Overview

Design for testability(DFT) makes it possible to:

- Assure the detection of all faults in a circuit.
- Reduce the cost and time associated with test development.
- Reduce the execution time of performing test on fabricated chips.

We will focus on DFT techniques for digital logic, although it is relevant for memory and analog/mixed-signal components as well.

An example chip level DFT technique is called **Built-in self-test** (BIST) (used for digital logic and memory.)

At the system level, DFT includes **boundary scan** and **analog test bus**.

The DFT techniques discussed focus on improving testability of SAFs.

DFT for other fault models, e.g., delay faults, is described in the literature.

Ad-hoc DFT

Two forms of DFT: *ad-hoc* and *structured*.

Ad-hoc DFT relies on "good" design practices:

- Avoid *asynchronous* logic feedbacks.

  Feedback can result in oscillation.

  ATPG are designed to work on acyclic combinational logic.

- Make FFs initializable, i.e., provide clear and reset.

- Avoid gates with a large fan-in.

  Large fan-in makes the inputs difficult to observe and the output difficult to control.

- Provide test control for difficult to control signals.

  For example, signals produced by a long counter require many clk cycles to control.

  This increases the length of the test sequence.

Structured DFT

Testability measures can be used to identify circuit areas that are difficult to test.

Once identified, circuit is modified or *test points* are inserted.

This type of *ad-hoc* strategy is difficult to use in large circuits:

- Testability measures are approximations and don't always work.
- Good fault coverage is not **guaranteed** from ATPG even after circuit modifications and test point insertion is performed.

Structured DFT involves adding extra logic and signals dedicated for test according to some procedure.

The circuit has two modes, normal and test mode.

The most commonly used structured methods are Scan and BIST.

Scan

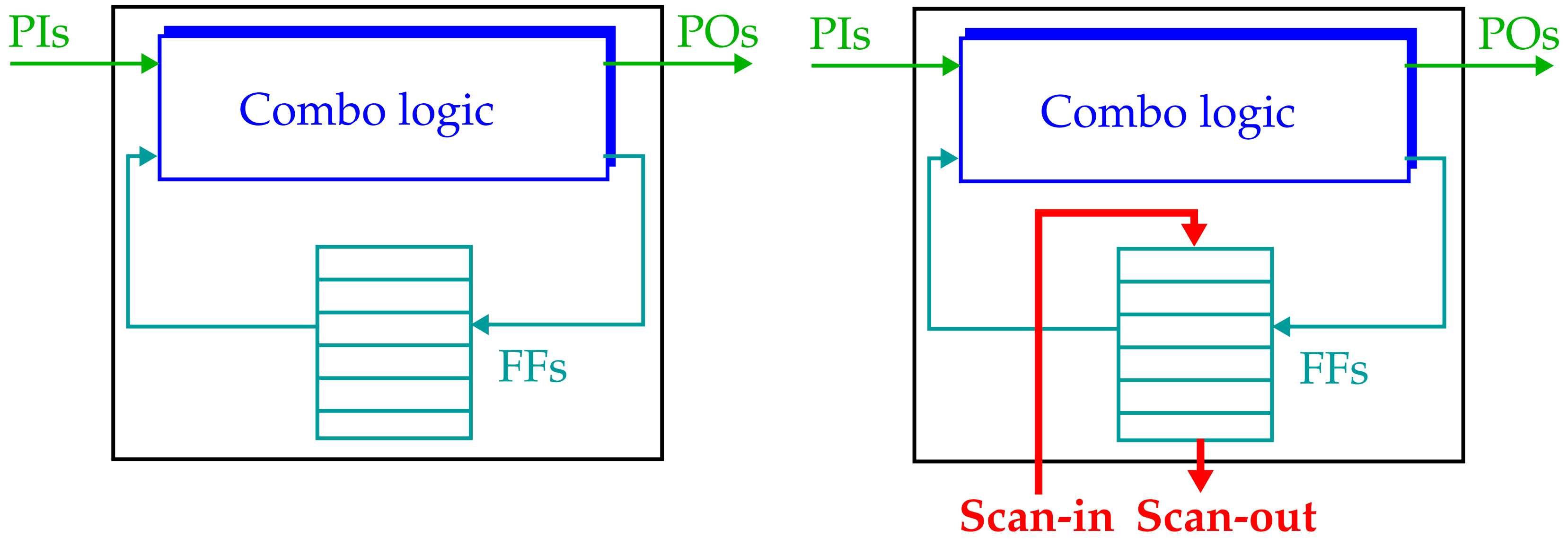
Scan proposed in ’73 by Williams and Angell.

Main idea is to obtain control and observability for FFs.

It reduces *sequential* TPG to *combinational* TPG.

With Scan, a synchronous sequential circuit works in two modes.

Normal mode and test mode:

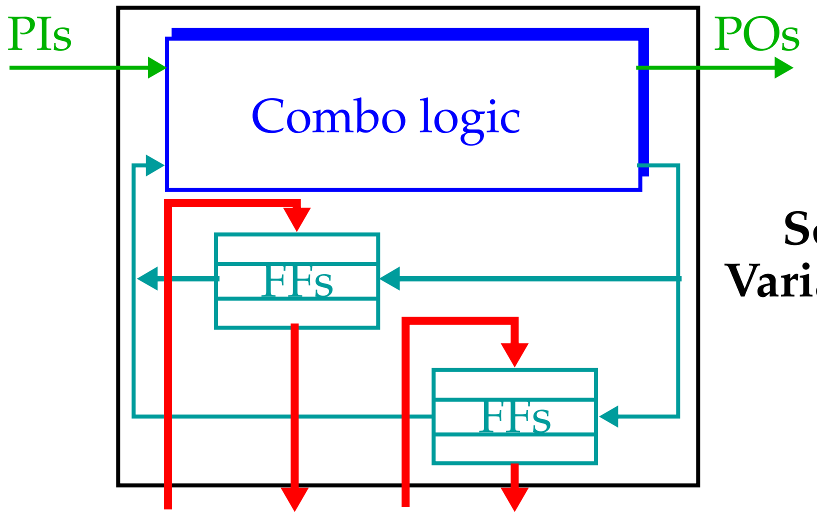


In **test mode**, all FFs are configured as a shift register, with Scan-in and Scan-out routed to a (possibly dedicated) PI and PO.

