ECSE-323
Digital System Design

Sequential Testing Lecture #1

Why test?

It's all about money!

$$$

Economics of testing

We should test whenever:

Cost of testing < Cost of not testing

or, in terms of the Benefit-Cost ratio:

Benefits of testing/Cost of testing > 1

Some *costs* of testing:

- *fixed costs*
  - test system hardware
  - factory space
  - test vector generation (done once)
- *variable costs*
  - operator salaries
  - hardware maintenance
  - electricity costs

Costs increase with circuit speed and complexity

Some *costs* of *not* testing:

- system maintenance
- system downtime
- lawsuits
- longer time-to-market
- loss of market share

Costs of not-testing increase rapidly as the point of failure moves to higher levels in the system hierarchy

(e.g. chip - pcboard - computer - local power controller - national power grid)

The lowest cost of ownership is achieved by finding defective units before they are shipped from the vendor.

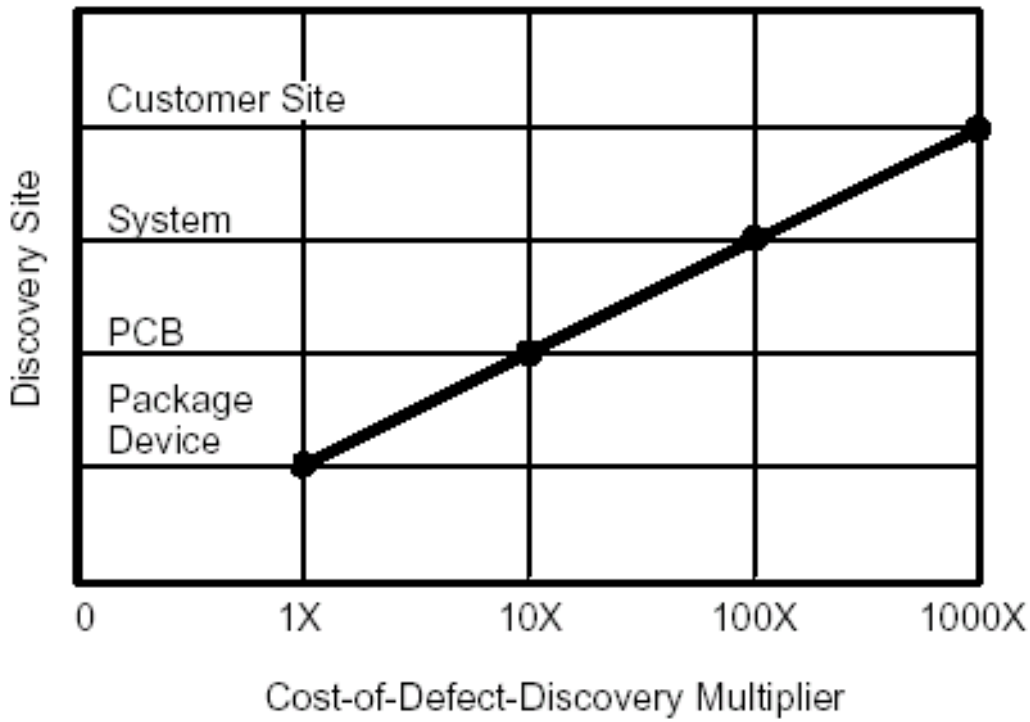


Figure 4-6. Cost of Ownership

(From the Texas Instruments IEEE std. 1149.1 (JTAG) Testability Primer)

ADVANTEST Model T6682 ATE

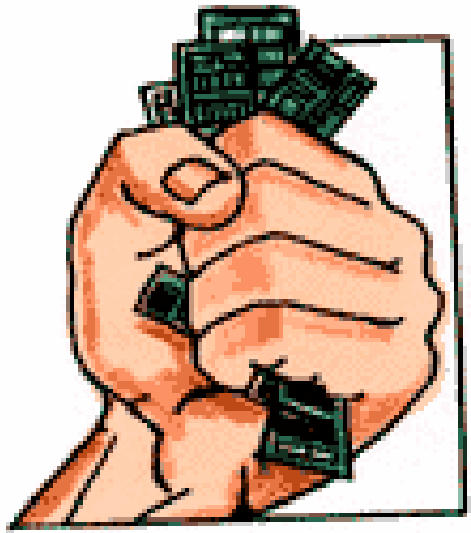
Some typical testing costs (as of 2000)

- 0.5-1.0GHz, 1024 digital pins: ATE purchase price
  - **= $1.2M + 1,024 pins x $3,000/pin = $4.272M**
- Running cost (five-year linear depreciation)
  - **= Depreciation + Maintenance + Operation**
  - **= $0.854M + $0.085M + $0.5M**
  - **= $1.439M/year**
- Test cost (24 hour ATE operation)
  - **= $1.439M/(365 x 24 x 3,600)**
  - **= 4.5 cents/second**

Types of Testing Procedures

- ***Verification testing*, *characterization testing*, or *design debug***
  - **Verifies correctness of design and of test procedure – usually requires correction to design**
- ***Manufacturing testing***
  - **Factory testing of all manufactured chips for parametric faults and for random defects**
- ***Acceptance testing* (*incoming inspection*)**
  - **User (customer) tests purchased parts to ensure quality**

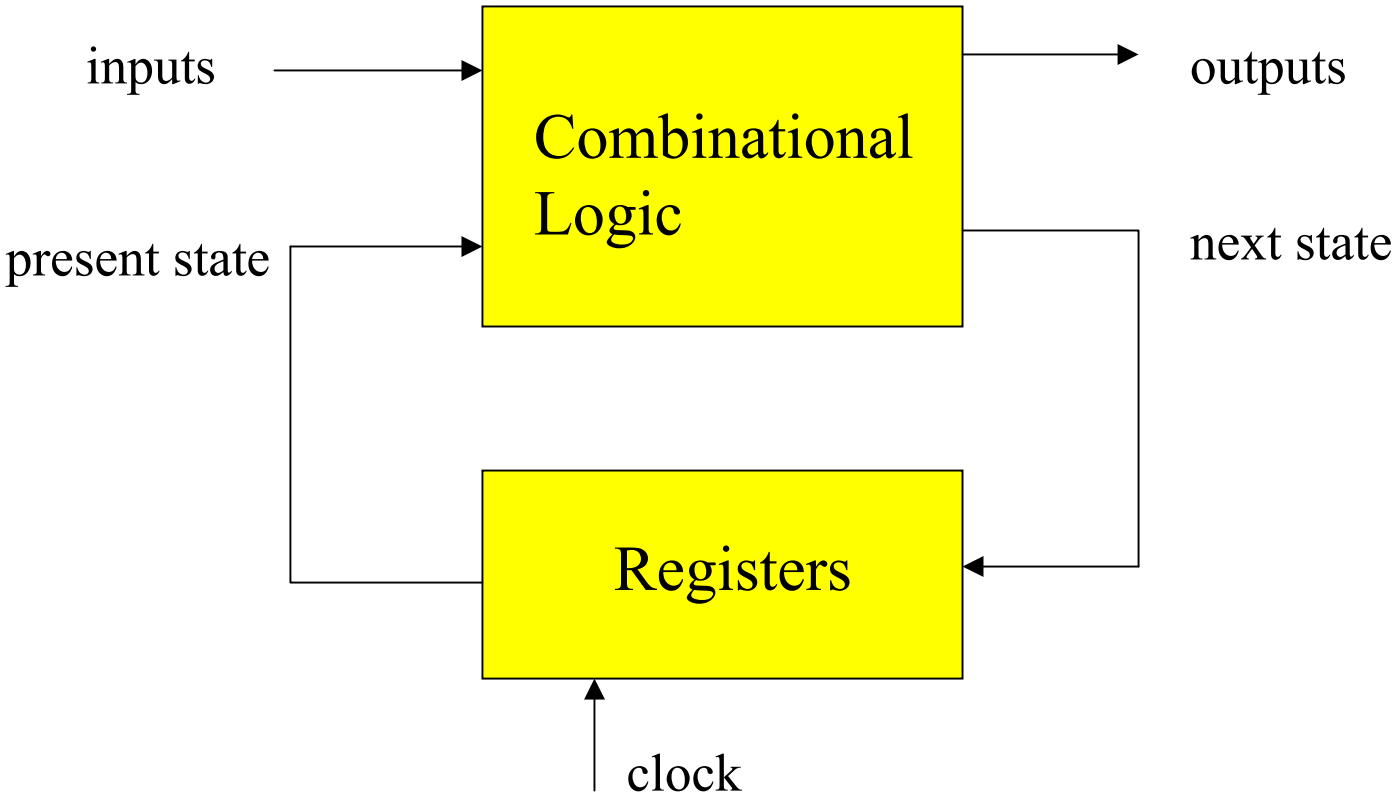
In this course we will focus on *manufacturing* and *acceptance testing*, i.e. testing of chips/systems after the design has been debugged.



In previous lectures, you learned about testing of ***combinational*** circuits.

Now we will look at the testing of ***sequential*** circuits.

Let us use the Finite State Machine structure as the ***canonical*** form of a sequential circuit.



