طراحی سیستم‌های تعبیه‌شده
Embedded System Design

فصل ششم ـ قسمت اول

اعتبارسنجی
Validation

کاظم فولادی
دانشکده‌ی مهندسی برق و کامپیوتر
دانشگاه تهران

kazim@fouladi.ir

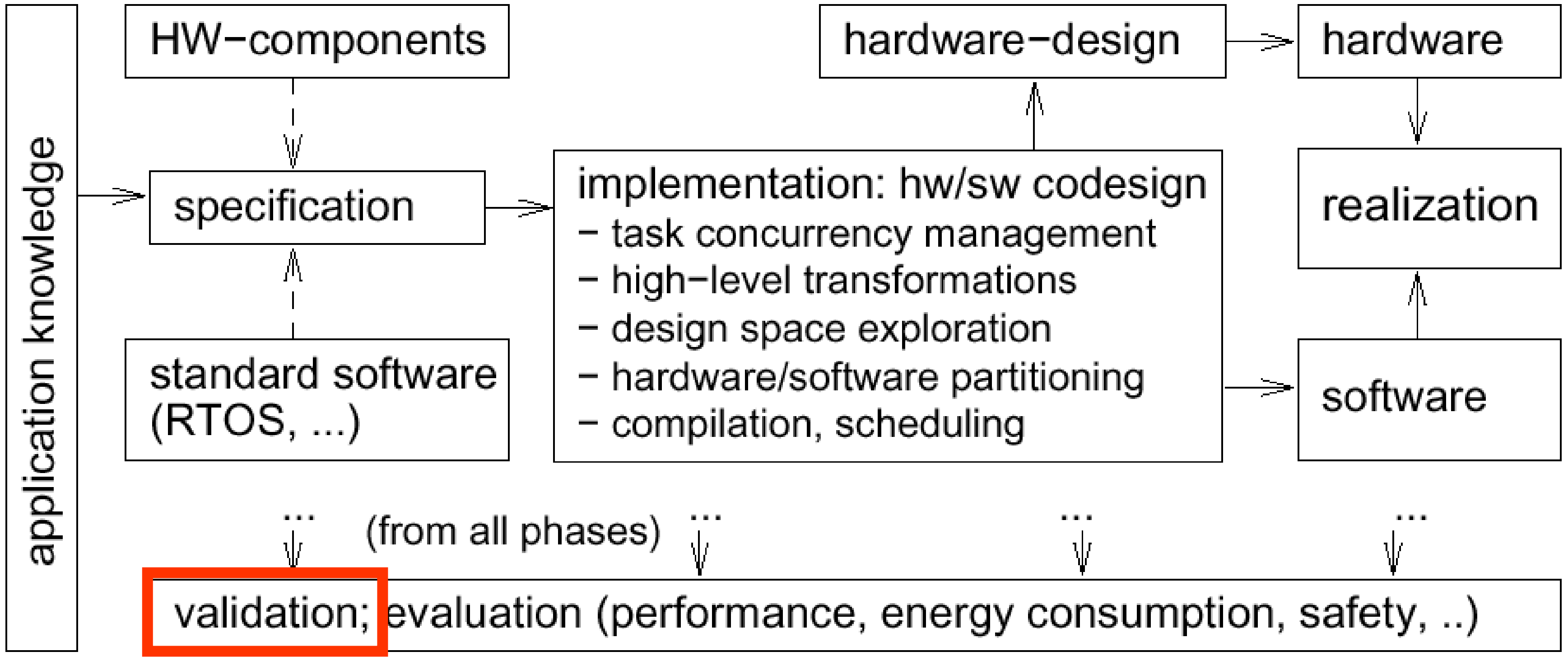
Validation

- Simulation and test pattern generation (TPG) –

اعتبارسنجی

ـ شبیه‌سازی و تولید الگوی آزمون ـ

اعتبارسنجی
Validation


application knowledge
HW−components
specification
standard software (RTOS, ...)
implementation: hw/sw codesign
− task concurrency management
− high−level transformations
− design space exploration
− hardware/software partitioning
− compilation, scheduling
hardware−design
hardware
realization
software
(from all phases)
validation; evaluation (performance, energy consumption, safety, ..)