Scan

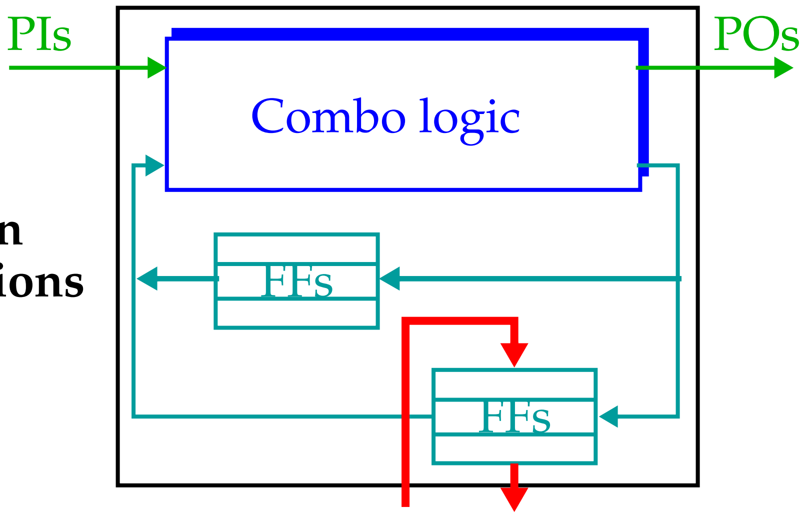
Once initialized, **normal mode** is used to apply a pattern to the PIs, and the results are latched in the FFs.

The circuit is put in test mode again and the results scanned out.



Multiple Scan chains

Scan Variations

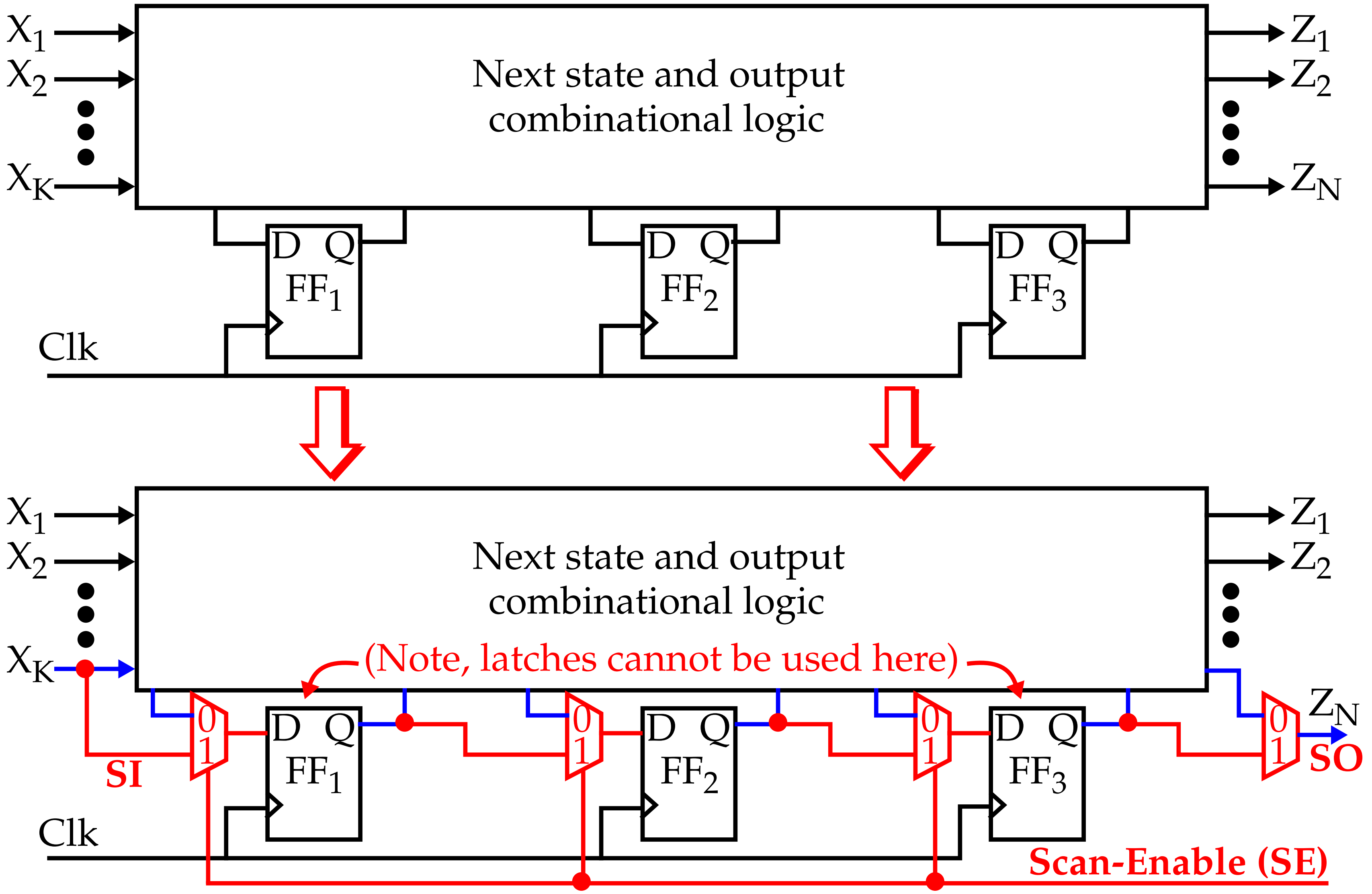


Partial Scan

Note that scan is usually inserted *after* the circuit is verified to be functionally correct.

Scan-Path Design

Any sequential circuit may be modeled as:



Scan Design Rules

A designer needs to observe four rules during functional design:

- Only *D-type master-slave* FFs should be used.

  No *JK*, *toggle* FFs or other forms of asynchronous logic.

- At least on PI must be available for test.

  As shown in previous circuit, the Scan-in and Scan-out pins can be multiplexed (only one additional MUX is needed at Scan-out).

  Therefore, the only required *extra* pin is Scan-Enable, SE (or Test Control, TC).

- All FFs must be controlled from PIs.

  Simple circuit transformations can be used to change FFs whose **Clk** is "gated" by an internal logic signal.

- Clocks must not feed data inputs of the FFs.

  A *race condition* can result in **normal** mode otherwise.

  This is generally considered good design practice anyway.