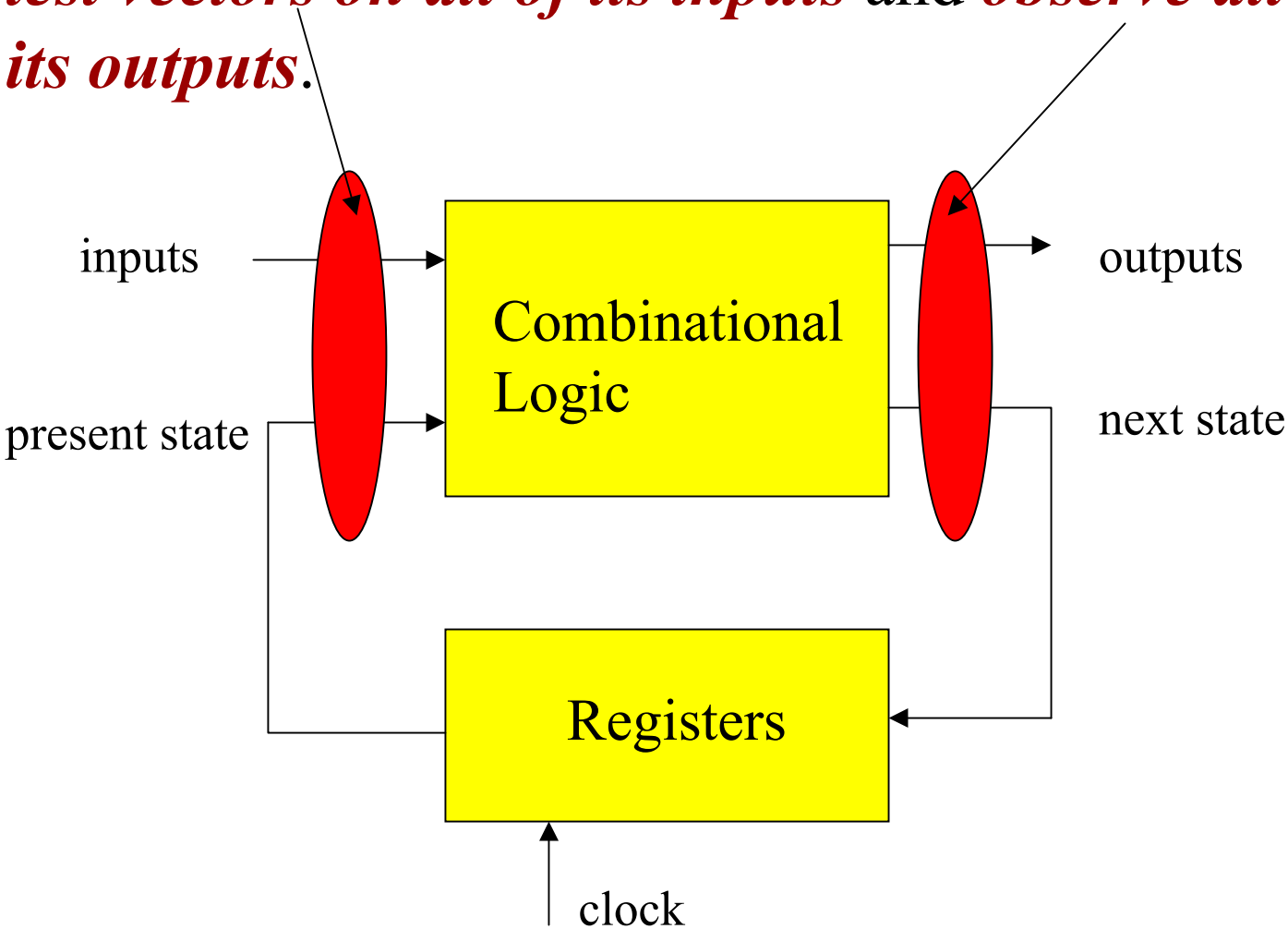
Testing of sequential circuits is *much more* ***difficult*** than testing combinational circuits

Why?

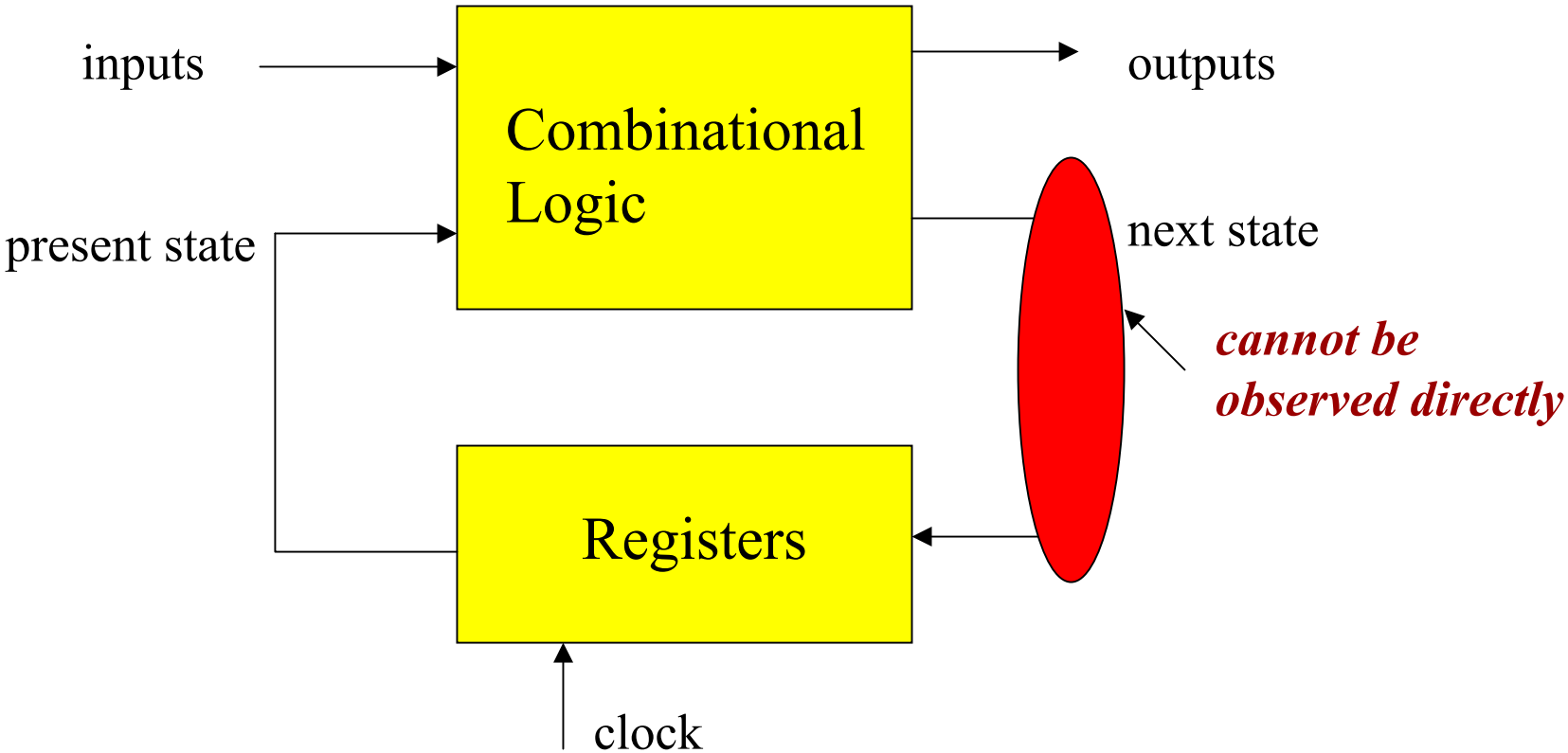
The difficulties lie in the twin problems of

- ***observability***
- ***controllability***

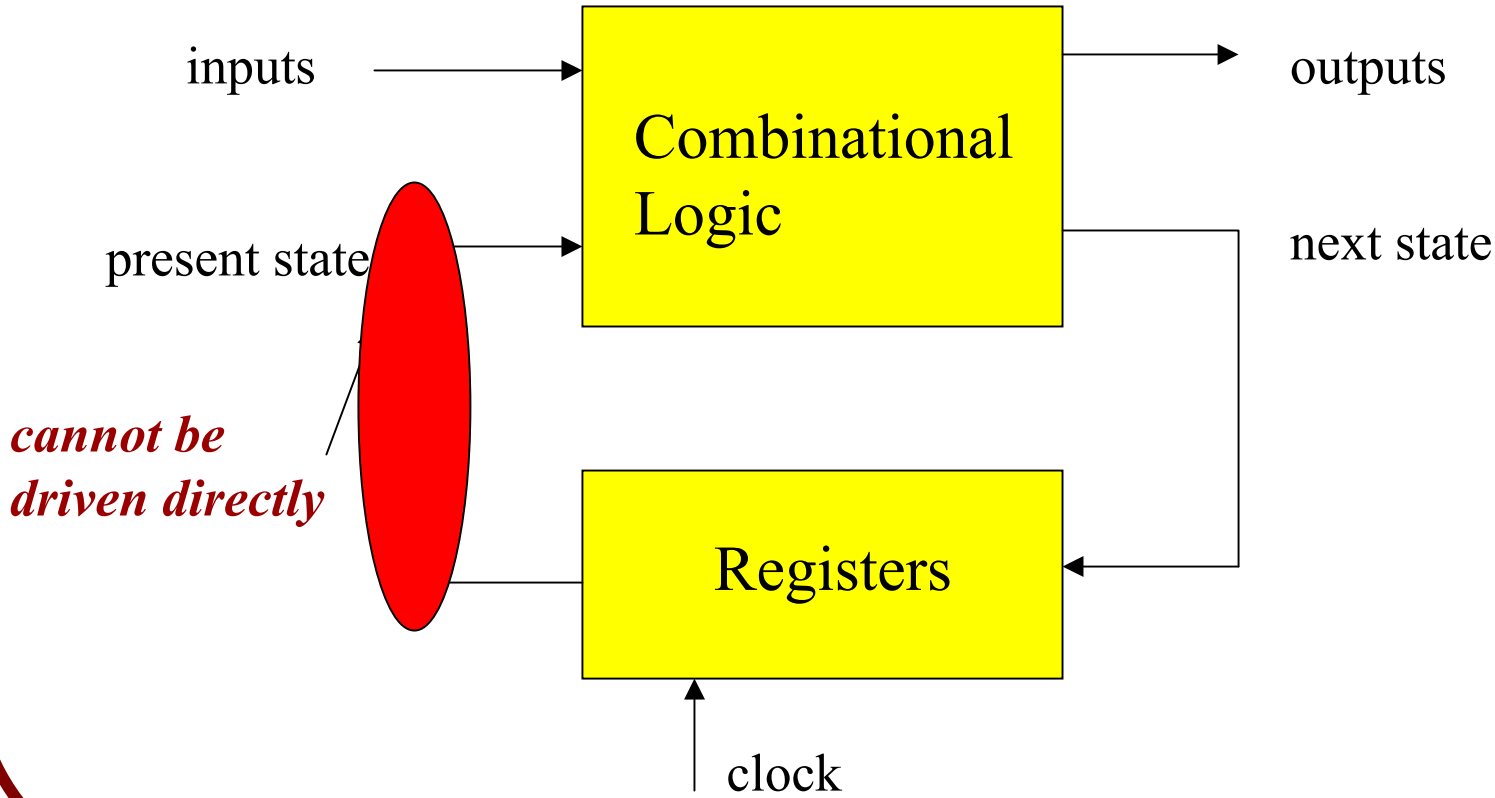
To test the combinational logic we must ***present test vectors on all of its inputs*** and ***observe all of its outputs***.



