Digital Programming: High Speed Digital Scan Applications

TERADYNE

PN 553-401-36 Rev B
February 2009

Limited Reproduction Rights for Teradyne Customers

Teradyne Inc.
600 Riverpark Drive
North Reading, MA 01864

Restricted Rights

Acknowledgements

UltraFLEX and FLEX training documents contain references to Windows NT, Windows 2000, Windows XP, and Microsoft Visual Basic.

High Speed Digital SCAN Applications

- High Speed Digital SCAN
  - Objectives
  - Scan terminology
  - HSD Scan Overview
    - Features
    - Basic Scan Pattern Requirements
  - HSD Scan Architecture
    - Two and three bit scan
  - Scan Programming
    - Programming Steps
    - Vector syntax
    - Pattern compiler
  - Scan Debug and Setup
  - Using CMEM for Deep Scan capture
- Summary/Review
- Scan programming and debugging Lab 8
- Appendix A: Device Scan Basics
- Appendix B: SN74BCT8245A Device Data Sheet

Objectives

- List and describe industry scan standards.
- Describe UltraFLEX Scan Instrument architecture.
- Identify possible scan configurations for the UltraFLEX Scan Instrument.
- Describe UltraFLEX Scan Chain data configurations.
- Describe the key differences between parallel and scan pattern data.
- Describe how VM pattern data is organized after loading and accessed at run time.
- List and describe the syntax necessary for creating Scan vectors.
- Observe and compile Scan patterns.
- Load and run an UltraFLEX scan pattern program, and use IG-XL PatternTools to view the scan pattern data.
- Observe program examples of using CMEM for Deep Scan Capture

SCAN Terminology

- **Alternate Data Bus (ADB)**: A bidirectional bus in a digital channel board that connects digital source and capture to the digital channels. SCAN hardware does not use ADB lines on the UltraFLEX.
- **Boundary scan**: A test technique that involves creating a long scan chain around the periphery of a device by connecting the I/O elements of the design. (See also: JTAG)
- **Capture memory (CMEM)**: A 64M x 16 DDR SDRAM memory. In SCAN, it can be used to capture per pin fails and fail data per cycle. You can move fail data from the capture memory to the host DSPs for fail processing. You can also conduct this capturing and moving concurrently.
- **Defect Physical**: An anomaly that occurs during IC manufacturing. A defect may cause a device to no longer comply with its functional or parametric design specifications.
- **DFT**: Design-for-test. DFT adds logic or other circuitry to a device design to enhance its testability.
- **Fault**: The logical or functional result of a defect.
- **Fault model**: A description of faulty behavior in terms of a specific circuit characteristic or specification.
- **UltraFLEX databus**: Databus that connects the computer running IG-XL to tester hardware.

SCAN Terminology

- **History RAM**: An UltraFLEX read-only memory that captures data about the state of the pattern generator and digital channels when a pattern runs.
- **JTAG**: Joint Test Action Group. This industry group promoted the industry standard for the IEEE Standard Test and Boundary Scan Architecture (IEEE Std. 1149.1) test interface.
- **Internal scan**: A test technique that uses existing flip-flops and latches in the design to support serial shifting of test data into a device through scan chains to test the device's internal logic circuits.
- **Parallel vector**: A vector that specifies data on all pins in a pattern (a traditional pattern vector, as opposed to a scan vector). When loaded in VM, parallel vectors require 3 bits of VM space for every pin cycle (as opposed to a UltraFLEX SCAN-compiled scan vector that can require as few as one bit for a pin cycle).
- **Path delay fault model**: A fault model that is used to test faulty path timing behavior such as a slow-to-rise or slow-to-fall pathways. This model can be used to verify the timing specification of an internal circuit.
- **SCAN:** The use of specialized UltraFLEX software and hardware to support features that cannot be supported using standard digital parallel vector memory hardware. SCAN features include VM data pooling across multiple channels and broadcast of SCAN-compiled vector data.

SCAN Terminology

- **Scan cell**: One element of a scan chain. During a design process commonly referred to as scan insertion, sequential elements such as flip-flops and latches are modified to support scan testing. After this process, scanable sequential elements are referred to as scan cells.
- **Scan chain**: During scan testing, a collection of scan cells is connected together in a shift register configuration to form a scan chain. Test pattern data is shifted into and out of the scan chains through device scan in and scan out pins respectively. An HSD Scan chain = 2 channels 1 scan in and 1 scan out.
- **Scan chain position**: An index specifying the position of a scan cell within a scan chain. Positions are specified relative to the node at the scan out pin for the scan chain.
- **Scan cycle**: One tester cycle during a scan vector. It corresponds to a single shift of the scan chain.
- **Scan in (pin)**: A device input pin that is used to drive scan test pattern data to drive a device scan chain.
- **Scan index**: The relative cycle number in a scan vector, starting with an index of 0 for the first cycle. This is generally related to scan cell positions in a scan chains, but the exact relationship depends on several factors. HRAM information is generally tester-cycle related, and a single tester cycle will not correspond with a single scan chain position on all scan-in pins (unless all scan chains are exactly the same length), so HRAM displays use scan index rather than scan chain position.

SCAN Terminology

- **Scan insertion**: A device design process where all sequential design elements such as flip-flops and latches are modified to support internal scan testing.
- **Scan node**: See scan cell.
- **Scan out (pin)**: A device output pin that is used to compare scan test pattern data being unloaded from a device scan chain
- **Scan pattern** or **scan test pattern**: An IG-XL pattern file that uses the scan microcode to indicate tester cycles where scan test data is shifted into (and out of) a device under test.
- **Scan pin**: A DUT pin connected to one end of one or more scan chains.
- **Scan shift**: The process of simultaneously loading and unloading test pattern data to and from device scan chains. Pattern data shifted (or scanned) into the scan chains through scan in pins. Data is shifted (or scanned) out of the device through scan out pins.
- **Scan vector**: A single state of all latches in a scan chain. Also a vector in a pattern file that specifies pattern data that is applied serially to certain pins referred to as scan pins. These vectors normally supply pattern data for the scan chains within a device. Often, special tester hardware is used to execute scan vectors to minimize the amount of tester memory needed to store data for the vector.

SCAN Terminology

- **SCAN VM**: Areas of shared VM that are used to store SCAN-compatible scan pattern data. Scan pattern data stored in SCAN VM is arranged differently than when it is stored in regular VM.
- **Serial vector**: A vector that applies drive or compare data serially to a device. See also scan vector.
- **SRM**: Subroutine Vector Memory. A fast-access memory that allows random access during pattern bursts. SRM executes pattern loops, subroutines, and memory test and instrument-specific microcodes
- **Stuck-at fault model**: A model used to test the structural characteristics of an integrated circuit. It is based on shorting transistor gate connections and wire routing to power and ground. This fault model is the most commonly used for internal scan testing.
- **Test computer I/O (TCIO)**: A bus that connects the computer running IG-XL to tester hardware.
- **Timing modes**: Modes that specify edge generation configuration for a given pattern burst. For HSD Scan, both single and dual timing modes are supported
- **Transition delay fault model**: This model tests the rise and fall specification of specific gates as opposed to point-to-point IC pathways. Similar to path delay fault model.
- **VM**: Vector Memory. UltraFLEX VM stores SCAN vector data, parallel vector data, on-board pin level data, and timing data.

HSD SCAN Terminology

- Scan chain: 2 channels, 1 scan in and 1 scan out
- 2 bit scan mode requires 1 bit for scan in and 1 bit for scan out
- 3 bit scan mode requires 1 bit for scan in and 2 bit for scan out
- 2 bit scan provides greater scan depth but has no “X” (don’t care) state
- Vector Memory (VM) is used for scan source and expect data
- Scan Packing: Compressing vector memory to increase VM depth
- Scan Stacking: Using the VM of other channels to increase scan depth
- HRAM and CMEM are memories used to capture scan out data for processing
- Megavector (MV) = Number of scan in/out cycles in millions

HSD Scan Instrument Basics

- The HSD Scan Instrument can provide large amounts of serial data for testing logic devices containing special circuits for testing only, commonly known as scan design circuits.
- Some common types of scan-designs for built-in self test are:
  - Scan Path
  - Level-Sensitive Scan Design (LSSD)
  - Boundary Scan
  - JTAG IEEE 1149.1
- Test systems with scan-based architectures, such as the UltraFLEX can facilitate functional testing of inaccessible internal blocks of a device, thereby reducing the test complexity and time it might take to reach these blocks by other means.
- Without scan test capability, buried logic could take a magnitude more vectors to set up and test, increasing test times and the cost of test.

HSD Scan Instrument Basics

- The HSD SCAN Instrument sends and receives serial data patterns into a device where the device uses the data to verify a certain part of the circuitry.
- In general, devices with scan designs are tested by serially inputting scan test vectors into the device through a designated **scan-in** pin (or pins).
- The scan circuit passes through special state registers, called scan registers (or latches), connected to the internal logic blocks that are to be tested.
- When the device is in test mode, data is passed into the scan registers and shifted through the scan registers according to the design of the device.
- Finally, the result at the **scan-out pin** (or pins) is compared to an expected output. Any failure data is then stored for later evaluation.
- UltraFLEX Scan tests can be used alone or with high-speed functional tests or memory tests.

HSD1000 Scan Instrument Features

- 250MHz Single Patgen Mode
- 500MHz Dual Patgen Mode
- **32 scan chains per board (64 channels per board)**
- Scan In: 1-bit, Scan Out: 1-bit or 2-bit
- Scan or parallel operation can be selected per PatGen cycle
- VM is used for scan sourcing and comparing
  - 384 Megavectors for 2 bit chains (in Dual mode)
  - 256 Megavectors for 3 bit chains (in Dual mode)
- VM can be stacked for greater depth
  - Up to 3GV for 2 bit scan mode (in Dual mode)
  - Up to 1.5GV for 3 bit scan mode (in Dual mode)
- 4GV deep Scan vectors using serial digsrc
- Selective fail capture
  - Capture only cycles that failed
- Interactive debug tools for failure diagnostic

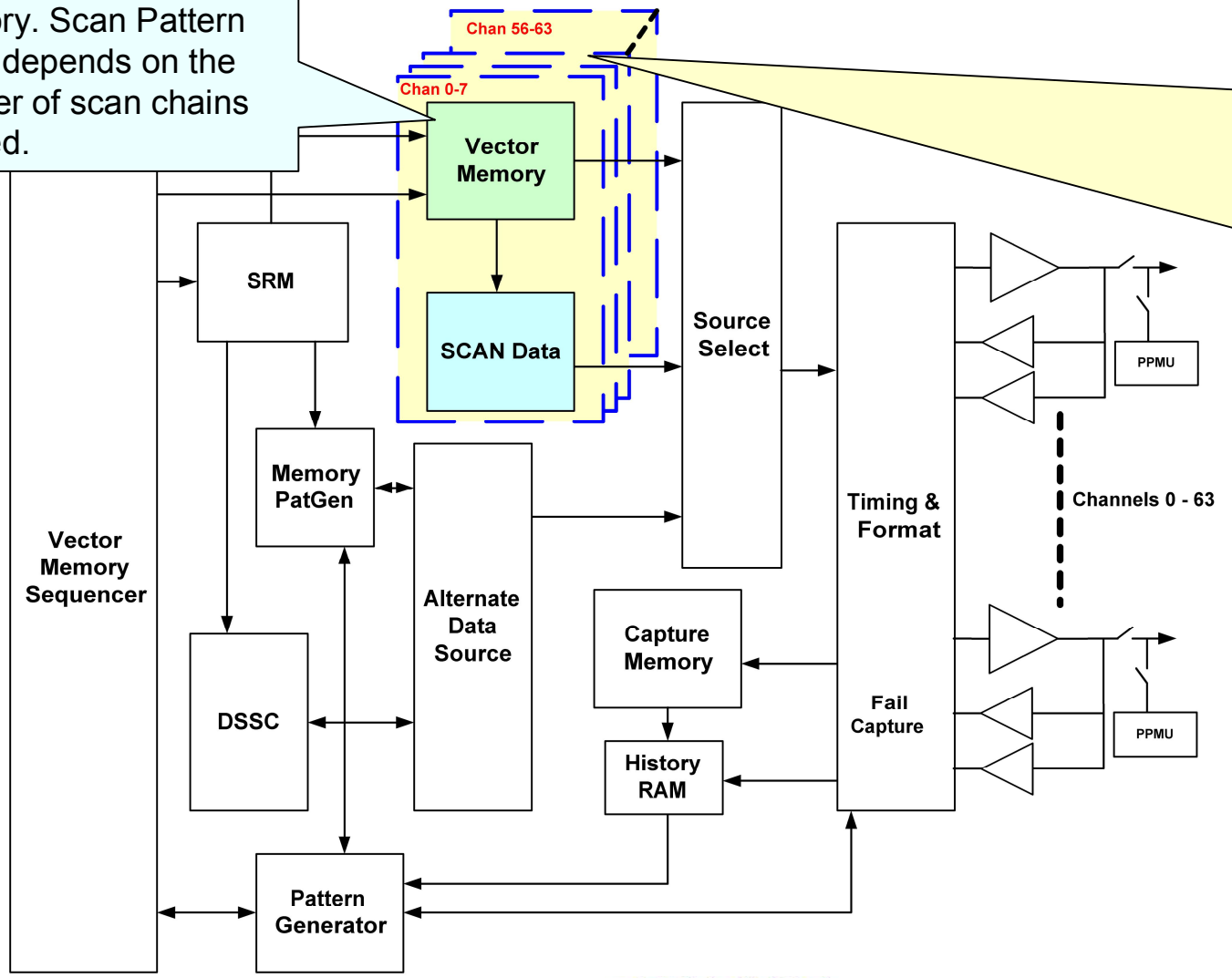
UltraPin800 Scan Instrument Features

- 128 DUT pins per board
  - 128 single-ended
  - 64 differential pairs
- Data rate: 800Mbps
- 256MV Pattern Memory
- **64 max scan chains per board**
- All pins HV capable (+13V, 10mA)
- 20mA active loads and 50 ohm load to VT per channel
- Per pin features
  - Freq Counter
  - MTO
  - Digital Signal Source & Capture (DSSC)
- Time domain per Instrument
- EPA: +150ps

UltraFLEX SCAN Architecture

HSD SCAN: Digital Subsystem SCAN Hardware

Scan Pattern Data is stored in Vector Memory. Scan Pattern depth depends on the number of scan chains needed.



Each HSD1000 instrument has 64 channels that are divided into **eight SCAN** regions of 8 channels each. With 64 channels and each "scan chain" requiring two channels (one for scan-in and the other for scan-out), each HSD1000 can have up to 32 scan chains.

HSD SCAN Chain Depths

- Three-bit Scan Chain Configurations per Digital Board

SCAN Type	HSD1000 Maximum Number of Scan Chains (N_{max_chains})	HSD800-128 Maximum Number of Scan Chains (N_{max_chains})	Maximum Scan Chain Depth (Mcycles) (D_{max_depth})	Maximum Data Rate (Mcycles/second)
X2	32	64	256	500
X4	16	32	512	500
X8	8	16	1024	500
Extended	5	10	1536	500

- Two-bit Scan Chain Configurations per Digital Board

SCAN Type	HSD1000 Maximum Number of Scan Chains (N_{max_chains})	HSD800-128 Maximum Number of Scan Chains (N_{max_chains})	Maximum Scan Chain Depth (Mcycles) (D_{max_depth})	Maximum Data Rate (Mcycles/second)
X3	32	64	384	500
X6	16	32	768	500
X12	8	16	1536	500
X24	4	8	3072	500

- Note: Before you select the SCAN type for your application, you must first know your system information and device requirements. See “Selecting SCAN Type” in the UltraFLEX User Manuals for more information.

HSD Scan Chain Density

- Maximum number of chains per test system:

UltraFLEX Configuration (Pin Count)	Maximum Scan Chains per Tester
UltraFLEX-HD (High Density)	12X
UltraFLEX-SC (Standard Count)	12X
UltraFLEX-SC (H2 Upgrade)	18X
UltraFLEX-HC (High Count)	32X

- The only differences between the HSD1000 and UltraPin800 are in the maximum number of chains per board and per system.
- Memory depth is the same on both the HSD1000 and UltraPin800.
- On the UltraPin800, the maximum number of chains per board doubles.
- Stacking rules are the same and are organized by groups of 8 channels.

HSD Vector Memory

Channel 0

	Bit 2	Bit 1	Bit 0
0			
1			
2			
		.	
		.	
		.	
		.	
64/128MV			

- Parallel vectors consist of 3 bits to represent 8 states (0, 1, L, H, M, V, X, -)
- Scan vectors only need 2 or 3 states
  - 0,1 for scan in
  - L, H or L, H, X for scan out
- Can “pack” Scan vectors into vector memory to increase depth
- Scan packing and stacking techniques are the same for both HSD1000 and UltraPin800 and are limited to groups of 8 channels.

SCAN Pattern Requirements

- Scan pin data using UltraFLEX Vector Memory “Parallel” only:
  - When you compile scan test patterns as parallel patterns (using the `-scan_parallel` compiler option), the following applies:
    - Scan pin data is the same as regular parallel pin vector data.
    - SCAN hardware is not used during pattern execution.
    - VM pattern memory cannot be shared across channels.
    - Pattern pin data requires three bits per cycle.
    - The maximum scan pattern depth is limited by the maximum VM depth which is 64M cycles running in Single PatGen mode or 128M cycles running in Dual PatGen mode.

SCAN Pattern Requirements

- Scan pin data using UltraFLEX SCAN hardware:
  - When you compile UltraFLEX test patterns as SCAN-compatible patterns (x2, x3, x4, x6, x8, x12, x24, extended), the following applies:
    - VM pattern memory can be integrated across several digital channels.
    - Scan-in pattern pin data requires only one bit per cycle (“0” or “1”)
    - Scan-out pattern pin data requires two bits per cycle (“L” ,“H” & “X”).
    - The maximum scan pattern depth is limited by the SCAN type defined in the pattern file “scan_type”.
    - Scan Patterns can be compiled for either Single or Dual PatGen timing mode.

SCAN Pattern Requirements

- Two-bit Scan Mode Options: **`x3`**, **`x6`**, **`x12`**, **`x24`**
  - These scan types specify scan packing methods that allow 2-bit scan chains.
  - The number after x specifies packing efficiency. Data of 3/6/12/24 scan cycles is packed into one parallel vector.
- Three-bit Scan Mode Options: **`x2`**, **`x4`**, **`x8`**, **`extended`**
  - These scan types specify scan packing methods that allow three-bit scan chains.
  - The number after x specifies packing efficiency. Data of 2/4/8 scan cycles is packed into one parallel vector.
- Even_Odd SCAN **General** Channel Rule:
  - For **three-bit** scan chains:
    - scan-in (input) pins must be assigned to even-numbered channels,
    - scan-out (output) pins must be assigned to odd-numbered channels.
  - For **two-bit** scan chains, scan-in and scan-out pins can be assigned to either even or odd channels.
  - Note: There are specific channel assignment rules for each three-bit Scan Chain Option that **must** be followed.