Storage Cells for Scan Designs

Common characteristics of all designs:

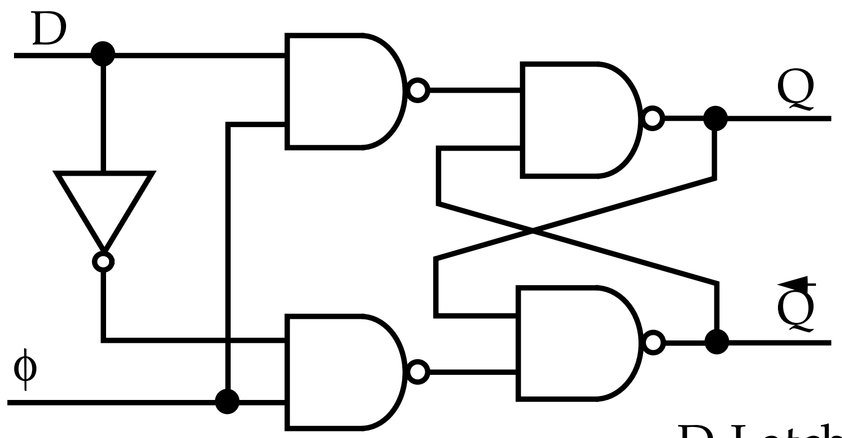
- A *normal input* and a *scan input*.

  The appropriate input can be selected using a multiplexer or by a two-clock system.

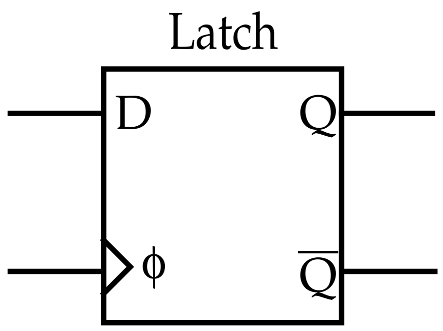
- A *storage cell*.

  The cell can be implemented using an edge-triggered FF, a master-slave FF or level-sensitive latches controlled by clocks having >=2 phases.

We'll focus only on D-FFs of the **master-slave** variety.

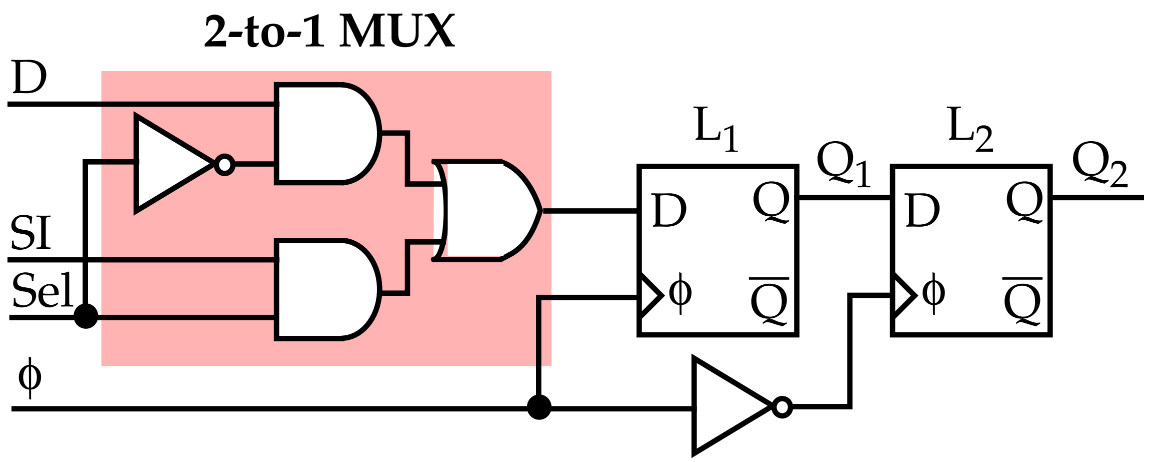


D-Latch

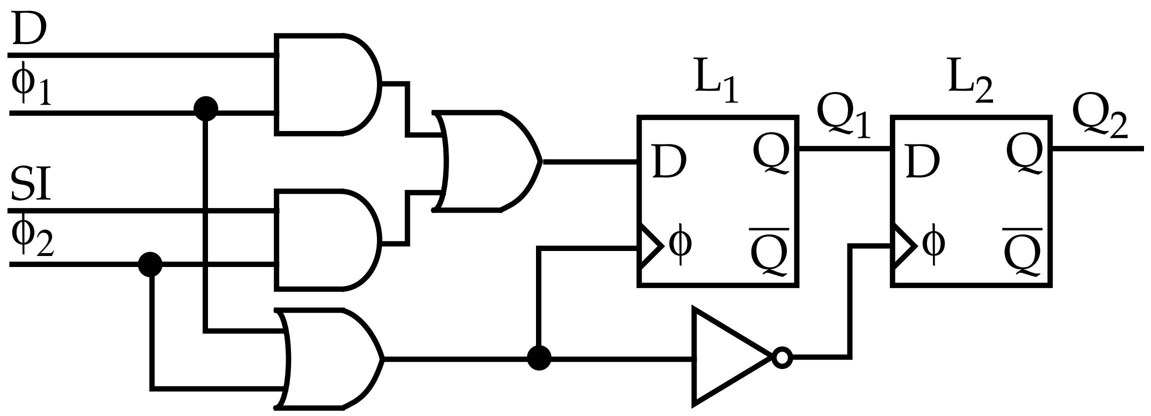


Storage Cells for Scan Designs

An implementation using two-port master-slave FF with a **MUX**.

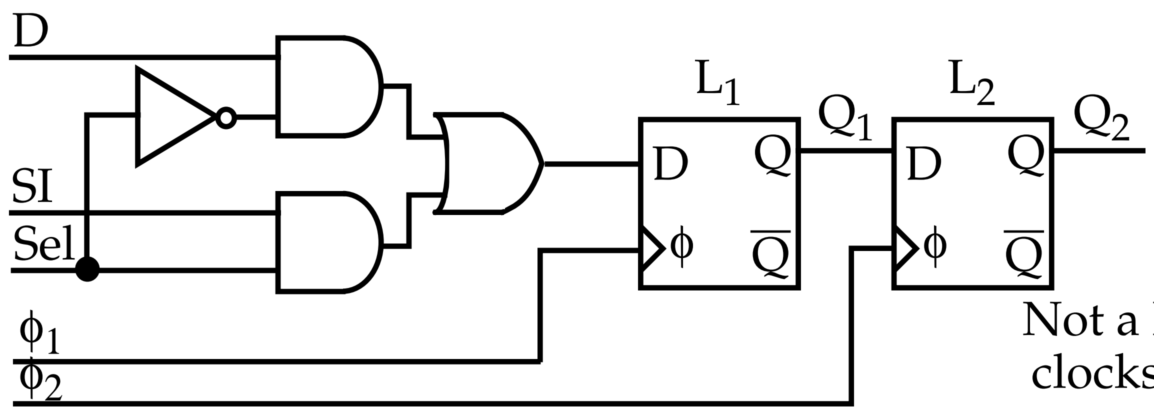


A **two-port** clocked FF implementation.