The ***observability*** problem is that some internal circuit nodes cannot be observed directly at the outputs.



The ***controllability*** problem is that some internal circuit nodes cannot be directly driven by external inputs.



One approach to sequential testing tries to find a ***sequence of inputs*** that will ***drive the FSM*** to the states associated with the desired test vectors.

But, the sequences:

- ***may be very difficult to generate***
- ***may not exist***
- ***may be quite long***

Instead of making testing difficult, why not make it *easier*?

One way to do this is to approach the system design process right from the start with the aim of making testing easier.

- ***Design for test*** **(DFT)** refers to those *design* techniques that make test generation and test application cost-effective.

- Some DFT methods for digital circuits:
  - ***Scan-Path***
  - ***Built-in self-test*** **(BIST)**
  - ***Boundary scan*** **(JTAG)**

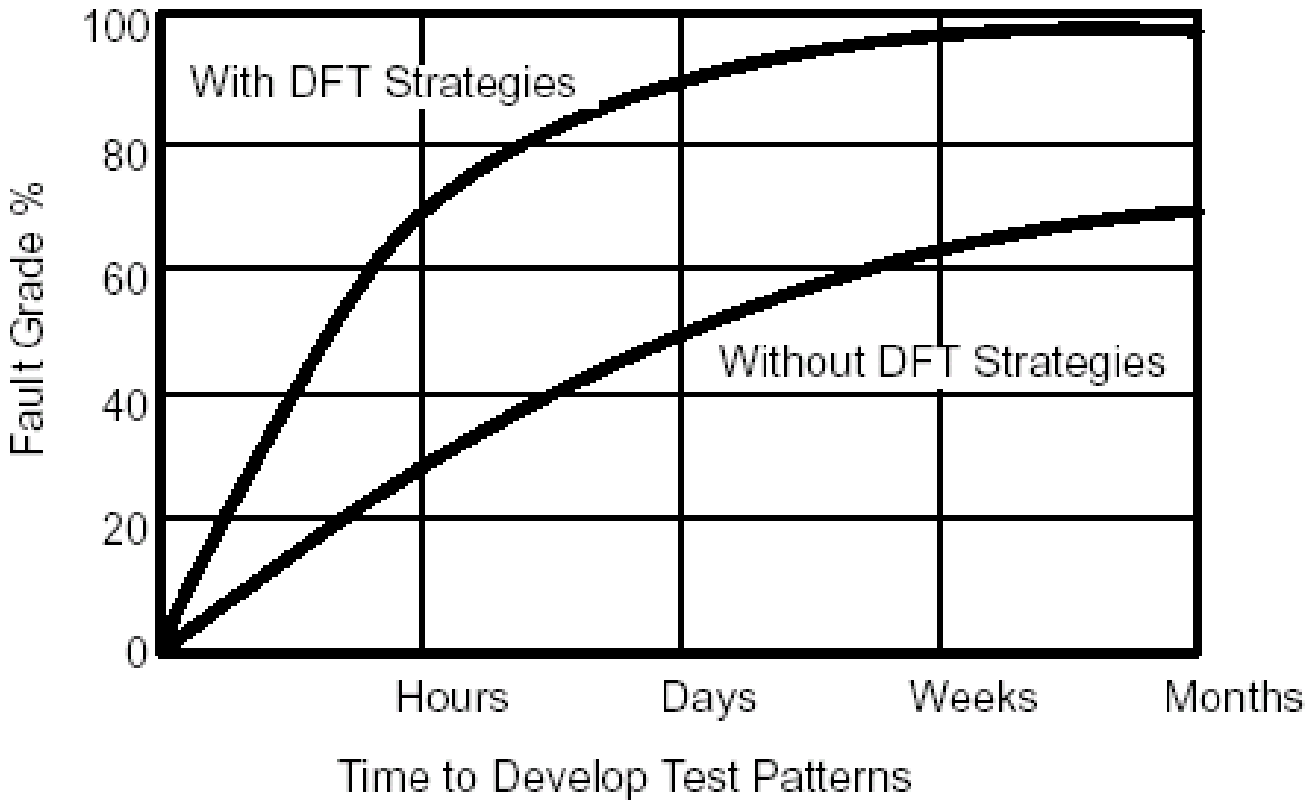
As circuits get more complex, the time needed to develop testing programs for them increases

Table 1-2. Time to Develop Test Programs (in Man-Months)†

1987–1980	3–6 months
1981–1983	6–12 months
1984–1986	9–18 months
1987–1990	12–24 months

†Source: Texas Instruments

DFT reduces product test generation times



(From the Texas Instruments IEEE std. 1149.1 (JTAG) Testability Primer)

Time is Money!

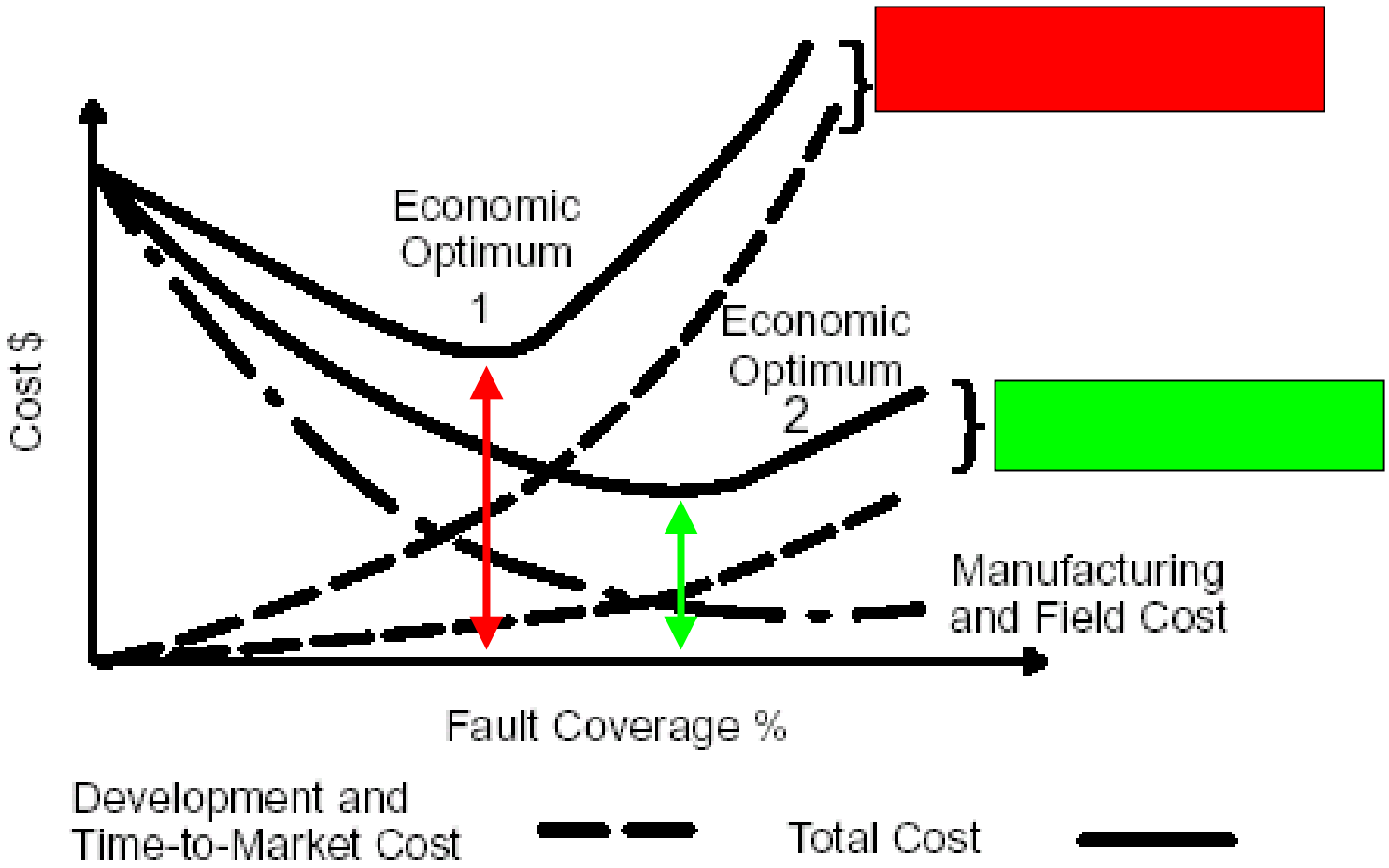
Reducing time-to-market greatly increases profit over the lifetime of the product.

