory cells, while the LSB region is fault-free. The proposed memory repair technique without redundancy is summarized in the Memory Repair Algorithm.

Algorithm 1 Memory Repair Algorithm (Mode, M, N, I)

```
 1: if (Mode = MBIST) then
 2:   for $i = 0$ to $I - 1$ do
 3:     row_address_fault[i][M-1:0] $\Leftarrow$ 0;
 4:     if (cur_err_out = 1) then
 5:       error_word[i] $\Leftarrow$ error_register[i][N-1:0];
 6:       MSB_region[i] $\Leftarrow$ error_word[i][sensitivity $\geq$ threshold];
 7:       LSB_region[i] $\Leftarrow$ error_word[i][sensitivity $<$ threshold];
 8:       if (|MSB_region[i] = 1 and |LSB_region[i] = 0) then
 9:         row_address_fault[i][(cur_row_address[i])] $\Leftarrow$ 1;
10:       end if
11:     end if
12:   end for
13: else
14:   while (Mode = Functional) do
15:     if (row_address_fault[i][(cur_row_address[i])] = 1) then
16:       swap (MSB_region, LSB_region);
17:     end if
18:   end while
19: end if
20: return row_address_fault[i]
```

The proposed memory repair algorithm initializes the row address fault register in test mode (steps 1-12) by setting a bit corresponding to the faulty row address to a “1” and checks the row address fault register in functional mode (steps 13-20) on every memory read and write operation to determine when to activate the permutation logic. *In MBIST mode*, the bit error location is captured by reading the error register (5), next the high and low sensitivity regions are examined for the presence of errors via a reduction-OR operator (6-8), and the row address is labeled faulty if errors are found in the high sensitivity (MSB) region, while the low sensitivity (LSB) region is error free (9). *In functional mode*, the row address fault register is accessed on every memory operation (15) and if the current row address is marked faulty, the MSB and LSB regions are permuted (16). Thus, the algorithm provides memory repair without redundancy in which sensitive-to-error data is stored in error-free memory locations, while less sensitive-to-error data is assigned to faulty memory locations.

3.6 Simulation Results

Figure 3.12 shows a simulated BER plot for an AWGN channel based on an ISDB-T receiver model. The simulation parameters are summarized in Table 3.3.

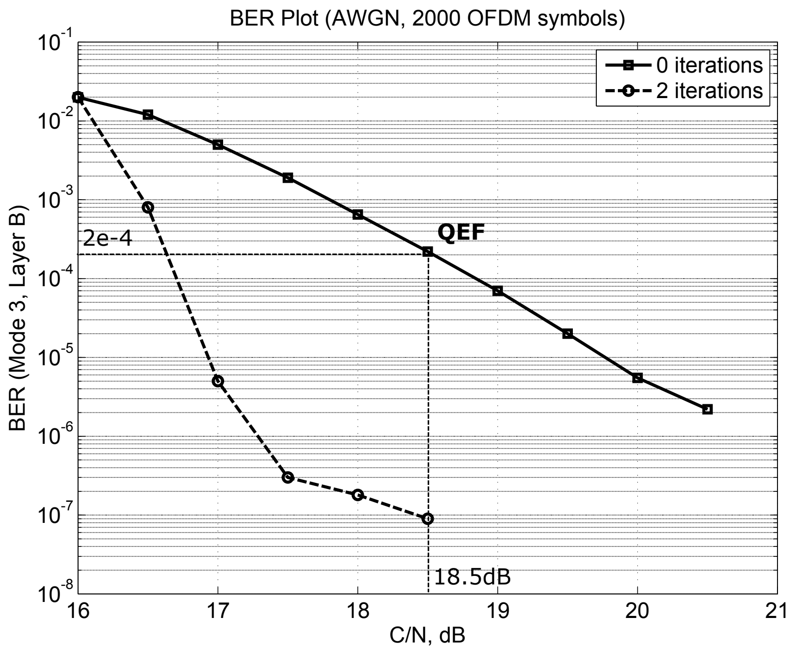


Figure 3.12: Simulated BER Plot (Mode 3, Layer B: 64-QAM, R=3/4, N_R=1, AWGN, 2000 OFDM symbols, FEC with 0 and 2 Iterations).

The Quasi-Error Free (QEF) operating point is defined as the maximum acceptable bit error rate for which the user does not perceive any degradation in performance. The ISDB-T QEF point for an AWGN channel is 2×10^{-4} at 18.5 dB carrier-to-noise (CN) ratio.

Parameter	Value	Description
Transmission Parameters		
Mode	3	5,617 sub-carriers
Layer A	1	Number of Segments
	QPSK	Modulation
	R=2/3	Code Rate
	I=4	Interleaver Length
Layer B	12	Number of Segments
	64-QAM	Modulation
	R=3/4	Code Rate
	I=2	Interleaver Length
Layer C	0	Number of Segments
	16-QAM	Modulation
	R=3/4	Code Rate
	I=1	Interleaver Length
Channel Parameters		
Channel Models	AWGN	Additive White Gaussian Noise
	Rayleigh	Fading Channel
CN	16 - 21 dB	Carrier to Noise Ratio
F_{offset}	0 Hz	Frequency Offset
T_{offset}	0 ppm	Sampling Offset
F_{drift}	0 Hz/sec	Frequency Offset Drift
Receiver Parameters		
N_R	1	Number of Receive Antennas
N_{FFT}	8,192	Fast Fourier Transform
N_{sym}	2,000	Number of OFDM symbols
N_{iter}	0, 2	Number of Iterations
N_{SA}	0 - 400	Number of Stuck-At Faults in Deinterleaver Memory

Table 3.3: Simulation Parameters.

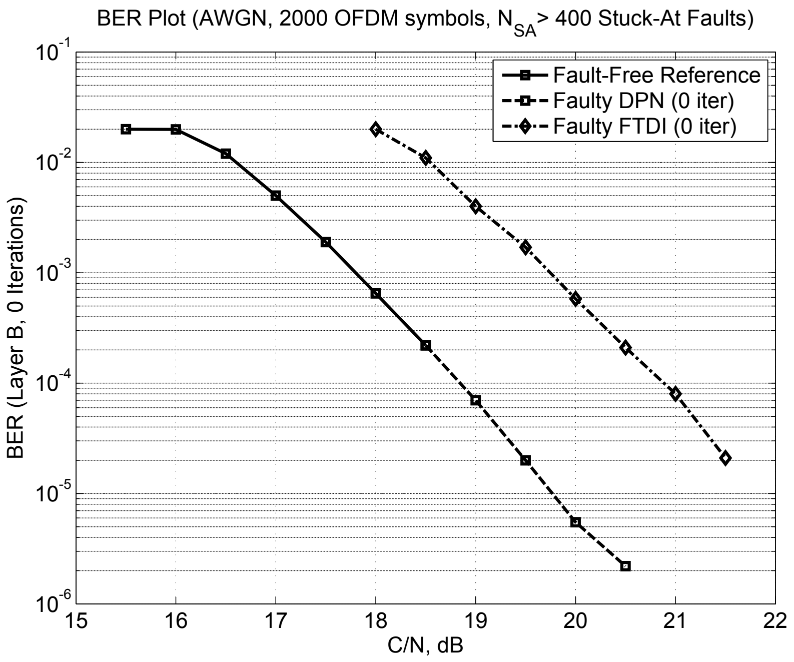
The BER drops to 1×10^{-7} with 2 turbo-like iterations of soft-output Viterbi and RS (204, 188) decoder [44]. The transmission parameters in Table 3.3 were selected to cover most of the signal processing blocks (Mode 3, Layer B), while keeping reasonable simulation time (e.g. $N_R = 1$). Figure 3.12 shows the nominal BER plot with zero stuck-at faults in the de-interleaver memory ($N_{SA} = 0$).

To analyze the impact of embedded memory faults on the BER, the two largest area blocks in Table 3.1 were selected: the frequency-time de-interleaver (FTDI) and the de-puncturer (DPN). N_{SA} stuck-at faults alternating between s-a-0 and s-a-1 were uniformly distributed every 190 OFDM symbols corresponding to the worst-case de-interleaver delay for Layer B, as illustrated in Example 3.4. The tests were divided into *high fault* ($N_{SA} \geq 400$) and *low fault* cases ($N_{SA} = 10$ to 20).

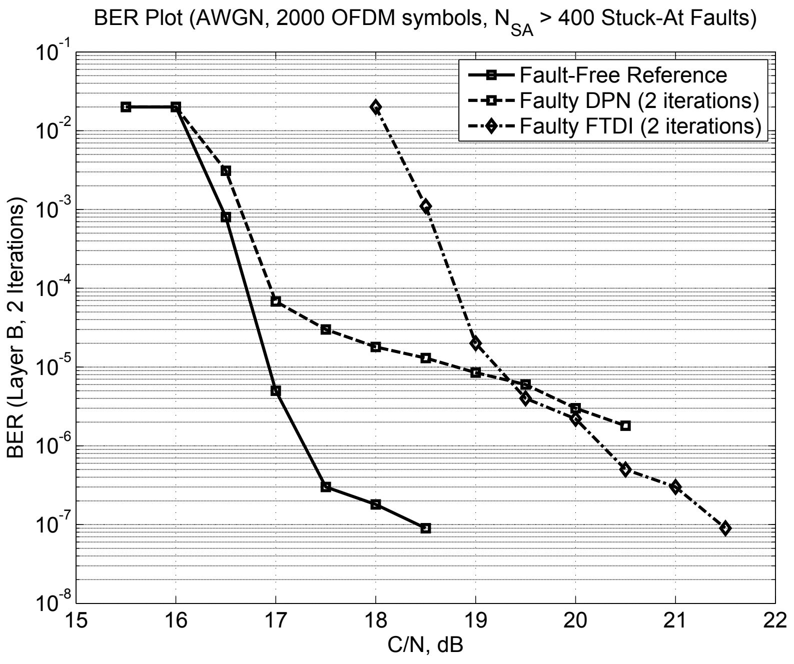
Figures 3.13(a) and 3.13(b) show high fault behavior for an AWGN channel with 0 and 2 decoding iterations, respectively. The de-puncturer (DPN) memory is bypassed in the 0 iteration case and the BER plot for DPN aligns with the fault-free reference. However, the BER plot for FTDI shows a 2 dB performance degradation at the 2×10^{-4} QEF point. In the 2 iteration case, the DPN memory is used and a degradation in BER is observed. The gain loss for FTDI is comparable to the 0 iteration case due to a very high N_{SA}.

Figures 3.14(a) and 3.14(b) show the high fault impact on the BER for an AWGN channel at $CN = 18.5$ dB QEF point for two different FTDI word formats. In Figure 3.14(a) N_{SA} faults were mapped onto each bit in $\{I[23:14],\ Q[13:4],\ CN[3:0]\}$ de-interleaver word, while zero faults were introduced in the de-puncturer. For a very high N_{SA}, the upper 5 bits of I and Q as well as the upper 2 bits of CN show highest sensitivity to memory faults, while the lower 5 bits of I and Q as well as the lower 2 bits of CN show low fault sensitivity. Similarly, Figure 3.14(b) shows the impact of $N_{SA} = 400$ on BER for a shorter word format: $\{I[14:9],\ Q[8:3],\ CN[2:0]\}$. In both cases, the de-interleaver memory can be divided into blocks of high and low fault sensitivity. The proposed memory repair exploits the observed difference in sensitivity and permutes the incoming data such that bits that are sensitive to faults are stored in the fault-free locations and vice versa.

