A New Low Energy BIST Using A Statistical Code

Sunghoon Chun, Taejin Kim and Sungho Kang
Yonsei University

Contents

Introduction

- Two challenges for testing complex SoCs
  - Test data volume
  - Excessive energy consumption
- Two issues are conflicting goals
  - Large test data and more effective test data
    - Guarantee higher fault coverage
    - But, increasing energy consumption
- Many approaches have been researched

Proposed Method

- Propose a new pattern generator
  - A modified input reduction
    - To reduce test data and test power consumption
  - A statistical and simple compression code
    - To generate more effective and smaller test sets
  - A new test sequence skipping technique
    - To reduce wasted energy consumption with a lower hardware overhead during the self-testing

Energy Consumption Model

- Dynamic energy dissipation
  - Caused by the switching activity accounts which accounts for over 90% of the total energy consumption

- Use the extended version of the WTM (ETM)
  - To estimate the energy consumption of the CUT
  - A scan vector $P = \{p_1, p_2, \ldots, p_i\}$
  - A response $R = \{r_1, r_2, \ldots, r_i\}$

$$ETM = \sum_{j=1}^{i-1} (p_j \oplus p_{j+1}) \cdot (k-j) + \sum_{j=1}^{i-1} (r_j \oplus r_{j+1}) \cdot (k-j)$$

Proposed Low Energy BIST

- Overall algorithm of the low energy BIST generation

```
Low_Energy_BIST()
{
  /* Phase 1: preparing an initial test set */

      T = generate_pseudo-random_pattern();
      T' = don't_care_identification();
      T'IR = input_reduction();


  /* Phase 2: generating a pattern generator using the MICRO code */

      determine_the_compression_block();
      generate_the_weighted_logic();
      generate_the_decompressor_for_the_MICRO();
      generate_pattern_generator_using_the_MICRO_code();

  /* Phase 3: generating a skipping logic for low energy test */

      determine_success_stages();
      reorder_the_success_stages();
      calculate_the_seeds_for_the_success_stages();
      synthesize_the_skipping_logic();

}
```

Preparing an Initial Test Set

- Modified input reduction
  - Process to identify the compatible inputs and the inverse-compatible inputs for test
  - Use an original test pattern set
  - Require don't care identification process
  - Test set after the modified input reduction T_{IR}

Original Test Set

I_1	I_2	I_3	I_4
1	0	1	0
0	1	0	0
1	1	1	0
0	0	1	1

Don't care identification

I_1	I_2	I_3	I_4
1	X	1	X
X	1	0	0
X	1	1	X
0	0	X	1

Compatible Inputs

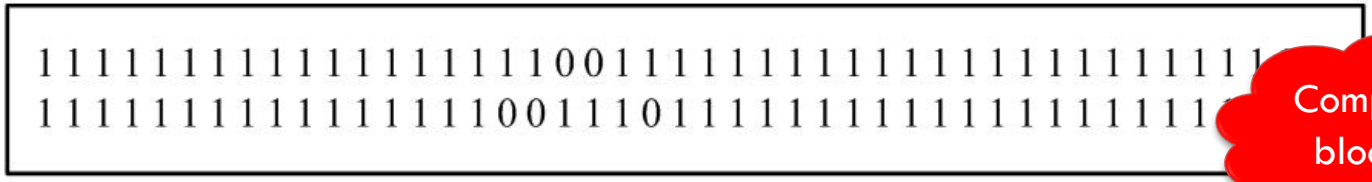
I_1	I_2	I_3	I_4
1	X	1	X
X	1	0	0
X	1	1	X
0	0	X	1

New Pattern Generator

- Huffman code
  - Close to entropy limits
  - Can be achieve the highest compression ratio among statistical codes for test data
  - High hardware overhead
- MICRO code
  - To appropriately satisfy both high compression ratio and low hardware overhead
  - The compression ratio is enhanced by increasing the occurrence frequency of one block

MICRO code

- Example of MICRO code
  - S935 ISCAS ‘89 benchmark circuits
  - Part of a test pattern