Table 1-1. High-Technology Product Scenarios†

	Product A	Product B	Product C
To Market:	on time	on time	6 mos. late
Budget:	on	50% over	on
Available Profit Over 5 Years:	100%	96%	66%

†Source:McKinsey & Company

With DFT the economic optimum occurs at lower costs *and* lower defect levels



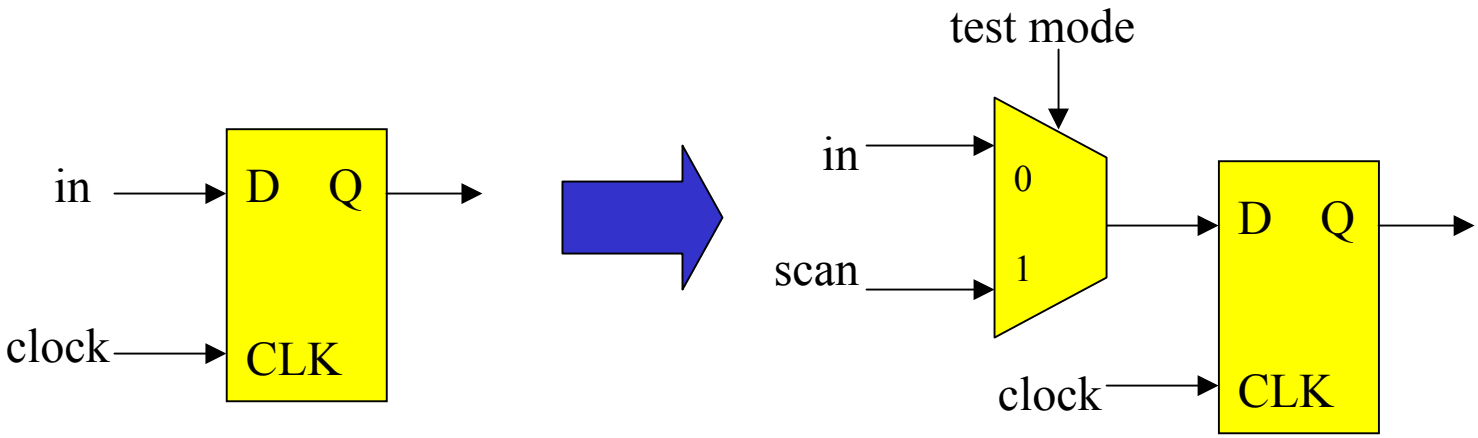
(From the Texas Instruments IEEE std. 1149.1 (JTAG) Testability Primer)

Scan-Path Design of Sequential Systems

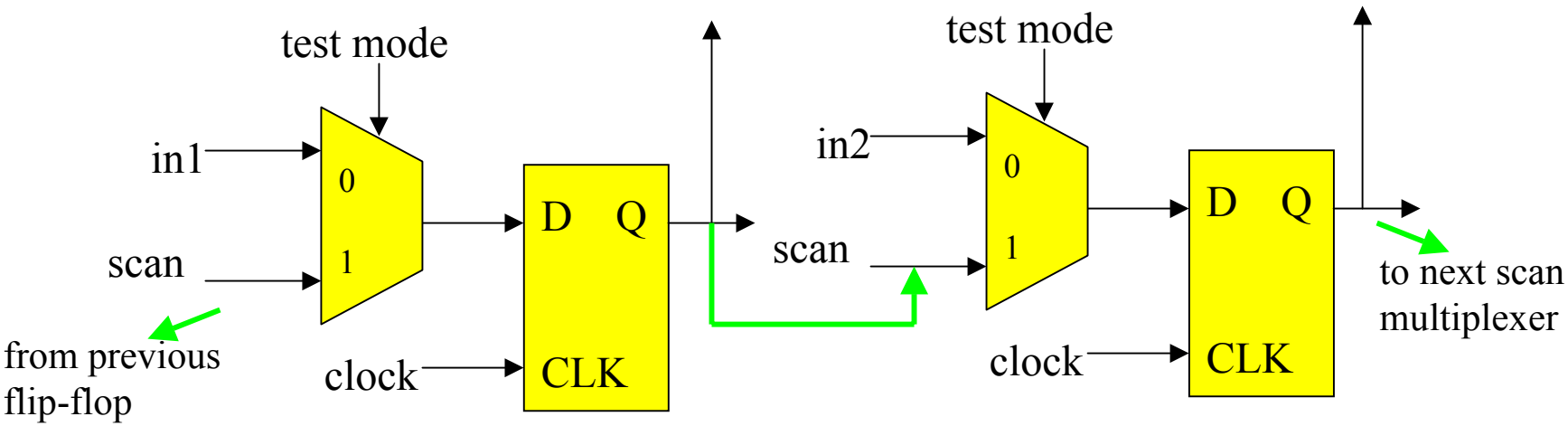
The basic idea behind scan-path design is to be able to ***reconfigure*** the system into completely accessible combinational logic blocks and ***long chains of flip-flops*** (referred to as the scan-path).

The scan path can be implemented in a very straightforward way:

Place a 2:1 multiplexer at the data inputs of every register and flip-flop.

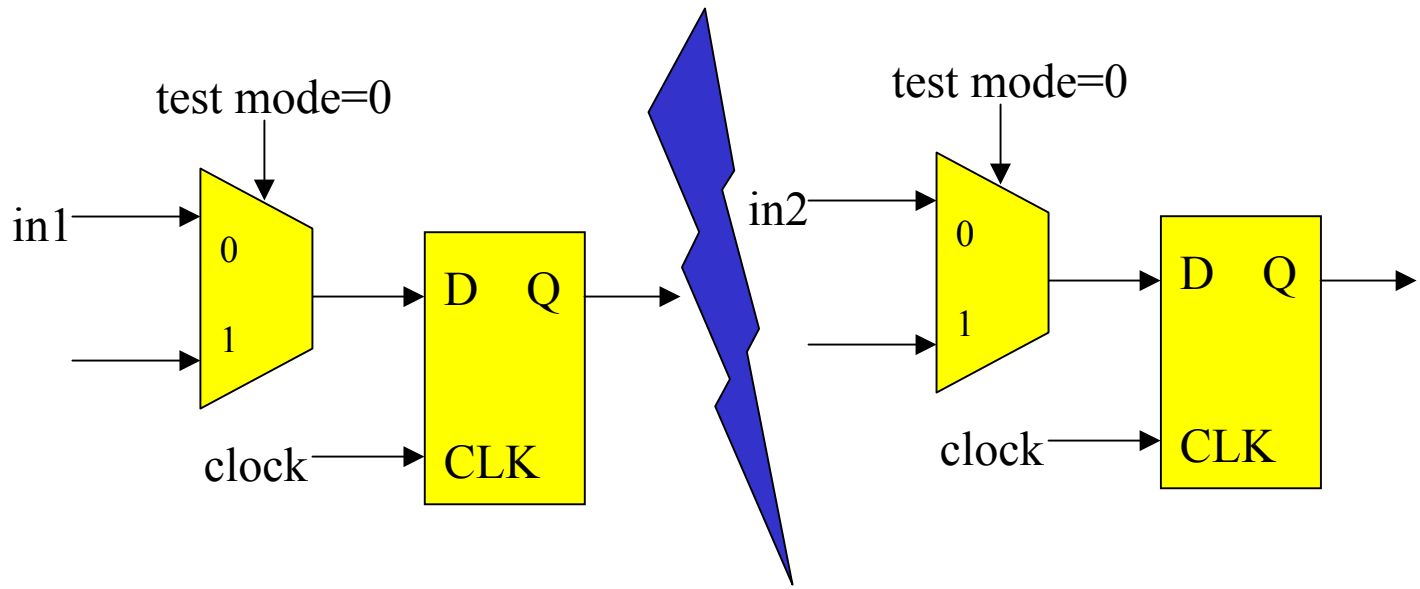


The scan inputs of each multiplexer are then connected to the output of another flip-flop, so that all of the flip-flips are connected together in a chain (the "scan-path").



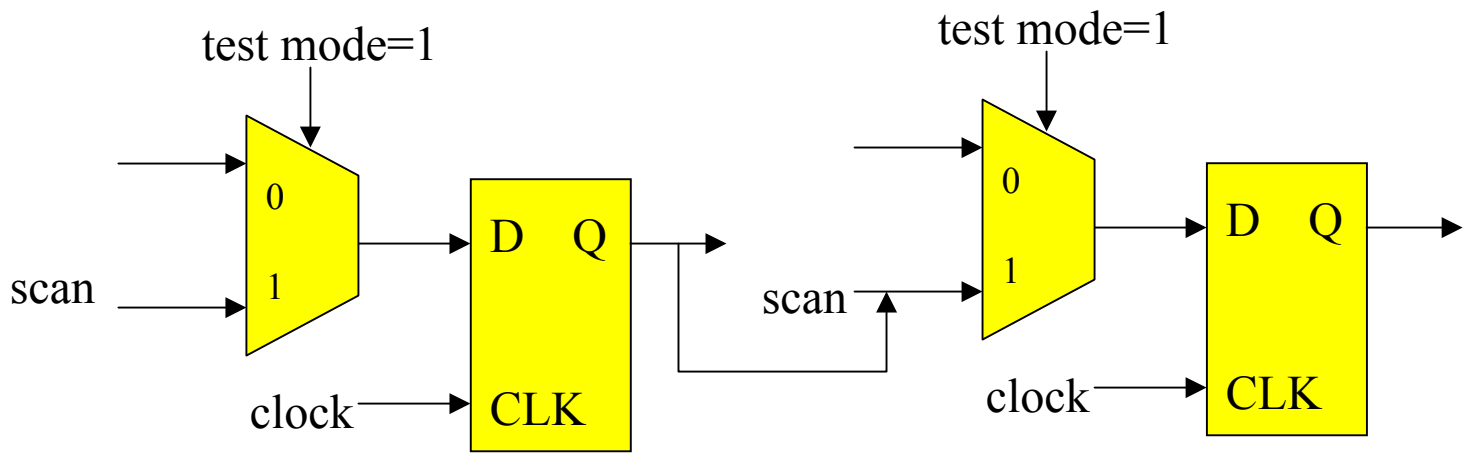
When test_mode = 0, the scan path is broken apart and the circuit functions normally.