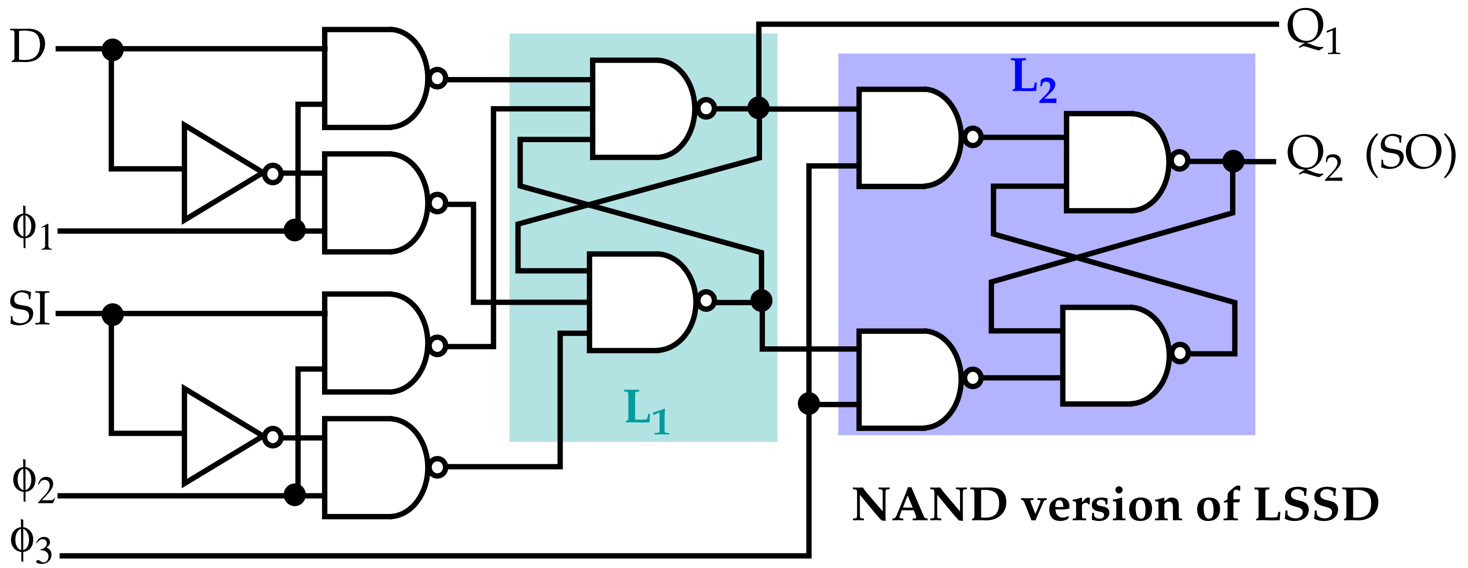
Storage Cells for Scan Designs

To ensure *race-free* operation, use a 2-phase nonoverlapping clk.



Not a FF, since two clocks are used.

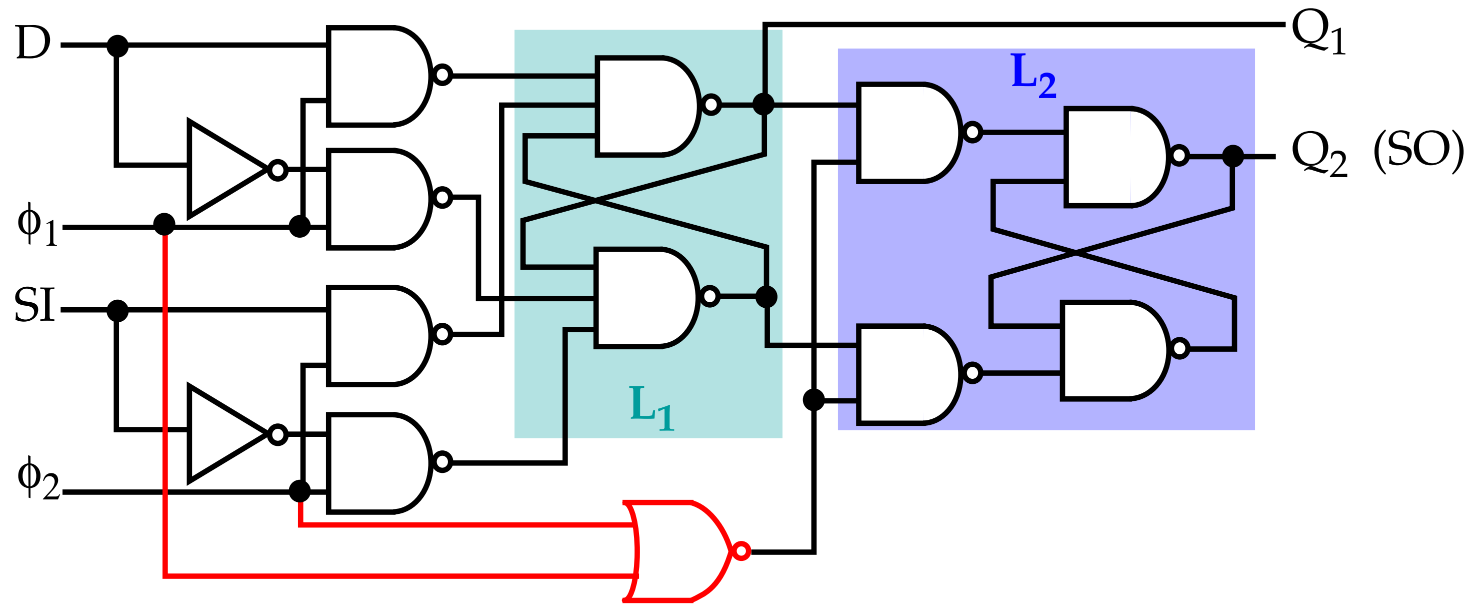
In order to avoid performance degradation introduced by the MUX.