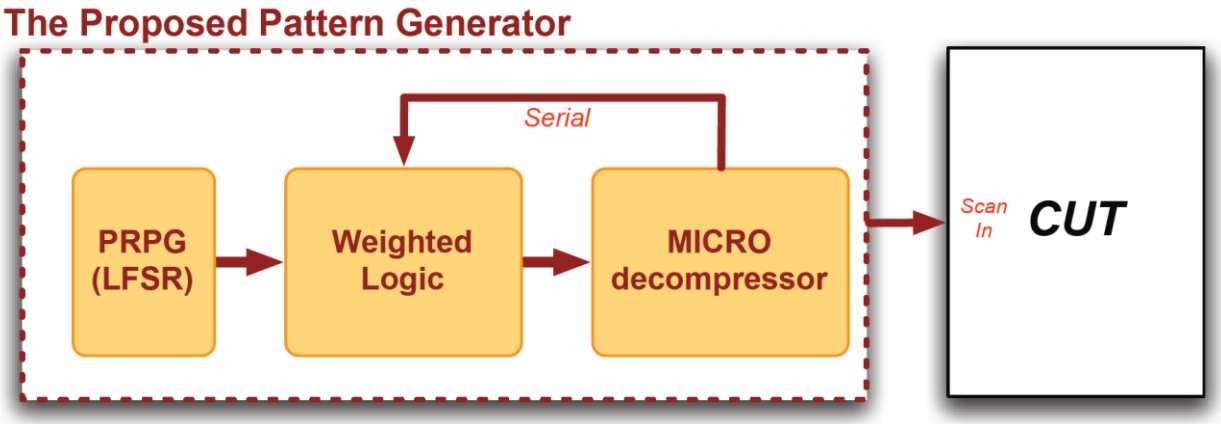


Compression block (*CB*)

Pattern Block	Huffman Code	Pattern Block	MICRO
1111	10	1111	0
1100	00	00	100
0011	110	01	101
1011	010	10	110
		11	111

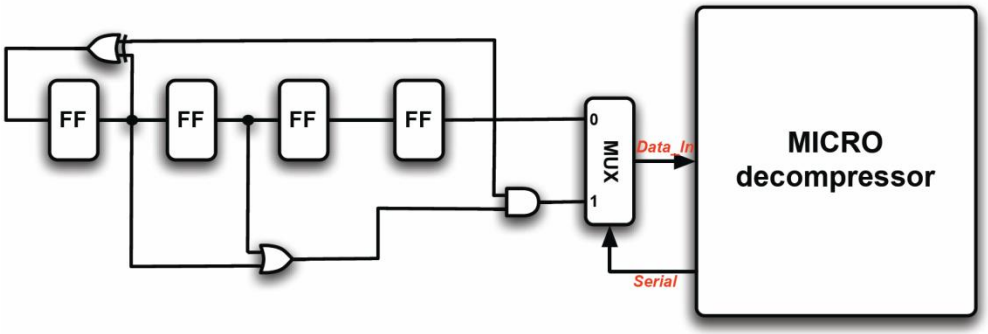
Pattern Generator Using MICRO Code

- General LFSR
- Weighted logic
  - Designed to make the compression block occur as much as the probability of the occurrence frequency of the *CB*
- MICRO decompressor
  - To generate test patterns from the test data encoded to MICRO code



Pattern Generator Using MICRO Code

- Example of the proposed pattern generator
  - Occurrence frequency of the compression block : 0.625
  - General LFSR : 4bit LFSR



- Advantages
  - Require fewer states than a general LFSR to achieve the same fault coverage
  - Can generate upwards of 2^n-1 stages
  - Therefore, it can reduce hardware overhead and energy consumption

Useless Patterns for Self-Testing

- Major sources of energy consumption during self-testing
  - Scan shifting for useless patterns

- Need a method to prevent unnecessary energy consumption

- Previous works
  - Inhibiting technique [Girard]
  - Filtering technique [Manich]
  - Still low energy reduction ratio
  - High hardware overhead for multiple LFSR inhibitions

Generating Skipping Logic (1)

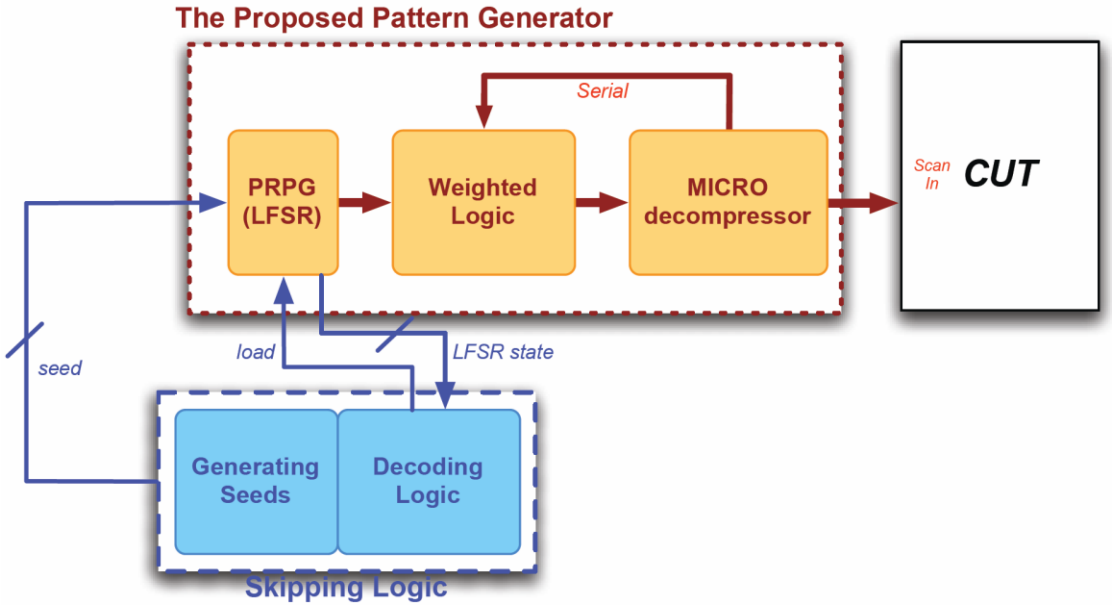
- Procedure of developing the skipping structure
  - Phase 1
    - Generating pseudo-random test sequence by the proposed pattern generator
    - Determining its single stuck-at fault coverage
    - Identifying the first and last vectors of each useless subsequence