The scan inputs are ignored in this mode.

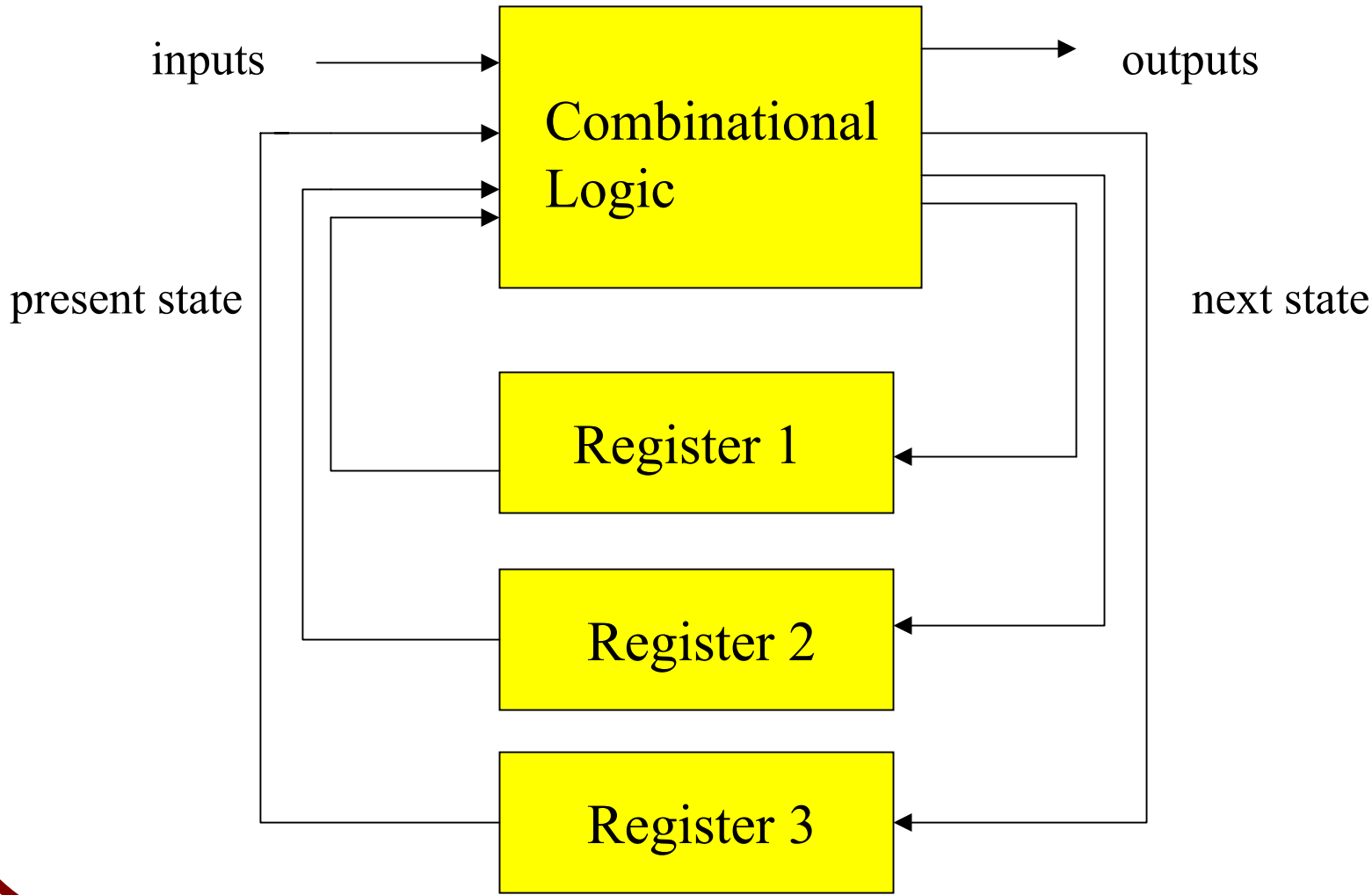


When test_mode = 1, the scan path is formed and the circuit functions as a shift register.

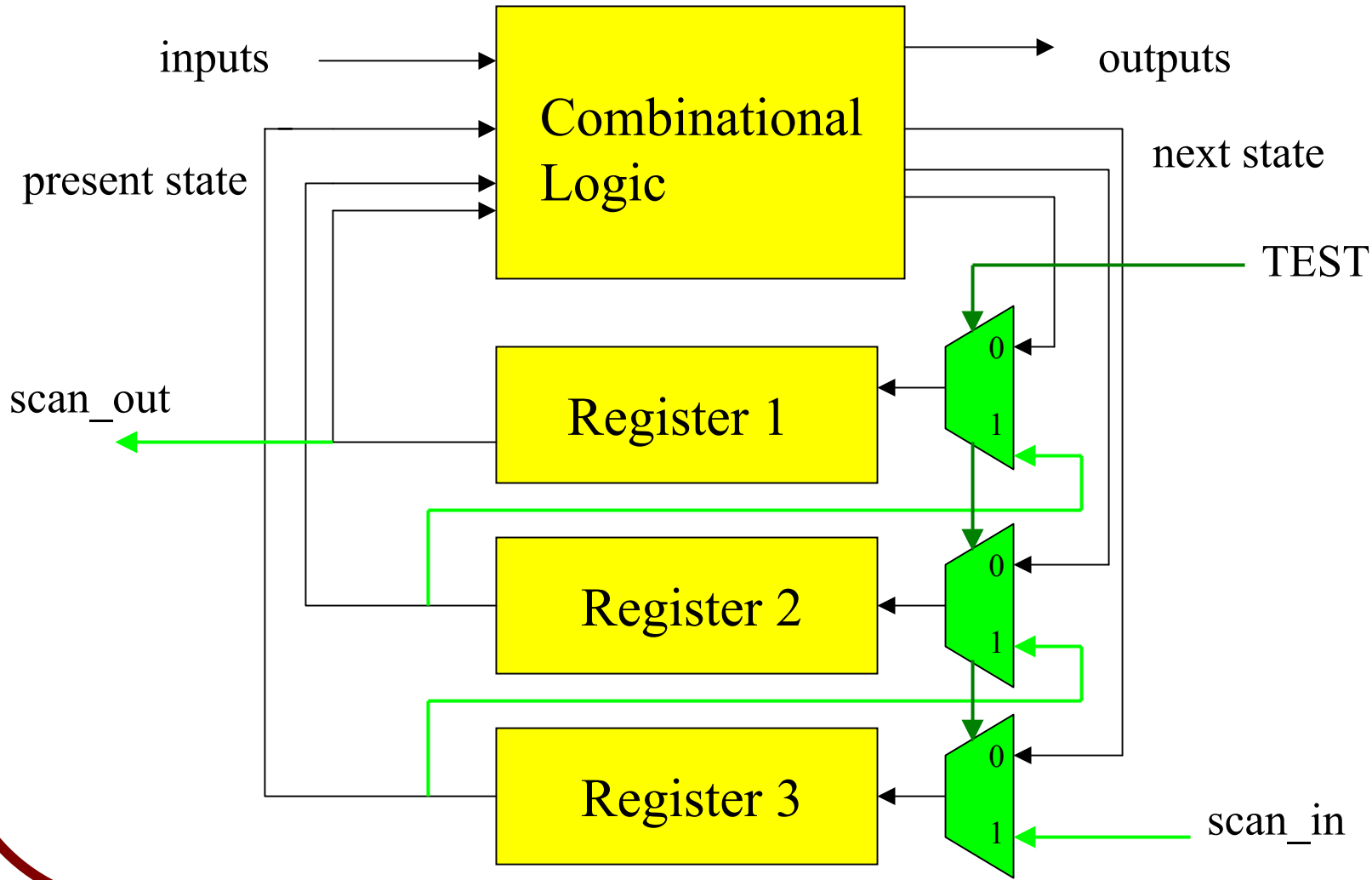
The inputs are ignored in this mode.