A sensitivity coefficient ζ in Figure 3.11 was computed based on the BER plot in

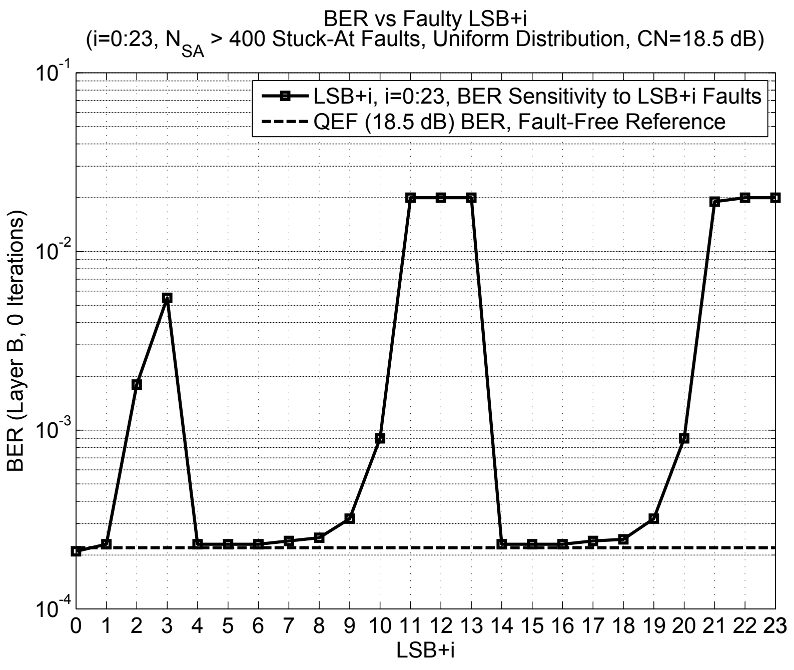


(a) 0 Iterations

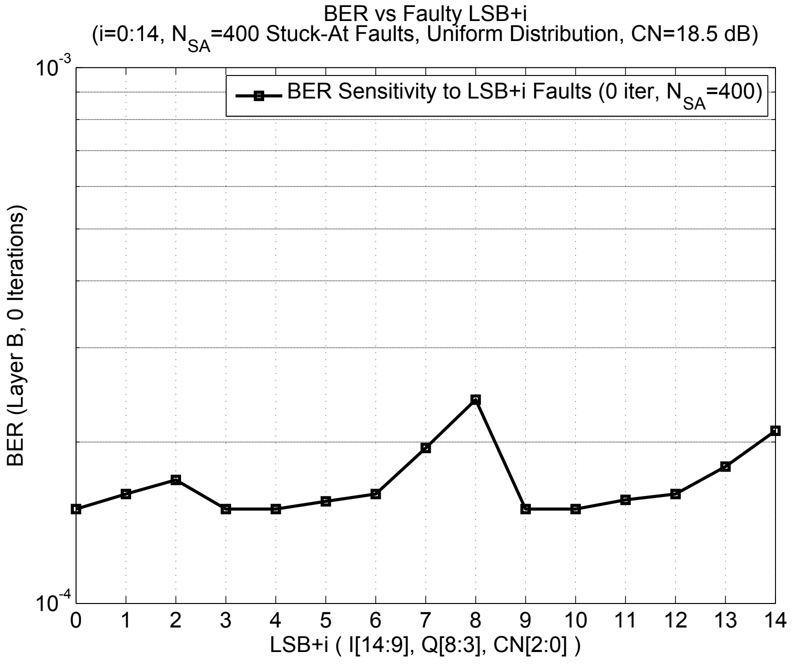


(b) 2 Iterations

Figure 3.13: Simulated BER Plot (Mode 3, Layer B: 64-QAM, R=3/4, N_R=1, AWGN, 2000 OFDM symbols).

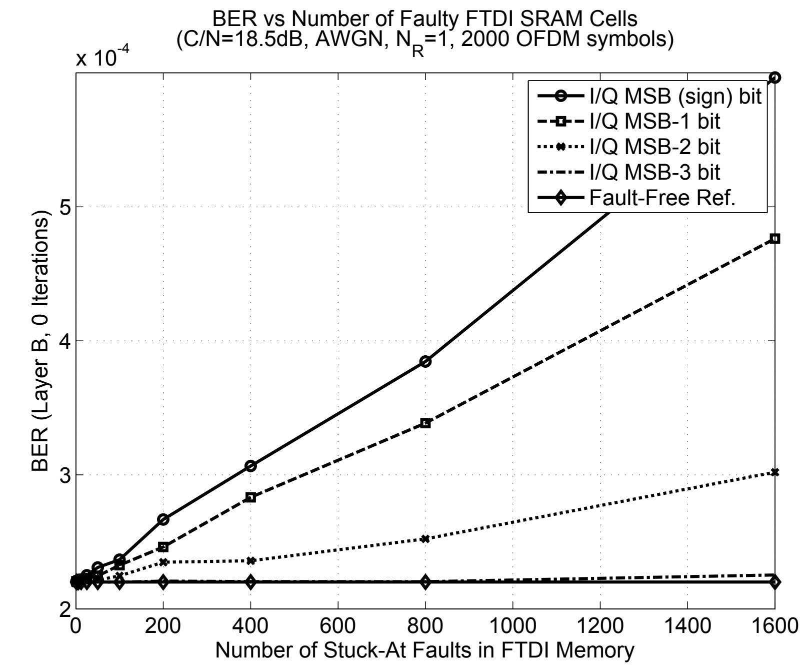


(a) FTDI Word: $\{I(23:14),\ Q(13:4),\ CN(3:0)\}$

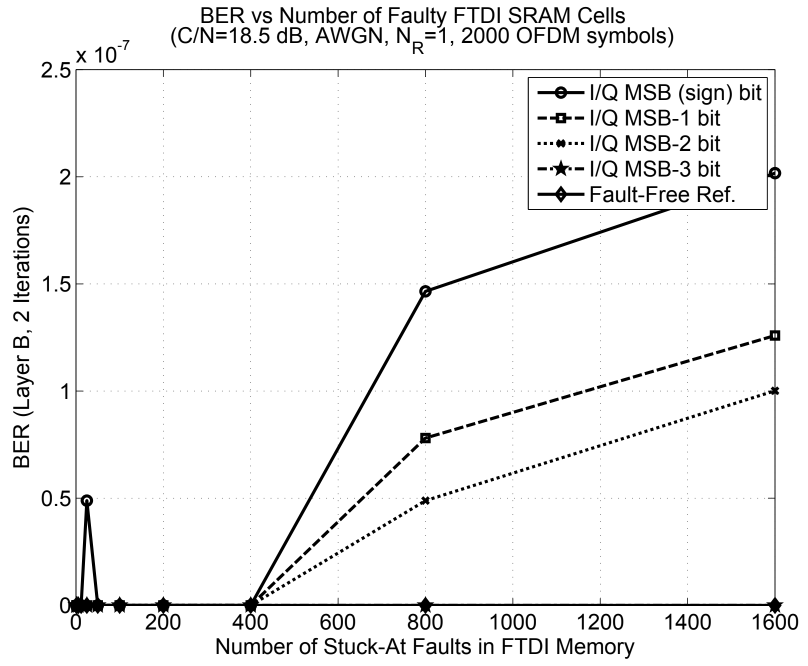


(b) FTDI Word: $\{I(14:9),\ Q(8:3),\ CN(2:0)\}$

Figure 3.14: Simulated BER Plot (Mode 3, Layer B: 64-QAM, R=3/4, N_R=1, AWGN, 2000 OFDM symbols, 0 iterations).



(a) 0 Iterations



(b) 2 Iterations

Figure 3.15: Simulated BER Plot (Mode 3, Layer B: 64-QAM, R=3/4, N_R=1, AWGN, 2000 OFDM symbols).

Figure 3.14(b). By setting a sensitivity threshold to 1.6×10^{-4}, or 7% above the fault-free reference, the high-sensitivity (MSB) region consists of the top 3-bits of I and Q and the MSB of CN: $\{I[14:12],\ Q[8:6],\ CN[2]\}$, while the low-sensitivity region of the same width contains: $\{I[11:9],\ Q[5:3],\ CN[1]\}$. If a memory fault is found in the MSB region and no faults were found in the LSB region, the permutation of the MSB and LSB regions in the case of Figure 3.14(b) results in 0.35 dB gain improvement at 2×10^{-4} BER over memory without repair at the QEF BER for an AWGN channel. The gain improvement ΔCN due to the proposed repair was found by dividing the maximum difference in the MSB BER in Figure 3.14(b) by the slope of the BER plot at the QEF BER=2×10^{-4} in Figure 3.12:

$$\Delta CN = \left.\frac{\Delta BER}{slope}\right|_{QEF} = \left.\frac{BER_{Q[5]} - BER_{Q[8]}}{slope}\right|_{QEF} = \frac{(0.00015 - 0.00024)}{(-2.6 \times 10^{-4}\ dB^{-1})} = 0.35\,dB.$$

The implementation cost of storing the repair information is equal to $\lceil M/i \rceil$ storage bits, where M is the number of memory rows and i is an integer between 1 and M equal to the number of rows assigned to a single bit of the row address fault register. Thus, with $M = 1024$ and $i = 4$, the implementation overhead is 256 bits.

Figures 3.15(a) and 3.15(b) show the BER at QEF $CN = 18.5\ dB$ as a function of the number of stuck-at faults N_{SA} for $\{I[19:12],\ Q[11:4],\ CN[3:0]\}$ de-interleaver word. In the 0 iteration case in Figure 3.15(a), the MSB (sign bit) shows higher sensitivity to a memory fault in comparison to MSB-3 bit, which is close to the fault-free reference. With 2 decoding iterations in Figure 3.15(b), $N_{SA} \leq 400$ stuck-at faults were corrected by the soft-output Viterbi and RS (204, 188) turbo-like iterative decoding.

3.7 Summary

The total yield model combines the gross yield and the random fault yield based on the generalized negative binomial distribution characterized by an average number of faults and clustering parameters for each process step. Embedded memories have a much denser layout in comparison to core logic and therefore higher fault densities.

Embedded memory faults can impact the functionality of the memory array, the address decoder or the read / write circuits. In each case, the faults can be mapped to the corresponding single or multi-bit faults in the memory array. SRAM faults can be characterized into: stuck-at faults (SAF), stuck-open faults (SOF), data retention faults (DRF), and coupling faults (CF). MBIST memory tests known as *march* tests can be used to discover the faults in each category and create a fault map of the memory array.

Memory repair strategies use the fault map information to correct or mitigate the impact of errors. Repair techniques with redundancy incur area overhead and require optimum allocation of the limited number of spare rows and columns. Repair techniques without redundancy, must be comparable in repair performance and introduce lower implementation overhead to be a competitive alternative to redundancy oriented repair techniques. The implementation of memory repair can be classified into hard repair, soft-repair, combinational and cumulative repair strategies based on how repair information is acquired and retrieved.

The proposed repair technique is a low-overhead, fault-tolerant scheme for reducing the impact of embedded memory errors in OFDM receivers without the use of redundancy. The proposed repair technique divides the memory array into blocks of high and low sensitivity to cell failures based on a sensitivity coefficient computed for each bit in a memory word as a difference in BER caused by the faulty bit compared to the fault-free bit. Thus, the data block with faulty cells is *permuted* with a data block of the same size only if the former has a higher sensitivity to error. The permutation results in 0.35 dB improvement in CN at 2×10^{-4} BER for an AWGN channel compared to memory without repair based on simulation results with zero decoding iterations, when $N_{SA} = 400$ stuck-at faults were randomly distributed across the deinterleaver memory.

The next chapter compares performance advantages of the proposed repair technique, presents a memory repair architecture in addition to BER measurements for AWGN and fading channel models.

4 VLSI Implementation

Knowledge is of no value unless you put it into practice.

ANTON CHEKHOV

4.1 Introduction

The proposed memory repair strategy is implemented as part of a larger Design-for-Test (DFT) on-chip infrastructure. The memory repair architecture interfaces with Built-In Self-Test (BIST) memory wrappers designed for increased *controllability* and *observability* of embedded memory faults. The repair architecture provides low latency and configurable area overhead. The effectiveness of the proposed repair strategy is evaluated on a multiple-FPGA platform connected to an ISDB-T signal generator through a wireless channel emulator. Measurement results are presented for an AWGN and TU-6/TU-12 fading channels.

4.2 DFT Architecture

Figure 4.1 shows the top-level DFT architecture [4]. It consists of STAR[1] Memory System (SMS) modules, a JPC[2]/SFP[3] server, an eFuse Box and 1149.1 JTAG and P1500 SECT interfaces. The SMS modules contain embedded memory wrappers controlled by a STAR processor [4]. The JPC/SFP server interfaces IEEE 1149.1 Joint Test Action Group (JTAG) with IEEE P1500 Standard for Embedded Core Test (SECT) and provides a connection to the one-time programmable eFuse Box used to store hard repair information.

[1] STAR: Self Test And Repair.
[2] JPC: JTAG to P1500 Converter.
[3] SFP: Shared Fuse Processor.

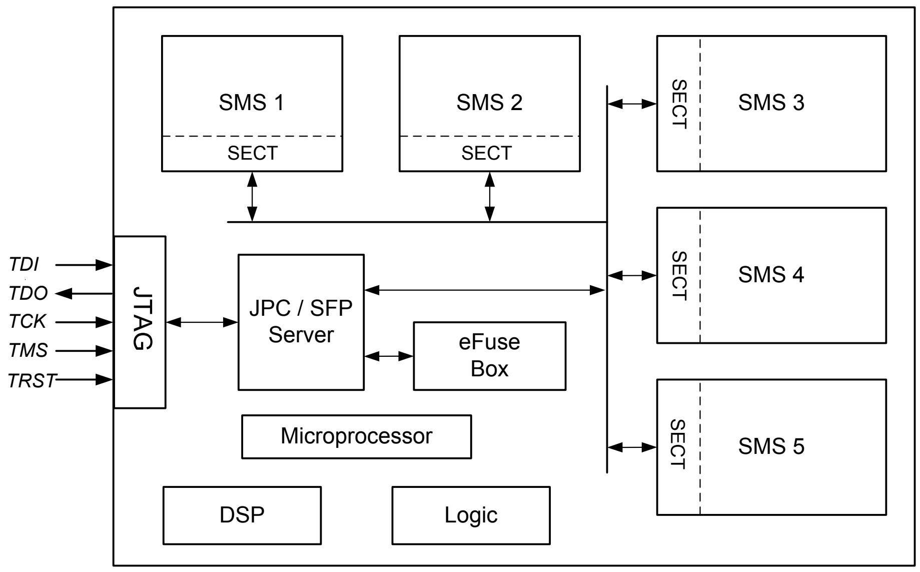


Figure 4.1: SoC Level DFT Architecture with JTAG IEEE 1149.1 Interface [4].