Generating Skipping Logic (2)

- Procedure of developing the skipping structure
  - Phase 2
  - New success stage reordering algorithm
    - To find an optimal test sequence with a low hardware overhead
    - State transition graph (V,E)
      - V: success stage S_i
      - E: skipping constraint value (SCV) = $\alpha \cdot N_{DF} + \beta \cdot N_{Tr}$
      - N_{DF} : the number of detected faults of V after a predecessor
      - N_{Tr} : the number of transitions between success stages
    - Find the minimum success stages with minimum transitions using the state transition graph
    - Can be extended to reseeding technique

Generating Skipping Logic (3)

- Procedure of developing the skipping structure
  - Phase 3
    - Synthesize the skipping logic
      - To prevent unnecessary pattern subsequences
      - To jump the first vectors of success stages as the order obtained in the second phase



Experimental Results

- 10 scan chains for each benchmark circuit
- Generating polynomial equations by using LBISTArchitect

Circuits	General LFSR	[Girard]	[Manich]	[Zhang]	Previous Works
	Test data (bits)	Test data (bits)	Test data (bits)	Test data (bits)	F. C. (%)
s5378	10,700,000	5,485,248	1,376,448	537,140	84.39
s9234	12,350,000	7,390,240	2,726,880	679,250	96.42
s13207	35,000,000	19,801,600	10,035,200	2,940,000	97.78
s15850	30,550,000	18,124,704	6,041,568	1,857,440	93.61
s38417	116,480,000	34,305,408	21,305,856	9,967,360	98.25
s38584	102,480,000	50,654,400	26,351,360	9,311,040	98.90

Experimental Results

- Test data of the proposed method

Circuits	F.C (%)	Test data (w/o IR) (bits)	Test data (w/ IR) (bits)	# of skips	Reduction ratio
s5378	99.0	471,014	407,296	33	96.2
s9234	97.3	941,811	671,232	95	94.6
s13207	95.8	4,767,000	3,677,440	67	89.5
s15850	98.0	2,611,414	2,090,048	53	93.2
s38417	99.3	21,482,240	17,557,920	78	84.9
s38584	99.1	17,099,520	12,988,160	66	87.3

Experimental Results

- Comparisons of energy reduction

Circuits	General LFSR	[Girard]	[Manich]	[Zhang]	Proposed
	ETM	Energy reduction (%)	Energy reduction (%)	Energy reduction (%)	Energy reduction (%)
s5378	6,205,783	54.2	86.8	86.8	97.2
s9234	6,151,931	41.3	73.7	73.7	95.9
s13207	10,489,489	5.4	45.3	45.3	85.2
s15850	9,153,358	2.8	60.2	60.2	89.9
s38417	58,217,693	57.7	70.7	70.7	81.3
s38584	51,214,830	58.0	75.4	75.4	90.5

Experimental Results

- Comparison of area overhead

Circuits	General LFSR	[Girard]	[Manich]	[Zhang]	Proposed	
	Required LFSR bits	Area (%)	Area (%)	Area (%)	LFSR bits	Area (%)
s5378	22	9.5	14. 1	28.3	11	16.6
s9234	22	7.7	11.4	37.6	11	21.5
s13207	24	4.0	6.1	21.8	12	11.3
s15850	23	3.7	5.7	16.1	11	8.3
s38417	25	1.4	3.2	8.8	12	3.8
s38584	25	1.4	2.6	8.4	13	4.2

Conclusions

- Developed a new pattern generator based on a statistical compression code
  - To minimize the energy consumption
  - To reduce the test data
  - To reduce the hardware overhead
- Experimental results
  - Energy reduction ratio : 81.%~97.2%
  - High test data compression
  - Low hardware overhead
- The proposed approach proved to be an attractive and an effective solution of BIST for low energy consumption