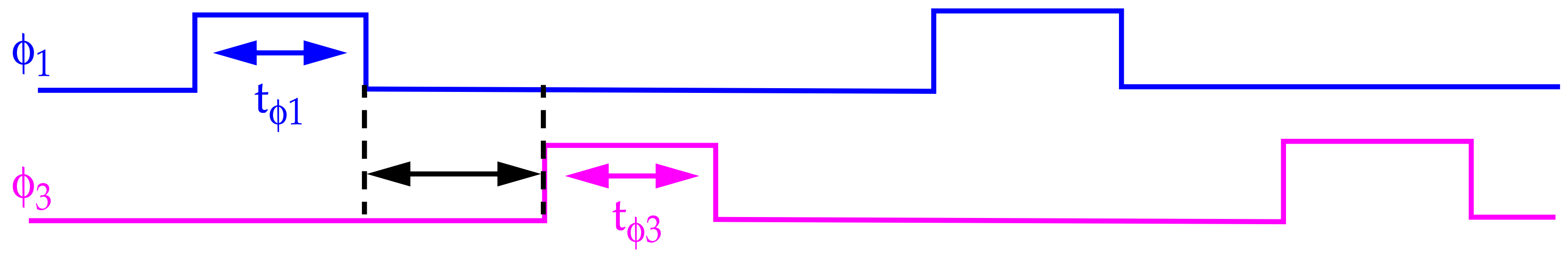
NAND version of LSSD

Storage Cells for Scan Designs

In LSSD, clocks ϕ_1 and ϕ_2 can be NORed together to drive L_2, replacing ϕ_3.



Note that in the 3-clock scheme, in order to prevent hazards, ϕ_1 and ϕ_3 and well as ϕ_2 and ϕ_3 MUST be non-overlapping.



Tests for Scan Circuits

Two phases:

- *Shift test*

  Set *TC*= 0, and shift toggle sequence 00110011... using Clk.

  The length is $n_{sff} + 4$, where n_{sff} are the number of scan flops.

  This sequence produces all 4 transitions, 0->0, 0->1, 1->1 and 1->0, catches all/most SA faults.

The *Shift test* can be used in either single-clock or two-clock designs.

A *Flush test* is also possible in two-clock designs:

  ϕ_1 (Master Clk) is held low while ϕ_2 and ϕ_3 are held high.

  This creates a continuous path between SI and SO for application of 0 and 1.

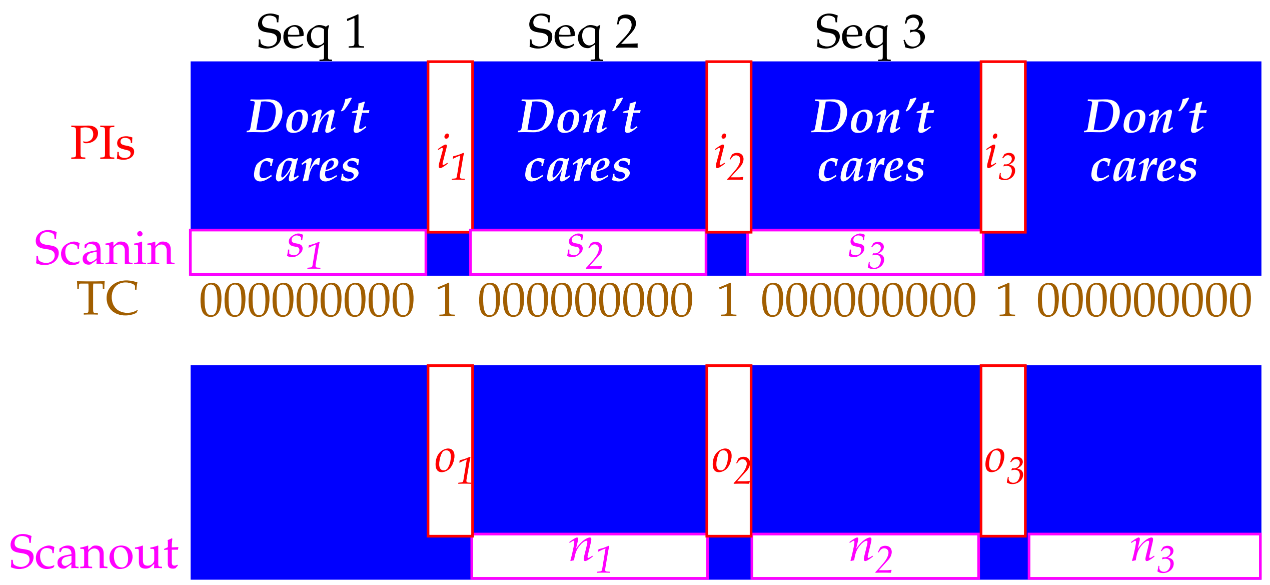
- *Combinational logic test*

  This phase allows the combination logic circuit to be tested for SA faults.

  An ATPG algorithm is used where outputs of Scan FFs are treated as *pseudo-PIs* (completely controllable) and inputs are treated as *pseudo-POs*.

Tests for Scan Circuits

Each vector contains two parts: i_x and s_x represent PIs and *pseudo-PIs* (state variables), o_x and n_x represent POs and *pseudo-POs* (next state variables).



9 scan vectors
1 normal vector

On 10th vector TC is set to 1.

39 Clks are needed for these 3 vectors.

The vectors are converted as shown above.

Don't care bits can be filled randomly or with a specific sequence.

Faults at POs under 1st vector are detected after 10th Clk.

However, faults captured in FFs for this vector are detected on 11th through 19th Clk, during scanin of 2nd vector.