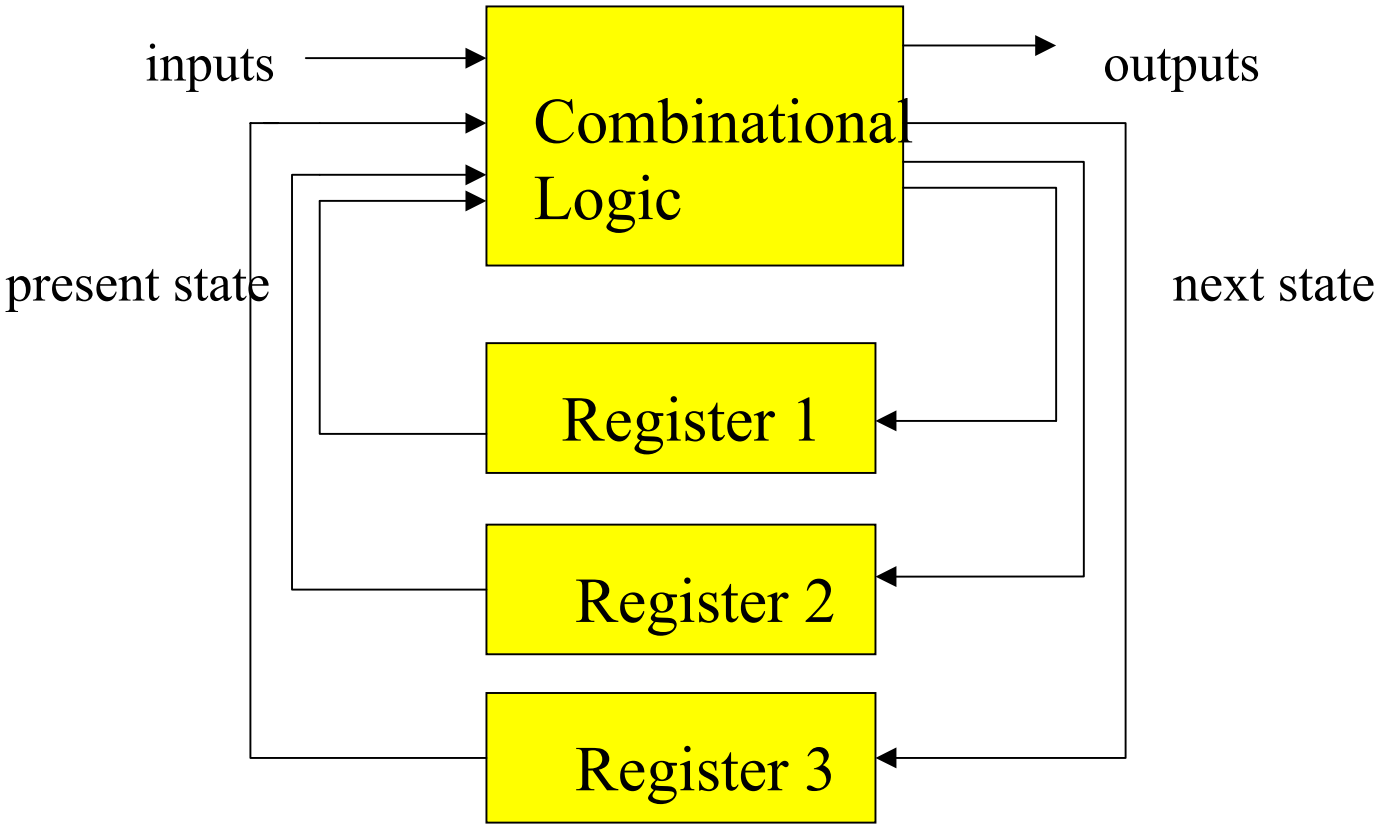
Example: Ordinary version of an FSM



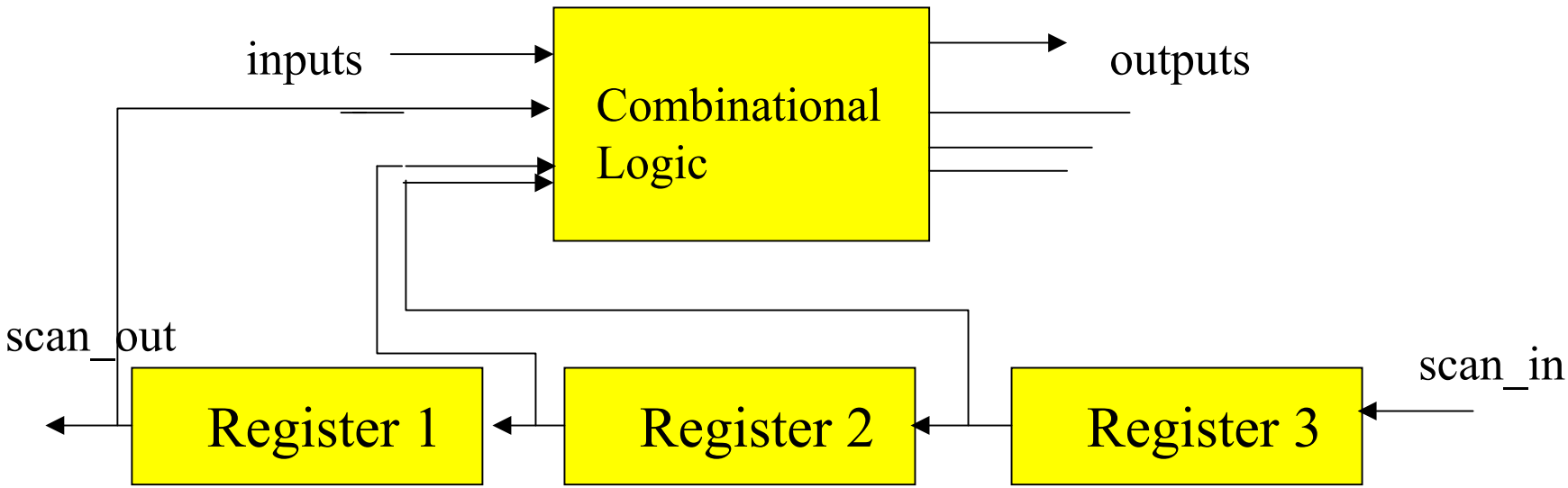
Example: Scan-Path version of an FSM



When TEST=0, the circuit looks like the ordinary, non-scan-path version of the FSM



When TEST=1, the circuit looks like a shift register



The scan-path solves the ***controllability*** problem, because we can use it to scan in the test data

It also solves the ***observability*** problem, because we can use it to scan out the test responses

The testing procedure for a scan-path circuit consists of two parts:

- Test the scan registers
- Test the combinational logic

To test the scan registers:

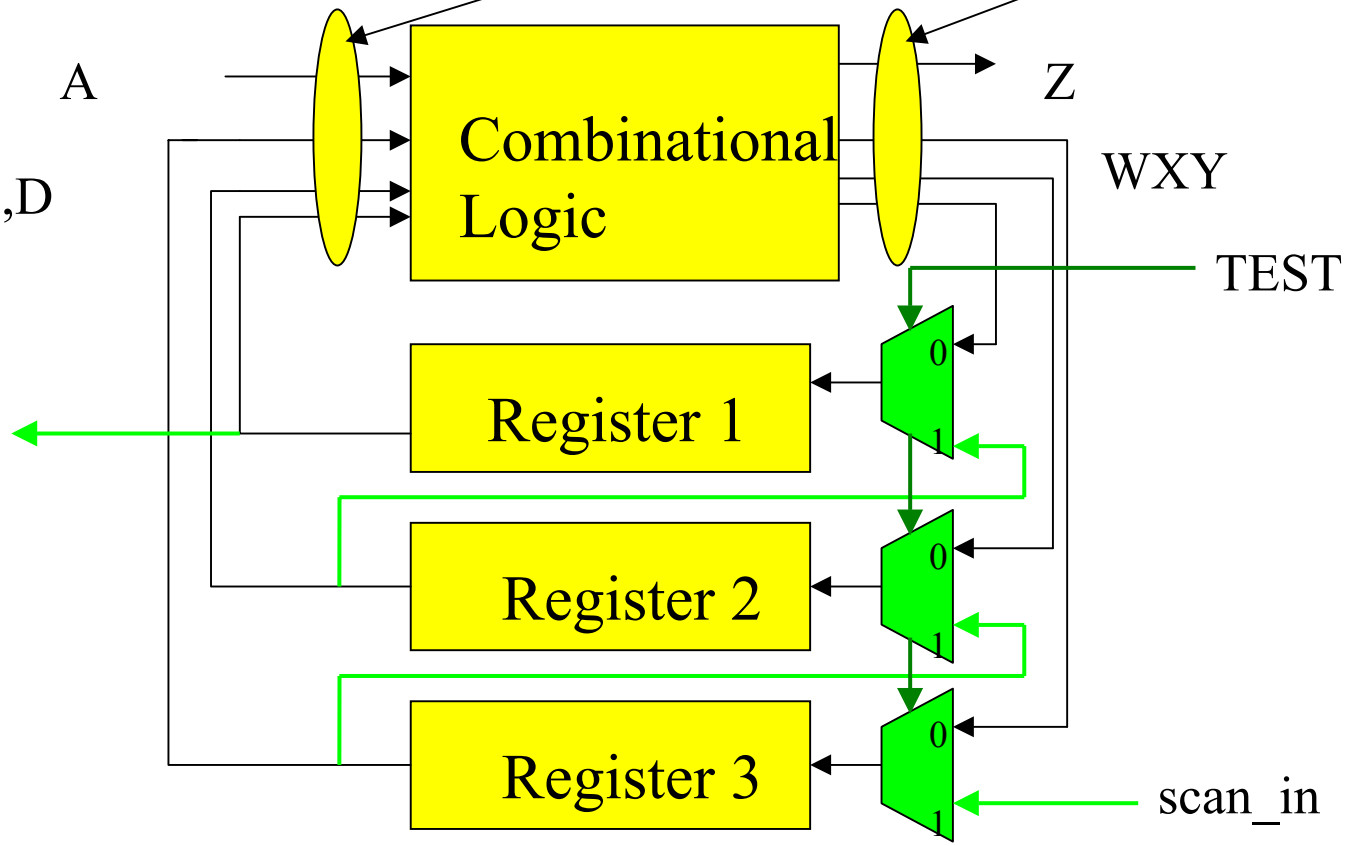
- Set TEST=1 to reconfigure the flipflops into the scan chain shift-register.
- Shift in N random bits, where N is the number of flip-flops.
- Shift out the N bits and compare with the values of the bits that were shifted in.