2 Bit SCAN Chain: X3

1 ch ⇨ 1 ch (1:1 stacking) X3 VM:

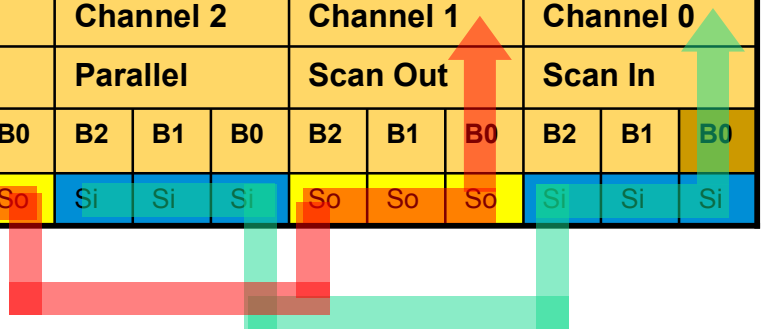
Channel 7			Channel 6			Channel 5			Channel 4			Channel 3			Channel 2			Channel 1			Channel 0		
Parallel			Parallel			Parallel			Parallel			Scan Out			Scan In			Scan Out			Scan In		
B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0
vm	vm	vm	vm	vm	vm	vm	vm	vm	vm	vm	vm	So	So	So	Si	Si	Si	So	So	So	Si	Si	Si

- Any channel can be parallel, scan drive, or scan compare
- 64M x 3 = 192Mb of scan drive or scan compare per pin (Single mode)
- 128M x 3 = 384Mb of scan drive or scan compare per pin (Dual mode)

2 Bit SCAN Chain: X6

2 ch ⇨ 1 ch (2:1 stacking) X6 VM:

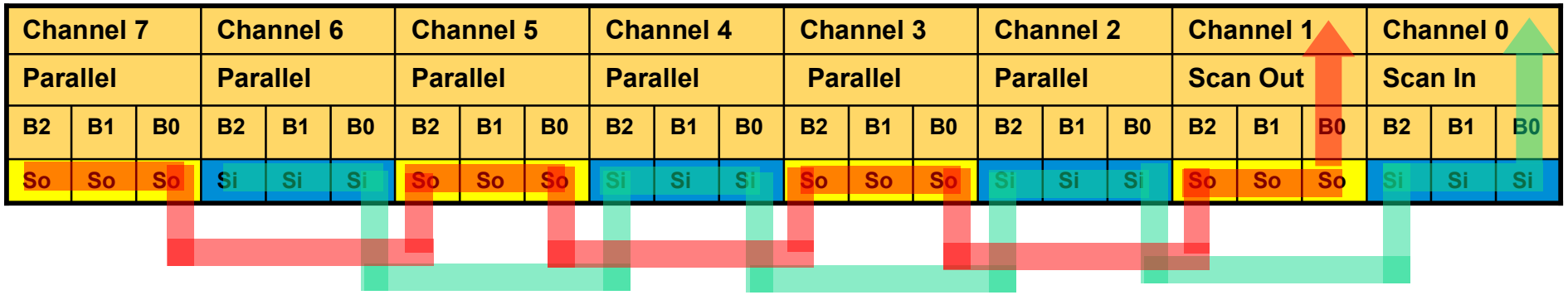
Channel 7			Channel 6			Channel 5			Channel 4			Channel 3			Channel 2			Channel 1			Channel 0		
Parallel			Parallel			Parallel			Parallel			Parallel			Parallel			Scan Out			Scan In		
B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0
vm	vm	vm	vm	vm	vm	vm	vm	vm	vm	vm	vm	So	So	So	Si	Si	Si	So	So	So	Si	Si	Si



- Any channel can be parallel, scan drive, or scan compare
- 64M x 6 = 384Mb of scan drive or scan compare per pin (Single mode)
- 128M x 6 = 768Mb of scan drive or scan compare per pin (Dual mode)

2 Bit SCAN Chain: X12

4 ch ⇨ 1 ch (4:1 stacking) X12 VM:



- Any channel can be parallel, scan drive, or scan compare
- 64M x 12 = 768Mb of scan drive or scan compare per pin (Single mode)
- 128M x 12 = 1536Mb of scan drive or scan compare per pin (Dual mode)

2 Bit SCAN Chain: X24

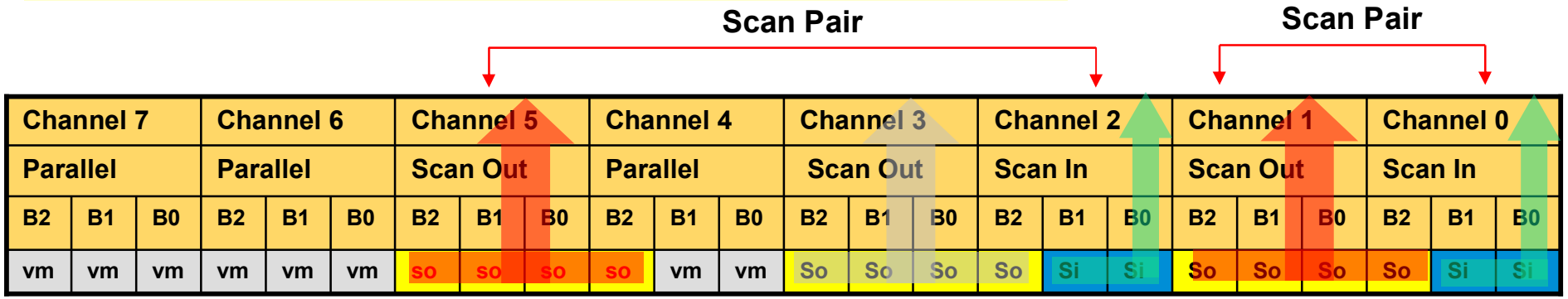
8 ch ⇨ 1 ch (8:1 stacking) X24 VM:

Channel 7			Channel 6			Channel 5			Channel 4			Channel 3			Channel 2			Channel 1			Channel 0		
Parallel			Parallel			Parallel			Parallel			Parallel			Parallel			Parallel			Scan Data		
B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0
Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc	Sc

- Any channel can be parallel, scan drive, or scan compare
- 64M x 24 = 1536Mb of scan drive or scan compare per pin (Single mode)
- 128M x 24 = 3072Mb of scan drive or scan compare per pin (Dual mode)

3 Bit SCAN Chain: X2

2 ch ⇨ 2 ch (1:1 stacking) X2 VM:



- I/O on any channel
- 64M x 2 = 128Mb per chain (Single mode)
- 128M x 2 = 256Mb per chain (Dual mode)
- Scan-in pins can be assigned to any even channels.
- Scan-out pins can be assigned to any odd channels.
- Up to four scan-in and four scan-out pins can be assigned to each SCAN region.
- Non-SCAN channels can be used for parallel (**scan_setup**) data pins.

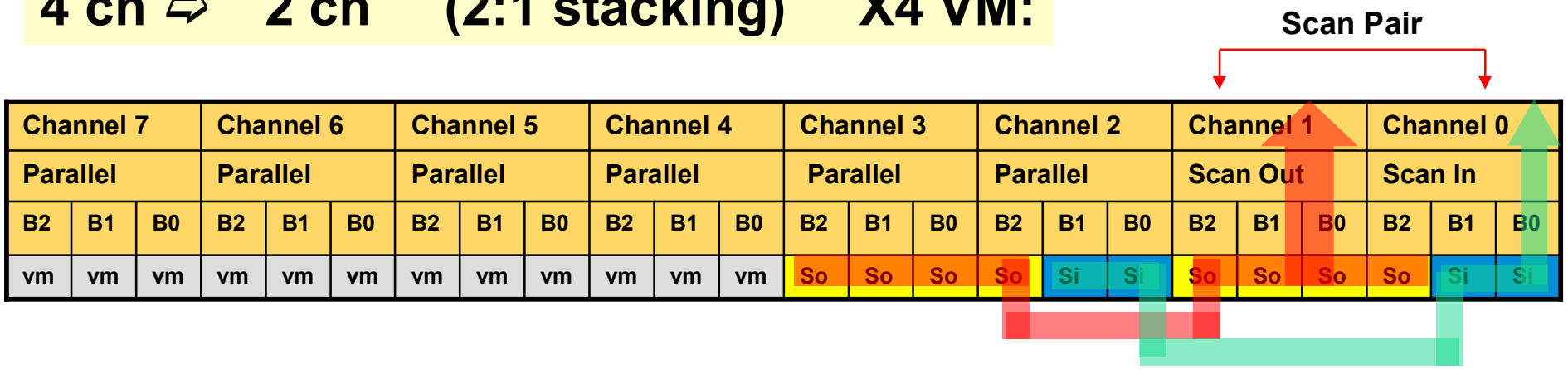
SCAN Channel Assignments: Rules for X2 SCAN

SCAN Region	Available for SCAN Pins							
	Scan In	Scan In	Scan In	Scan In	Scan Out	Scan Out	Scan Out	Scan Out
1	Ch0	Ch2	Ch4	Ch6	Ch1	Ch3	Ch5	Ch7
2	Ch8	Ch10	Ch12	Ch14	Ch9	Ch11	Ch13	Ch15
3	Ch16	Ch18	Ch20	Ch22	Ch17	Ch19	Ch21	Ch23
4	Ch24	Ch26	Ch28	Ch30	Ch25	Ch27	Ch29	Ch31
5	Ch32	Ch34	Ch36	Ch38	Ch33	Ch35	Ch37	Ch39
6	Ch40	Ch42	Ch44	Ch46	Ch41	Ch43	Ch45	Ch47
7	Ch48	Ch50	Ch52	Ch54	Ch49	Ch51	Ch53	Ch55
8	Ch56	Ch58	Ch60	Ch62	Ch57	Ch59	Ch61	Ch63

- See “Allocating SCAN Channels” in the UltraFLEX User Manuals for more information on the Scan Channel Assignment rules including the assignments for the UltraPin800.

3 Bit SCAN Chain: X4

4 ch ⇨ 2 ch (2:1 stacking) X4 VM:



- I/O on any channel
- 64M x 4 = 256Mb per chain (Single mode); 128M x 4 = 512Mb per chain (Dual mode)
- Up to two scan-in and two scan-out pins can be assigned to each SCAN region.
- Scan-in pins can assigned to the first and fifth channels in each SCAN region.
- Scan-out pins can be assigned to the second and sixth channels in each SCAN region.
- Non-SCAN channels can be used for parallel (**scan_setup**) data pins.

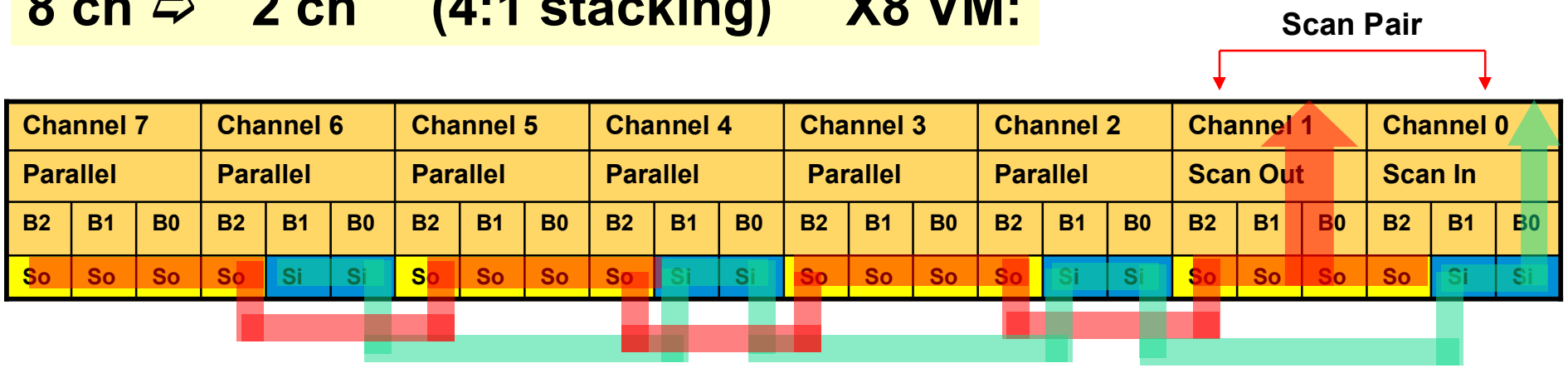
SCAN Channel Assignments: Rules for X4 SCAN

SCAN Region	Available for Scan Pins			
	Scan In	Scan In	Scan Out	Scan Out
1	Ch0	Ch4	Ch1	Ch5
2	Ch8	Ch12	Ch9	Ch13
3	Ch16	Ch20	Ch17	Ch21
4	Ch24	Ch28	Ch25	Ch29
5	Ch32	Ch36	Ch33	Ch37
6	Ch40	Ch44	Ch41	Ch45
7	Ch48	Ch52	Ch49	Ch53
8	Ch56	Ch60	Ch57	Ch61

- See “Allocating SCAN Channels” in the UltraFLEX User Manuals for more information on the Scan Channel Assignment rules including the assignments for the UltraPin800.

3 Bit SCAN Chain: X8

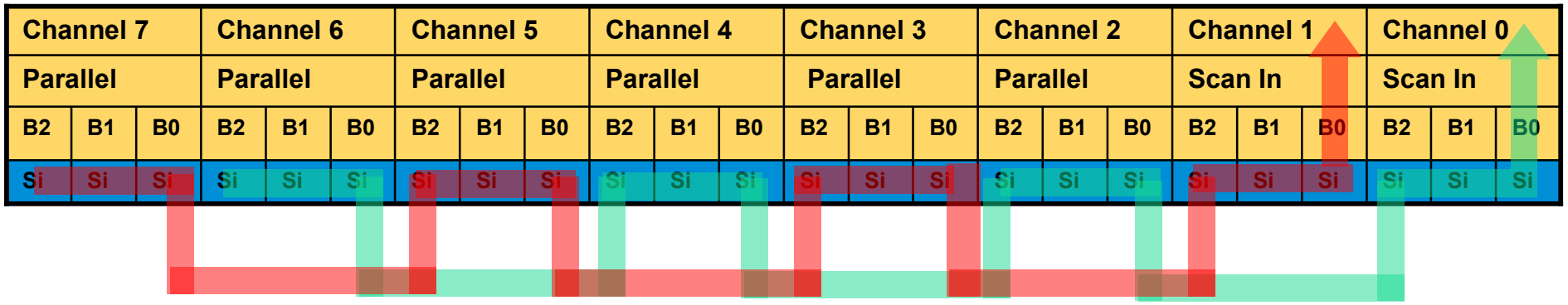
8 ch ⇨ 2 ch (4:1 stacking) X8 VM:



- Scan drive on the even channel
- Scan compare on the odd channel
- I/O on any channel
- 64M x 8 = 512Mb per chain (Single mode); 128M x 8 = 1024Mb per chain (Dual mode)
- Scan-in pins can be assigned to the **first channel** of a SCAN region.
- Scan-out pins can be assigned to the **second channel** of a SCAN region.
- Non-SCAN channels can be used for parallel (**scan_setup**) data pins.

3 Bit SCAN Chain: Extended

8 ch ⇨ 2 Drive ch (Extended)



- Two drive channels on one HSD Timing Generator
- Scan drive on Channel 0 and Channel 1
- 64M x 12 = 768Mb per chain (Single mode)
- 128M x 12 = 1536Mb per chain (Dual mode)

3 Bit Extended SCAN

8 ch ⇨ 1 compare channel (Extended)

Channel 7			Channel 6			Channel 5			Channel 4			Channel 3			Channel 2			Channel 1			Channel 0		
Parallel			Parallel			Parallel			Parallel			Parallel			Parallel			Scan Out			Parallel		
B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0	B2	B1	B0
So	So	So	So	So	So	So	So	So	So	So	So	So	So	So	So	So	So	So	So	So	So	So	So

- One scan compare channel on one HSD Timing Generator
- Scan compare on channel 1
- 64M x 12 = 768Mb per chain (Single mode); 128M x 12 = 1536Mb per chain (Dual mode)
- **Do not** assign scan-in and scan-out pins to the same SCAN region.
- Scan-in pins can be assigned to the first and second channels of a SCAN region.
- Scan-out pins can be assigned to the second channel of another SCAN region.
- Non-SCAN channels can be used for parallel (**scan_setup**) data pins. Scan-out pins can be assigned to the second channel of a SCAN region.

SCAN Pattern File Example

```
digital_inst = hsdm;
opcode_mode = dual;
import tset tset_scan;
scan_type = x2;
// tdi = scan input pin
// tdo = scan output pin
//-------------------------------------------------------------------

scan_pins = {
        tdi,
        tdo
        }

vm_vector ( $tset, dir:S, oe:S, portA:S, portB:S,   tdi:S, tdo:S, tck:S, tms:S )
{
begin_scan_vec:
//Initialize TAP-controller state to Shift-IR
                      > tset_scan      X    X    .X        .X       1        X       1        1; //1. Extra cycle to get TCK
                      > tset_scan      X    X    .X        .X       1        X       1        1; //2. Start Reset Sequence
                      > tset_scan      X    X    .X        .X       1        X       1        1; //3. Reset Sequence
repeat 30             > tset_scan      X    X    .X        .X       1        X       1        1; //4. Reset Sequence
// Shift in test seq 1001101100, shift out test seq 1001101100 thru BSR (10-bit long)
scan_setup
          > tset_scan     X    X          .X           .X                0  L    1  0 ; //65-92.  EXTENDED Scan
scan 26
(si tdi:S > 1001101100111111111111111111)
(so tdo:S > XXXXXXXXXXXXXXXXXXHLLHHLHH);

          > tset_scan   X    X    .X        .X       1  L  1  0 ; //SMS last scan cycle needs go to parallel
          > tset_scan   X    X    .X        .X       1  L  1  1 ; //93 Last TDO bit - To Exit1-DR
          > tset_scan   X    X    .X        .X       1  X  1  1 ; //94. To Update-DR
          > tset_scan   X    X    .X        .X       1  X  1  0 ; //95. To Run-Test/Idle
          > tset_scan   X    X    .X        .X       1  X  1  1 ; //96. To Select-DR-Scan
          > tset_scan   X    X    .X        .X       1  X  1  1 ; //97. To Select-IR-Scan
          > tset_scan   X    X    .X        .X       1  X  1  0 ; //98. To Capture-IR
```

Vector Pattern header requirements for SCAN identifying:

1. Dual PatGen mode (or Single)
2. the scan type
3. the scan pins declarations

Standard parallel vectors

Scan vectors

UltraFLEX SCAN Programming

SCAN Programming Steps

- Do the following to support SCAN-compatible test patterns:
  1. Identify your system configuration:
     - The number of HSD digital channel boards in the test system (see the system configuration file, CurrentConfig.txt, for this information)
     - The number of device scan chains for your application
     - The required scan test pattern memory depth
     - The type of scan chains (two-bit or three-bit)
     - Note: Use the reference table in IG-XL SCAN Help to determine the SCAN type for your application based on the HSD board's scan chain configurations. Also refer to the DIB Design Guidelines for scan design rules.
  2. After determining the SCAN type for your application, follow the channel assignment rules for X2 – 12 & Extended SCAN to allocate digital channels.
  3. Develop the pin map & channel map and the levels and timing sheets and create the functional test instances and the flow sheet.
  4. Create (ATPG or hand-generated) the SCAN Patterns as follows:

Creating SCAN Vectors

Vector #	Vector type
0	parallel vector
1	parallel vector
----- <----------	**vector group boundary**
2	parallel vector
3	parallel vector
----- <----------	**vector group boundary**
4	**scan vector**
5	parallel vector
6	parallel vector
----- <-------	**vector group boundary**
7	**scan vector**
8	parallel vector
9	parallel vector
----- <-------	**vector group boundary**

1. Scan Pattern organization for Dual PatGen Mode:
   - A scan vector is considered to be an entire vector group. That is, it must occur as a vector immediately following a vector group boundary, and the number of cycles must be a multiple of the vector group size (two for dual mode).
   - The next vector is considered the start of a vector group. Dual mode scan vectors must have a cycle count that is divisible by two.
   - This affects scan vector placement. For example, in dual mode, you need a multiple of two parallel vectors before the first scan vector (this multiple can be zero). The number of parallel vectors between scan vectors must be a multiple of 2 (0, 2, 4, 6….)
   - When padding scan vectors to have an even number of cycles, the first cycle of a scan-in pin is padded with any drive high or low pin data. The last cycle of a scan-out pin is padded with don't care (X) data. An implication of using scan vector padding is that the parallel vectors must be properly grouped.