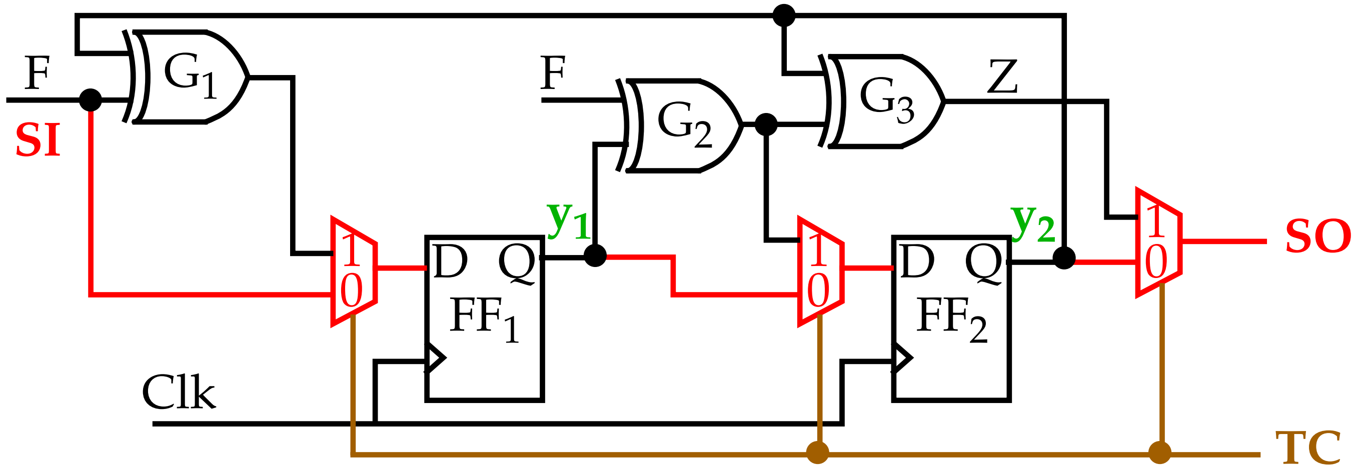
Tests for Scan Circuits

The general formula for the length of the test (which includes *Shift test*) is:

$$\text{Scan test length} = n_{sff} + 4 + (n_{sff} + 1)n_{comb} + n_{sff}$$
$$= (n_{comb} + 2)n_{sff} + n_{comb} + 4 \ \text{clock periods}$$

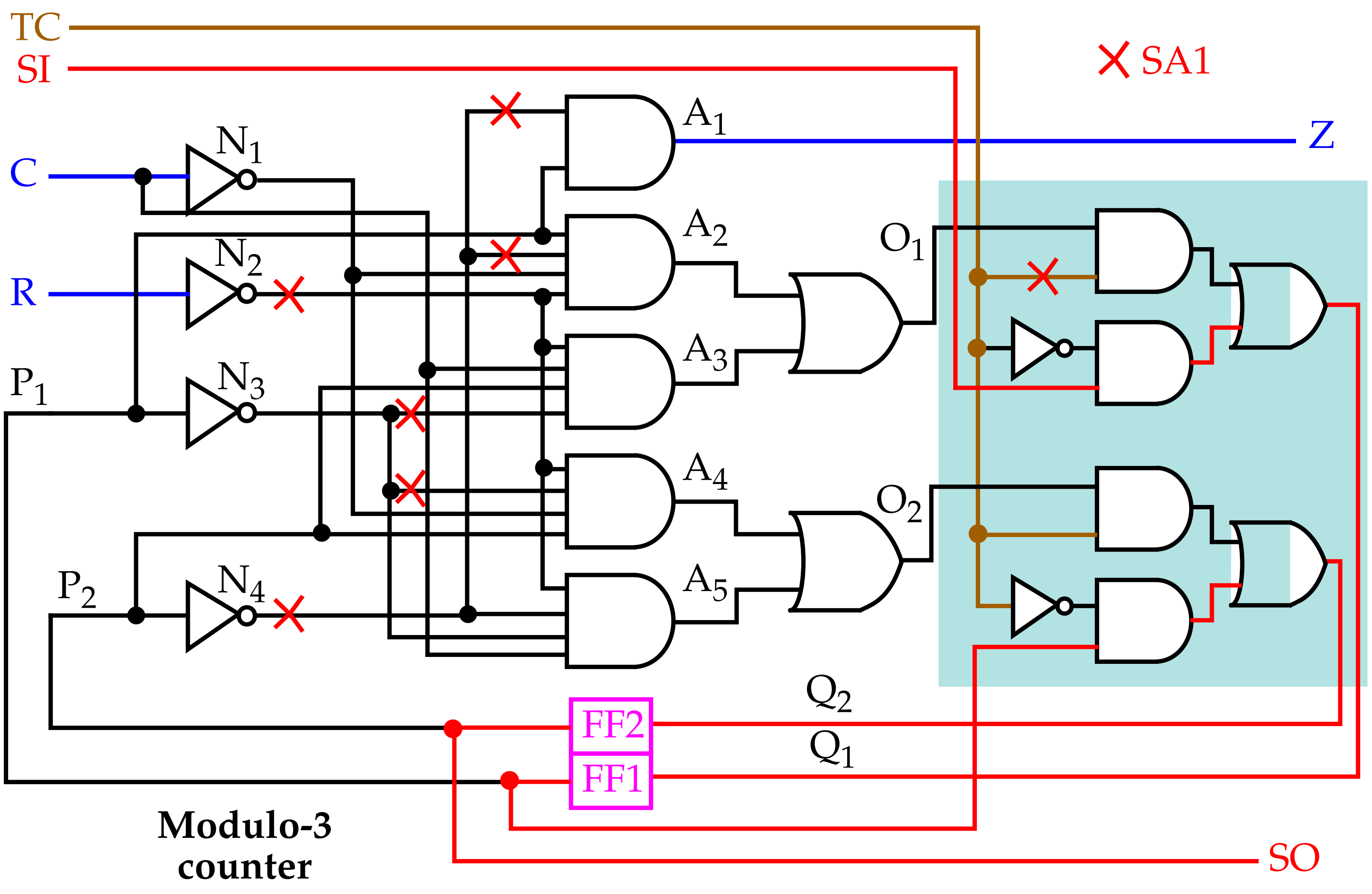
For a circuit with 2,000 FFs and 500 vectors, 1,004,504 Clks needed.

Also, the *pseudo-PIs/POs* are **dispersed**, improving observability/controllability, effectively partitioning the CUT into smaller circuits.



Two partitions: G_1 (*pseudo-PI* y_2) and G_2/G_3 (*pseudo-PIs* y_1/ y_2).

Example Application



With R=0, counter is reset to 00. With R=0, C=1, sequence is 00->01->10->00.

Example Application

The output *Z* becomes 1 only in state 10, otherwise it's zero.
With *R*=0, *C*=0, the couter retains its state.

The combinational logic is *combinationally irredundant*.
However, without scan, the sequential circuit has 6 **untestable** SA1 faults.

A sequential ATPG algorithm generated 35 tests to detect 36 of the 42 faults in the non-scan version.