To test the combinational logic:

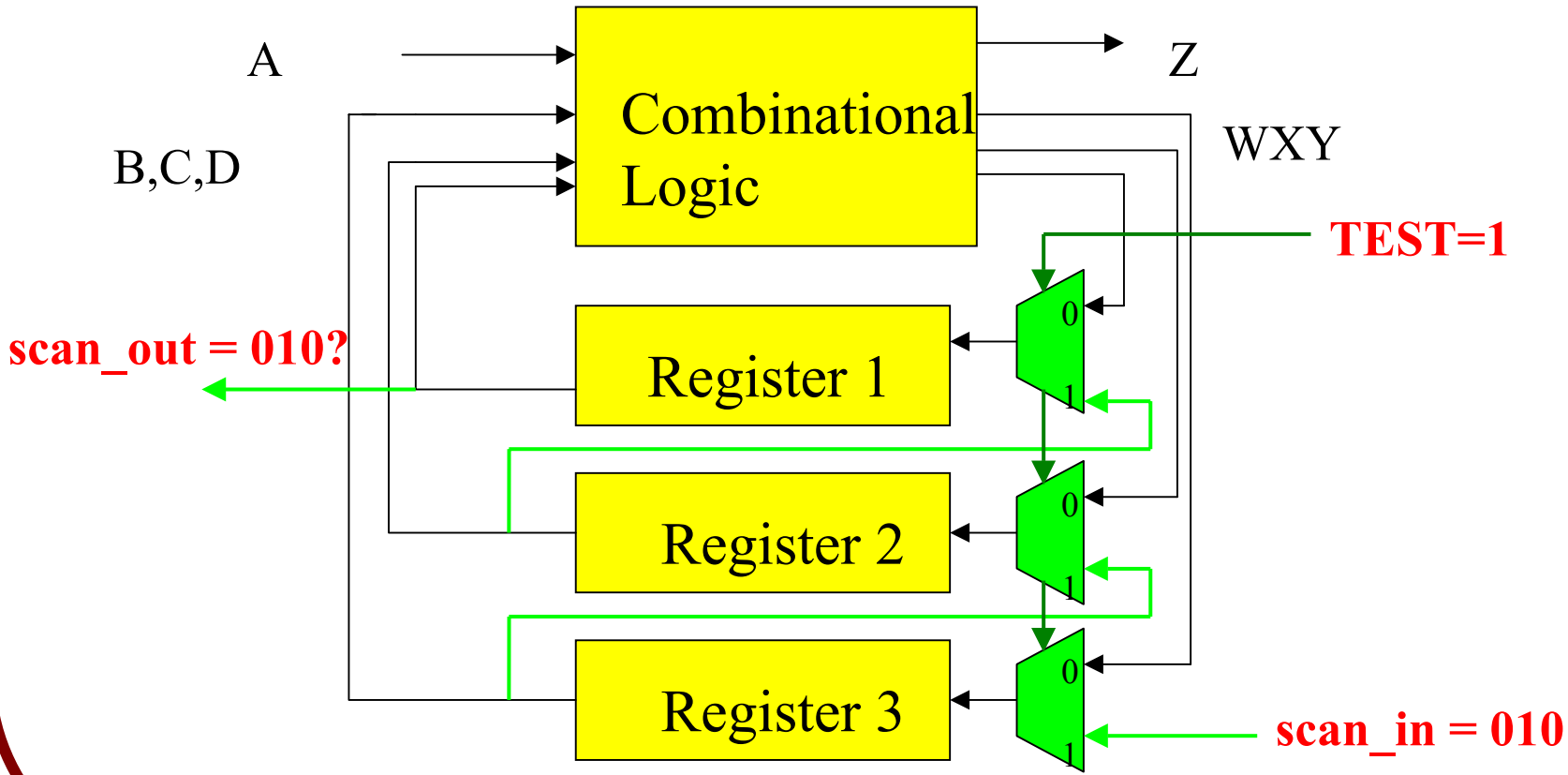
- First, derive a Fault-Detection-Test-Set (FDTS) for the combinational logic block
- For ***every*** test vector in the FTDS do the following:
  - set TEST=1 to connect the scan-path registers, and shift in the bits corresponding to the values of the internal inputs as required by the test vector
  - set TEST=0 to put the system into normal mode
  - set the external inputs as required by the test vector
  - clock the registers once to load the internal responses into the registers.
  - check the external outputs against the expected response to the test vector
  - set TEST=1 and shift out the scan-path contents - check these against the expected responses for the internal outputs

Steps 1 and 6 can be combined,
that is, we can shift in the next test vector
while shifting out the previous responses.

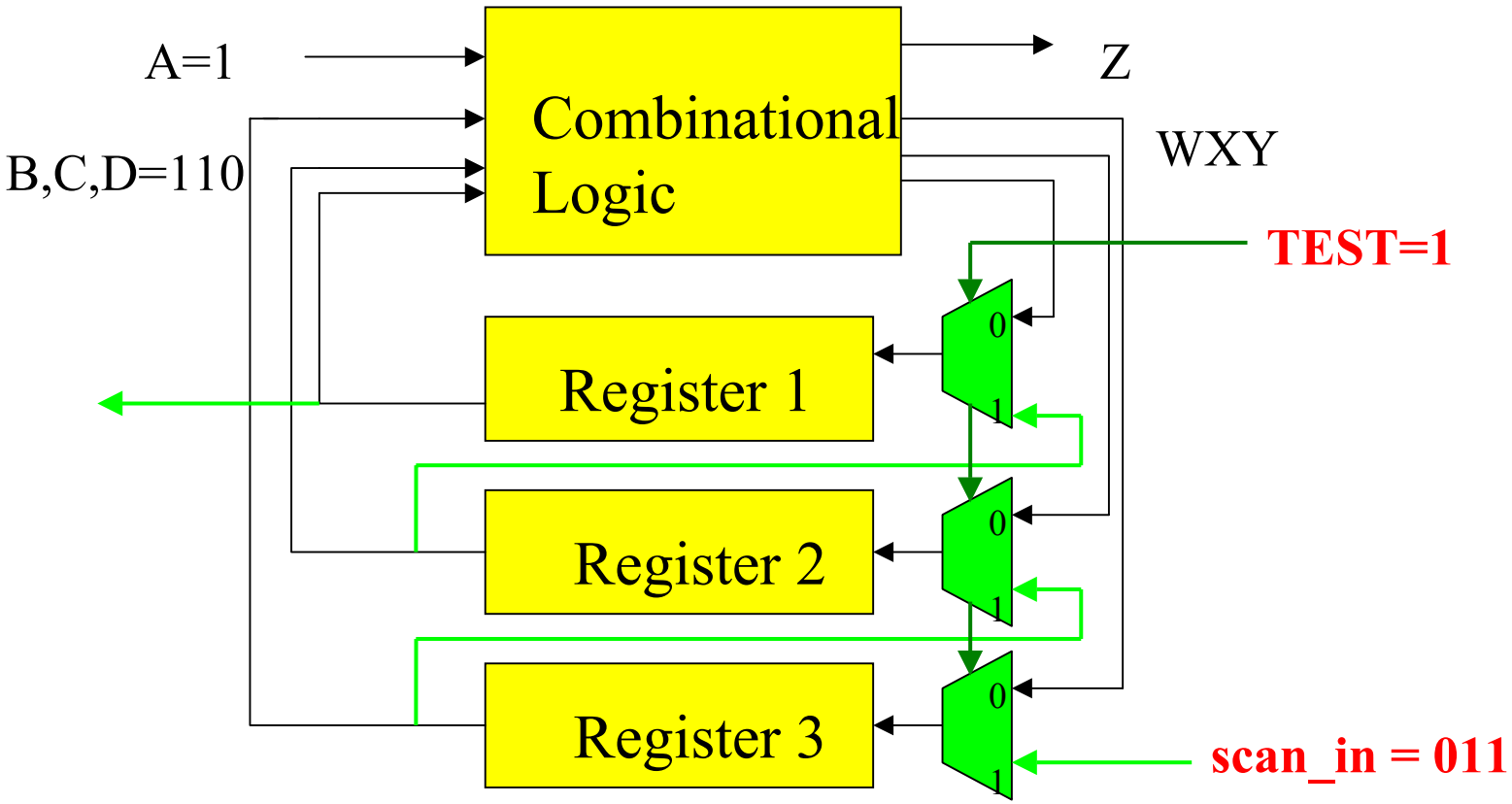
Example - present test vector (ABCD=1110) and check the response on lines (WXYZ)