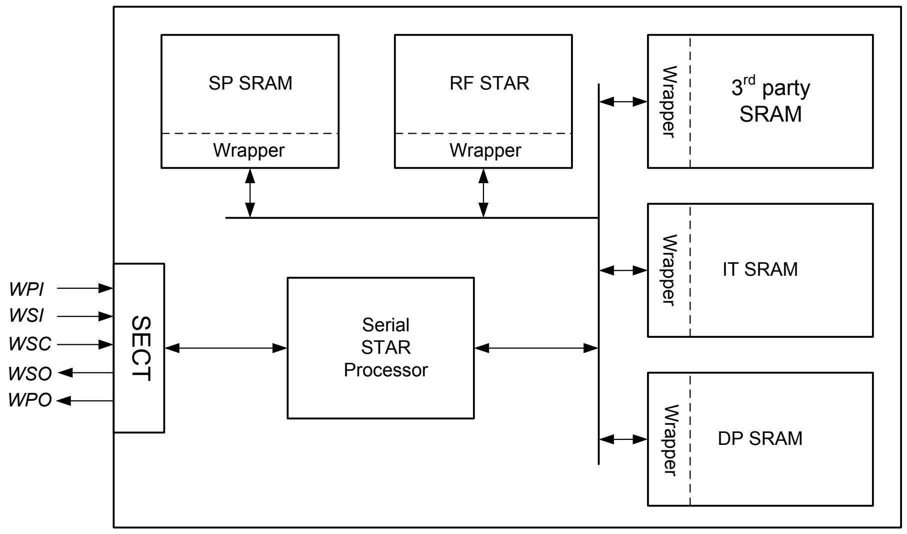
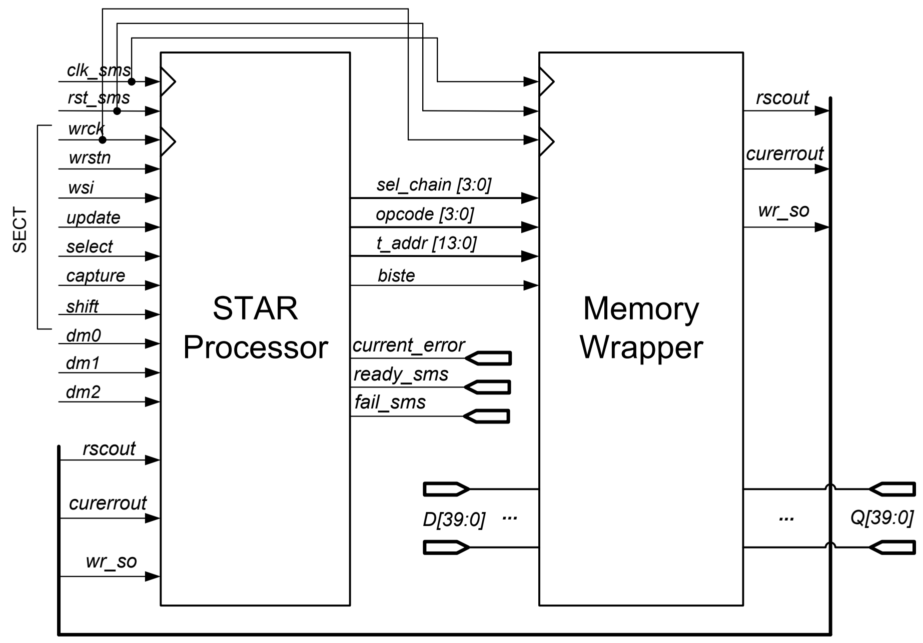
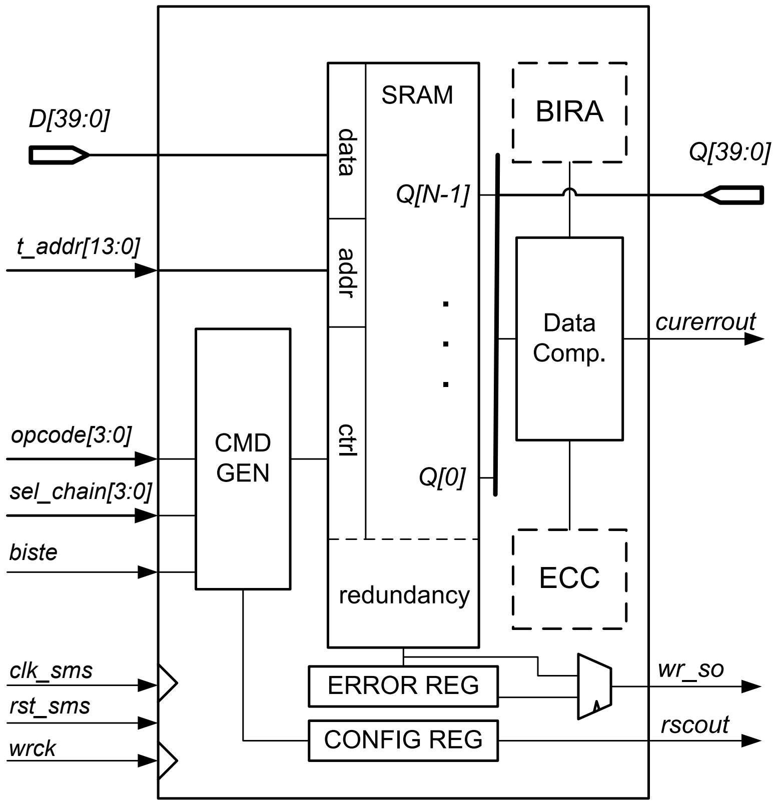


Figure 4.2: SMS (STAR Memory System) Block Level Architecture with SECT IEEE P1500 Interface [4].



(a) STAR Processor and Memory Wrapper Interface.



(b) Memory Wrapper Architecture.

Figure 4.3: STAR Processor and Memory Wrapper Architecture [4].

Figure 4.2 shows the block level architecture of a single SMS module composed of a STAR processor and BIST memory wrappers. The processor directs the execution of march tests carried out in parallel by the memory wrappers. The results of MBIST tests are communicated back to the processor and can be accessed externally via the JTAG interface. The architecture of the memory wrapper and the proposed repair technique are generic enough to support multiple memory types[4] and memory BIST vendors.

The STAR processor to memory wrapper interface is shown in Figure 4.3(a). The internal registers of the wrappers can be configured and accessed via wrapper control signals. If a mismatch is detected during a march test, the *curerrout* flag is pulsed and memory fault information is accumulated in an accumulative fault register. The accumulative fault register can be accessed externally via the serial output *wr_so* of the wrapper or in certain cases via the $Q[39:0]$ output of the memory. The memory wrapper architecture is shown in Figure 4.3(b). It consists of an SRAM memory block, a command generator, a data comparator as well as auxiliary registers. An optional Built-In Repair Analysis (BIRA), SEC/DED Error Correction Coding (ECC), and redundancy blocks can be instantiated by the memory compiler.

4.3 Memory Repair Architecture

Figure 4.4 shows the architecture of the proposed soft memory repair without redundancy. The proposed repair algorithm in Section 3.5 initializes the row address fault register in test mode (steps 1-12) by setting a bit corresponding to the faulty row address to a “1” and checks the row address fault register in functional mode (steps 13-20) on every read and write operation to determine when to active the permutation logic. *In MBIST mode*, the error capture and repair enable logic is used to: (i) capture externally the serial output of the accumulative fault register; (ii) examine its contents for the location of faults in both high and low sensitivity regions; and (iii) set the corresponding bit of the row address fault register if the higher sensitivity region has at least one error while the lower sensitivity region is error-free. *In functional mode*, the row address fault register is accessed on every read and write

[4] Supported memory types include Single Port (SP), Dual Port (DP), Register File (RF), Integrated Test (IT) as well as 3-rd party SRAM memories.

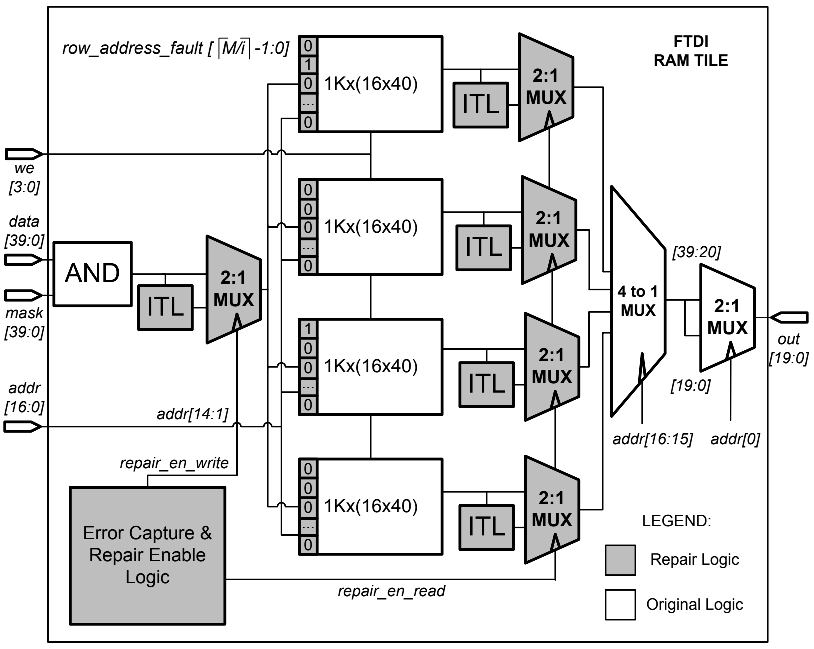


Figure 4.4: Proposed Memory Repair Architecture.

operation, and if the current row address is labeled faulty, the regions of high and low sensitivity are swapped by the fault repair interleave (ITL) logic and output through the 2-to-1 MUX controlled by repair enable signals. During a memory write, the 2-to-1 input MUX selects between the original data or the permuted data based on the corresponding bit of the row address fault register before writing the data into one of the memory blocks. During a memory read, the 2-to-1 output MUX selects either the original or the de-permuted data based on the corresponding bit of the row address fault register and forwards the data to the output bus.

The FTDI memory is organized internally into 1K rows and 16 $\times$ 40-bit columns. Therefore, the maximum size of the row address fault register is 1024. However, to reduce area overhead, a single bit in the row address fault register can be used to track multiple rows as shown in Figure 4.5. The external-to-memory row address fault register consists of $\lceil M/i \rceil$ flip-flops, where M is the total number of memory rows and i is an integer between 1 and M equal to the number of memory rows assigned to a single bit of the row address fault register. The values of M and i are commonly expressed as powers of two. In the extreme cases: (i) when $i = 1$ the size of the row address fault register is M, which means that every row has a dedicated bit that

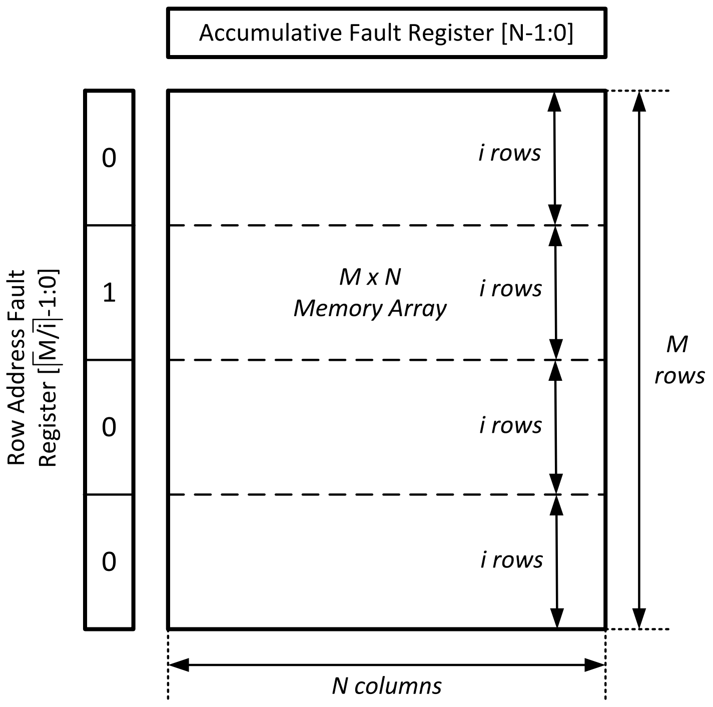

Accumulative Fault Register [N-1:0]
Row Address Fault Register [⌈M/i⌉-1:0]
0
1
0
0
M x N Memory Array
i rows
i rows
i rows
i rows
M rows
N columns

Figure 4.5: Configurable Row Address Fault Register Architecture.