مقدمه (۱)
Introduction (1)

تعریف: **اعتبارسنجی** فرایندی است که در آن بررسی می‌شود آیا یک طراحی مشخص (احتمالاً نیمه تمام) برای هدف آن مناسب است، همه‌ی محدودیت‌ها را ارضا می‌کند و به صورتی که مورد انتظار است عمل خواهد کرد.

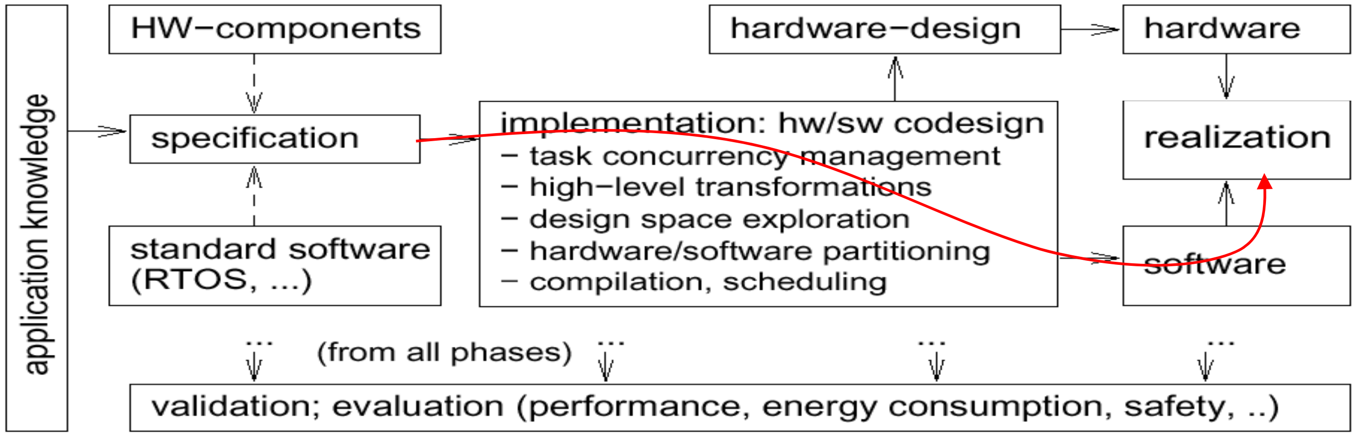
تعریف: اعتبارسنجی با دقت ریاضی، وارسی (رسمی) نام دارد.

Definition: *Validation* is the process of checking whether or not a certain (possibly partial) design is appropriate for its purpose, meets all constraints and will perform as expected.

Definition: Validation with mathematical rigor is called *(formal) verification*.

مقدمه (۲)

Introduction (2)


application knowledge
HW−components
specification
standard software (RTOS, ...)
implementation: hw/sw codesign
− task concurrency management
− high−level transformations
− design space exploration
− hardware/software partitioning
− compilation, scheduling
hardware−design
hardware
realization
software
(from all phases)
validation; evaluation (performance, energy consumption, safety, ..)

Ideally: Formally verified tools transforming specifications into implementations ("*correctness by construction*").