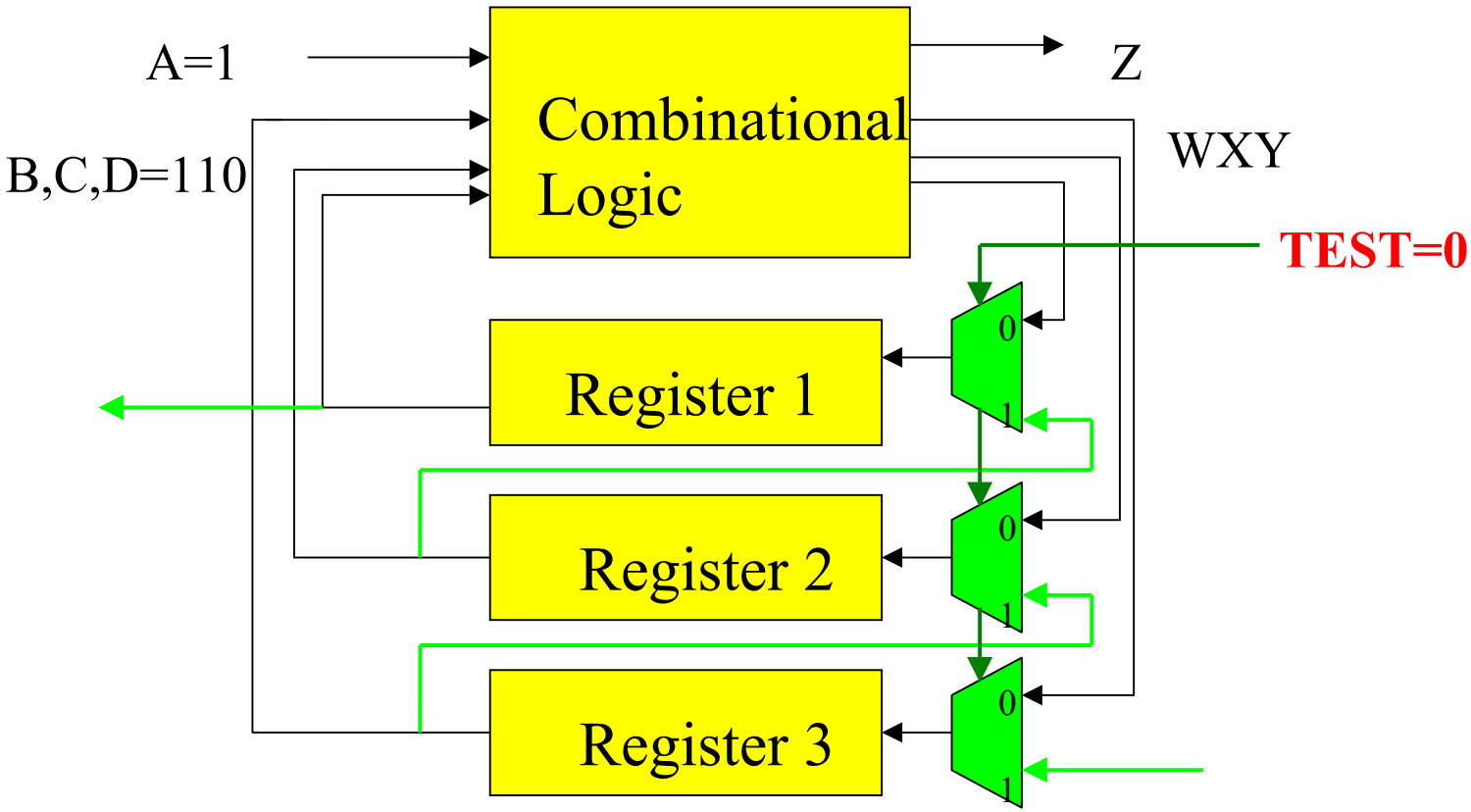
Step 1 - set test=1, shift in 010 to test flip-flops



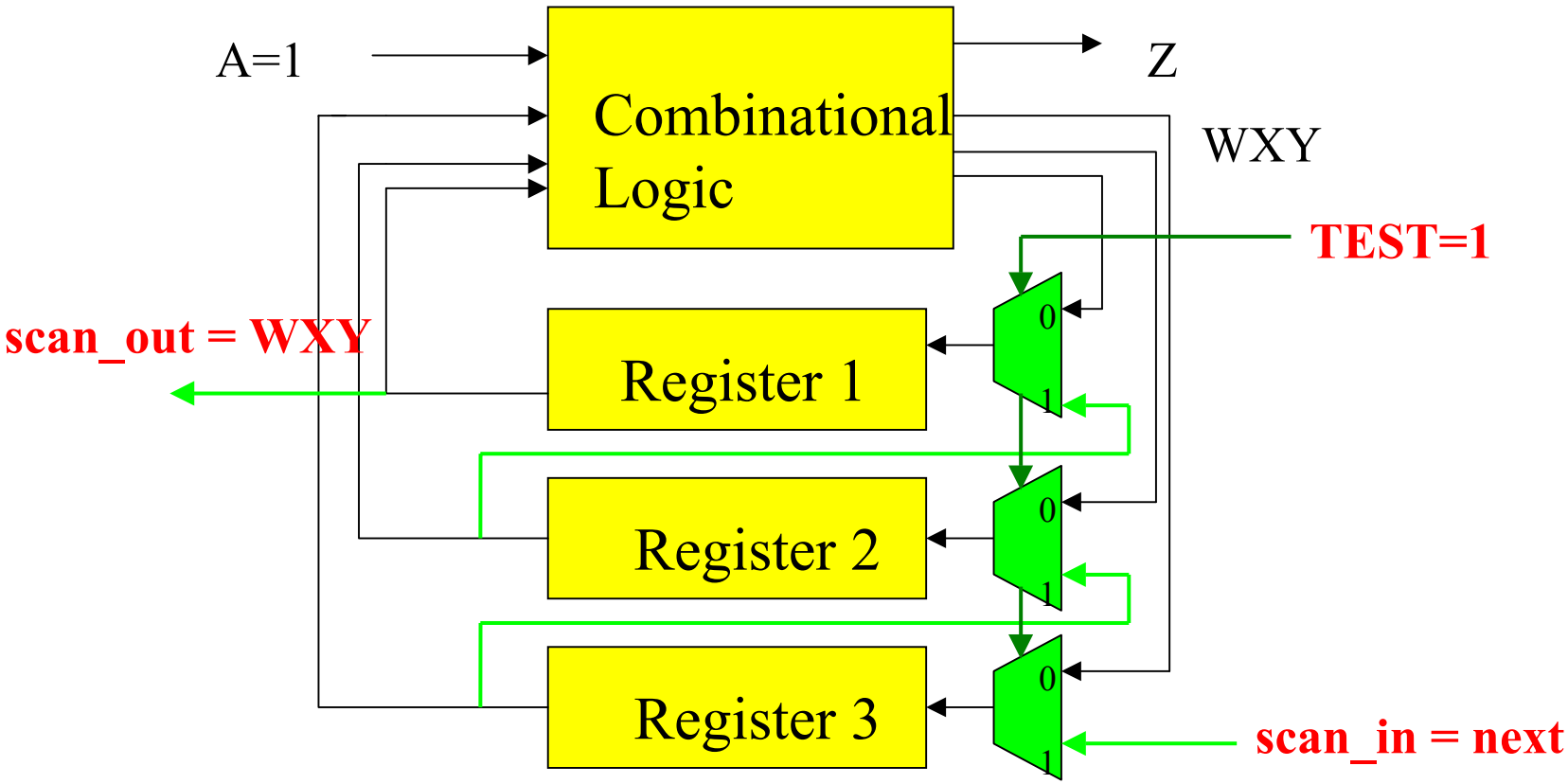
Step 2 - set test=1, shift in the BCD values of the test vector - 011 (bit D is entered first, and the A value is presented externally)



Step 3 - set test=0, clock once to store the values WXY into the registers. Examine the value of Z



Step 4 - set test=1, shift out the responses WXY through scan_out. At the same time the next test vector can be shifted in.



Drawbacks of Scan-Path Design

- Circuit may be *larger*, due to the added multiplexers

- Circuit may be *slower*, again due to the added multiplexers

- Designer is *limited* to a fully synchronous system (no asynchronous elements allowed)

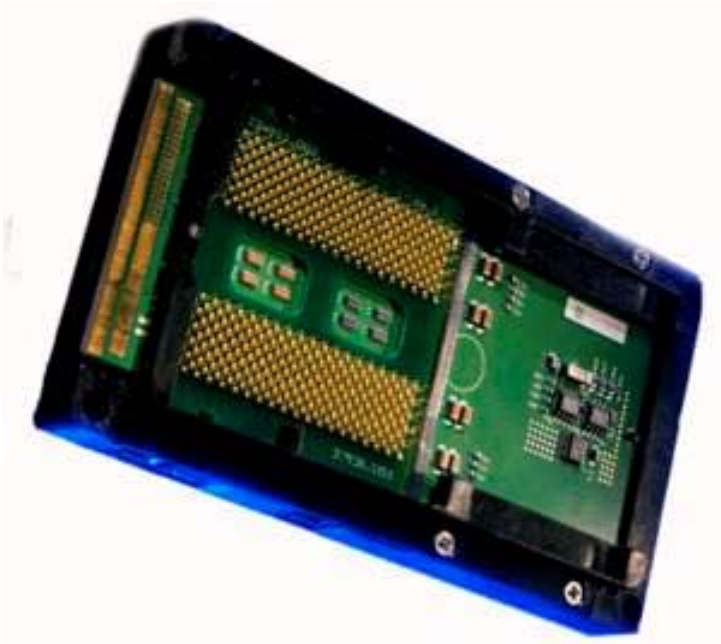
The savings in testing costs generated by using scan-path design generally outweigh the added chip costs.

So, scan-path design is used extensively in modern ICs.

Example - Intel/HP McKinley microprocessor

(also known as Itanium 2)

Introduced in 2002, it has 221 million transistors



Test Methodology for the *McKinley* Processor, by Don Douglas Josephson, Hewlett-Packard Company, Steve Poehlman, Intel Corporation, Vincent Govan, Hewlett-Packard

- **The design contains *51 scanpaths***
- **Approximately *136,000 state elements* are accessed via these scanpaths**
- **The longest scanpath is *4800 bits* long.**
- **An estimate of the *total state element count* for the processor would be roughly *160,000*; therefore, approximately *85% of the internal state elements* (exclusive of memory arrays and register files) are scan accessible.**
- ***5 scanpaths* accessing *24,000 non-destructive observe-only scanlatches* allow access to internal state while the chip continues to operate. The longest of these scanpaths is 8000 bits long. These are used for special observability purposes during silicon debug.**