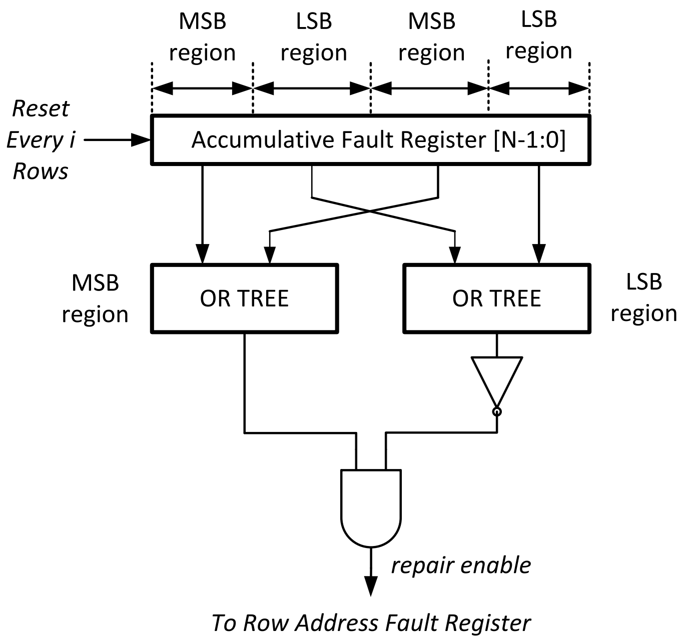


Figure 4.6: Memory Repair Enable Architecture for an N-bit Word with Two Sensitivity Regions.

controls whether to enable or disable the permutation of high sensitivity region for that row; (ii) when $i = M$, the size of the row address fault register is a single bit, which means that the in-coming data is permuted at the input and de-permuted at the output for the entire memory instance. Thus, the value of i can be adjusted to reflect the expected number of memory faults λ for a given technology process. For example, in Figure 4.5, the value of i is set to $M/4$, such that the row address fault register introduces an area overhead of only 4 flip-flops. By tuning the parameter i, one can trade off area overhead with the effectiveness of the proposed memory repair technique.

The expected number of memory faults in the 19.4 Mbit memory according to Figure 3.2 is less than 2. Thus, in the low-fault case, there is a higher chance that either the MSB or the LSB regions but not both are affected by memory faults. This causes a greater difference in the sensitivity coefficients between the two regions, and as a result the repair permutation is enabled. As the number of faults increases, the MSB and the LSB regions are more likely to both be affected by memory faults, and in that case the repair permutation is disabled. Thus, reducing the size of the row address fault register sacrifices repair performance in the high fault case.

Figure 4.6 shows the circuit used to activate the permutation logic. The accumulative fault register counts the total number of faults in a given memory column during an MBIST march test. A mismatch between the expected and the actual data results in an increment of the fault counter of the accumulative fault register corresponding to the column in which a memory fault occurred. The accumulative fault register is analyzed for presence of memory faults in the regions of high and low fault sensitivity via OR trees. If a region with high fault sensitivity contains a memory fault and a region of low fault sensitivity is fault-free, the two regions are permuted and the corresponding bit of the row address fault register is set to enable the permutation in functional mode. The accumulative fault register is reset every i rows.

Figure 4.7(a) shows the memory repair permutation or interleave (ITL) block that permutes high sensitivity data regions. In its simplest form, the permutation can be hard-wired. For example, in the proposed technique, the MSB and LSB regions of 8-bit I, 8-bit Q, and 4-bit CN fields of each memory word are permuted by interchanging the wires as shown in Figure 4.7(a). More generally, the mapper can be

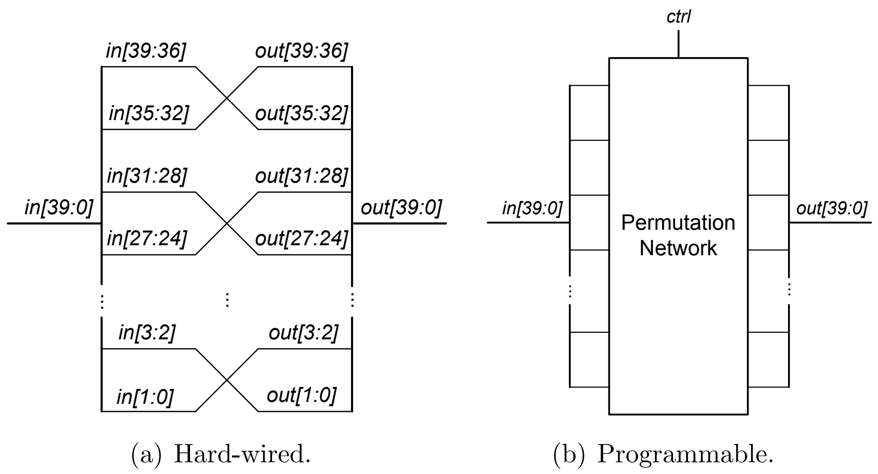


Figure 4.7: Interleave (ITL) Block Architecture for 8-bit I, 8-bit Q and 4-bit CN.

made programmable via a permutation network controlled by a *ctrl* signal as shown in Figure 4.7(b), in order to adapt the permutation to a variety of data formats and to account for potential difference between the logical address and the corresponding physical memory location that may arise due to memory layout constraints.

The proposed repair architecture does not introduce additional clock cycles required to access the memory. It provides minimum timing overhead since the only data path delay is due to the 2-to-1 MUX during write and read operations, while the ITL logic performs a zero delay permutation operation. The implementation costs of the proposed repair technique based on the worst-case PVT synthesis in 65-nm CMOS are presented in Table 4.1. The repair overhead is summarized under Δ_i columns for $i = 1,\ 2,\ 4$, where i is the number of memory rows assigned to a single bit of the row address fault register. Thus, the proposed memory repair can be configured to introduce 1.7 % of area overhead, when $i = 4$ due to the fault register with $\lceil M/i \rceil = \lceil 1024/4 \rceil = 256$ flip-flops.

Table 4.2 compares implementation performance of different SRAM memory repair techniques. The proposed strategy introduces a single multiplexer latency delay on read and write operations and 1.7% of area overhead ($i = 4$), dominated by external-to-memory registers used to store repair information. The proposed repair technique is different in the sense that it seeks to minimize the impact of embedded memory

65nm CMOS	FTDI SRAM (*original*)	FTDI SRAM (*i=1*)	Δ_1	FTDI SRAM (*i=2*)	Δ_2	FTDI SRAM (*i=4*)	Δ_4
Clock Rate (MHz) (Spec'd 64 MHz)	69.4	69.4	4ps w.c. slack	69.4	4ps w.c. slack	69.4	4ps w.c. slack
Number of Cells Cell Area (μm^2)	14,074 13,148,617	144,031 13,826,323	129,957 5.2%	78,646 13,481,993	64,572 2.5%	42,943 13,368,199	28,869 1.7%
Dynamic (μW) Leakage (μW) Total Power (μW) @ 69.4 MHz	31,060 887 31,947	39,413 1,055 40,468	8,352 168 27%	32,845 987 33,832	1,785 100 5.9%	32,387 928 33,315	1,327 41 4.3%

Table 4.1: ASIC Synthesis Results of the Proposed Memory Repair in 65 nm CMOS for the Worst-Case PVT: SS, 1.08V, 125°C. Δ_i: Repair Overhead When the Faulty Row Address is Recorded for Every i Memory Rows.

Parameter	[45]-2006	[46]-2007	[47]-2010	[4]-2011	This Work-2011*
Memory Size [Mbits]	0.5	0.5	0.5	19.4	19.4
Memory Area [mm^2]	N/A	6.79	5.05	12.9	12.9
Area Overhead	6.5%	2.8%	2.3%	1.7%	1.7%
Technology	180nm	180nm	180nm	65nm	65nm
Gate Count [kGE]	6.3	8.3	N/A	N/A	38.1
Redundant Rows	4	3	6	0	0
Redundant Cols	2	3	6	4	0
Error Correction	No	No	No	No	Yes
Clk Frequency [MHz]	N/A	N/A	N/A	69.4	69.4
Repair Strategy	soft	soft	soft	hard	soft

Table 4.2: SRAM Memory Repair Performance Comparison. *Based on i = 4 in Table 4.1.

faults relying on the downstream Viterbi and RS decoder blocks for error correction rather than using costly redundancy or local ECC logic as memory repair mechanism. The repair technique in [4] introduces a comparable area overhead with four redundant columns integrated internally in the layout of the main memory. The repair performance in [4] with redundancy is limited by the number of spare rows and columns, while the proposed strategy is capable of permuting data for all faulty memory rows. However, the choice of using registers as a means of storing repair information for every memory instance increases the gate count in comparison to [45], [46]. The large gate count is a result of the row address fault registers of size $\lceil M/i \rceil$ for each of the 30 de-interleaver memory instances with $M = 1024$ and $i = 1,\ 2,\ 3, \ldots,\ M$, where i is the number of memory rows assigned to a single bit of the row address fault register. Alternatively, the series of registers can be implemented as a block of memory of size $\lceil M/i \rceil$ for each instance or as a dedicated memory used to store a fixed set of the 14-bit fault addresses. The timing penalty of a single multiplexer delay on read and write operation is comparable to [45], however, no write buffer is required since the data is permuted via combinational logic before it is written to or read from the memory. The proposed repair technique is integrated with a commercial BIST infrastructure, similar to [46]; however, it is generic enough to be used with multiple memory BIST vendors.

4.4 Memory Repair Verification

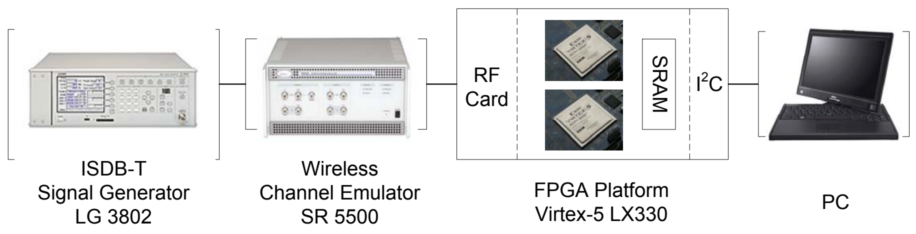


Figure 4.8: Memory Repair Verification Platform.

The test set-up used to verify the proposed memory repair strategy is shown in Figure 4.8. The signal generator modulates a pseudo-random binary number (PN) sequence according to the ISDB-T transmission specifications [3] summarized in Ap-

Virtex-5 LX330 FF1760-1	Original Build w/o BIST, w/o Repair	New Build w/o BIST, w/ Repair + Err. Gen.	Δ
Speed (MHz)	8.082	8.296	0.214
Area			
Num of Slices	68% (35,257 of 51,840)	69% (35,971 of 51,840)	1% (714)
LUTs	47% (98,255 of 207,360)	48% (101,538 of 207,360)	1% (3,283)
Slice FFs	21% (45,254 of 207,360)	22% (46,414 of 207,360)	1% (1,160)

Table 4.3: FPGA Post Place and Route Resource Summary.

pendix A. The wireless channel emulator generates programmable time-varying channel characteristics such as multi-path delay spread, fading, and channel loss. The channel emulator is connected to a custom made platform with two Virtex-5 LX330 FPGAs via an RF tuner card. An external to FPGA synchronous SRAM chip is used to store the de-interleaver memory data due to its large memory requirements. The OFDM receiver register data is communicated to the PC via the I^2C interface and analyzed in real-time.

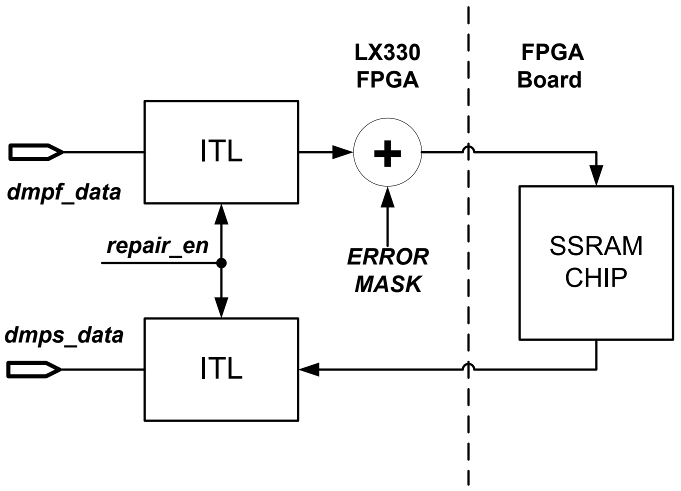


Figure 4.9: Memory Repair FPGA Prototype.

Figure 4.9 shows a schematic of the proposed memory repair prototype consisting of error generation and repair logic. The error mask introduces bursts of alternating s-a-0 and s-a-1 faults at the output of the de-mapper first before it is written into a functional SRAM memory chip acting as a faulty de-interleaver memory. The maximum error counter is set to 190 OFDM symbols corresponding to the worst-case de-interleaver delay. Therefore, to summarize, bursts of length N_{SA} are introduced every 190 OFDM symbols. The fault repair interleave (ITL) logic is activated by the repair enable signal. A 32-bit I^2C register is used to enable or disable repair, introduce bit faults via a 20-bit error mask and keep track of the number of faults via an 11-bit counter. While a DFT netlist with memory BIST wrappers is not typically prototyped on an FPGA, the DFT library can be made FPGA synthesizable [48] to prepare and execute DFT tests on an FPGA.