Creating SCAN Vectors

2. Ensure the pattern file header has the following:
   - For SCAN-compatible patterns, you must specify the HSD board and the timing mode.
   - Note that in the current IG-XL release PatGen mode can be either dual or single.
   - For example:
     - `digital_inst = HSDM;`
     - `opcode_mode = dual; // or single ;`
   - Specify the workbook that contains the pin map for this pattern:
     - `pinmap_workbook = "my_job.xls";`
   - Import the time set used in your pattern file:
     - `import tset tset1, tset2;`
   - Set up your SCAN type statement, for example:
     - `scan_type = X2;`
   - List the SCAN pins.

Creating SCAN Vectors: SCAN Pins Declaration

```
scan_pins = {
    <scan_in_pin>,
    <scan_out_pin >,
    <scan_in_pin>,
    <scan_out_pin
}
```

Use this format for non-adjacent pin-channels.

- This statement indicates what pins are used as scan pins.
- The scan_pins statement does not accept pin group names or channel groups. Each scan pin or channel must be listed individually.
- Example:

```
scan_pins = {
 tdi,tdo, //ch0 scan in with ch1 scan out
 }
```

SCAN Pattern File Example

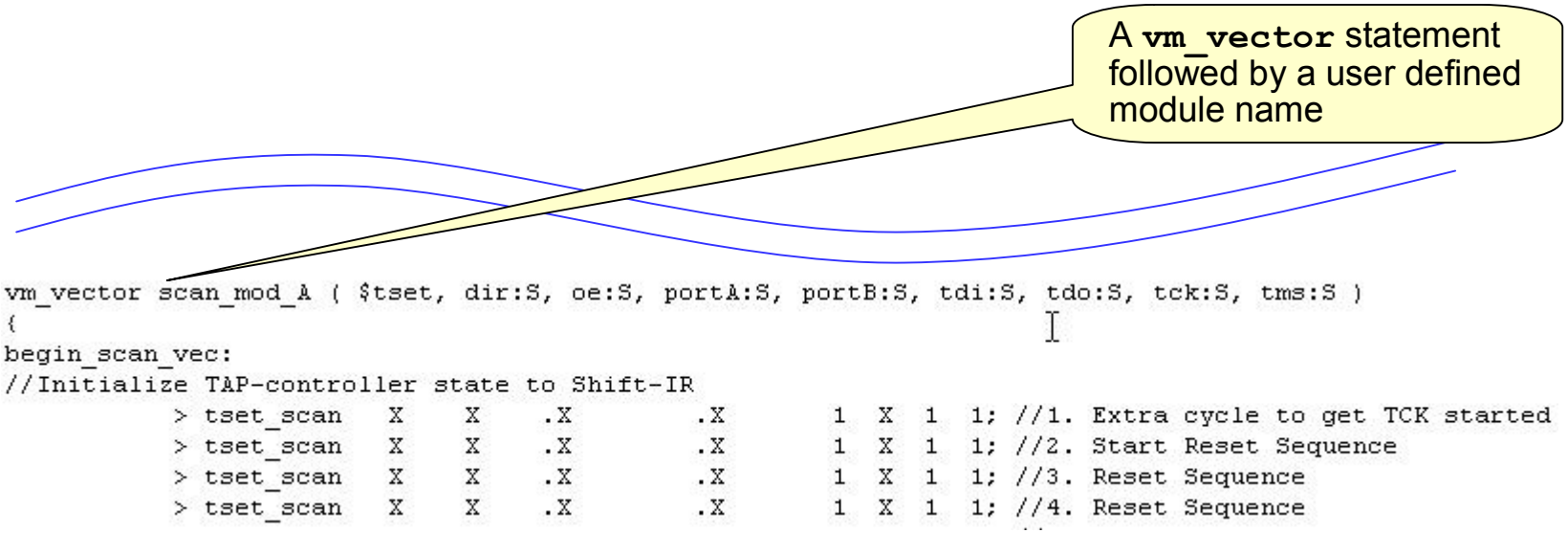
scan_type depends on the use of either two or three bit chains

Although pin group names cannot be used in the `scan_pins` declaration, pin groups names can be used as part of the vector statement.

```
digital_inst = hsd
pinmap_workbook =      \bct8245aFP.xls";
opcode_mode = du
import tset tset_scan;
scan_type = x2;
// tdi = scan input pin
// tdo = scan output pin
//-------------------------------------------------------------------------------------

scan_pins = {
          tdi, tdo,  //ch0 scan in with ch1 scan out
         }

vm_vector scan_mod_A ( $tset, dir:S, oe:S, portA:S, portB:S, tdi:S, tdo:S, tck:S, tms:S )
{
begin_scan_vec:
//Initialize TAP-controller state to Shift-IR
          > tset_scan   X    X    .X        .X       1  X  1  1; //1. Extra cycle to get TCK started
          > tset_scan   X    X    .X        .X       1  X  1  1; //2. Start Reset Sequence
          > tset_scan   X    X    .X        .X       1  X  1  1; //3. Reset Sequence
          > tset_scan   X    X    .X        .X       1  X  1  1; //4. Reset Sequence
```

Creating SCAN Vectors

3. Specify the memory to which the scan pattern is loaded.
   - You must indicate whether you are loading patterns to the VM or the SRM for parallel vectors.
   - For SCAN-compatible vectors, you must use the VM. To do this, use the following statement:
     - **`vm_vector [MODULE NAME]`**



Creating SCAN Vectors

4. A. Create your scan setup vectors:
   - The **`scan_setup`** microcode establishes the parallel vector state that is repeated during the next scan vector; it does not generate a tester cycle.
   - The **`scan_setup`** vector looks like a normal parallel vector containing the following:
     - Any parallel channel data codes on parallel pins
     - Any tset name.
     - Any extended microcode (auxiliary PatGen command, subject to timing mode restrictions)
   - The **`scan_setup`** vector can appear anywhere in the pattern, and the state applies to all future scan vectors until the next **`scan_setup`** statement.

Creating SCAN Vectors

4. A. Scan setup vectors (continued):
   - Since this vector does not generate a tester cycle, no labels are allowed on this vector.
   - Datacodes specified in a scan_setup vector for pins used in subsequent scan vectors are ignored.
     - Dummy data for scan pins must appear in this vector, although the data is not used
   - Non-scan pins will have their data applied using the formatting and timing of the specified tset.

SCAN Pattern File Example

- A vector with the **`scan_setup`** opcode does not generate a tester cycle.
- Labels **cannot** be used on scan_setup vectors. The pattern compiler gives an error.

```
scan_setup
          > tset_scan  X       X .X .X 0 L 1 0; //35-44.
scan 10
(si tdi:S > 1001101100)
(so tdo:S > LHLLHHLHHL);

          > tset_scan   X    X    .X       .X       1  L  1  1 ; //45. Last TDO bit - To Exit1-DR
          > tset_scan   X    X    .X       .X       1  X  1  1 ; //46. To Update-DR -> noop
          > tset_scan   X    X    .X       .X       1  X  1  0 ; //47. To Run-Test/Idle
          > tset_scan   X    X    .X       .X       1  X  1  1 ; //48. To Select-DR-Scan
          > tset_scan   X    X    .X       .X       1  X  1  1 ; //49. To Select-IR-Scan
          > tset_scan   X    X    .X       .X       1  X  1  0 ; //50. To Capture-IR
// Shift in instruction sample (00000010)
          > tset_scan   X    X    .X       .X       1  X  1  0 ; //51. To Shift-IR
          > tset_scan   X    X    .X       .X       0  X  1  0 ; //52. IR0
          > tset_scan   X    X    .X       .X       1  X  1  0 ; //53. IR1
          > tset_scan   X    X    .X       .X       0  X  1  0 ; //54. IR2
          > tset_scan   X    X    .X       .X       0  X  1  0 ; //55. IR3
repeat 2  > tset_scan   X    X    .X       .X       0  X  1  0 ; //56-57. IR4-IR5
          > tset_scan   X    X    .X       .X       0  X  1  0 ; //58. IR6
          > tset_scan   X    X    .X       .X       0  X  1  1 ; //59. IR7
          > tset_scan   X    X    .X       .X       1  X  1  1 ; //60. To Update-IR -> sample mode
```

Creating SCAN Vectors

4. B. Create your scan vectors:
   - Use the scan statement to define the number of cycles (max = 65K) for which scan data should be used, the scan pin type (scan-in or scan-out), the scan pins to be programmed, their radices, and the data for the scan pins.
   - Example:

```
scan 10
(si tdi:S > 1001101100)
(so tdo:S > LHLLHHLHHL);
```

Creating SCAN Vectors: Scan Vector Syntax

```
scan <cycles>
    si <scan_pin> [:<radix>] > <scan_data> //scan in
    so <scan_pin> [:<radix>] > <scan_data> ;//scan out
```

- **cycles**: The number of scan cycles to execute for this vector.
- **si**: The scan in pin.
- **so**: The scan out pin.
- **scan_pin**: The name of a scan pin or scan channel (as declared in the scan_pins statement.
- **radix**: Either S (symbolic), X or H (Hexadecimal). Default is symbolic.
- **>**: a limiter/delimiter between the scan pin and scan data.
- **scan_data**: The data for the scan pin.

SCAN Pattern File Example

```
scan_setup
            > tset_scan  X       X .X .X 0 L 1 0; //35-44.
scan 10
(si tdi:S > 1001101100)
(so tdo:S > LHLLHHLHHL);

            > tset_scan   X    X    .X        .X       1  L  1  1 ; //45. Last T
            > tset_scan   X    X    .X        .X       1  X  1  1 ; //46. To Upd
            > tset_scan   X    X    .X        .X       1  X  1  0 ; //47. To Run
            > tset_scan   X    X    .X        .X       1  X  1  1 ; //48. To Sel
            > tset_scan   X    X    .X        .X       1  X  1  1 ; //49. To Sel
            > tset_scan   X    X    .X        .X       1  X  1  0 ; //50. To Cap
// Shift in instruction sample (00000010)
            > tset_scan   X    X    .X        .X       1  X  1  0 ; //51. To Shift-IR
            > tset_scan   X    X    .X        .X       0  X  1  0 ; //52. IR0
            > tset_scan   X    X    .X        .X       1  X  1  0 ; //53. IR1
            > tset_scan   X    X    .X        .X       0  X  1  0 ; //54. IR2
            > tset_scan   X    X    .X        .X       0  X  1  0 ; //55. IR3
repeat 2    > tset_scan   X    X    .X        .X       0  X  1  0 ; //56-57. IR4-IR5
            > tset_scan   X    X    .X        .X       0  X  1  0 ; //58. IR6
            > tset_scan   X    X    .X        .X       0  X  1  1 ; //59. IR7
            > tset_scan   X    X    .X        .X       1  X  1  1 ; //60. To Update-IR -> sample mode
// sample/boundary reg/length = 18
            > tset_scan   X    X    .X        .X       1  X  1  0 ; //61. To Run-Test/Idle
            > tset_scan   X    X    .X        .X       1  X  1  1 ; //62. To Select-DR-Scan
            > tset_scan   X    X    .X        .X       1  X  1  0 ; //63. To Capture-DR
            > tset_scan   X    X    .X        .X       1  X  1  0 ; //64. To Shift-DR

// Shift in test seq 1001101100, shift out test seq 1001101100 thru BSR (18-bit long)

            > tset_scan X X .X .X 0 L 1 0; //65-92.  EXTENDED Scan from 27-->28
scan 28
(si tdi:S > 10011011001111111111111111111)
(so tdo:S > XXXXXXXXXXXXXXXXXXHLLHHLHHLL);
```

The number of symbolic scan characters for a pin must match exactly the number of scan cycles specified for that scan vector

Scan Pattern Programming Syntax

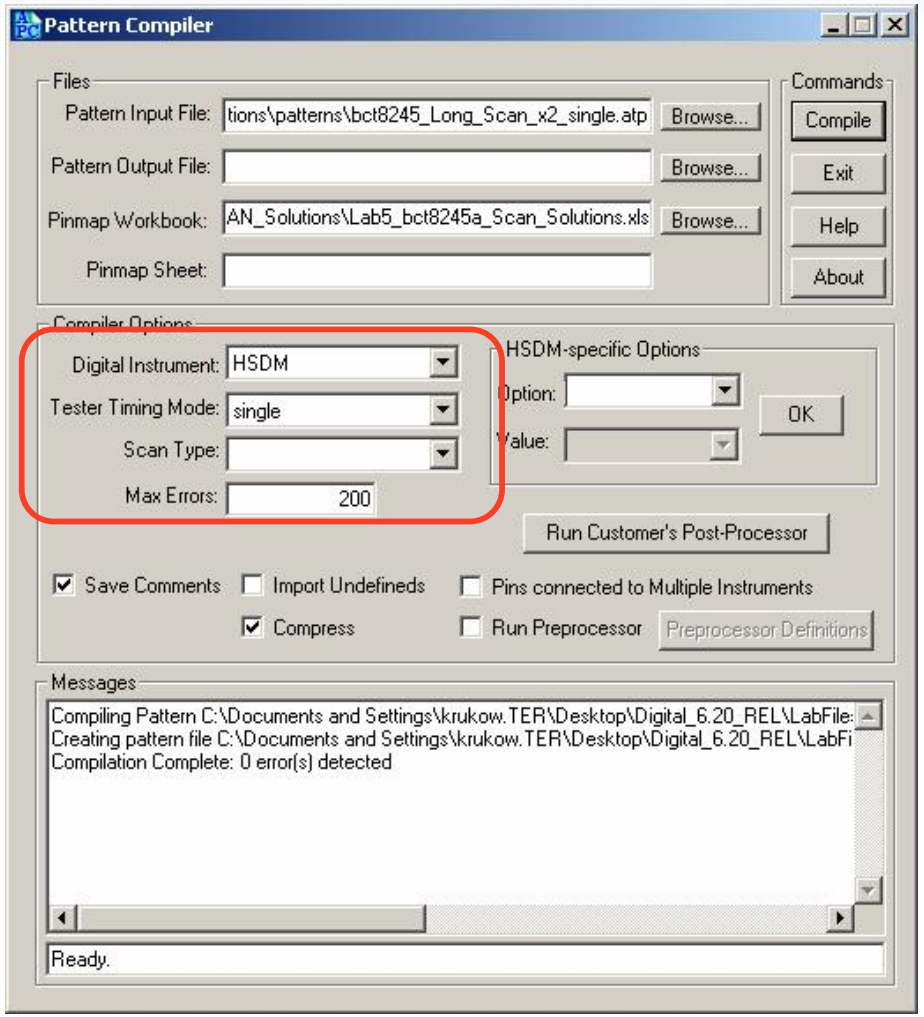
- Shorthand for repeating strings
  - Syntax: [n-string]
  - Example: [5-01] = 0101010101
  - Can be used with symbolic or hexadecimal data
  - Shorthand format is not preserved in the PatternTool or on reverse compile
  - Cannot be nested
- White space can be inserted anywhere in the scan data to improve readability.
- White space must be used if the number of cycles in the scan vector is greater than 1024, since the pattern compiler does not handle single strings with more than 1024 characters.

Scan Pattern Programming Examples

- `Scan 256 (si pin07:X > [64-f]);`
- `Scan  12 (si pin07 > 010111001111)`
  `(so pin02 > LHLHHHLLHHHH);`
- `Scan 40 (si edat:X > f0f0f0f0f0)`
  `(so odat:X > abcdef0120)`

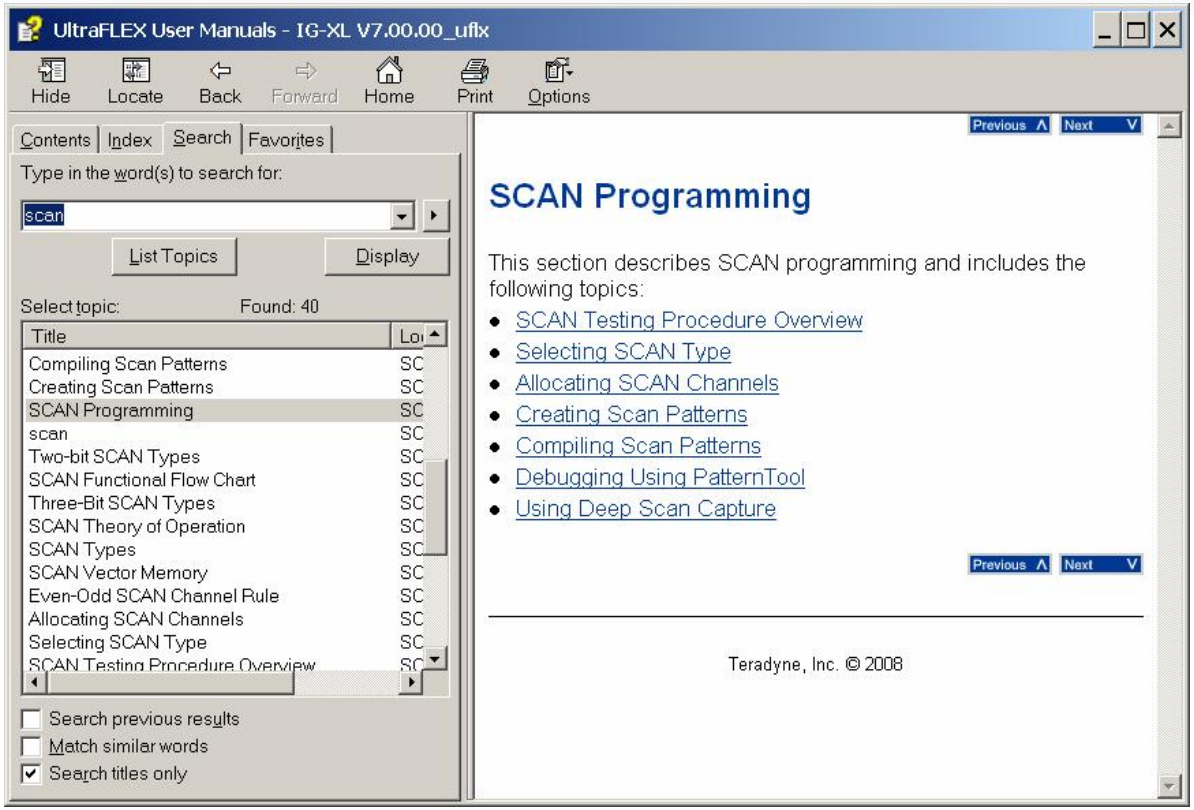
Pattern Compiler Options

- In the Pattern Compiler window, use the following options to compile SCAN vectors:
  - Tester Timing Mode
    - Choose single or dual
  - Scan Type
    - parallel = Expand scan vectors into parallel vectors
    - x2
    - x3
    - x4
    - x6
    - x8
    - x12
    - x24
    - extended



SCAN Programming: UltraFLEX User Manuals

- For more information on SCAN pattern programming, see the UltraFLEX User Manuals (IG-XL Help).