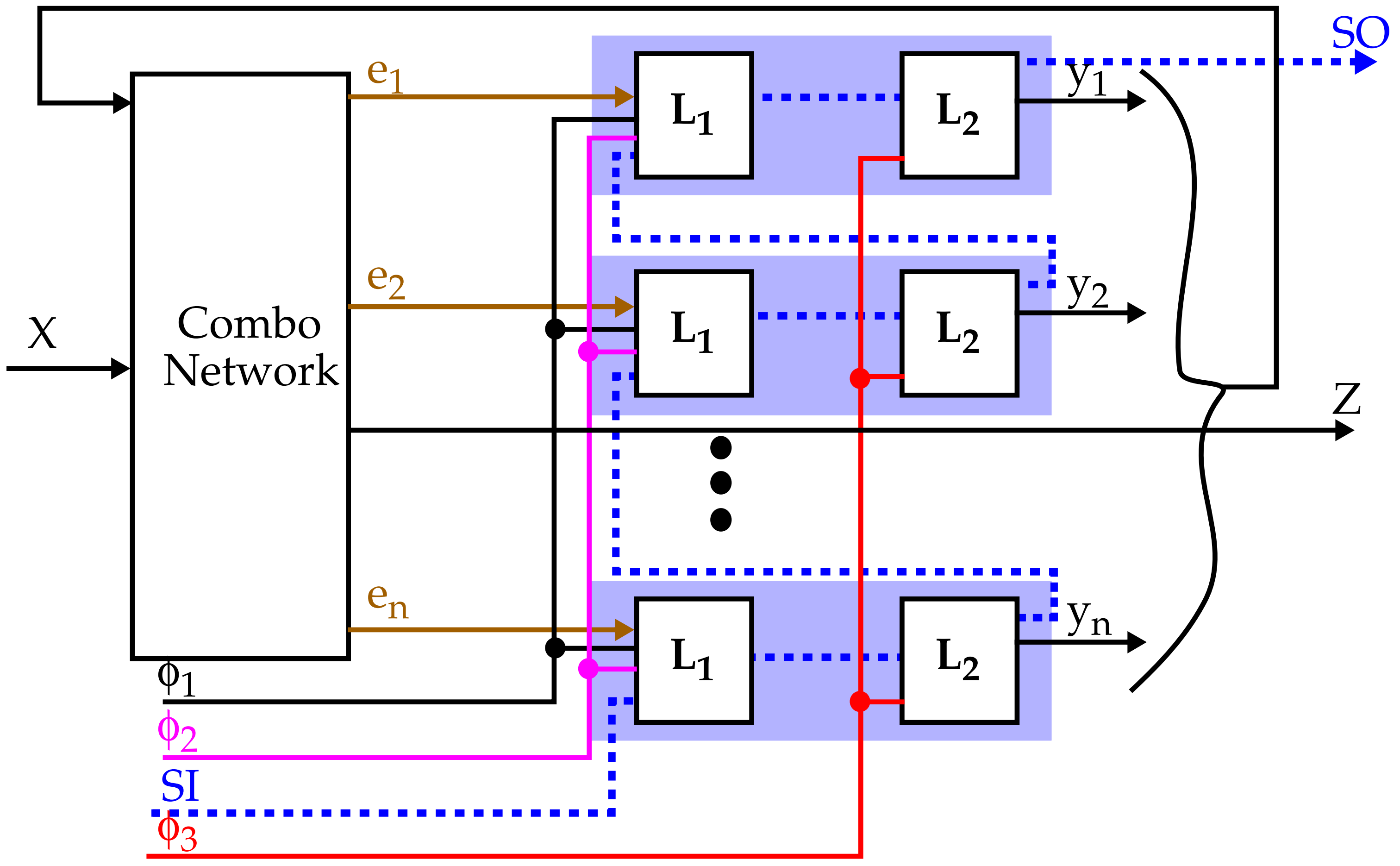
Another combinational ATPG algorithm generated 12 vectors for the combinational part with *C*, *R*, *P1* and *P2* as inputs and *Z*, *Q1* and *Q2* as outputs.

Once converted to scan sequences with a 6 vector *Shift* test, 44 vectors result.

Fault simulation indicates that all faults are detected, including the 6 untestable SA1 faults and the MUX fault.

Scan Architectures

General structure of an **LSSD double-latch** design.