In practice: Non-verified tools and manual design steps

☞ validation of each and every design required

Unfortunately has to be done at intermediate steps and not just for the final design

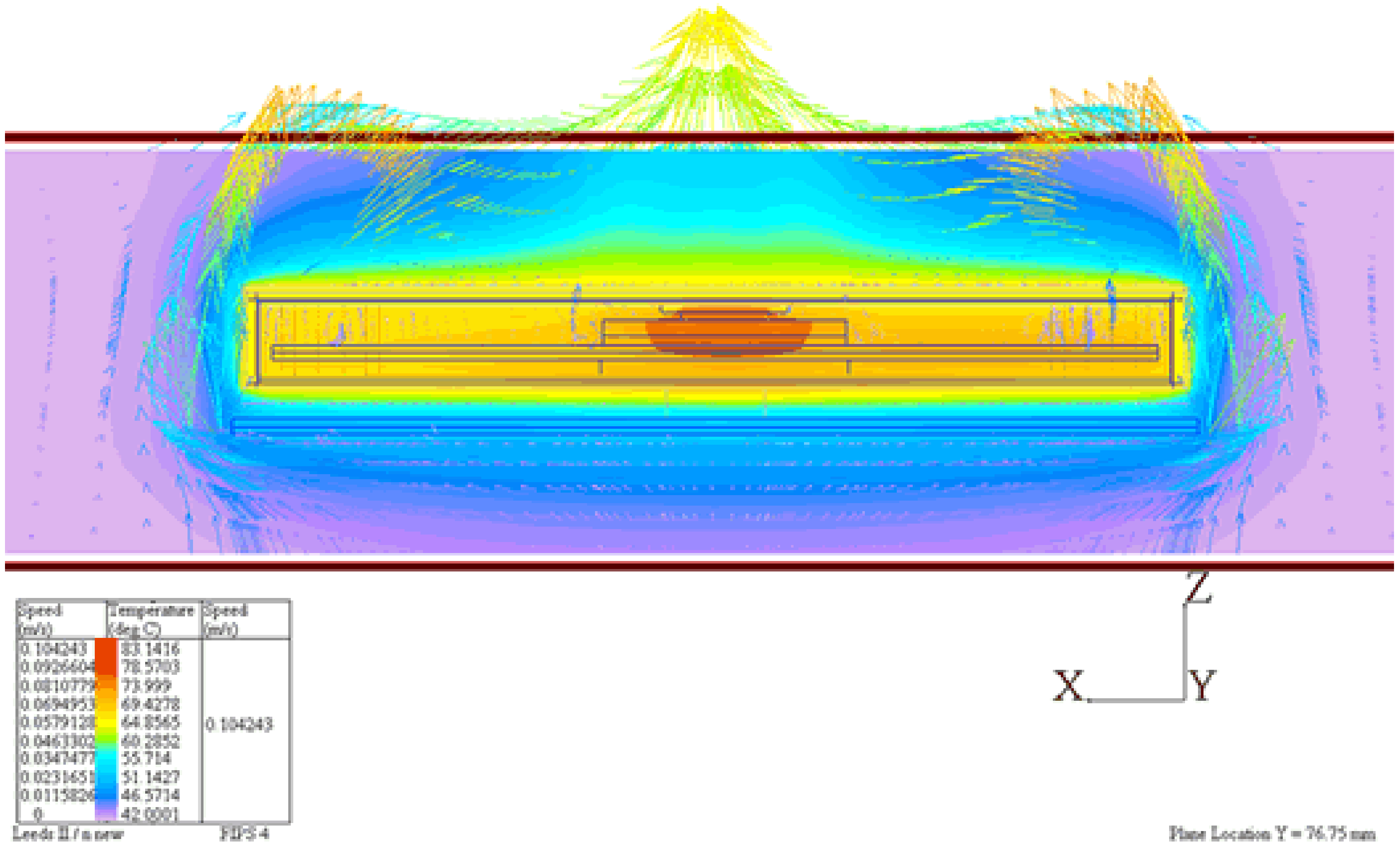
☞ Major effort required.

شبیه‌سازی‌ها
Simulations

- Simulations try to imitate the behavior of the real system on a (typically digital) computer.
- Simulation of the functional behavior requires executable models.
- Simulations can be performed at various levels.
- Some non-functional properties (e.g. temperatures, EMC) can also be simulated.