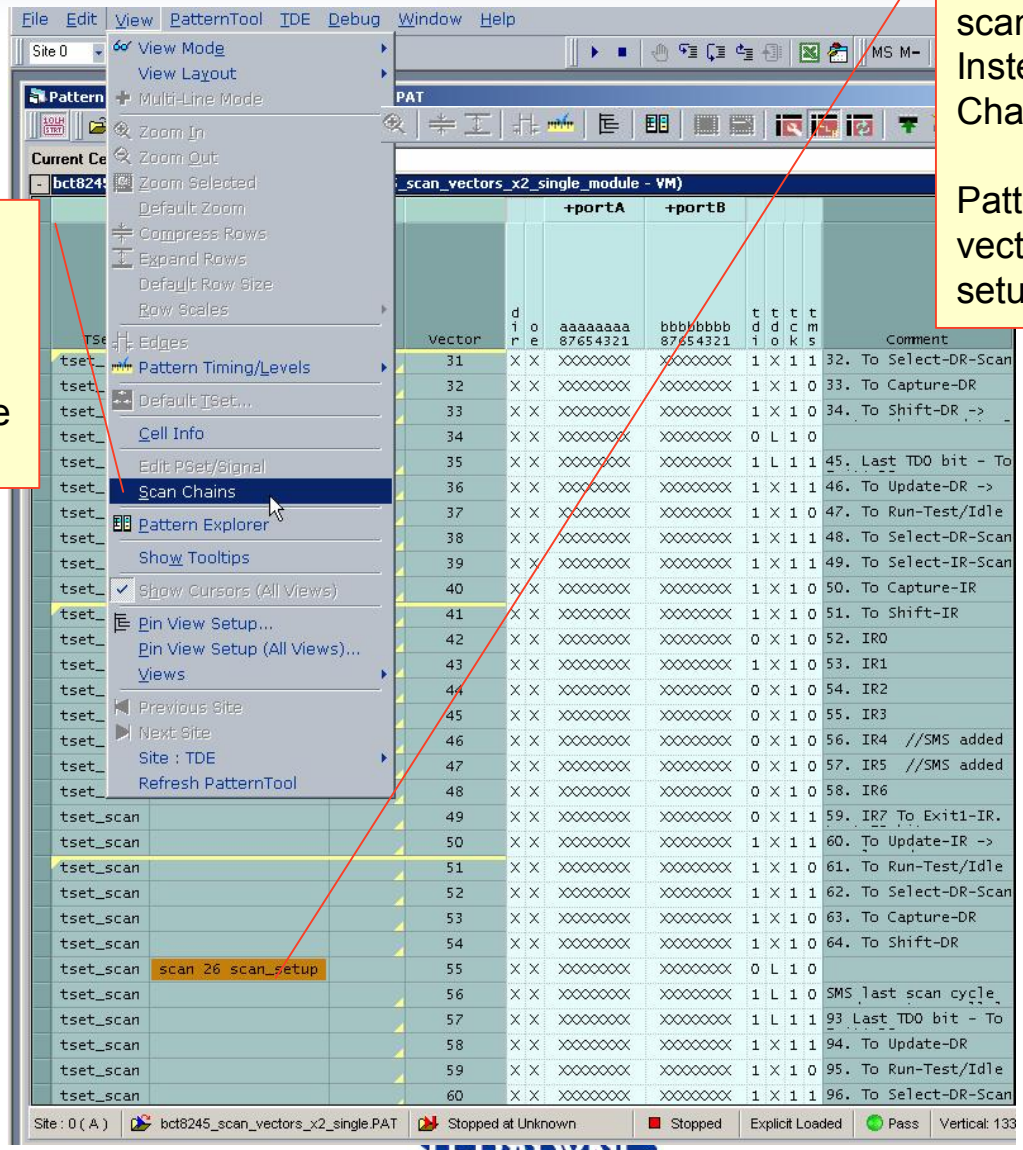
Scan Debugging Tools

Fail Processing

- History RAM (HRAM)
  - 1024 cycles deep
  - Capture modes:
    - Capture all
    - Capture fails only
    - Capture pass only
- Per Pin Capture Memory (CMEM)
  - 64M/128Mb per pin
  - Can capture selected central data or pin data using CMEM VBT
  - Move captured data to Host or DSP's for processing
- 32 bit fail counter per pin

Scan Pattern Tools

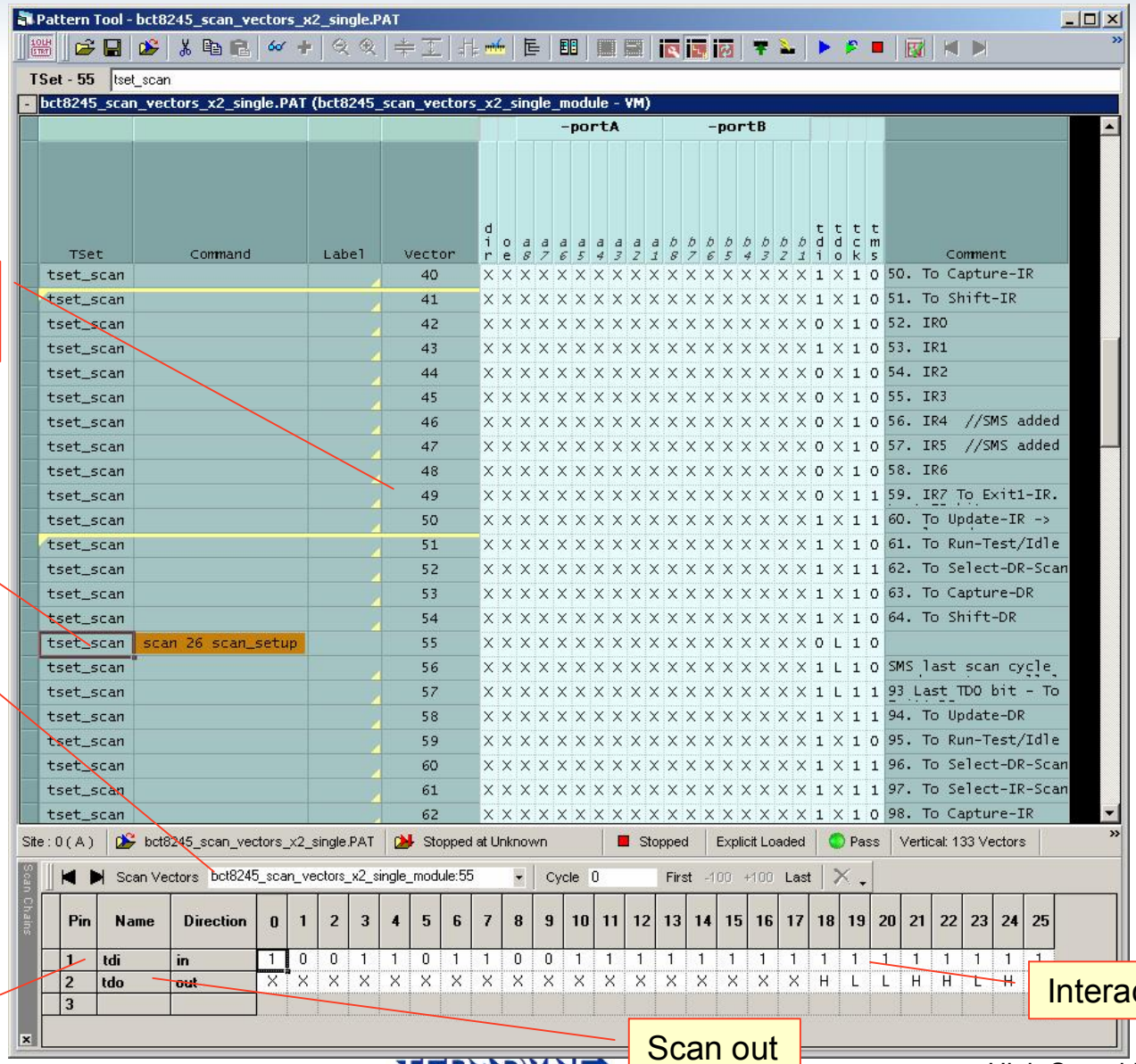
To view or edit scan data, use **View>Scan Chains**. The Scan Chains pane appears in the lower part of the PatternTool window.



PatternTool does not display scan data in its main vector grid. Instead, it uses a special Scan Chains pane.

PatternTool uses the main vector grid to display the scan setup data

Scan Chain Pane Using PatternTool



Parallel vectors

Scan vector

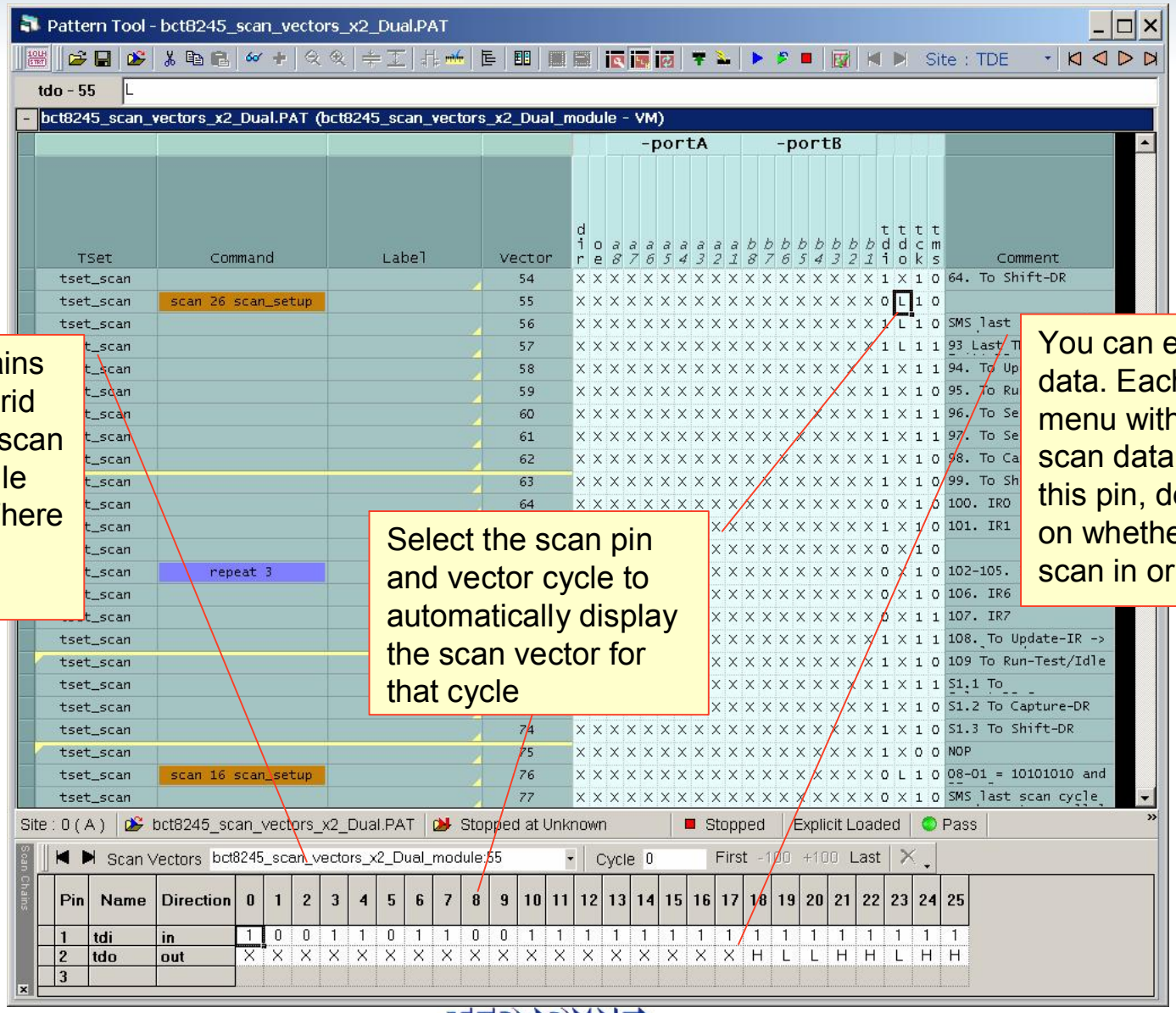
Scan Vector cycle number in scan chain view.

Scan in

Interactive editing

Scan out

Scan Pattern Tools: Scan Chain View (Vertical)

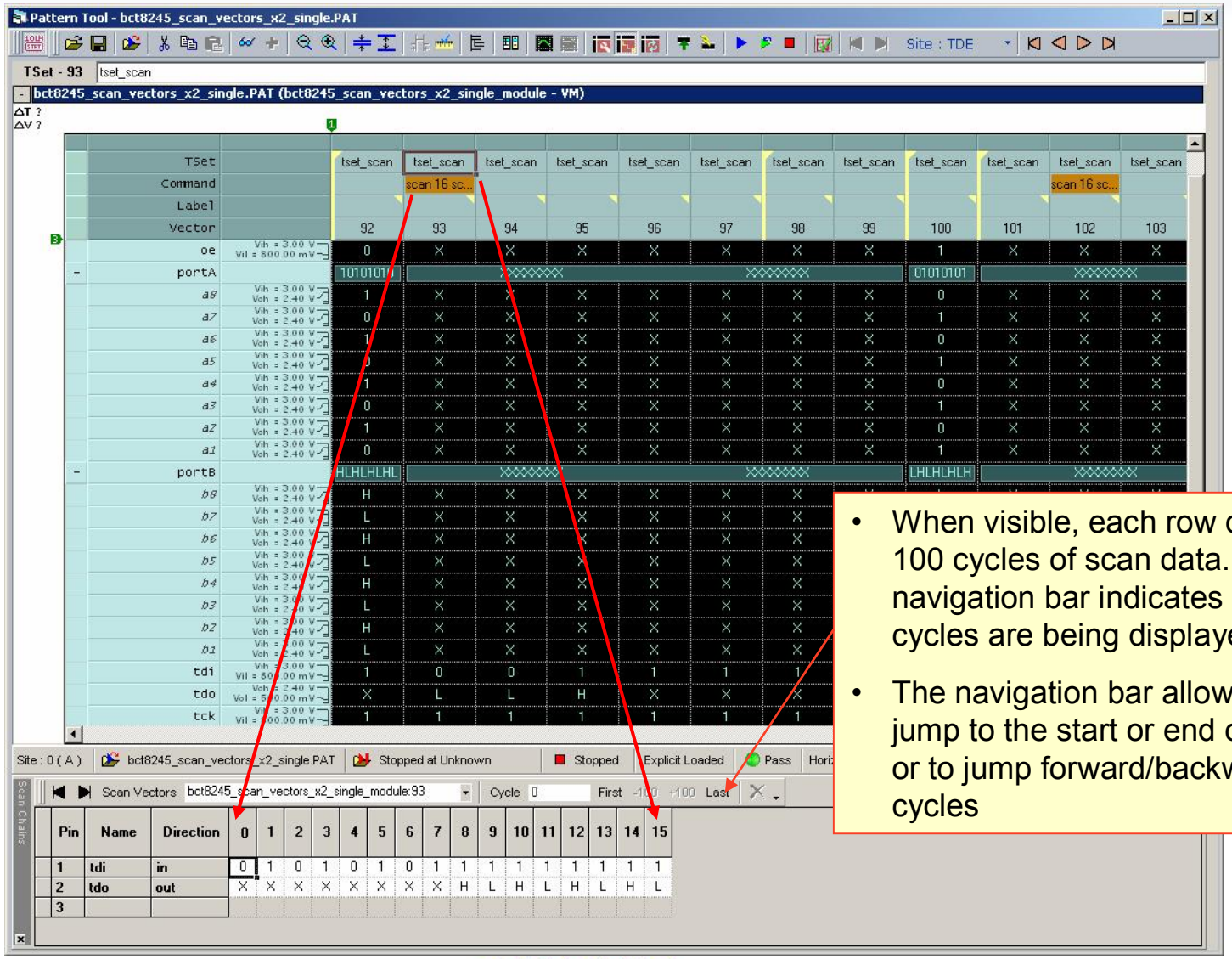


The Scan Chains pane uses a grid to display the scan data for a single scan vector. There is one row for each scan pin

Select the scan pin and vector cycle to automatically display the scan vector for that cycle

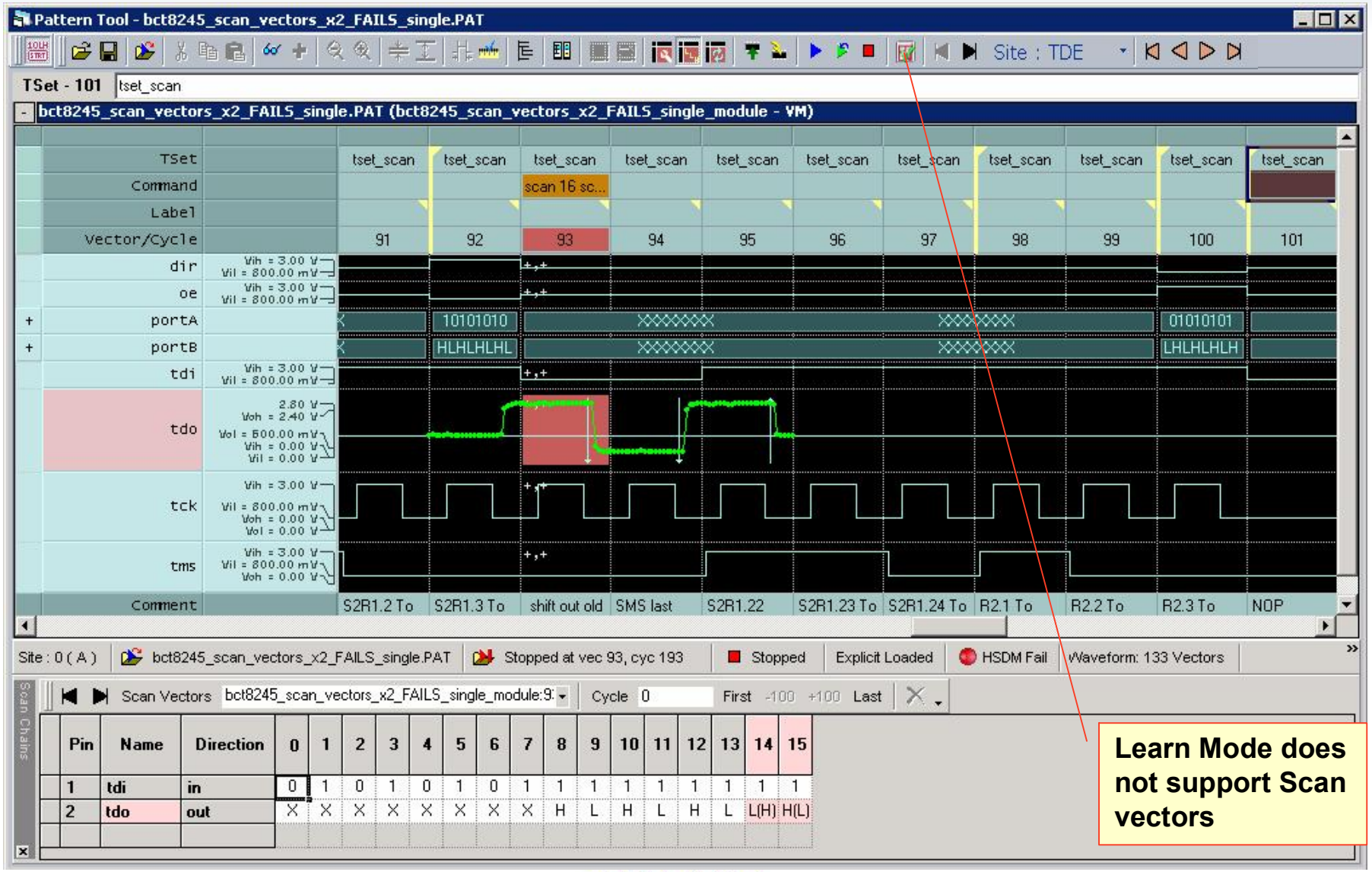
You can edit scan data. Each cell has a menu with the valid scan data values for this pin, depending on whether the pin is scan in or scan out.

