**EECS 579: Built-in Self-Test 1**

- System is partitioned into subcircuits suitable for BIST
- Scan design can provide controllability/observability access
- Key design goals: high fault coverage; low hardware overhead

**Built-in Self-Test**

**Economic Case for BIST**

- Overcome pin limitations (high logic-to-pin ratio) and related controllability and observability problems
- Reduce ATPG and test application times
- Lower testing cost by reducing or eliminating external ATE
- Provide “at-speed” testing for high-speed ($\geq 1$Ghz) circuits
- Improve testing quality (fault coverage and yield)
- Address shortage of skilled test engineers

**Classification**

- Concurrent (on-line)
- Nonconcurrent (off-line)
Concurrent BIST

Testing occurs “on-line” during normal operation

**Design Methods**
- Self-checking based on ED/EC codes
- Replicated circuits with equality checkers
- On-chip electrical monitoring

**Characteristics**
- No explicit test-pattern generation
- Low error detection latency
- Fault coverage often inadequate
- Limited applicability

---

Error-Detecting/Correcting Codes

**Code Types**
- Parity (Hamming)
- m-out-of-n
- Arithmetic
- etc.

**Code Selection Criteria**
- Error-detecting or correcting
- Number and types of errors to cover
- Separable or nonseparable
- Overhead in logic
- Overhead in delay (latency)
Parity Codes

- A parity check bit \( p \) is appended to \( n - 1 \) information bits \( x_1x_2 \ldots x_{n-1} \) to create an \( n \)-bit codeword \( x_1x_2 \ldots x_{n-1}p \).
- An error is detected if the parity
  \[ x_1 \oplus x_2 \oplus \ldots \oplus x_{n-1} \oplus p \]
deviates from its normal value (even or odd).
- Parity bit generation and checking are implemented by XOR logic.
- Single-bit parity code covers single-bit errors, no double-bit errors, some unidirectional errors, and some multiple-bit errors.
- With multiple parity bits, more error coverage and/or error correction can be obtained.
- Major applications: buses and RAMs.

Parity Codes

\begin{figure}
\centering
\includegraphics[width=\textwidth]{parity_diagram}
\caption{Parity generator and checker diagram.}
\end{figure}
**Arithmetic Codes**

- Use arithmetic operations such as addition for encoding/decoding
- General property: with code $C$ and arithmetic operation $\ast$
  \[ C(N_1 \ast N_2) = C(N_1) \ast C(N_2) \]
- **3N Code**: Data word is $N$ encoded by multiplying by 3 and decoded by dividing by 3. Nonseparable and single error detecting (SED) code. Easy to implement. Why?
- **Residue Code**: Separable code. A check word is computed as $R(N) = N \pmod{m} = \text{remainder on division by } km$. Also SED if $m$ is odd.

**Self-Checking Circuits**

- Circuit $C$ is **self-testing** if every fault produces a non-codeword output for at least one input
- $C$ is **fault secure** if no fault produces an output that corresponds to an incorrect codeword
  
  Self-checking = self-testing + fault secure

- Little is known about designing general self-checking circuits; many specific cases are known.
Nonconcurrent BIST

Testing occurs “off-line” during special test mode

**Design Methods**
- Random or exhaustive test generation with output response compaction
- Algorithmic or deterministic test generation with prestored (compacted or uncompacted) test data

**Characteristics**
- High fault coverage achievable
- Applicable to most circuit types
- Long error latency
- *(Pseudo)Random*: very long tests sequences, uncertain fault coverage
- *Deterministic*: high circuit overhead

---

**Test Generator**
- **Random**:
  - Linear feedback shift register (LFSR)
  - Cellular automaton (CA)
- **Deterministic**:
  - ROM with prestored test data
  - Specially-designed FSM to generate test vectors

**Response Monitor**
- Direct comparison of raw responses with hardware- or software-generated reference (good) responses. Often uses multiple-input linear shift register or MISR (“miser”)
- Compression of test responses into a compact signature and comparison with prestored response signatures: “signature analysis”
Compression-Based BIST

- CUT can produce a vast amount of response data \( R \)
- Signature \( S(R) \) is used to compare the expected and observed responses.
- Some possible compression function \( S \):
  - One’s counting
  - Transition counting
  - “Signature Analysis” [Hewlett-Packard]: cyclic code checking with LFSRs
- LFSRs are also used for pseudorandom test generation.