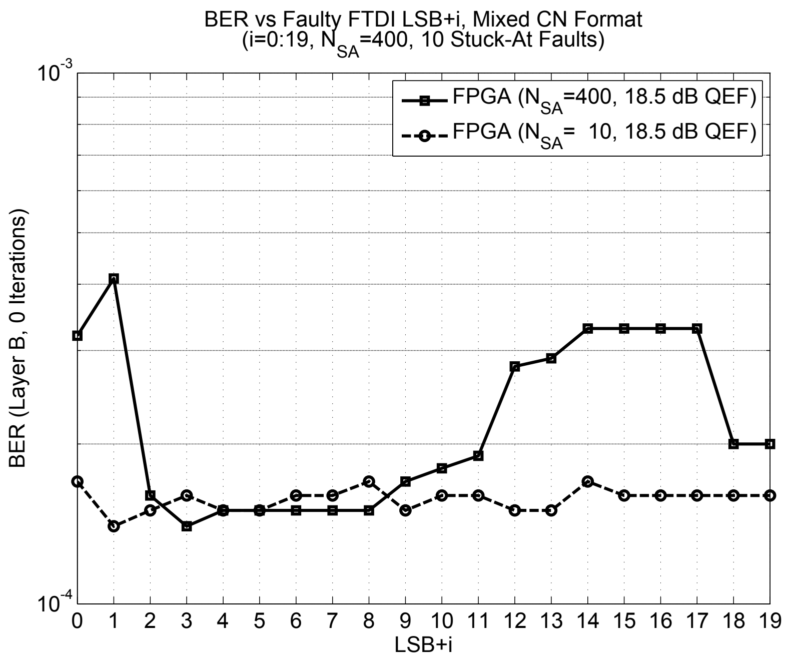
A resource summary based on the post place-and-route netlist of the proposed memory repair prototype is shown in Table 4.3. The new FPGA build was optimized for speed at the expense of available area by distributing error generation and repair logic across block boundaries and configuring synthesis options, such as enabling register duplication and re-timing, while disabling resource sharing [49].

4.5 Measurement Results

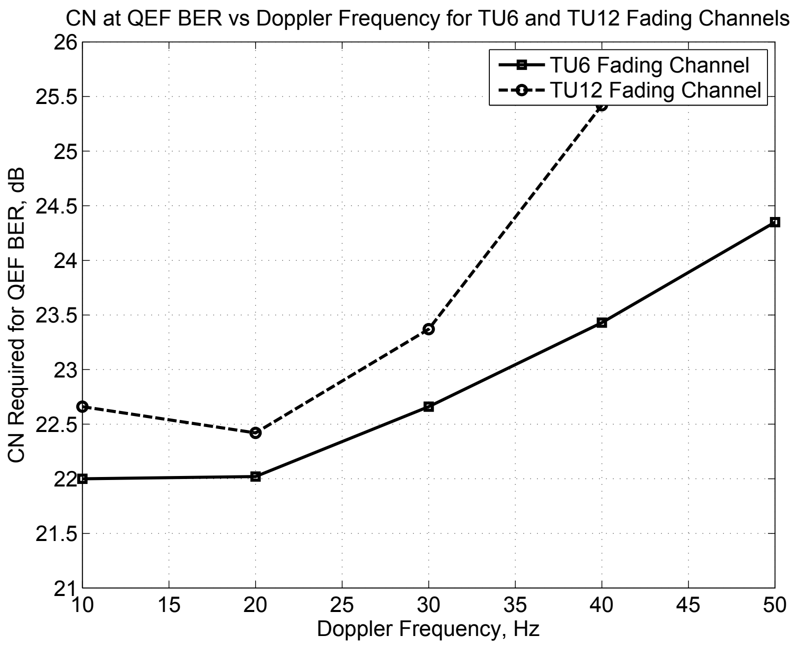
Measurement results were recorded by reading the internal registers of the OFDM receiver via an I^2C interface according to the experimental set-up in Figure 4.8. Each point on the BER curve is based on the average value of the BER register over a three minute interval, corresponding to the transmission of a payload of approximately $20\ Mbps \times 180\ s = 3.6\ Gbits$.

Figure 4.10(a) shows the effect of stuck-at faults on the BER for an AWGN channel at QEF $CN = 18.5\ dB$ with a burst of $N_{SA} = 10$ and $N_{SA} = 400$ alternating s-a-0 and s-a-1 memory faults for a mixed de-interleaver word format. An order of magnitude increase in the number of memory stuck-at faults results in a 1.6-fold increase in the BER at $CN = 18.5\ dB$ for an AWGN channel.

Figure 4.10(b) shows the CN required to achieve QEF BER vs Doppler frequency for TU-6 and TU-12 fading channels in the nominal, fault-free case. The wireless channel emulator was configured with multi-path delay and gain parameters for the 6 (TU-6) and 12 (TU-12) tap fading channels according to Appendix A. A static frequency shift model was used to emulate the Doppler effect: $f_d = f_c\ (\nu/c)$, where f_c is the

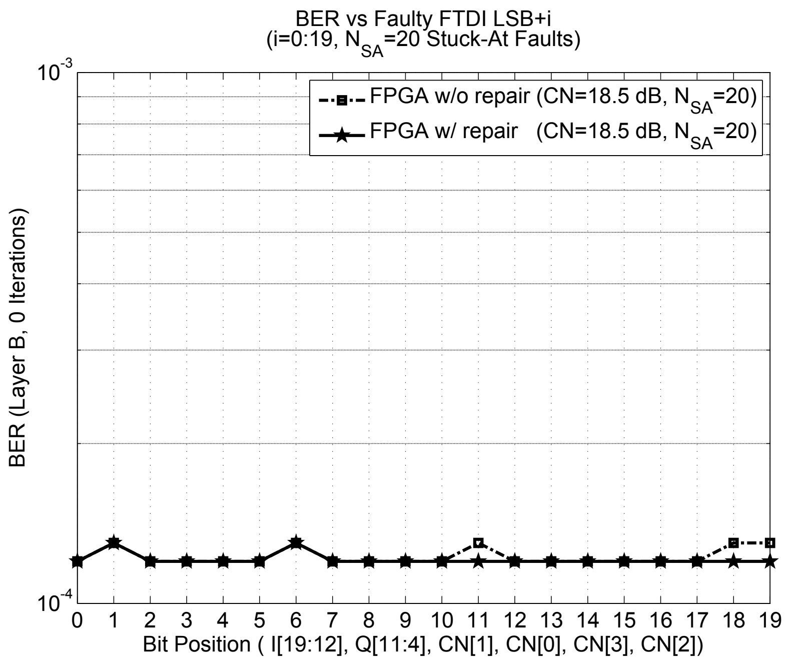


(a) BER vs Faulty FTDI LSB+i Data Comparison for $N_{SA} = 10$ and $N_{SA} = 400$ Stuck-At Faults.

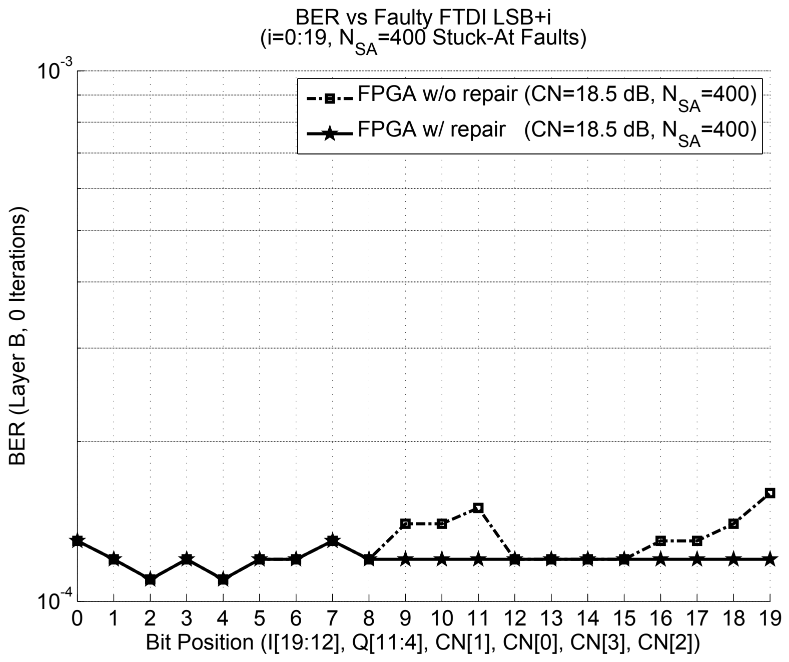


(b) CN required for QEF BER vs Doppler Frequency for TU-6 and TU-12 Fading Channels.

Figure 4.10: Measured Results (FPGA).

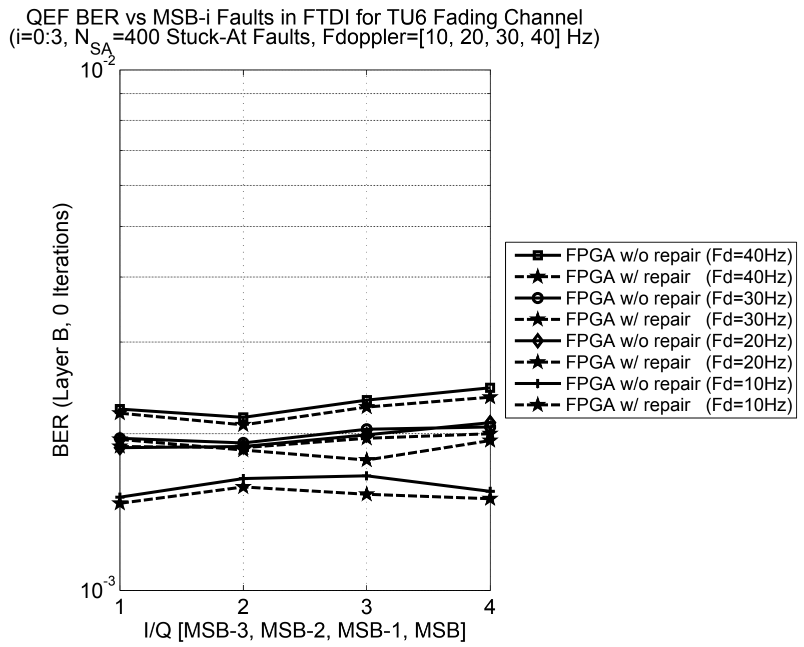


(a) $N_{SA} = 20$ Stuck-At Faults.

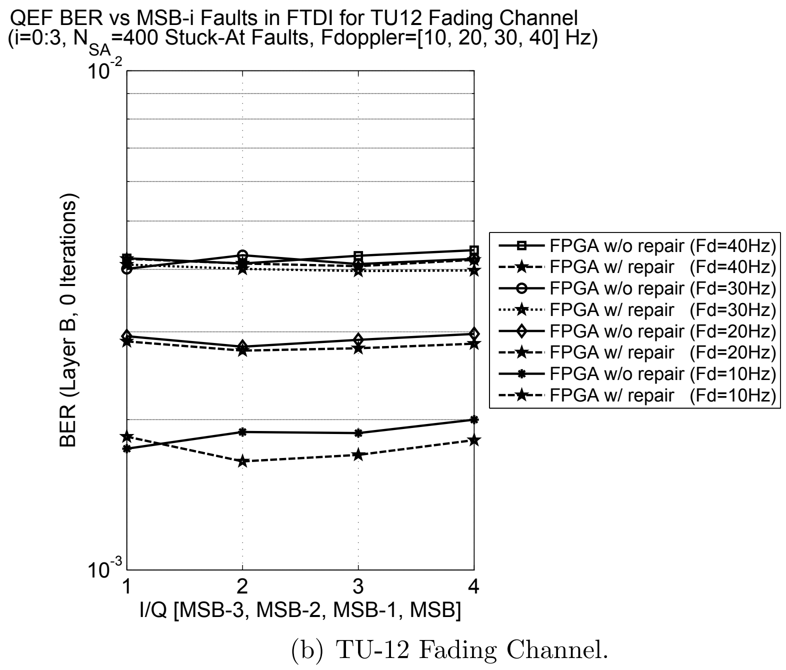


(b) $N_{SA} = 400$ Stuck-At Faults.

Figure 4.11: Measured Results (FPGA): BER vs Faulty LSB+i Data for an AWGN Channel.



(a) TU-6 Fading Channel.



(b) TU-12 Fading Channel.

Figure 4.12: Measured Results (FPGA): QEF BER vs MSB-i Faults for TU-6 and TU-12 Fading Channels.

carrier frequency, ν is the receiver velocity, and c is the speed of light ($\approx 3\times10^8\ m/s$). As expected, higher CN values are needed to compensate for mobility of the receiver.

Figures 4.11(a) and 4.11(b) show the deviation in QEF BER for high ($N_{SA} = 400$) and low ($N_{SA} = 20$) fault cases, with and without proposed memory repair for a single deinterleaver word format and an AWGN channel. In both cases, the MSB region consists of $\{I[19:16],\ Q[11:8],\ CN[3:2]\}$, while the LSB region is defined as $\{I[15:12],\ Q[7:4],\ CN[1:0]\}$. The dashed line shows the increase in BER for deinterleaver memory with N_{SA} faults. The solid line represents the BER when the proposed memory repair is enabled. As a result of the permutation of MSB and LSB regions, the proposed repair strategy achieves the fault sensitivity exhibited by the LSB region for MSB region data, whenever the MSB region has faults and the LSB region is fault-free.