Scan Pattern Tools: Scan Chain View (Horizontal)

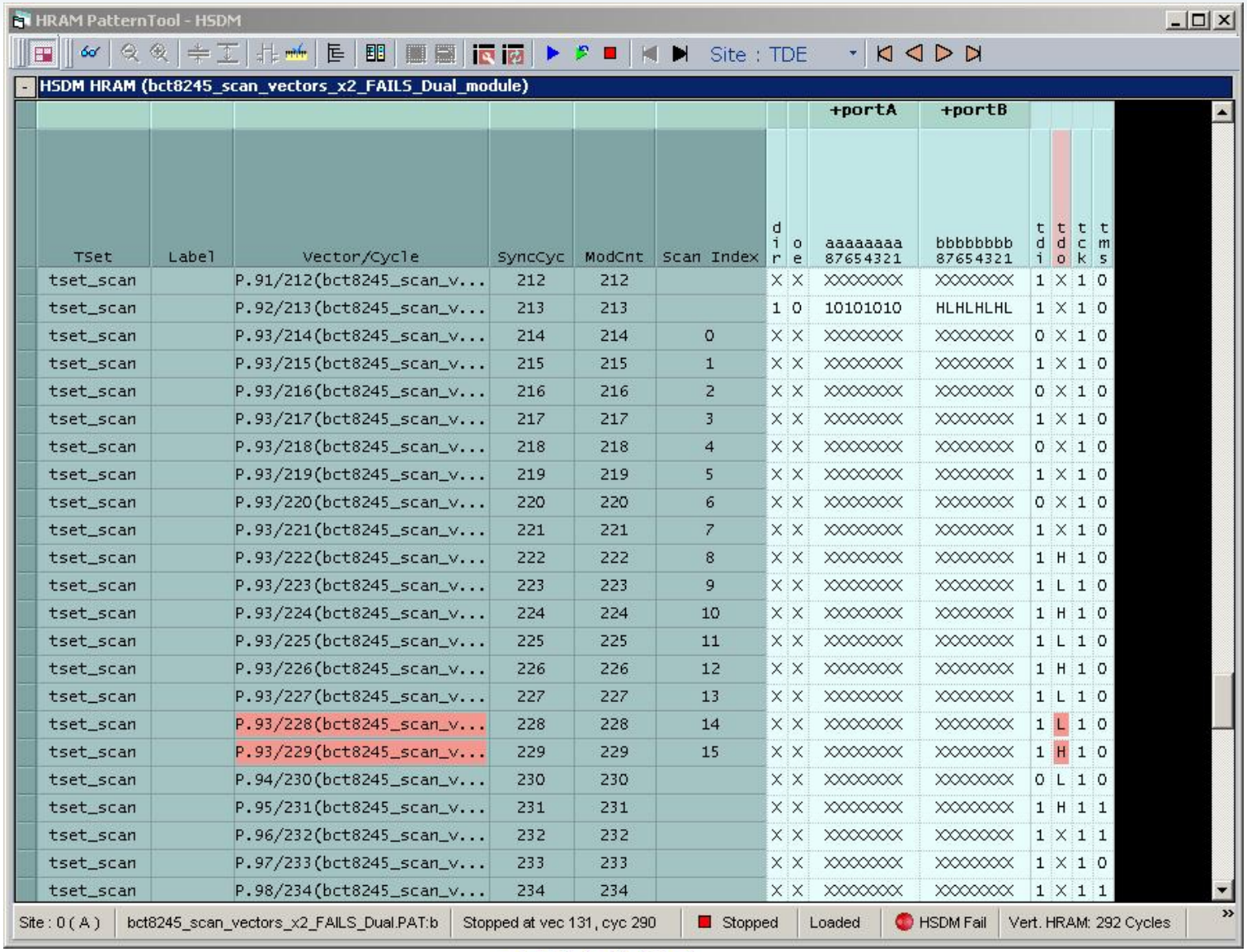


- When visible, each row displays 100 cycles of scan data. The navigation bar indicates which 100 cycles are being displayed.
- The navigation bar allows you to jump to the start or end of the data, or to jump forward/backward 100 cycles

Scan Pattern Tools: Scan Chain View FAILS

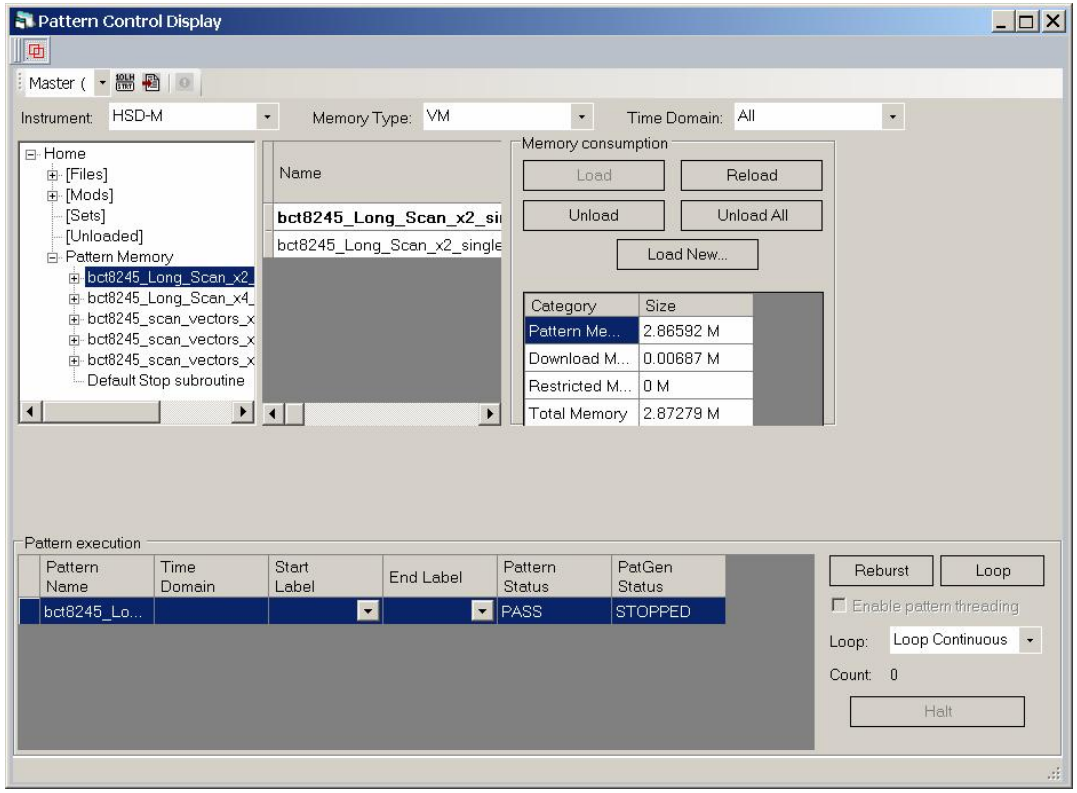


Scan Pattern Tools: HRAM Capture All



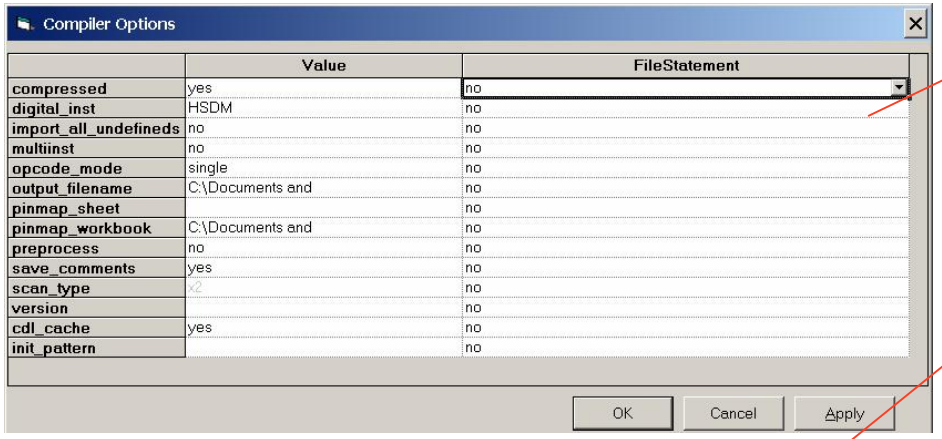
Scan Pattern Tools: Pattern Control Display

- This display provides useful information on all vector modules used in the current job along with size, opcode mode, and file location. and the display provides a total memory consumption status.
- Additionally the PCD provides the ability to load new patterns and either burst or loop new or existing scan or parallel vector patterns.

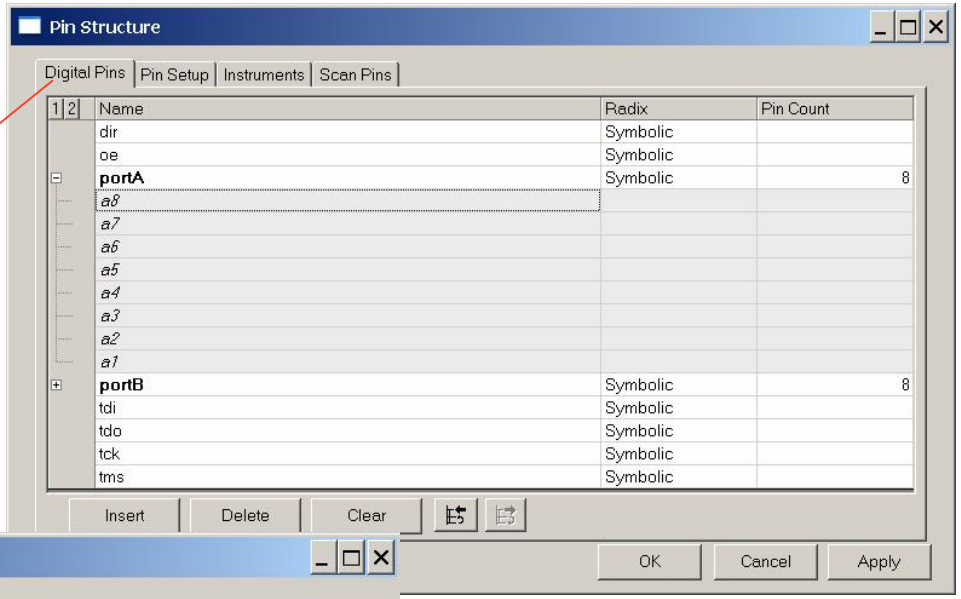


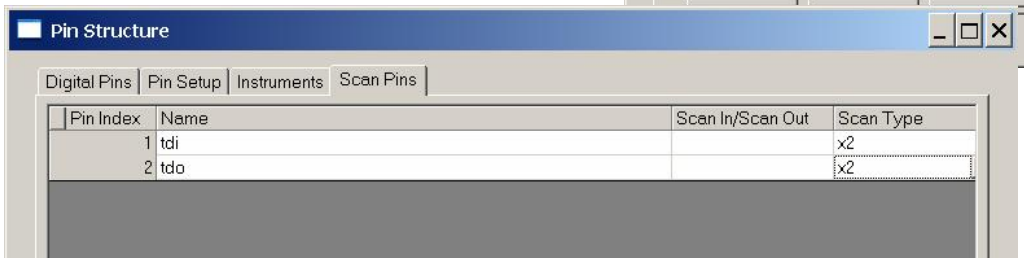
Scan Pattern Tools

In TDE, select **File>Options** and view compiler setting such as scan_type, opcode_mode, etc.



In TDE, select **Edit>Pin Structure** and view the Digital and Scan Pins setup.



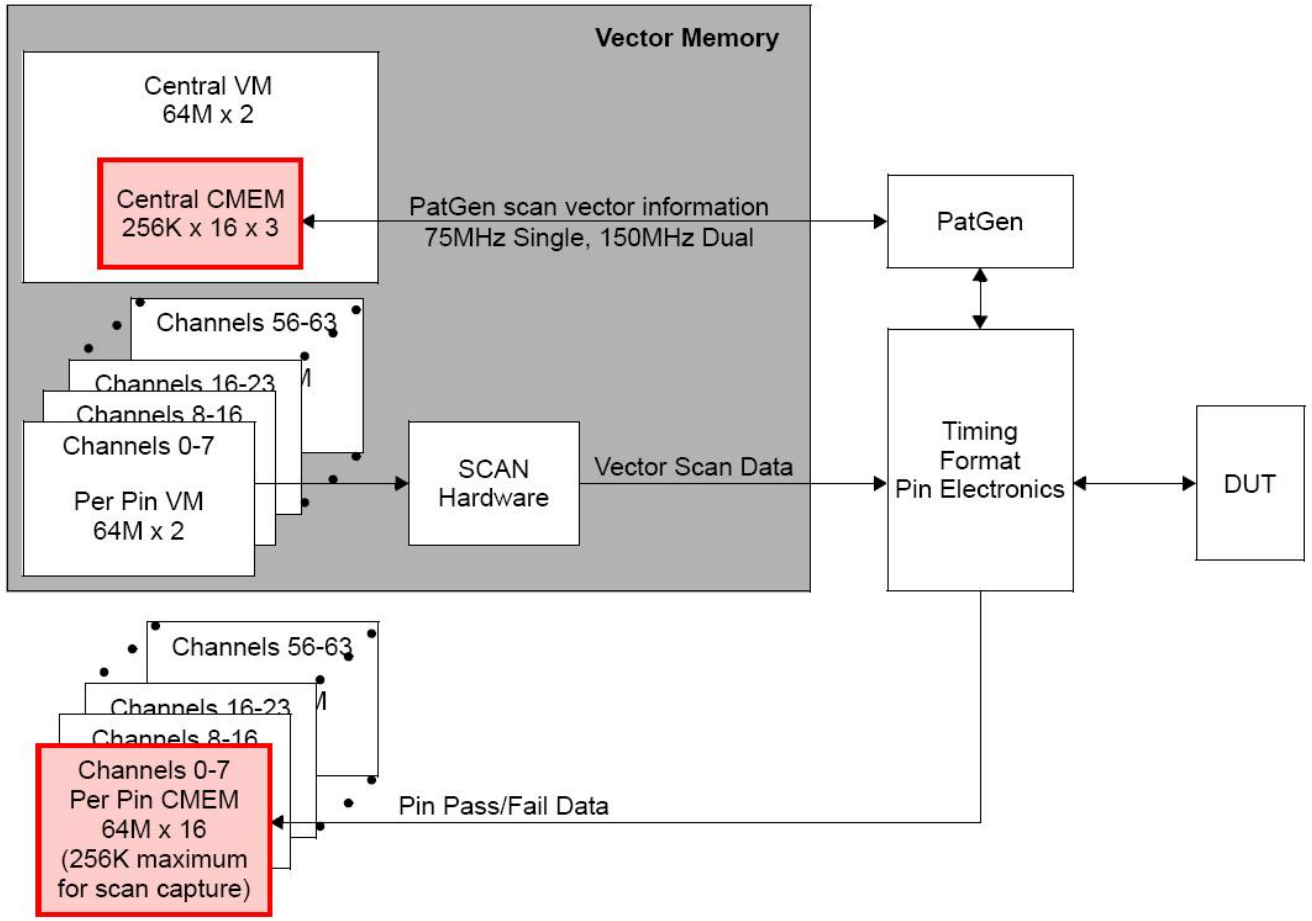


Setting IG-XL Offline License Options for Scan

- When working offline on a workstation, the IG-XL simulator fully enables all features and capabilities.
- You can configure the simulator to enforce the same license limits that are in effect on your test systems. This allows you to verify that your tester has sufficient licenses to run your test program.
- To have the simulator enforce licenses, copy the <LICENSE_ENABLERS> section from the CurrentConfig.txt file on the tester you want to emulate to your workstation's SimulatedConfig.txt file. CurrentConfig.txt is always stored in the tester directory. If you performed a standard IG-XL installation, the files are located in:
  - C:\Program Files\Teradyne\IG-XL\<version>\tester
- CurrentConfig.txt Determines instrument locations in test head slots. Defined by software in:
  - \IG-XL\<version>\tester\CurrentConfig.txt

Deep Scan Capture Programming

- HSD1000 and UltraPin800 digital instruments allow you to store scan fail data into LFVM (CMEM Capture Memory), far greater than the 512/1024 HRAM cycles to be captured.

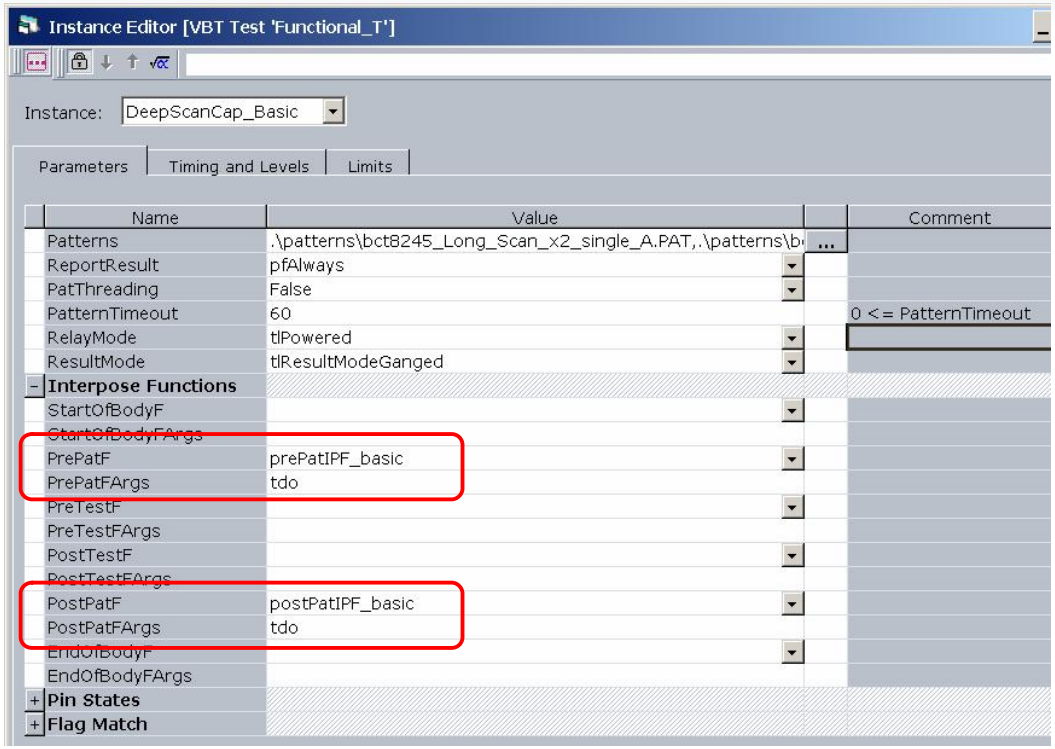


Deep Scan Capture Programming

- Two main types of data can be read from CMEM:
  - Central Data: cycle count, vector offset, repeat count, scan index, and pattern name
  - Pin Data: pass or fail, expected state, and actual state of the pin
- The central data that HRAM currently captures can now be stored into CMEM, storing up to 256K, storing three 16 bit fields in single mode and 512K in dual mode.
- You must specify which central fields to store. The choices are:
  - Absolute Cycle Count
  - Module Cycle Count
  - Sync Cycle Count
  - Pattern Name (VM Only)
  - Vector Offset (VM Only)
  - Scan Index
  - Loop Repeat Count

Deep Scan Capture Programming

- Capture of data into CMEM is independent of data captured into HRAM.
- Each memory can store different types of data and different amounts of data.
- A typical use of Deep Scan Capture is to implement pre-pattern and post-pattern interpose functions used with a scan test that captures failing cycles, reads back the data from CMEM, and writes the data to an output file that can be used by design engineers for fault analysis.



Deep Scan Capture Programming: Supported Use Cases

- Failing Scan pattern captures the first 10K of fails as absolute cycle count, module count, or sync cycle count with pins that fail.
  - With the above use case, multiple sites are enabled (1 PatGen per site) along with Pattern Threading enabled or disabled, scan fail output files are created and sent to the design engineer for analysis with EDA tools.
- Failing Scan pattern captures fails (greater than HRAM) as absolute cycle count, module count, or sync cycle count with pins that fail and the scan fail data is saved into an STDF file and sent to the design engineer. A GDR (Generic Data Record) is created for scan fails.
- Failing Scan pattern captures fails (greater than HRAM) with pattern files/modules contained in a Pattern Set. Scan data is captured in CMEM as a single output file.
- Similar to the above with a separate scan fail output file for each module in the pattern set.
- Similar to the first use case, except that an overflow of central CMEM will cause an overflow alarm.