Compression-Based BIST

**System Architecture**
Compression-Based BIST

Characteristics

• Applicable to moderately large combinational logic blocks including the combinational part of sequential circuits with scan

• Good response data compression: from $n$ to $\log_2 n$ (ones/transition counting) and from $n$ to a constant $k$ ($k = 16$ for HP’s original Signature Analysis method)

• Loss of fault coverage due to aliasing, that is, the faulty signature = the good signature), especially with pseudorandom tests

• Aliasing can be hard to detect and eliminate

• Hard to apply to an entire system on a chip such as a microprocessor or microcontroller

One’s Counting

• Apply $m$ vectors to the CUT and monitor the one’s count at each output $z_i$. If the response $R$ at $z_i$ has $r$ ones, the signature is $1C(R) = r$

• For every single-output combinational CUT with $m$ tests for any fault set $F$ of interest, there is a test set of at most $m^2$ tests that detect $F$ after compression by one’s counting.

• To construct a complete, deterministic $1C$ test set for any $F$ in a single-output combinational circuit.

  Let $T = \{T^0, T^1\}$, where $m = |T|$ and $T^d$ denotes all tests that produce output $d$. Construct a test set $S$ containing one copy of each vector in $T^0$ and $|T^0| + 1$ copies of each vector in $T^1$.

  $S = \{S^0, S^1\}$ is the required test set.

  **Proof:** $S$ fails to detect an $f$ in $F$ if the number of faulty responses to $S^0$ is the same as the number of faulty responses to $S^1$.

  This is impossible since at most $|T^0|$ responses to $S^0$ can be faulty while at least $|T^0| + 1$ responses to $S^1$ must be faulty.
**One’s Counting (contd.)**

- $1C(R)$ is independent the order of $R$
- Good tests produce small or large values of $1C(R)$
- Aliasing probability can be readily estimated

---

**One’s Counting (contd.)**

- Data compression is typically used with pseudorandom tests, so it is of interest to know the probability of aliasing in such cases

- Probability $P(r)$ that random tests produce $R$ with $r$ ones is $\binom{m}{r}/2^m$

- If all $2^m - 1$ error response sequences are equiprobable (an unrealistic assumption!), then the masking probability for a given $r$ is

  $$P_{1C}(M|m,r) = \frac{\text{No. of masking sequences (aliases)}}{\text{No. of error sequences}} = \frac{\binom{m}{r} - 1}{2^m - 1}$$

- The overall masking probability $P_{1C}(M) = \sum_{r = 0}^{m} P_{1C}(M|m, r)P(r)$ which approaches $(\pi m)^{-0.5}$ asymptotically as $m$ increases.

- The normalized $1C$ signature $r/2^n$ resulting from applying all test vectors to a circuit (exhaustive testing) is called the **syndrome**.
**Transition Counting**

- Apply $m$ vectors to the CUT and monitor the transition count at each output $z_i$, i.e., the number of times a 0-to-1 or 1-to-0 change occurs.
- If the response stream is $r_1, r_2, \ldots, r_m$ the signature is given by

$$\text{TC}(R) = \sum_{r=1}^{m} r_i \oplus r_{i+1}$$

- Unlike one’s counting, the TC signature is affected by the order in which the test vectors are applied.

![Transition Counting Diagram]

**Transition Counting (contd.)**

- For every single-output combinational CUT with $m$ tests for any fault set $F$ of interest, there is a test set of at most $2m$ tests that detect $F$ after compression by transition counting.
- To construct a complete, deterministic TC test set for SSL faults in a single-output combinational circuit.

Let $T = \{T^0, T^1\}$, where $m = |T|$ and $T^d$ denotes all tests that produce output $d$. Construct a test sequence $S$ containing a vector from $T^0$ and $T^1$ in alternating positions. Repeat vectors from the smaller set, if necessary. $S$ is the required test set.

**Proof:** (To be discussed in class)

- The overall masking probability $P_{TC}(M)$ again approaches $(\pi m)^{-0.5}$ asymptotically as $m$ increases.