The gain improvement ΔCN due to the proposed repair is calculated by dividing the maximum difference in the MSB BER in Figure 4.11(b) by the slope of the BER plot at the QEF BER=2×10^{-4} in Figure 3.12:

$$\Delta CN = \left.\frac{\Delta BER}{slope}\right|_{QEF} = \left.\frac{BER_{I[15]} - BER_{I[19]}}{slope}\right|_{QEF} = \frac{(0.00012 - 0.00016)}{(-2.6\times10^{-4}\ dB^{-1})} = 0.15\,dB.$$

The measured ΔCN is 0.2 dB smaller the simulated ΔCN in Section 3.6. The 0.2 dB loss could be attributed to the RF tuner card, which was not modeled in the simulation.

Figures 4.12(a) and 4.12(b) show the deviation in QEF BER for a TU-6 and TU-12 fading channel models in the presence of $N_{SA} = 400$ memory burst faults every 190 OFDM symbols with and without the proposed memory repair for the top 4 bits of I and Q. The dashed line representing the proposed memory repair shows a smaller QEF BER degradation in comparison to memory without repair over all Doppler frequencies. In the case of the MSB fault and $F_d = 40\ Hz$ for TU-6 channel, the proposed repair reduces the BER from 0.00245 to 0.00235 or 4.1% decrease with zero decoding iterations. For the TU-12 channel, the MSB BER with $F_d = 40\ Hz$ is reduced from 0.00437 to 0.00418 or 4.4% decrease with zero decoding iterations.

4.6 Summary

The proposed memory repair strategy is implemented as part of a larger Design-for-Test (DFT) on-chip infrastructure and interfaces with Built-In Self-Test (BIST) memory wrappers. Each memory wrapper is controlled by a Self-Test-And-Repair (STAR) Processor that directs the parallel execution of march tests and communicates fault information externally via SECT and JTAG interfaces.

The proposed VLSI architecture of the memory repair strategy provides a configurable area overhead of $\lceil M/i \rceil$ bits where M is the number of memory rows and i is an integer between 1 and M equal to the number of memory rows assigned to a single bit of the row address fault register. The memory repair architecture introduces a single multiplexer data-path latency and zero additional clock cycles required to access the memory. The frequency-time de-interleaver memory with repair operates at 69.4 MHz and dissipate 1,885 μW based on the worst-case (SS, 1.08V, 125C) synthesis results in 65-nm CMOS. The memory repair can be configured to occupy 1.7% of the 19.4 Mbit de-interleaver memory area by assigning every $i = 4$ memory rows to a single bit of the row address fault register.

The repair strategy was verified on a multiple-FPGA platform interfaced with an ISDB-T signal generator and a wireless channel emulator. Measurements results for an AWGN channel show a gain improvement of 0.15 dB in the presence of a burst of $N_{SA} = 400$ stuck-at memory faults in the de-interleaver with zero FEC decoding iterations. Measurement results for TU-6 / TU-12 fading channels show a 4% improvement in BER in the presence of $N_{SA} = 400$ deinterleaver faults, when the proposed repair is enabled compared to the deinterleaver memory without repair.

5 Conclusion

5.1 Summary of Contributions

A memory repair strategy is proposed for an embedded memory SoC OFDM receiver. The proposed repair strategy saves implementation cost by eliminating redundancy or local error correction in favor of forward error correction and improves decoding performance in the presence of memory faults by permuting stored data so as to minimize the impact of memory faults on system performance as measured by the bit error rate.

The repair strategy was investigated on a 19.4-Mbit frequency-time de-interleaver embedded SRAM memory occupying 12.17 mm^2 or 62% of the OFDM receiver SoC core area in 65-nm CMOS. The impact of stuck-at faults on the bit error rate of the receiver as well as the probability of their occurrence in the de-interleaver memory was evaluated. A measured degradation of 0.15 dB at 2×10^{-4} QEF BER was observed for an AWGN channel in the presence of $N_{SA} = 400$ uniformly distributed stuck-at faults with zero forward error correction decoding iterations. The proposed memory repair introduces a configurable area overhead of $\lceil M/i \rceil$ bits, where M is the number of memory rows and i is an integer between 1 and M, inclusive. The de-interleaver memory is located before forward error correction blocks illustrated by the soft-output Viterbi and Reed-Solomon decoders, and as a result, it has a higher degree of fault tolerance. Thus, costly redundancy or local error correction can be replaced with system-level forward error correction, thereby saving implementation cost.

To improve bit error performance in the presence of memory faults, a soft repair technique without redundancy is proposed. The proposed repair technique assigns a sensitivity coefficient for each memory cell based on the impact of cell fault on a system performance metric such as the bit error rate. Thus, the memory array is divided into blocks of high and low sensitivity to memory faults. To minimize the impact of memory faults, the data block is permuted such that bits with higher fault sensitivity coefficient are assigned fault-free memory locations while bits with lower fault sensitivity coefficient are assigned faulty memory locations.

The proposed repair architecture does not introduce additional clock cycles required to access the memory. It provides minimum latency overhead since the only additional data path delay is introduced by a 2-to-1 MUX during write and read operations.

The proposed memory repair introduces a configurable area overhead due to an external row address fault register of size $\lceil M/i \rceil$ bits used to store the repair information, where M is the number of memory rows and i is the number of rows assigned to a single bit of the row address fault register. In the case of de-interleaver memory with $M = 1024$, a repair overhead of 1.7% of memory area can be achieved with $i = 4$ based on the worst-case PVT (SS, $1.08V$, 125°C) synthesis results in 65-nm CMOS. The proposed repair strategy is generic enough to support multiple Memory Built-In Self-Test (MBIST) hardware. The proposed repair strategy is integrated with Virage DFT architecture of an ISDB-T OFDM receiver in 65-nm CMOS and verified on a Virtex-5 LX330 multiple-FPGA platform for AWGN and TU-6 / TU-12 fading channels.

5.2 Future Directions

Storage of Error Information. The proposed soft-repair technique stores error information in a row address fault register, where each bit marks one or more memory rows as faulty. The size of the register can be adjusted to match the expected number of faults according to empirical fault data for a given memory process technology. Alternative ways of storing error information can be developed based on a different set of design constraints. For instance, a separate memory can be used to store the repair information for all deinterleaver repairable memory blocks, or alternatively, a single column can be added to each repairable deinterleaver memory block to record repair information. If the number of memory access cycles is allowed to increase, a stage of address comparators can be added in the data-path to determine when to activate the repair logic. Furthermore, spare eFuses can be utilized to store row address fault information generated by a diagnostic MBIST test. In the case of a programmable eFuse box, an accumulative repair strategy can be adopted; otherwise, spare one-time programmable eFuses may be used to store the location of memory errors to save soft repair implementation costs.

Support of Multiple Fault and Memory Types. The proposed repair strategy focuses on the dominant stuck-at faults in the embedded SRAM memory. The repair strategy can be augmented to include coupling faults, stuck open faults, transition and data retention faults. For example, permutation of data across multiple rows can be added to resolve fault coupling in the vertical direction. Furthermore, the repair technique can be extended to other volatile and non-volatile memories such as DRAM and Flash.

Empirical Fault Distribution. The proposed soft-repair technique can be adjusted to suit a particular fault distribution. An accurate fault map and fault description can increase the effectiveness of memory repair customized for a particular memory process. Memory process parameters and error distribution may be obtained from the foundry or from memory characterization tests averaged over a statistically significant number of fabricated SoCs.

Generalizing Sensitivity Coefficient. The idea of assigning a sensitivity coefficient for each bit in a memory word can be generalized and included as part of a memory compiler. For example, it is always true that the MSB is more sensitive to error than the LSB. Therefore, data permutation logic can be integrated with existing memory either externally or as part of memory layout, taking into account the differences between logical and physical memory cell locations, to further reduce implementation overhead. In addition, as bits travel through the signal processing chain, their fault sensitivity coefficients are modified. In the case of de-interleaver memory, the bits are processed by an identity function, i.e. bits are not modified over time; however, other signal processing block such as FFT, Viterbi and Reed-Solomon decoders perform linear and non-linear arithmetic operations on collections of bits over time. Thus, a generalized theory may be possible that identifies subsystems within the SoC that are most likely to benefit from memory repair.

A ISDB-T Standard

A.1 ISDB-T Overview

Integrated Services Digital Broadcasting - Terrestrial (ISDB-T) is a digital broadcasting multi-carrier system with a variety of reception features and formats [3]. The ISDB-T features three transmission modes for video, audio and data transmission over a bandwidth of 5.575MHz (Table A.1). The transmission band is divided into 13 OFDM segments each segment having a bandwidth of 6/14 MHz (Table A.2). Each OFDM segment consists of 108, 216, or 432 sub-carriers depending on the transmission mode. An ISDB-T receiver is capable of demodulating QPSK, DQPSK, 16-QAM and 64-QAM constellations; it supports inter-segment and intra-segment frequency and time de-interleaving and guard intervals of 1/4, 1/8, 1/16 and 1/32 of the symbol length to accommodate different broadcasting network configurations and Doppler shifts arising in mobile reception. The ISDB-T receiver uses a concatenation of convolutional inner code and Reed-Solomon outer code for error correction and decodes information at a bit rate of 3.65 Mbps to 23.23 Mbps.

A.2 OFDM Transmission Parameters

Table A.1 summarizes OFDM segment transmission parameters. OFDM signal transmission parameters are presented in Table A.2.

Mode	Mode 1		Mode 2		Mode 3	
Bandwidth	3000/7 = 428.57··· KHz					
Spacing between carrier frequencies	250/63 = 3.968···KHz		125/63 = 1.98841··· KHz		125/126 = 0.99206··· KHz	
Total Sub-carriers	108	108	216	216	432	432
Data Sub-carriers	96	96	192	192	384	384
Scattered Pilots	9	0	18	0	36	0
Continual Pilots	0	1	0	1	0	1
Control (TMCC)	1	5	2	10	4	20
Auxiliary Channel 1	2	2	4	4	8	8
Auxiliary Channel 2	0	4	0	9	0	19
Carrier Modulation Scheme	QPSK 16QAM 64QAM	DQPSK	QPSK 16QAM 64QAM	DQPSK	QPSK 16QAM 64QAM	DQPSK
Symbols Per Frame	204					
Effective Symbol Length	$252\mu s$		$504\mu s$		$1008\mu s$	
Guard Interval	$63\mu s(1/4)$ $31.5\mu s(1/8)$ $15.75\mu s(1/16)$ $7.875\mu s(1/32)$		$126\mu s(1/4)$ $63\mu s(1/8)$ $31.5\mu s(1/16)$ $15.75\mu s(1/32)$		$252\mu s(1/4)$ $126\mu s(1/8)$ $63\mu s(1/16)$ $31.5\mu s(1/32)$	
Frame Length	$64.26ms(1/4)$ $57.834ms(1/8)$ $54.621ms(1/16)$ $53.0145ms(1/32)$		$128.52ms(1/4)$ $115.668ms(1/8)$ $109.242ms(1/16)$ $106.029ms(1/32)$		$257.04ms(1/4)$ $231.336ms(1/8)$ $218.484ms(1/16)$ $212.058ms(1/32)$	
IFFT Sampling Frequency	$512/63 = 8.12698\cdots MHz$					
Inner Code	Convolutional Code (1/2, 2/3, 3/4, 5/6, 7/8)					
Outer Code	RS (204, 188)					

Table A.1: ISDB-T: OFDM Segment Parameters [3].

Mode	Mode 1	Mode 2	Mode 3
Number of OFDM Segments Ns	13		
Bandwidth	$3000/7(kHz) \times Ns + 250/63(kHz) = 5.575 \cdots MHz$	$3000/7(kHz) \times Ns + 125/63(kHz) = 5.573 \cdots MHz$	$3000/7(kHz) \times Ns + 125/126(kHz) = 5.572 \cdots MHz$
Number of segments of differential modulations	n_d		
Number of segments of synchronous modulations	$n_s(n_s + n_d = N_s)$		
Spacing between carrier frequencies	$250/63 = 3.968 \cdots kHz$	$125/63 = 1.984 \cdots kHz$	$125/126 = 0.992 \cdots kHz$
Total Subcarriers	$108 \times N_s + 1 = 1405$	$216 \times N_s + 1 = 2809$	$432 \times N_s + 1 = 5617$
Data Subcarriers	$96 \times N_s$=1248	$192 \times N_s = 2496$	$384 \times N_s = 4992$
Scattered Pilots	$9 \times n_s$	$18 \times n_s$	$36 \times n_s$
Continual Pilots	n_d+1	n_d+1	n_d+1
Control (TMCC)	$n_s + 5 \times n_d$	$2n_s + 10 \times n_d$	$4n_s + 20 \times n_d$
Auxiliary Channel 1	$2N_s = 26$	$4N_s = 52$	$8N_s = 104$
Auxiliary Channel 2	$4 \times n_d$	$9 \times n_d$	$19 \times n_d$
Carrier Modulation Scheme	QPSK, 16QAM, 64QAM, DQPSK		
Symbols Per Frame	204		
Effective Symbol Length	$252\mu s$	$504\mu s$	$1008\mu s$
Guard Interval	$63\mu s(1/4)$ $31.5\mu s(1/8)$ $15.75\mu s(1/16)$ $7.875\mu s(1/32)$	$126\mu s(1/4)$ $63\mu s(1/8)$ $31.5\mu s(1/16)$ $15.75\mu s(1/32)$	$252\mu s(1/4)$ $126\mu s(1/8)$ $63\mu s(1/16)$ $31.5\mu s(1/32)$
Frame Length	$64.26ms(1/4)$ $57.834ms(1/8)$ $54.621ms(1/16)$ $53.0145ms(1/32)$	$128.52ms(1/4)$ $115.668ms(1/8)$ $109.242ms(1/16)$ $106.029ms(1/32)$	$257.04ms(1/4)$ $231.336ms(1/8)$ $218.484ms(1/16)$ $212.058ms(1/32)$
Inner Code	Convolutional Code (1/2, 2/3, 3/4, 5/6, 7/8)		
Outer Code	RS (204, 188)		

Table A.2: ISDB-T: OFDM Transmission Signal Parameters [3].