Deep Scan Capture Programming

- Setting up Deep Scan Capture:
  - To set the number of CMEM cycles to be captured and the type of capture data:
    - TheHdw.Digital.CMEM.SetCaptureConfig
  - To set up which data storage fields to capture in central CMEM:
    - TheHdw.Digital.CMEM.CentralFields
- Central CMEM Get Current Setup:
  - To retrieve the setup information:
    - theHdw.Digital.CMEM.GetCaptureConfig

Deep Scan Capture Programming: Reading Capture Data

- Get the pattern generator information stored in CMEM:
  - **`TheHdw.Digital.CMEM.PatGenInfo`**
- Get the pattern module names stored in CMEM:
  - **`TheHdw.Digital.CMEM.PatternName`**
- Returns the per-site fail cycle scan data:
  - **`TheHdw.Digital.Pins.CMEM.StoredCycleData`**
  - Two SiteVariants are returned:
    - Index: Cycles in CMEM that contain a failure
    - Data: Bitmask representing which pins failed on a given cycle

Deep Scan Capture Programming

- The prepattern interpose function is used to set the capture config for Deep Scan Capture:

```
Public Function prePatIPF(argc As Integer, argv() As String)
    Call TheHdw.Digital.CMEM.SetCaptureConfig(-1, CmemCaptFail)
        TheHdw.Digital.CMEM.CentralFields = tlCMEMAbsoluteCycle
End Function
```

- A “-1” passed into SetCaptureConfig indicates that the full size of CMEM will be used:
  - When Central fields are not stored CMEM is 64M major cycles deep
  - When Central Fields are stored CMEM is 256K major cycles deep

Deep Scan Capture Programming

- There are limitations on the amount of central data field information that can be stored which corresponds to the number of HSD Patgens required.
- “Field” values can be OR’d together to allow multiple fields to be stored at the same time.
- CMEM central data storage field requirements:

Field	Number of bits	Number of 16 bit fields required	Number of HSD patgens required
Absolute cycle count	40	3	1
Module Cycle Count	40	3	1
Sync cycle count	40	3	1
VM offset	32	2	1
VM Pattern	20	2	1
Scan Index	28	6	2
Loop/Repeat count	28	2	1

Deep Scan Capture Programming

- Central data is read in the following way:

```
' Read back the central data.
Dim vectorData() As Long
Dim scanIndexData() As Long
vectorData = TheHdw.Digital.CMEM.PatgenInfo(tlCMEMVMVectorOffset)
scanIndexData = TheHdw.Digital.CMEM.PatgenInfo(tlCMEMScanIndex)
```

Deep Scan Capture Programming

- Central data is read in the following way (continued):

```
Dim sitePinData As New SiteVariant
Dim siteIndexData As New SiteVariant
Dim maxIndex As Long
Dim indexArr() As Long
Dim pinDataArr() As Double
'Read the failing pins per site
Call TheHdw.Digital.Pins(argv(0)).CMEM.StoredCycleData( siteIndexData,
                                                        sitePinData, 10000
maxIndex = -1
Dim site As Variant
For Each site In TheExec.Sites
     pinDataArr = sitePinData
     indexArr = siteIndexData
     If(indexArr(UBound(indexArr)) > maxIndex) Then
         maxIndex = indexArr(UBound(indexArr))
     End If
Next site
```

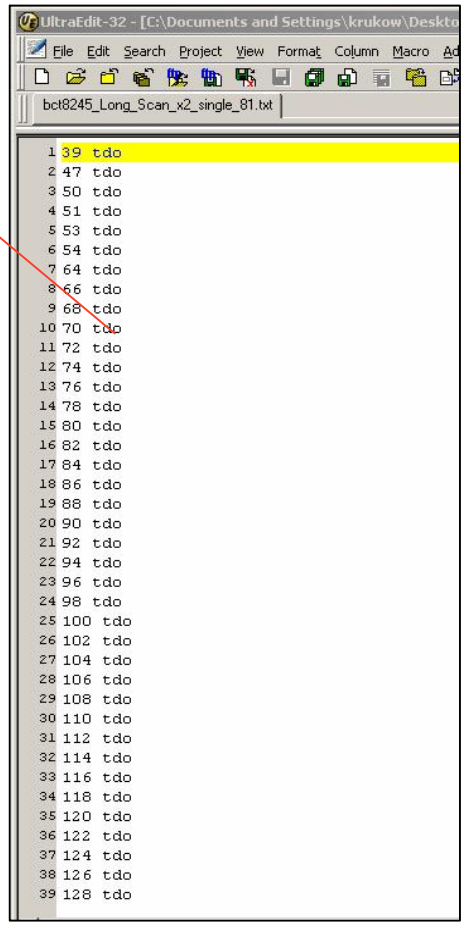
Deep Scan Capture Programming

- Two main types of data can be read from CMEM and sent to an output data file:
  - Central Data: cycle count, vector offset, repeat count, scan index, and pattern name
  - Pin Data: pass or fail, expected state, and actual state of the pin

```
Number   Site  Test Name       Pattern              1st Failed Cycle  Total Failed Cycles
400      1     LongScan_x2_Single bct8245_Long_Scan_x2_single_module (F) 39               512

  PINS
                        doaaaaaaaabbbbbbbbtttt
                        ie87654321876543211ddcm
                        r                  ioks
 Cycle Relative  Repeat


     39      10       0 XXXXXXXXXXXXXXXXX1H10
-> (F)                   GGGGGGGGGGGGGGGGGGMGG

     47      18       0 XXXXXXXXXXXXXXXXXOH10
-> (F)                   GGGGGGGGGGGGGGGGGGMGG

     50      21       0 XXXXXXXXXXXXXXXXX1H10
-> (F)                   GGGGGGGGGGGGGGGGGGMGG

     51      22       0 XXXXXXXXXXXXXXXXX1H10
-> (F)                   GGGGGGGGGGGGGGGGGGMGG

     53      24       0 XXXXXXXXXXXXXXXXX1H10
-> (F)                   GGGGGGGGGGGGGGGGGGMGG

     54      25       0 XXXXXXXXXXXXXXXXX1H10
-> (F)                   GGGGGGGGGGGGGGGGGGMGG

     64      34       0 XXXXXXXXXXXXXXXXXOH10
-> (F)                   GGGGGGGGGGGGGGGGGGMGG

     66      34       0 XXXXXXXXXXXXXXXXXOH10
-> (F)                   GGGGGGGGGGGGGGGGGGMGG

     68      34       0 XXXXXXXXXXXXXXXXXOH10
-> (F)                   GGGGGGGGGGGGGGGGGGMGG
```

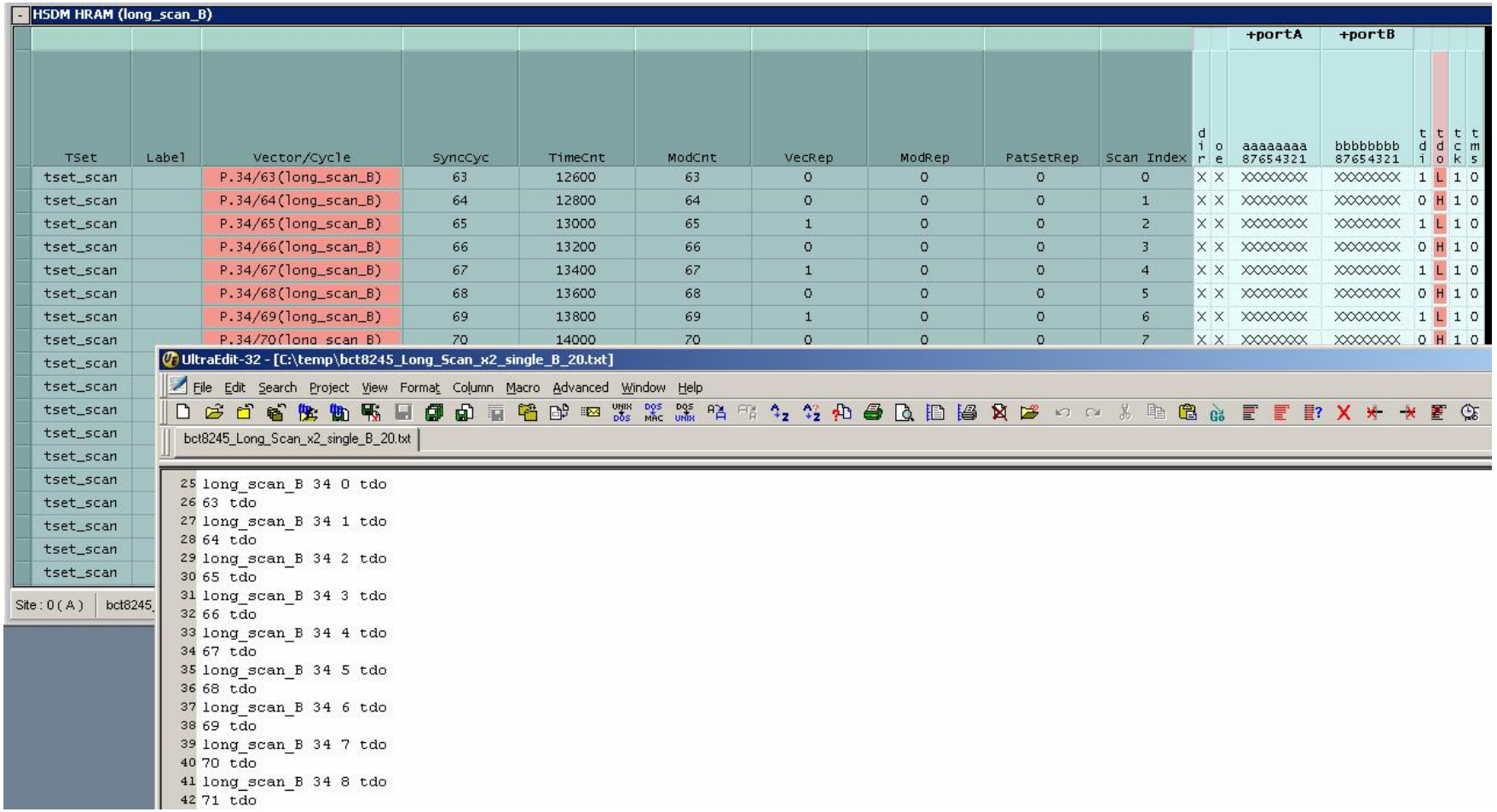
The scan output file is used for design engineering analysis containing fail information greater than 512/1024 locations for both central data and pin data from CMEM

The data collect output file contains HRAM fail information but is limited to fails no greater than 512/1024



Deep Scan Capture Programming

- CMEM setup to capture and send to an output file: pattern name, vector offset, cycle count and scan index:



Deep Scan Capture Programming

- Notes on using Central CMEM:
  - Since Central CMEM data is stored in the same memory as vector data, the pattern is limited to 75MHz in Single mode and 150MHz in Dual mode.
  - The VM must source and capture simultaneously, causing a bandwidth bottleneck. As a result, deep scan capture is guaranteed at frequencies only up to 75 MHz in single mode and 150 MHz in dual mode
    - You can disable this error check and run faster, but you do so at your own risk.
  - Central CMEM must be set up after ApplyLevelsTiming has been called (i.e., PrePattern Interpose function).
  - The test system must have sufficient hardware resources
    - For example, storing Scan Index into Central CMEM requires two HSD boards.
- See the IG-XL User Manuals for more information on Using Deep Scan Capture

Summary/Review

1. Define an HSD Scan chain:
   - 2 channels,1 scan in and 1 scan out
2. Describe the HSD 2 bit and 3 bit Scan modes:
   - The 2 bit scan mode requires 1 bit for scan in and 1 bit for scan out and the 3 bit scan mode requires 1 bit for scan in and 2 bits for scan out. Using the 2 bit scan provides greater scan depth but has no “X” (don’t care) state.
3. Describe the difference between Scan Packing and Scan Stacking:
   - Scan Packing: Compressing vector memory to increase VM depth
   - Scan Stacking: Using the VM of other channels to increase scan depth
4. When using the HSD in parallel scan mode, what are the number of bits per cycle and what is the maximum VM depth?
   - Pattern pin data requires three bits per cycle. The maximum scan pattern depth is limited by the maximum VM depth which is 64M cycles running in Single PatGen mode or 128M cycles running in Dual PatGen mode.

Summary/Review

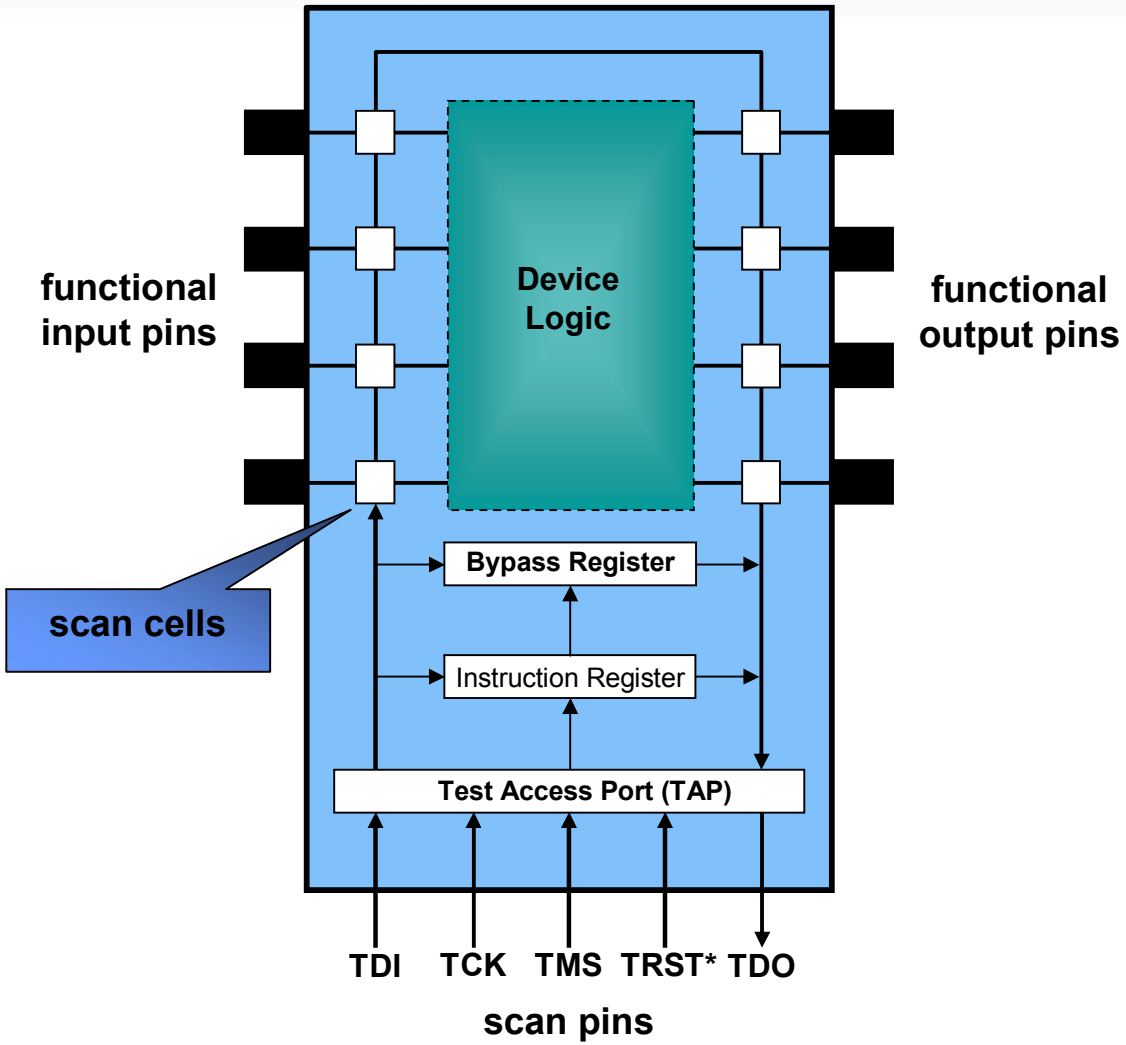
4. What Scan In and Scan out channels are available for scan region 4 using the X4 mode?
   - Scan In channels available would be Channels 24 & 28; Scan Out channels available would be Channels 25 & 29.
5. In Scan pattern programming, when must white space be used?
   - If the number of cycles in the scan vector is greater than 1024, since the pattern compiler does not handle single strings with more than 1024 characters.
6. PatternTool does not display scan data in its main vector grid. How do you view or edit the scan data?
   - Use View>Scan Chains. The Scan Chains pane appears in the lower part of the PatternTool window.