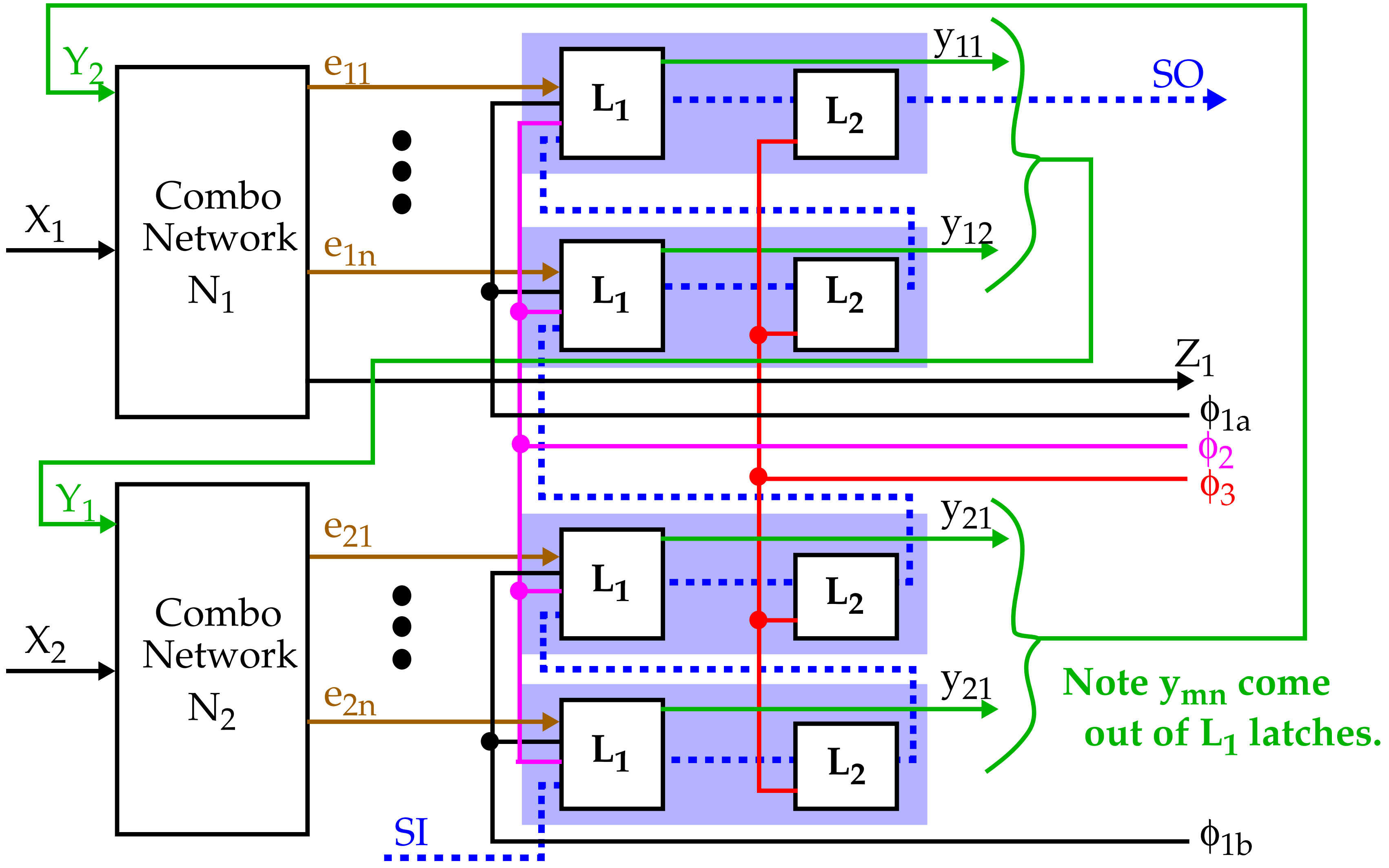


The outputs, y_i, come from the output of the L_2 latches.

Normal mode, ϕ_1 and ϕ_3 are used, test mode, ϕ_2 and ϕ_3 are used.

Scan Architectures

In pipelined systems, combinational blocks are separated by only **1 latch**.



Note y_{mn} come out of L_1 latches.

Scan Architectures

During normal mode, ϕ_{1a} and ϕ_{1b} are nonoverlapping clocks, and clocks ϕ_2 and ϕ_3 are not used.

Note the L_2s are **not** used during normal operation and represents overhead.

For the double latch design, operation proceeds as follows:

- Test the latches:

  Set ϕ_2=ϕ_3=1.
- Apply 0 and 1 alternatively at *SI*
- Clock ϕ_2, then ϕ_3, n times.
- Initialize:

  Shift in the initial values into the FFs.
- Repeat for all patterns:

  Apply a pattern to the PIs.

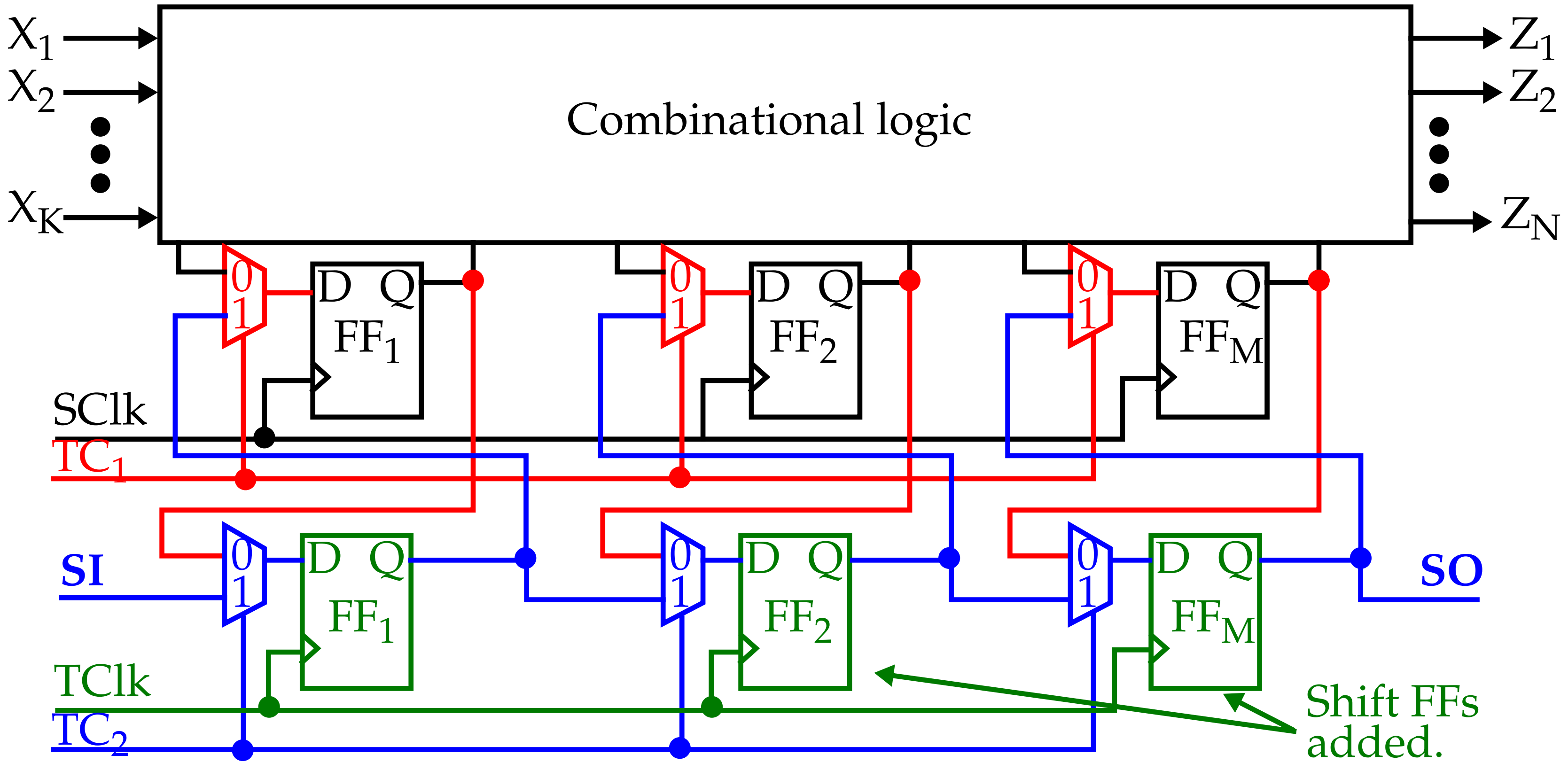
  Clock ϕ_1 and observe results at POs.

  Shift out the response and initialize for next pattern.

  Clock ϕ_2, then ϕ_3, n times.

Scan Architectures

Scan-Set Architecture offers **on-line** test capability.



TClk and *SI* used to initialize FFs with TC_1/TC_2=1, then *SClk* used to load data into upper FFs in parallel.

Test pattern applied to PIs, TC_1/TC_2=0, then *SClk* transfers data to lower FFs, *TClk* used to scan data out.