A.3 Channel Models

Model	Path Delay, μs	Path Gain, dB
Urban	delay[0]=0.0	power[0]=-3.0
	delay[1]=0.2	power[1]= 0.0
	delay[2]=0.5	power[2]=-2.0
	delay[3]=1.6	power[3]=-6.0
	delay[4]=2.3	power[4]=-8.0
	delay[5]=5.0	power[5]=-10.0
Rural	delay[0]=0.0	power[0]= 0.0
	delay[1]=0.1	power[1]=-4.0
	delay[2]=0.2	power[2]=-8.0
	delay[3]=0.3	power[3]=-12.0
	delay[4]=0.4	power[4]=-16.0
	delay[5]=0.5	power[5]=-20.0
Bad Urban	delay[0]=0.0	power[0]=-2.5
	delay[1]=0.3	power[1]= 0.0
	delay[2]=1.0	power[2]=-3.0
	delay[3]=1.6	power[3]=-5.0
	delay[4]=5.0	power[4]=-2.0
	delay[5]=6.6	power[5]=-4.0

Table A.3: Channel Models.

Model	Path Delay, μs	Path Gain, dB
Portable Indoor	delay[0]=0.0	power[0]= 0.0
	delay[1]=0.1	power[1]=-6.4
	delay[2]=0.2	power[2]=-10.4
	delay[3]=0.4	power[3]=-13.0
	delay[4]=0.6	power[4]=-13.3
	delay[5]=0.8	power[5]=-13.7
	delay[6]=1.0	power[6]=-16.2
	delay[7]=1.6	power[7]=-15.2
	delay[8]=8.1	power[8]=-14.9
	delay[9]=8.8	power[9]=-16.2
	delay[10]=9.0	power[10]=-11.1
	delay[11]=9.2	power[11]=-11.2
Portable Outdoor	delay[0]=0.0	power[0]= 0.0
	delay[1]=0.3	power[1]=-1.5
	delay[2]=1.0	power[2]=-3.8
	delay[3]=1.6	power[3]=-7.3
	delay[4]=5.0	power[4]=-9.8
	delay[5]=6.6	power[5]=-3.3
	delay[6]=5.0	power[6]=-5.9
	delay[7]=6.6	power[7]=-20.6
	delay[8]=5.0	power[8]=-19.0
	delay[9]=6.6	power[9]=-17.7
	delay[10]=5.0	power[10]=-18.9
	delay[11]=6.6	power[11]=-19.3

Table A.4: Channel Models.

B Interleaver Pseudocode

Algorithm 2 Block Interleaver (out, m, n, in)

```
1: for $i = 1$ to $m$ do
2:   A[$i$, $1 : n$] $\Leftarrow$ in[$(i-1) \times n + 1$, $i \times n$]; // fill rows
3: end for
4: for $j = 1$ to $n$ do
5:   out[1, $(j-1) \times m + 1 : j \times m$] $\Leftarrow$ A[$1 : m$, $j$]; // read columns
6: end for
7: return out[1, $m \times n$]
```

Algorithm 3 Block Deinterleaver (out, m, n, in)

```
1: for $j = 1$ to $n$ do
2:   A[$1 : m$, $j$] $\Leftarrow$ in[$(j-1) \times m + 1$, $j \times m$]; // fill columns
3: end for
4: for $i = 1$ to $m$ do
5:   out[1, $(i-1) \times n + 1 : i \times n$] $\Leftarrow$ A[$i$, $1 : n$]; // read rows
6: end for
7: return out[1, $m \times n$]
```

Algorithm 4 Convolutional Interleaver (out, m, n, d, in)

```
1: for $i = 1$ to $m$ do
2:   shift_reg_i[$(i-1) \times d$, 0] $\Leftarrow$ [shift_reg_i[$(i-1) \times d - 1$, 1], in[i]]; // shift input
3:   shift_out_i $\Leftarrow$ shift_reg_i[$(i-1) \times d$]; // shift output
4:   shift_out $\Leftarrow$ [shift_out, shift_out_i];
5: end for
6: return shift_out[1, $m$]
```

Algorithm 5 Convolutional Deinterleaver (out, m, n, d, in)

```
1: for $i = 1$ to $m$ do
2:   shift_reg_i[$(m - i) \times d$, 0] $\Leftarrow$ {shift_reg_i[$(m - i) \times d - 1$, 1], in[i]}; // shift input
3:   shift_out_i $\Leftarrow$ shift_reg_i[$(m - i) \times d$]; // shift output
4: end for
5: return shift_out[1, $m$]
```

Algorithm 6 Helical Interleaver (out, m, ngrp, in)

```
1: for $j = 1$ to $m$ do
2:   // init $m$ columns with $ngrp$ elements in a helical fashion
3:   A[$j : ngrp + j - 1$, $j$] $\Leftarrow$ in[$(j - 1) \times ngrp + 1$, $j \times ngrp$];
4: end for
5: for $i = 1$ to $ngrp + m - 1$ do
6:   out[1, $(i - 1) \times m + 1 : i \times m$] $\Leftarrow$ A[$i$, $1 : m$]; // read rows
7: end for
8: return out[1, $m \times (ngrp + m - 1)$]
```

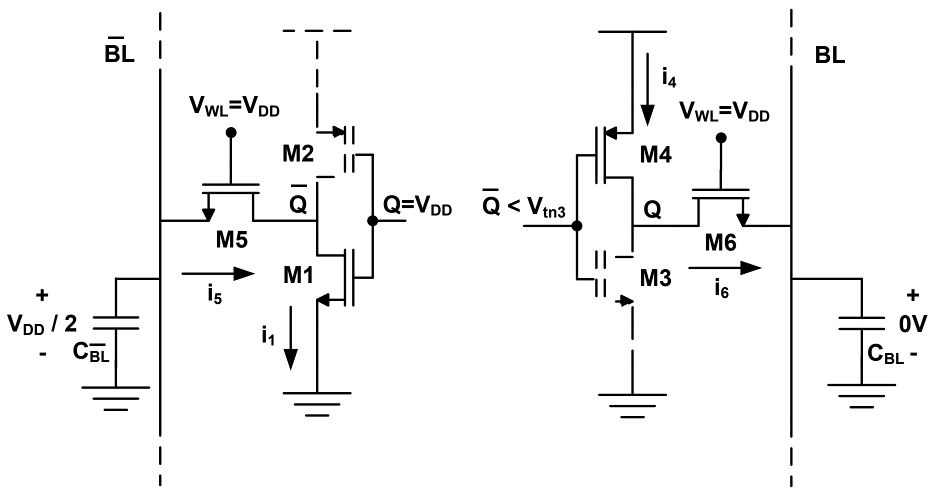
Algorithm 7 Helical Deinterleaver (out, m, ngrp, in)

```
1: for $i = 1$ to $ngrp + m - 1$ do
2:   A[$i$, $1 : m$] $\Leftarrow$ in[$(i - 1) \times m + 1 : i \times m$]; // fill rows
3: end for
4: for $j = 1$ to $m$ do
5:   // read $m$ columns with $ngrp$ elements in a helical fashion
6:   out[$(j - 1) \times ngrp + 1$, $j \times ngrp$] $\Leftarrow$ A[$j : ngrp + j - 1$, $j$];
7: end for
8: return out[1, $m \times (ngrp + m - 1)$]
```

C SRAM Embedded Memory

C.1 The 6-T SRAM Cell

The 6-T SRAM cell first inroduced in Figures 3.6(a) and 3.6(b) consists of two cross-coupled inverters and two access transistors. The cross-coupled inverters regenerate the data through positive feedback, while access transistors provide bi-directional current flow between the bit lines and the memory cell, once activated by the word line.

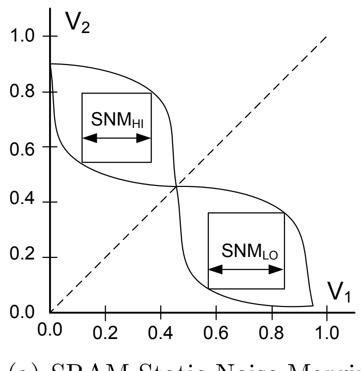


(a) Reading a “1”. (b) Writing a “0”.

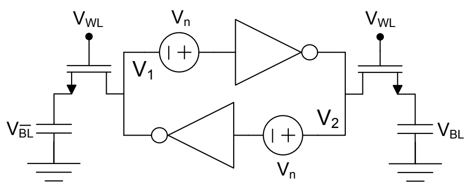
Figure C.1: 6-T SRAM Read Stability and Write Ability.

In the design of an SRAM cell, it is important to maintain *read stability* and *write ability*. Figure C.1(a) shows the active half-circuit during a read operation, assuming that the cell is storing a logic “1”, i.e. $V_Q = V_{DD}$. Before the read operation, the $\overline{BL}$ is precharged to an intermediate voltage of $V_{DD}/2$. Because the Q output is equal to a logic “1”, $|V_{gs2}| > |V_{tp2}|$ and M2 is in the cut-off region; M5 is in saturation ($V_{gs5} > V_{tn5}$, $V_{ds5} \approx V_{DD}/2 \geq V_{gs5} - V_{tn5}$); and M1 is in triode ($V_{gs1} > V_{tn1},\ V_{ds1} \leq V_{gs1} - V_{tn1}$). To maintain read stability, voltage $V_{\overline{Q}}$ must be designed to stay below $V_{DD}/2$ for a symmetric inverter to prevent the cell from changing its state during a read operation. Therefore, M1 is sized larger than M5, i.e. $(\frac{W}{L})_1 > (\frac{W}{L})_5$. As a result, $C_{\overline{BL}}$ is discharged with a current $i_{5,SAT} = i_{1,TRI}$. The time it takes to sense the voltage difference ΔV on $C_{\overline{BL}}$ in addition to the rise time due to the RC time constant of the word line is approximately equal to the memory read time: $t_{read} \approx t_{BL} + t_{WL} \geq C_{\overline{BL}}\Delta V/i_5 + \tau_{WL}$.

Figure C.1(b) shows the half-circuit during a write operation, assuming the cell is storing a logic "1". To write a logic "0", the BL is lowered to $0V$, while $\overline{BL}$ is raised to V_{DD}. Having sized the access and pull down transistors for read stability, we expect the V_{ds} of M1 to stay below the threshold voltage V_{tn3} of M3. Therefore, to change the stored information bit, instead of attempting to raise the $V_{\overline{Q}}$ to V_{DD}, we need to reduce V_Q below the threshold voltage V_{tn1} of M1. When $V_Q = V_{tn1}$, M6 operates in the triode region ($V_{gs6} = V_{DD} > V_{tn6},\ V_{ds6} \approx V_{tn1} \leq V_{gs6} - V_{tn6}$), and M4 operates in saturation ($|V_{gs4}| = V_{DD} - V_{tn3} > |V_{tp4}|,\ |V_{ds4}| \approx V_{DD} - V_{tn1} \geq |V_{gs4}| - |V_{tp4}|$). Thus, the capacitance at node Q is discharged via a current $i_{4,SAT} = i_{6,TRI}$. Since the pull-up transistor M4 opposes the lowering of the voltage at node Q, it must be designed smaller then the access transistor M6, i.e. $(\frac{W}{L})_4 < (\frac{W}{L})_6$. When V_Q reaches $V_{DD}/2$, the regenerative feedback will cause the cell to switch state. The time it takes C_Q to discharge to $V_{DD}/2$ in addition to the rise time of WL and the time required for the propagation delay of the inverter is approximately equal to the memory write time: $t_{write} \approx t_{BL} + t_{WL} + t_{PD} \geq C_{\overline{Q}}(V_{DD}/2)/i_6 + \tau_{WL} + t_{PD}$. Therefore, to achieve both



(a) SRAM Static Noise Margin (SNM).



(b) SRAM Cell SNM Test-Bench.