Digital Programming: Lab 8

- Scan programming and debugging
- Lab 8 objectives:
  - Observe and compile scan patterns
  - Load and run an UltraFLEX scan pattern program and use IG-XL Pattern Tools to view the scan pattern data.
- Approximate time: 90 minutes
- Offline/tester lab
- Go to Lab 8 in Digital Lab Guide

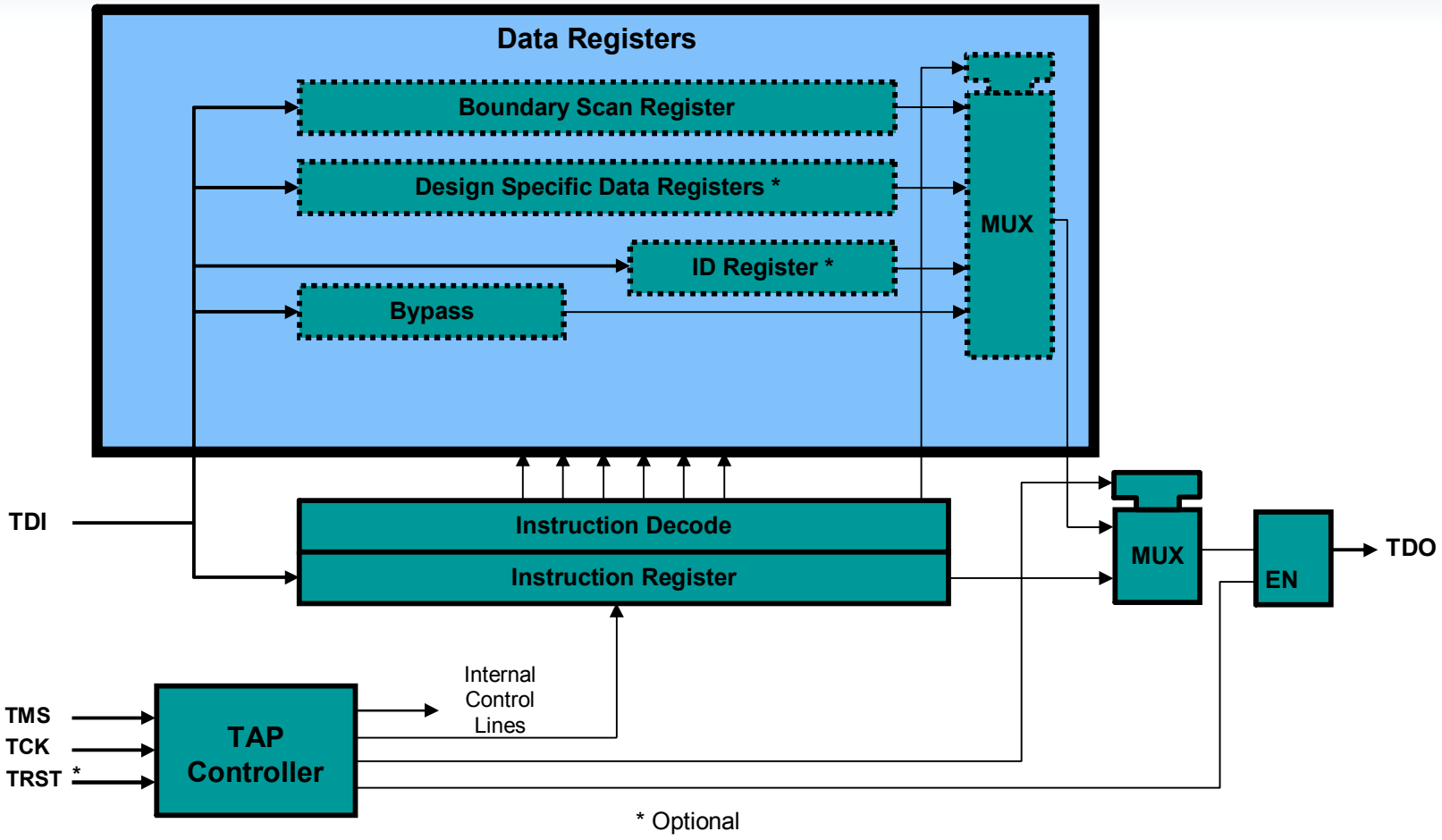
Appendix A:
Device Scan Basics

Boundary-Scan Architecture



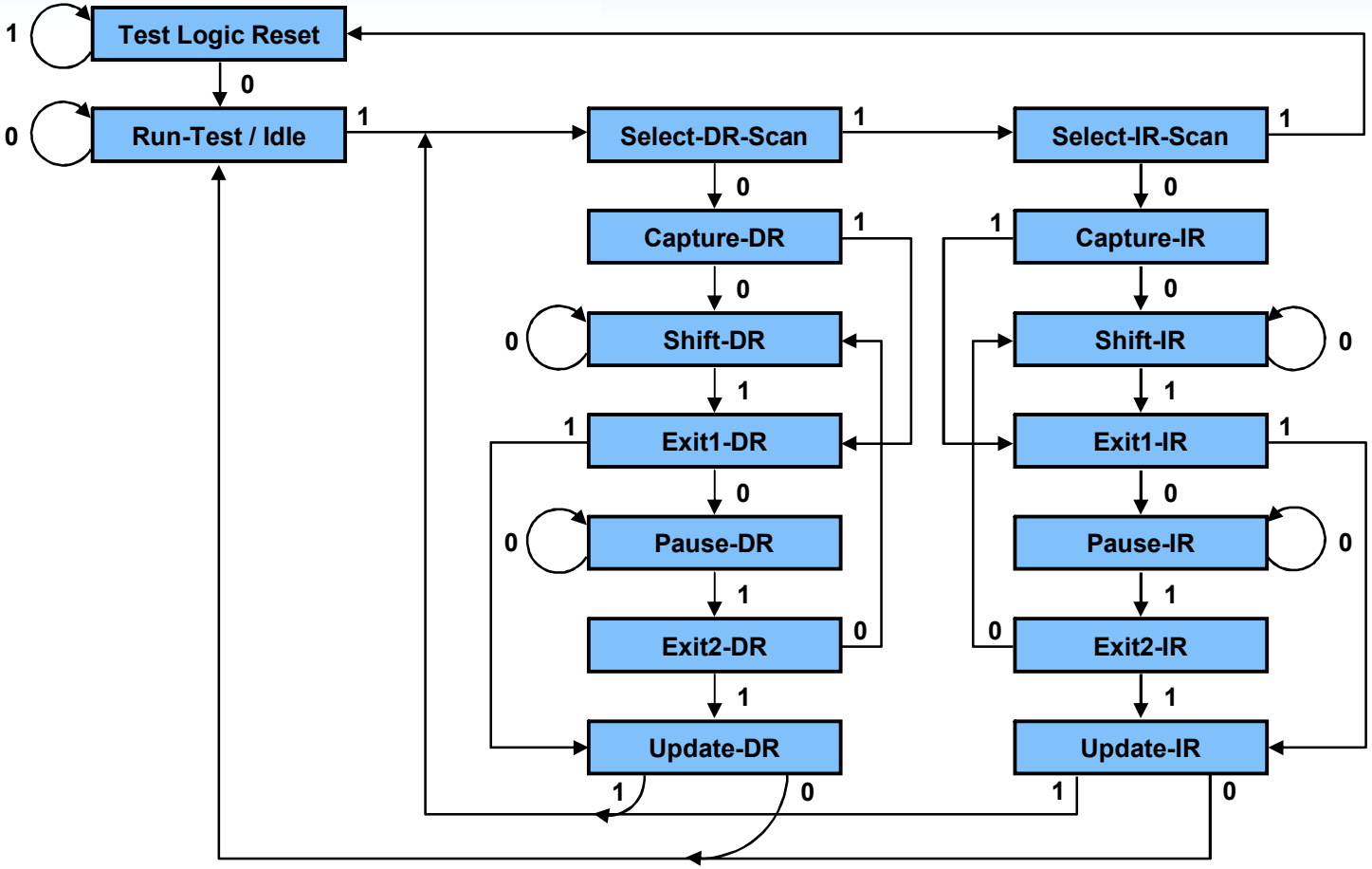
SN74BCT8245A – Scan Test Device with Octal Bus Transceivers

Boundary-Scan Registers



SN74BCT8245A – Scan Test Device with Octal Bus Transceivers

TAP Controller State Diagram



SN74BCT8245A – Scan Test Device with Octal Bus Transceivers

TAP Signals

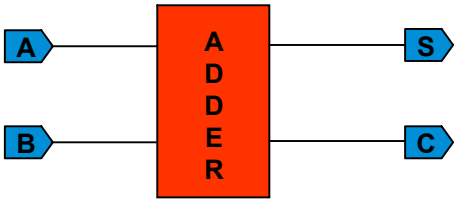
- Test Clock (TCK): This input provides the clock for the test logic. TCK is a dedicated input that allows the serial test data path to be used independent of component-specific system clocks. It also permits shifting of test data concurrently with normal component operation.
- Test Data Input (TDI): Provides serial input for test instructions shifted into the Instruction Register and for data shifted through the Boundary Register or other data registers. Values are clocked into the selected register on a rising edge of TCK.
- Test Data Output (TDO): The serial output for test instructions and data from the Boundary Register or other data registers. The contents of the selected register (instruction or data) are shifted out on the falling edge of TCK.

TAP Signals (Continued)

- Test Mode Select (TMS): The logic level of TMS, along with a rising edge applied to TCK, cause the movement from one state to another through the TAP controller. This, in turn, allows movement of data and TAP instructions through the state machine.
- Test Reset (TRST*): This optional input provides asynchronous initialization of the TAP Controller, which in turn causes asynchronous initialization of other test logic included in the design. The reset places the device in the normal operating mode and makes the Boundary Register inactive.

SCAN Definitions

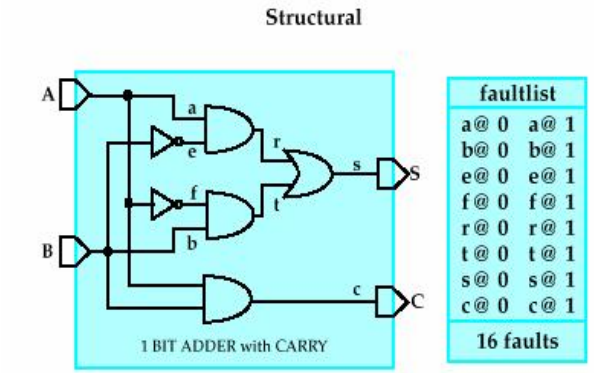
- Functional Test:
  - Targeted to verify that the circuit performs its intended end-use behavior for logic, timing, and power.
- Structural Test:
  - Verify the topology of the chip. (i.e., all gates are functioning and all interconnects are hooked up)
  - Targeted specifically for manufacturing defects of the chips.



1 Bit Adder

A + B = S
0 + 0 = 0 with C = 0
0 + 1 = 1 with C = 0
1 + 0 = 1 with C = 0
1 + 1 = 0 with C = 1

In functional test, 1 bit adder functionality is verified by validating correct output behavior under normal intended input patterns.



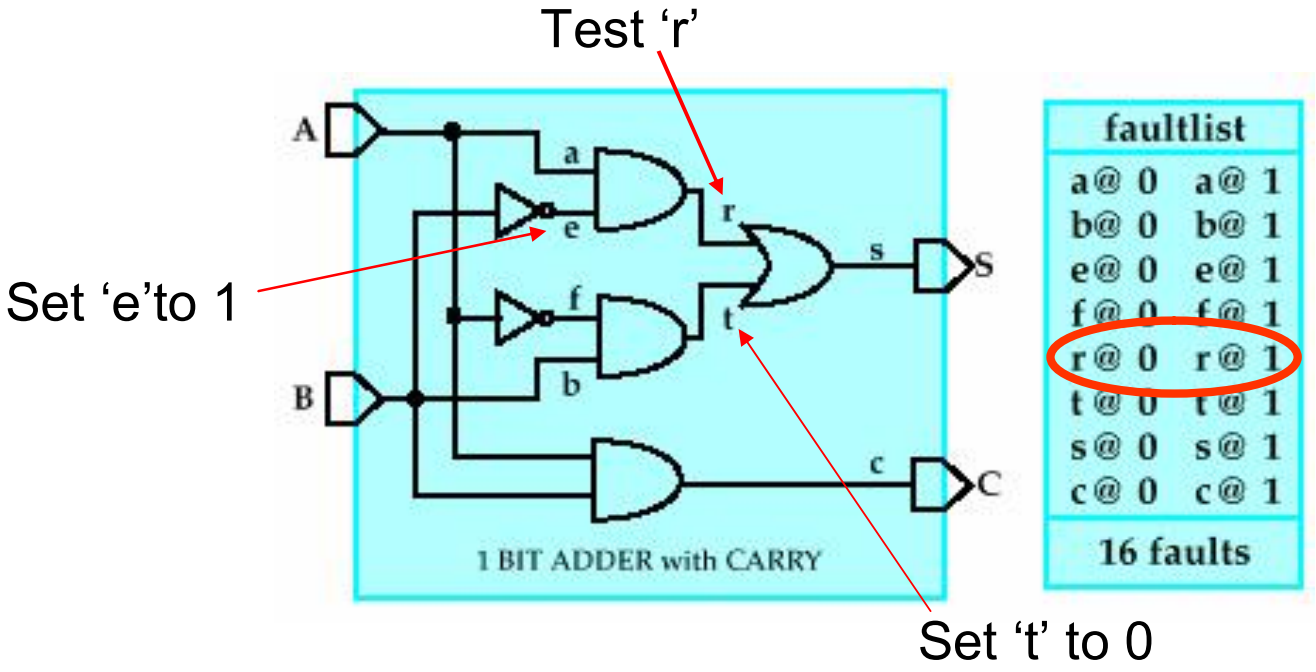
In structural test, fault list is created at the lowest level primitives (gate level) for the 1 bit adder, and test patterns are created to catch faults in the fault list

SCAN Definitions

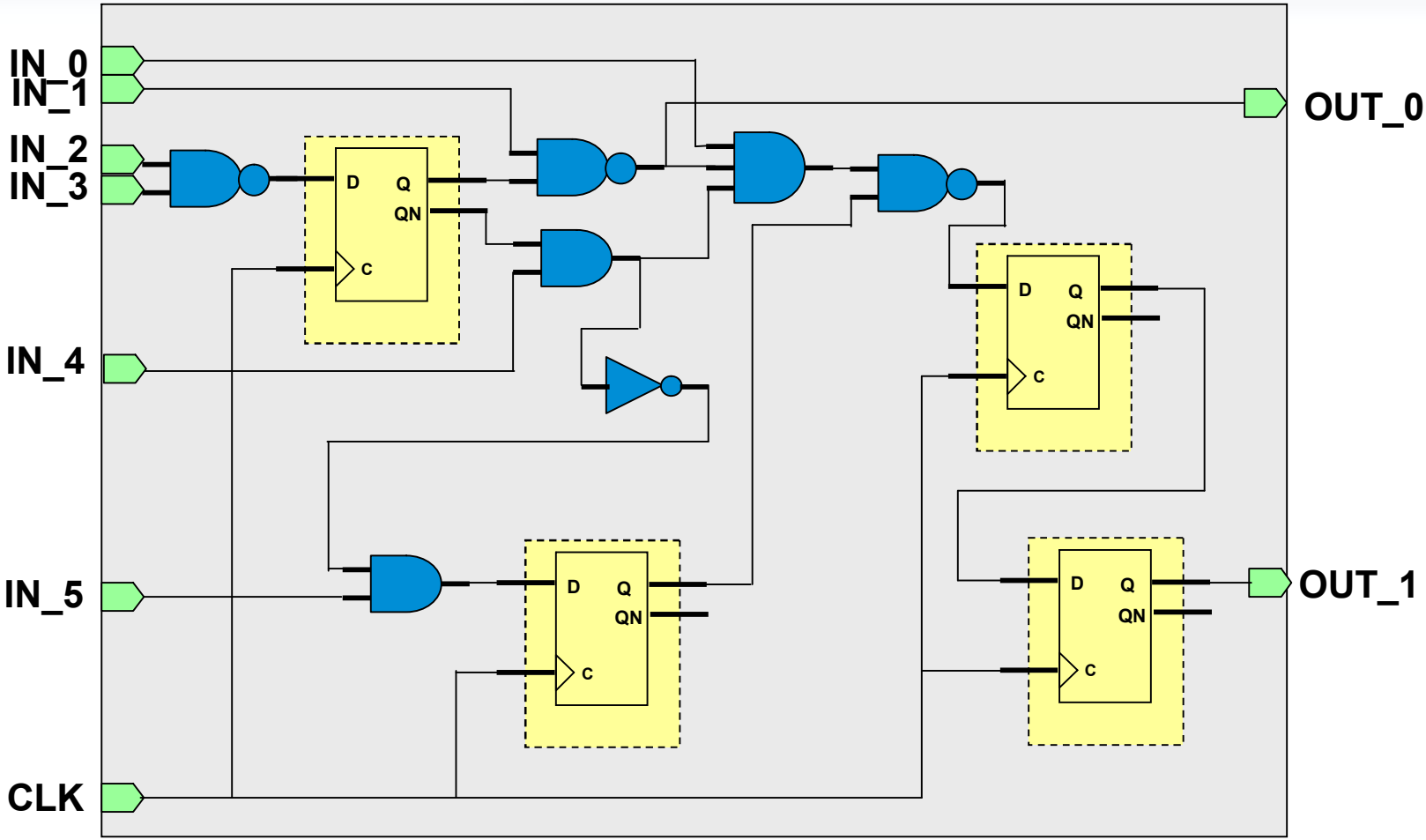
- DFT (Design for Test):
  - The act of adding logic or features to circuit to enhance the testability is often called DFT.
  - Required for Structural Test:
    - DFT Test and Structural Test is often used synonymously for this reason
  - Commonly Used DFT Techniques:
    - SCAN
    - BIST (Built In Self Test):
      - Logic Bist,
      - mBIST (memory BIST)
    - Current based test: Iddq

Controllability and Observability

- Controllability: Ability to control internal nodes to desired states
- Observability: Ability to observe states of internal nodes
- In this example, to test node ‘r’:
  - **Controllability** to set node ‘e’ to 1 is needed
  - **Observability** of node ‘r’ is achieved by setting node ‘t’ to 0

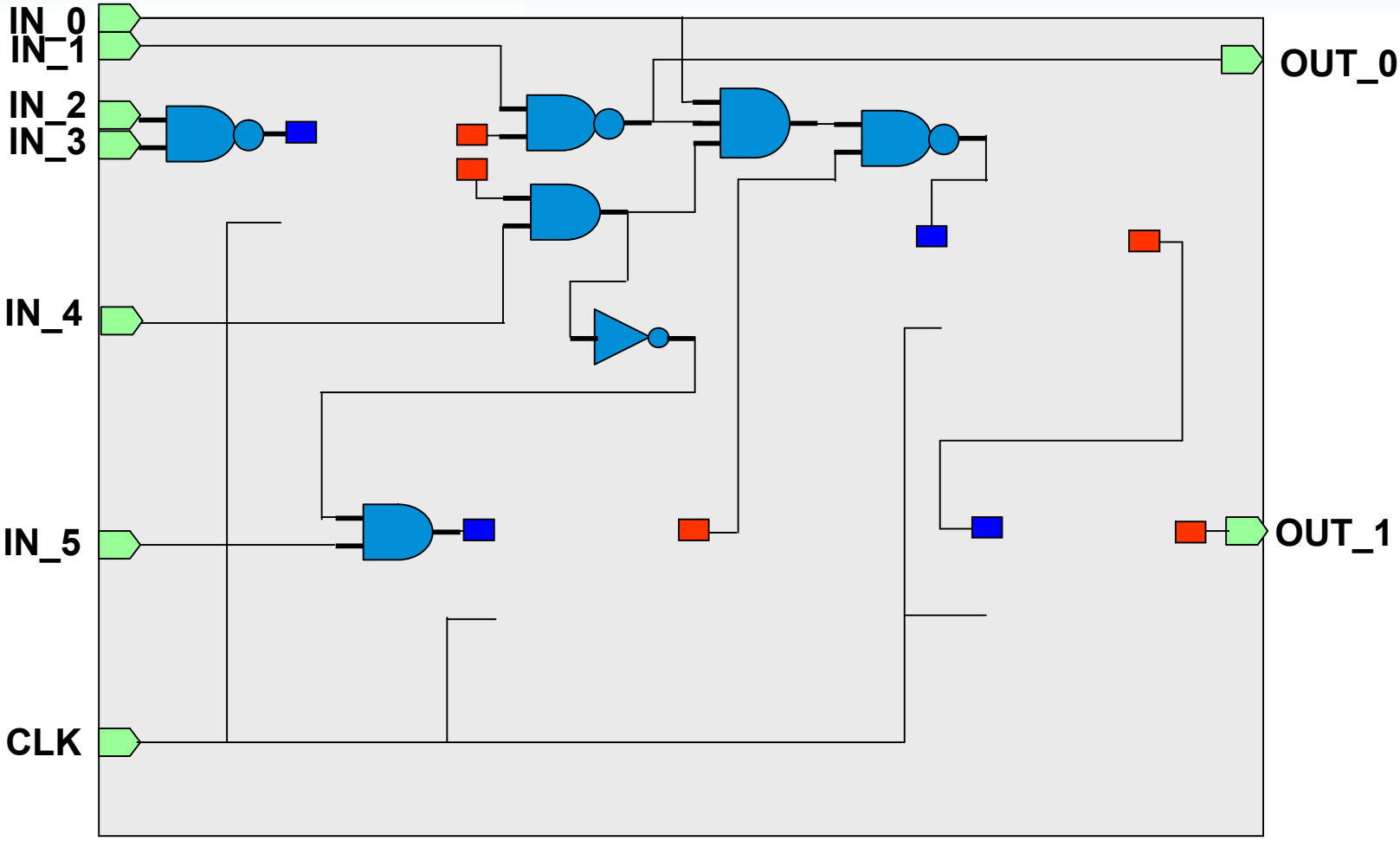


Simple Device: Functional View



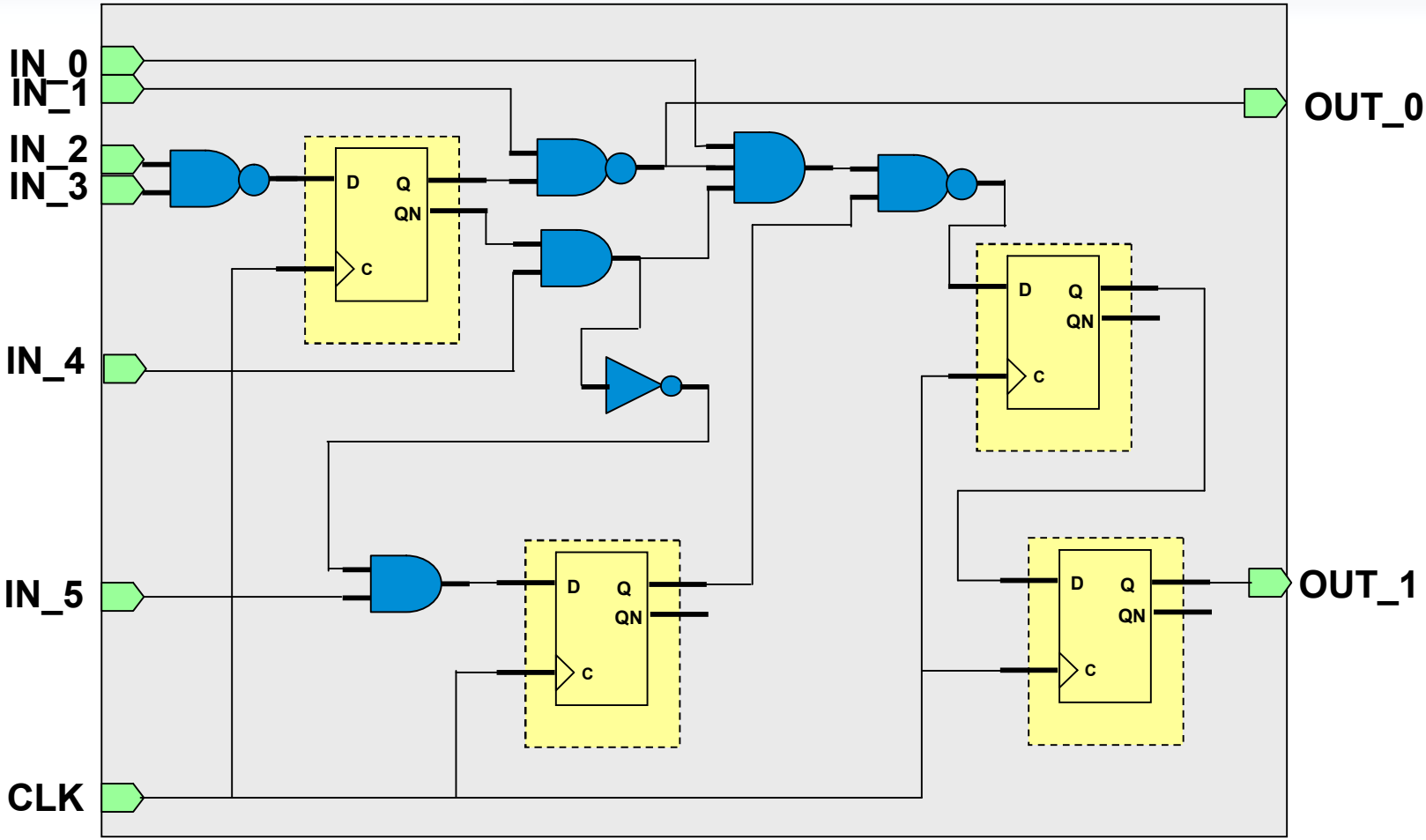
4 Stages of Pipeline (Sequential Depth of 4)
6 Inputs (Combinational Width of 6) & 2 Outputs
2^{4+6} = 1024 Vectors

Simple Device: Scan View



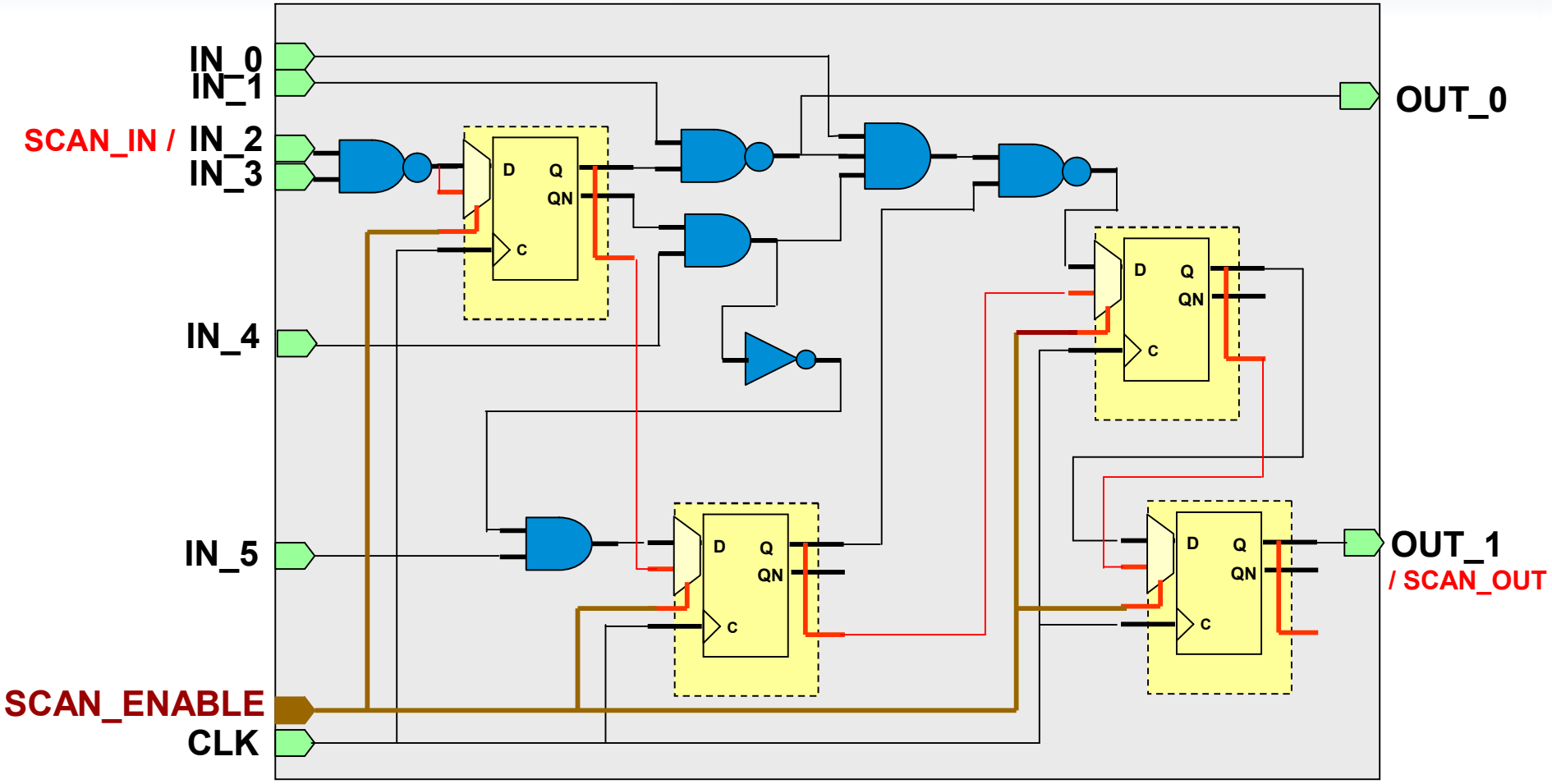
0 Stage Pipeline (No Sequential Elements)
6 Inputs, 2 Outputs, 5 Pseudo Inputs, 4 Pseudo Outputs
<<< 1024 test vectors

Mechanics of Scan



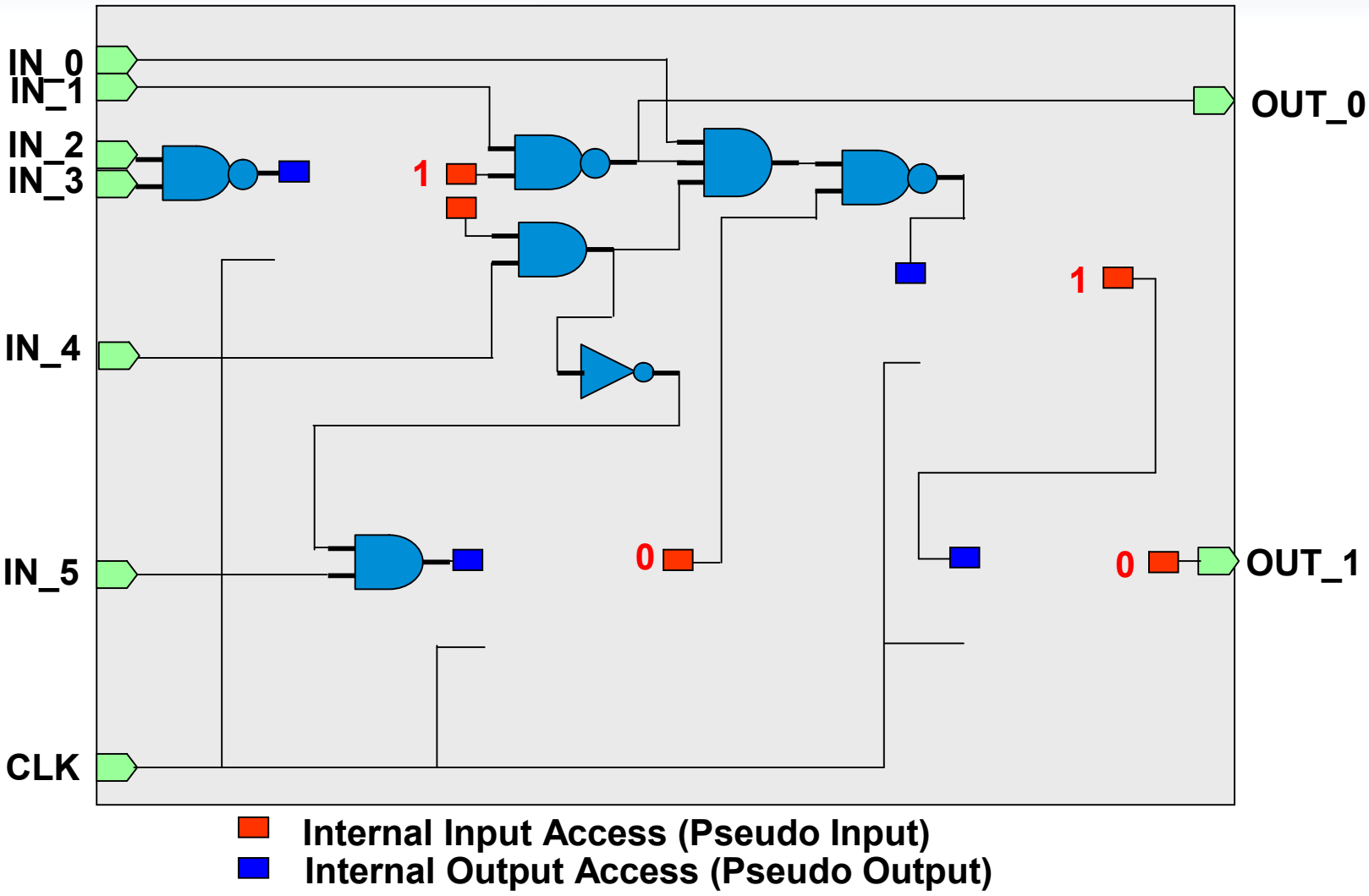
4 Stages of Pipeline (Sequential Depth of 4)
6 Inputs (Combinational Width of 6) & 2 Outputs
2^{4+6} = 1024 Vectors

Mechanics of Scan: Scan Insertion

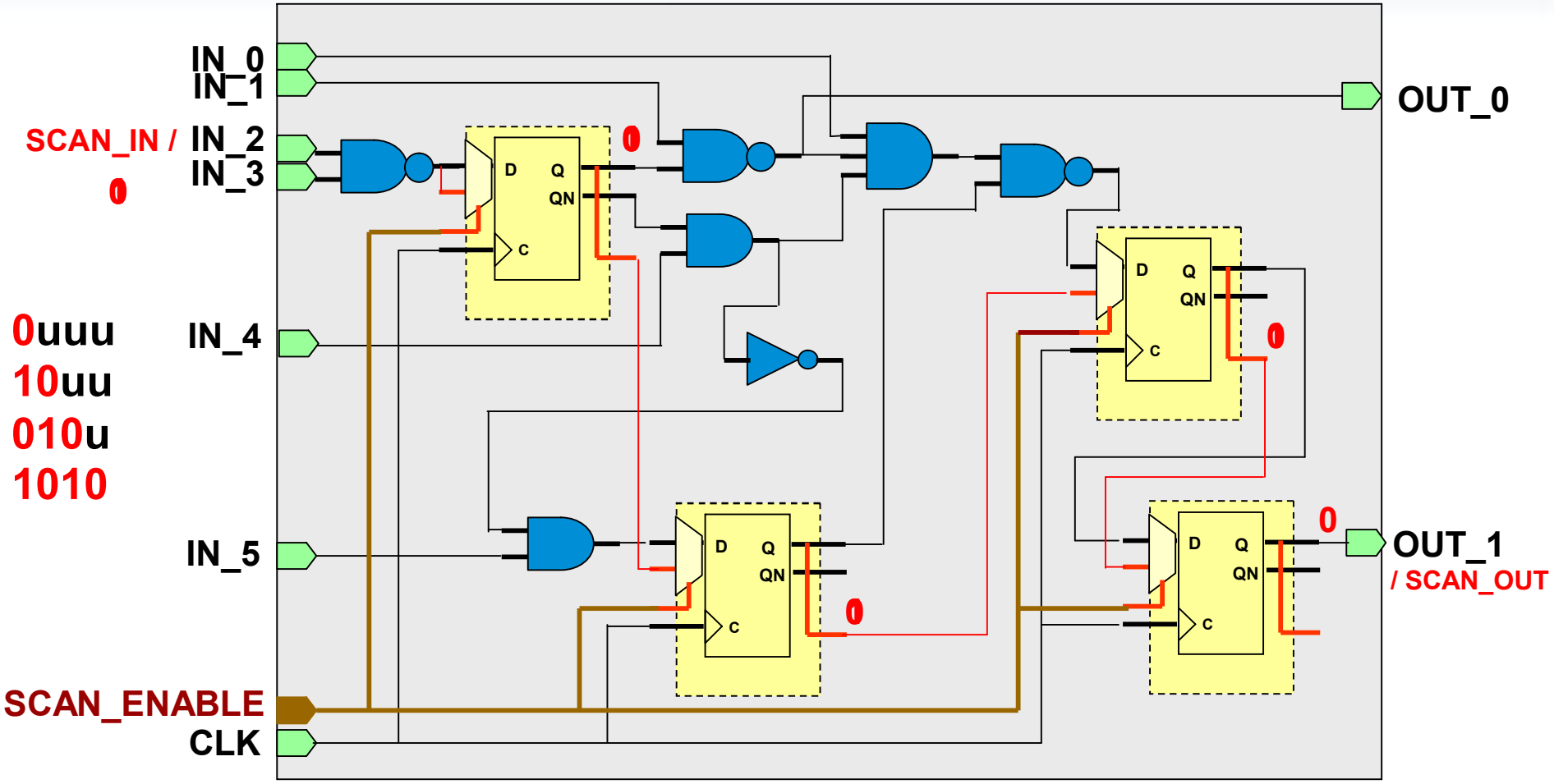


Replace Flip-Flops to Scan Flip-Flops
Hook Up Scan Chains
Hook Up Scan Enable

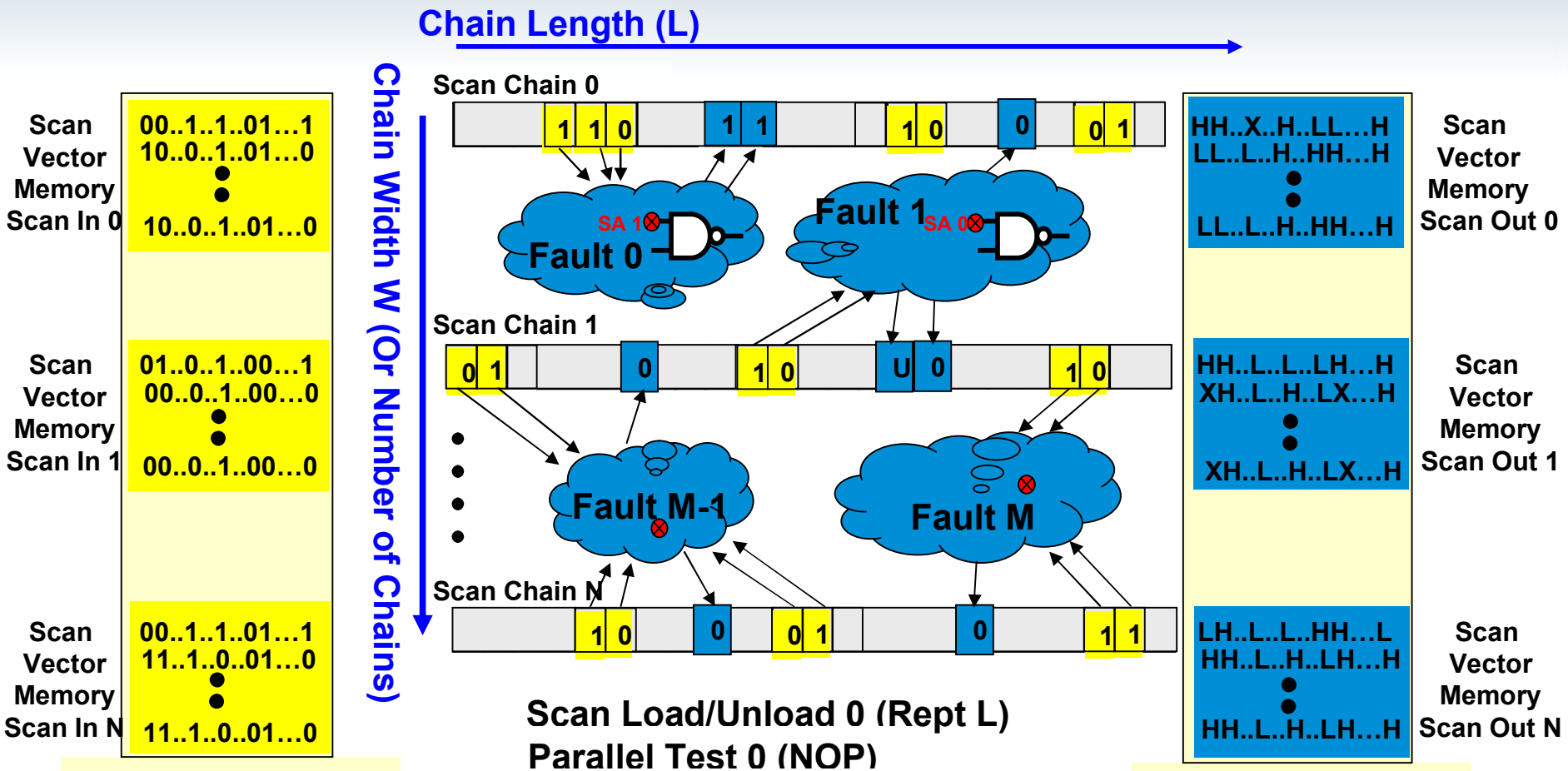
ATPG: Operation



Mechanics of Scan Shift Operation



Basics of Scan ATPG Operation



Scan Vector Memory For Scan Inputs

Scan Vector Memory For Scan Outputs

Scan Requires Large ATE Vector Memory!

Scan Load/Unload 0 (Rept L)
Parallel Test 0 (NOP)
Scan Load/Unload 1 (Rept L)
Parallel Test 1 (NOP)
○
○
Scan Load/Unload Z (Rept L)
Parallel Test Z (NOP)

Appendix B:
SN74BCT8245A Device Data Sheet