Figure C.2: 6-T SRAM Static Noise Margin.

read stability and write ability, the inverter pull-up (p), pull-down (n) and access (a) transistors are sized such that: $(\frac{W}{L})_p < (\frac{W}{L})_a < (\frac{W}{L})_n$ [50].

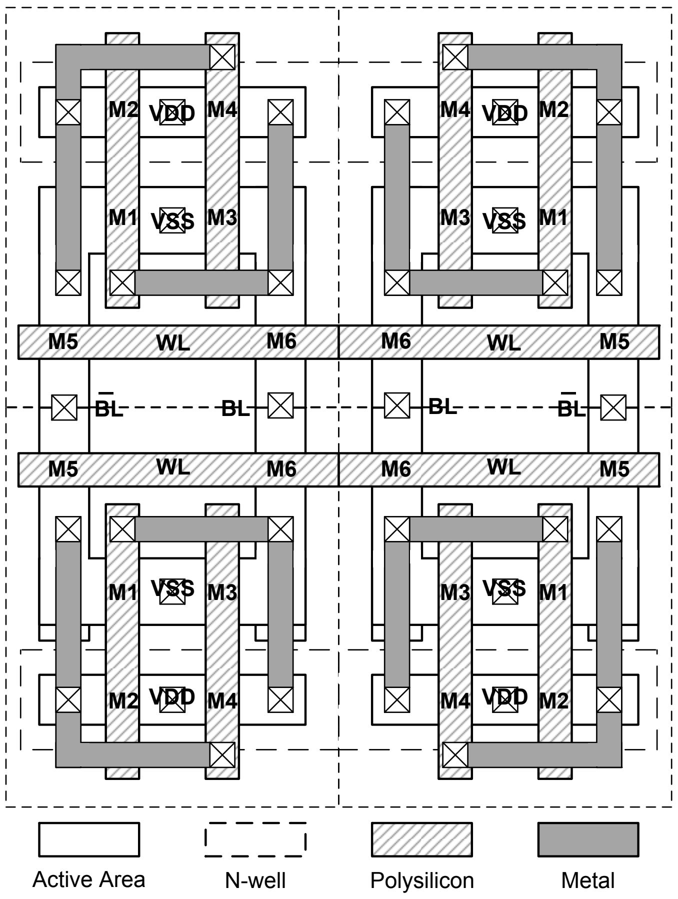


Figure C.3: 6-T SRAM 2 × 2 Memory Array Layout.

The stability and write ability of an SRAM cell are quantified by static noise margins in various modes of operation. The *static noise margin* (SNM) measures how much noise can be applied to the inputs of the cross-coupled inverters before a stable state is lost (during hold or read) or switched (during write) [51]. Figure C.2(a) shows a butterfly diagram for a static hold margin along with the test-bench in Figure C.2(b). To determine the hold margin, V_{WL} and V_{BL} are set to $0V$ and V_2 is plotted against V_1 and vice versa. If the inverters are identical the DC curves are inverse functions of each other and therefore symmetric around $V_2 = V_1$. The static noise margin is determined as the length of the side of the maximum square that can be inscribed between the curves. In case the inverters are not identical, the SNM is taken as min(SNM_{HI}, SNM_{LO}). The word line, bit line, and the noise voltage sources can be adjusted to imitate read and write cell conditions to obtain butterfly diagrams for read and write static noise margins. Furthermore, the noise margins can be qualified by *dynamic* conditions such as cell access time and recovery time [52].

A number of layout techniques can be used to minimize the area occupied by an SRAM memory array. Figure C.3 shows an area-efficient layout consisting of four 6-T SRAM cells [53]. The SRAM cell area can be reduced by observing minimum design rules and sharing power, ground and bit line contacts between the two inverters. The layout of a single SRAM cell can be re-used to minimize mismatch. Moreover, a symmetric layout in which the SRAM cell is mirrored around the horizontal axis allows the sharing of N-well regions between adjacent cell rows, thereby greatly reducing memory area. A symmetric layout introduces regularity and simplifies routing complexity, in addition to reducing the impact of process parameter gradients and common mode noise in the case of differential signal processing. Furthermore, a capacitive coupling between adjacent bit lines can be reduced through bit line twisting, thereby increasing memory speed. Finally, the dimensions of the memory layout that approach a square provide balanced lengths and therefore balanced capacitances and delays of word lines and bit lines.

References

[1] ITRS, "Design Report Update," *International Technology Roadmap for Semiconductors*, pp. 1–6, 2000.

[2] ITRS, "Design Report," *International Technology Roadmap for Semiconductors*, pp. 1–42, 2009.

[3] ARIB, "Transmission System for Digital Terrestrial Television Broadcasting," *STD-B31 Version 1.6*, November 2005.

[4] V. Logic, "STAR Memory System," *Virage Logic Product Training*, 2011.

[5] A. van de Goor, C. Jung, S. Hamdioui, and G. Gaydadjiev, "Low-Cost, Customized and Flexible SRAM MBIST Engine," *International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, pp. 382–387, 2010.

[6] S. M. Al-Harbi and S. K. Gupta, "An Efficient Methodology for Generating Optimal and Uniform March Tests," *VLSI Test Symposium*, pp. 231–237, 2001.

[7] Y. Zorian, "Embedded Memory Test and Repair: Infrastructure IP for SOC Yield," *ITC International Test Conference*, pp. 340–348, 2002.

[8] J. E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Transactions on Electronic Computers*, vol. 8, no. 3, pp. 330–334, 1959.

[9] P. K. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 Years of CORDIC: Algorithms, Architectures, and Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 9, pp. 1893–1907, September 2009.

[10] C. D. Thompson, "Fourier Transforms in VLSI," *IEEE Transactions on Computers*, vol. 32, no. 11, pp. 1047–1057, November 1983.

[11] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, 1st ed. Prentice Hall, June 1975.

[12] P. Duhamel and M. Vetterli, "Fast Fourier Transforms: A Tutorial Review and a State of the Art," *Signal Processing*, vol. 19, no. 4, pp. 259–299, April 1990.

[13] A. J. Viterbi, "Convolutional Codes and Their Performance in Communication Systems," *IEEE Transactions on Communications Technology*, vol. 19, no. 5, pp. 751–772, October 1971.

[14] J. Hagenauer and P. Hoeher, "A Viterbi Algorithm with Soft-Decision Outputs and its Applications," *Global Telecommunications Conference*, vol. 3, pp. 1680–1686, November 1989.

[15] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *SIAM Journal of Applied Mathematics*, vol. 8, no. 2, pp. 300–304, June 1960.

[16] J. L. Massey, "Shift Register Synthesis and BCH Decoding," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, January 1969.

[17] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A Method for Solving Key Equation for Decoding Goppa Codes," *Information and Control*, vol. 27, pp. 87–99, 1975.

[18] R. T. Chien, "Cyclic Decoding Procedures for Bose-Chaudhuri-Hocquenghem Codes," *IEEE Transactions on Information Theory*, vol. 10, no. 4, pp. 357–363, October 1964.

[19] D. Divsalar, H. Jin, and R. J. McEliece, "Coding Theorems for "Turbo-Like" codes," *Proceedings of Allerton Conference on Communication, Control and Computing*, pp. 201–210, 1998.

[20] S. ten Brink, "Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes," *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, October 2001.

[21] C. M. Rader, "Memory Management in a Viterbi Decoder," *IEEE Transactions on Communications*, vol. 29, no. 9, pp. 1399–1401, September 1981.

[22] G. Feygin and P. G. Gulak, "Survivor Sequence Memory Management in Viterbi Decoders," *IEEE International Symposium on Circuits and Systems*, vol. 5, no. 3, pp. 2967–2970, June 1991.

[23] W. J. Gross, "Implementation of Algebraic Soft-Decision Reed-Solomon Decoders," Ph.D. dissertation, University of Toronto, 2003.

[24] W. J. Gross, F. R. Kschischang, and P. G. Gulak, "Architecture and Implementation of an Interpolation Processor for Soft-Decision Reed-Solomon Decoding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 3, pp. 309–318, March 2007.

[25] J. L. Ramsey, "Realization of Optimum Interleavers," *IEEE Transactions on Information Theory*, vol. 16, no. 3, pp. 338–345, May 1970.

[26] D. Forney, "Burst-Correcting Codes for the Classic Bursty Channel," *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 772–781, October 1971.

[27] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed. McGraw-Hill, 2008.

[28] E. R. Berlekamp and P. Tong, "Interleavers for Digital Communications," *U. S. Patent 4,559,625*, 1985.

[29] M. Sasaki, "Technologies and Services of Digital Broadcasting," *Broacast Technology*, no. 20, pp. 14–19, Autumn 2004.

[30] Z. Wang and Q. Li, "Very Low-Complexity Hardware Interleaver for Turbo Decoding," *IEEE Transactions on Circuits and Systems-II: Express Briefs*, vol. 54, no. 7, pp. 636–640, July 2007.

[31] K. K. Parhi, "Approaches to Low-Power Implementations of DSP Systems," *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, vol. 48, no. 10, pp. 1214–1224, October 2001.

[32] S. Singh, "Fault Mapping Apparatus for Memory," *U. S. Patent 5,233,614*, 1993.

[33] N. S. Jayant and J. C. Lagarias, "Method and Apparatus for Providing Error-Tolerant Storage of Information," *U. S. Patent 5,892,775*, 1999.

[34] F. J. Aichelmann and L. K. Lange, "Memory Address Permutation Apparatus," *U. S. Patent 4,506,364*, 1985.

[35] C. Stapper, F. Armstrong, and K. Saji, "Integrated Circuit Yield Statistics," *Proceedings of the IEEE*, vol. 71, no. 4, pp. 453–470, April 1983.

[36] M. Ottavi, L. Schiano, X. Wang, Y.-B. Kim, F. J. Meyer, and F. Lombardi, "Evaluating the Yield of Repairable SRAMs for ATE," *IEEE Transactions on Instrumentation and Measurement*, vol. 55, no. 5, pp. 1704–1712, October 2006.

[37] I. Koren and Z. Koren, "Defect Tolerance in VLSI Circuits: Techniques and Yield Analysis," *Proceedings of the IEEE*, vol. 86, no. 9, pp. 1819–1836, September 1998.

[38] W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *Journal of Applied Physics*, vol. 19, no. 1, pp. 55–63, January 1948.

[39] R. Dekker, F. Beenker, and L. Thijssen, "Fault Modeling and Test Algorithm Development for Static Random Access Memories," *IEEE International Test Conference*, pp. 343–352, 1988.

[40] V. D. Agrawal, C. R. Kime, and K. Saluja, "A Tutorial on Built-In Self-Test. Part 1: Principles," *IEEE Design and Test of Computers*, vol. 10, no. 1, pp. 73–82, March 1993.

[41] V. D. Agrawal, C. R. Kime, and K. Saluja, “A Tutorial on Built-In Self-Test. Part 2: Applications,” *IEEE Design and Test of Computers*, vol. 10, no. 2, pp. 69–77, June 1993.

[42] S.-Y. Kuo and W. K. Fuchs, “Efficient Spare Allocation in Reconfigurable Arrays,” *23rd Design Automation Conference*, pp. 385–390, 1986.

[43] S. S. Skiena, *The Algorithm Design Manual*, 2nd ed. Springer, April 2008.

[44] S. Nakahara, M. Okano, M. Takada, and T. Kuroda, “Digital Transmission Scheme for ISDB-T and Reception Characteristics of Digital Terrestrial Television Broadcasting System in Japan,” *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 563–570, 1999.

[45] L.-M. Denq, T.-C. Wang, and C.-W. Wu, “An Enhanced SRAM BISR Design with Reduced Timing Penalty,” *15th Asian Test Symposium*, pp. 25–30, 2006.

[46] C.-D. Huang, J.-F. Li, and T.-W. Tseng, “ProTaR: An Infrastructure IP for Repairing RAMs in System-on-Chips,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1135–1143, October 2007.

[47] S.-K. Lu, C.-L. Yang, Y.-C. Hsiao, and C.-W. Wu, “Efficient BISR Techniques for Embedded Memories Considering Cluster Faults,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 2, pp. 184–193, February 2010.

[48] V. Smolyakov, P. G. Gulak, and J. Narsinghani, “Mixed ASIC/FPGA Scan-Based DFT Flow with Verigy 93K SoC Production Tester,” *Canadian Microelectronics Corporation (CMC) Application Note*, pp. 1–38, August 2010.

[49] S. Kilts, *Advanced FPGA Design: Architecture, Implementation, and Optimization*, 1st ed. Wiley, June 2007.

[50] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, 5th ed. Oxford University Press, 2004.

[51] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Addison Wesley, 2010.

[52] M. Sharifkhani and M. Sachdev, “SRAM Cell Stability: A Dynamic Perspective,” *IEEE Journal of Solid-State Circuits*, vol. 44, no. 2, pp. 609–619, February 2009.

[53] A. van de Goor and I. Schanstra, “Address and Data Scrambling: Causes and Impact on Memory Tests,” *Proceedings of the First IEEE International Workshop on Electronic Design, Test, and Applications*, pp. 128–136, January 2002.