مثالی از شبیه‌سازی گرمایی (۱)
Examples of thermal simulations (1)

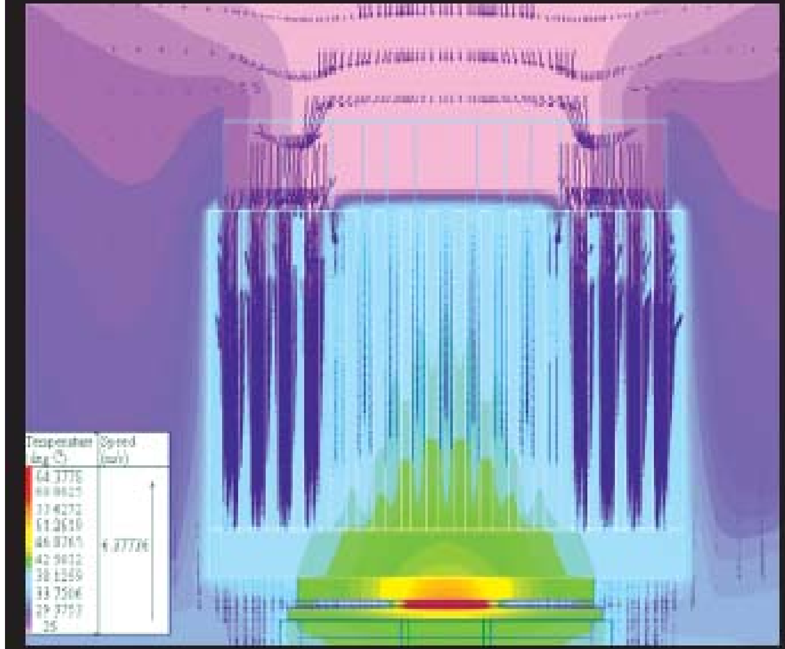
Encapsulated cryptographic coprocessor:



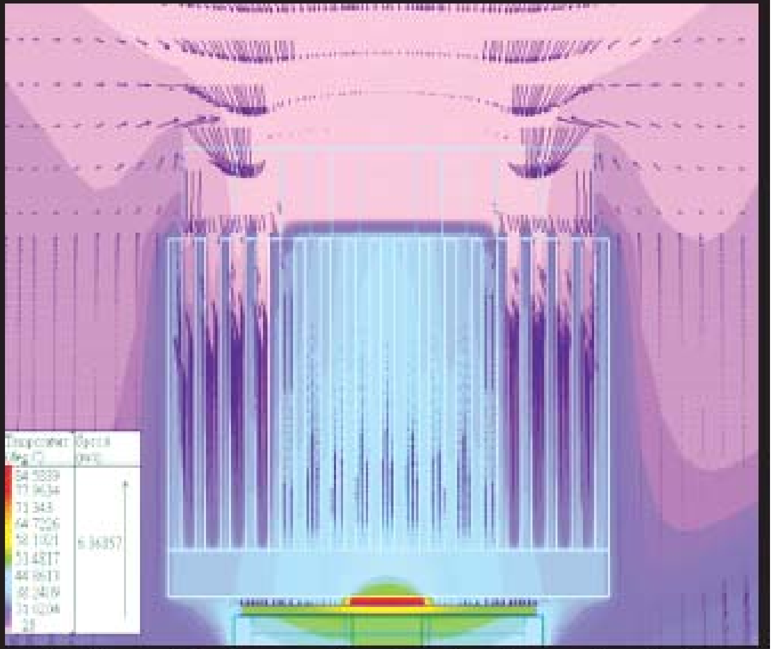
Source: http://www.coolingzone.com/Guest/News/NL_JUN_2001/Campi/Jun_Campi_2001.html

مثالی از شبیه‌سازی گرمایی (۲)
Examples of thermal simulations (2)

Microprocessor



▲ Flomerics image showing the thermal solution with a metal lid.

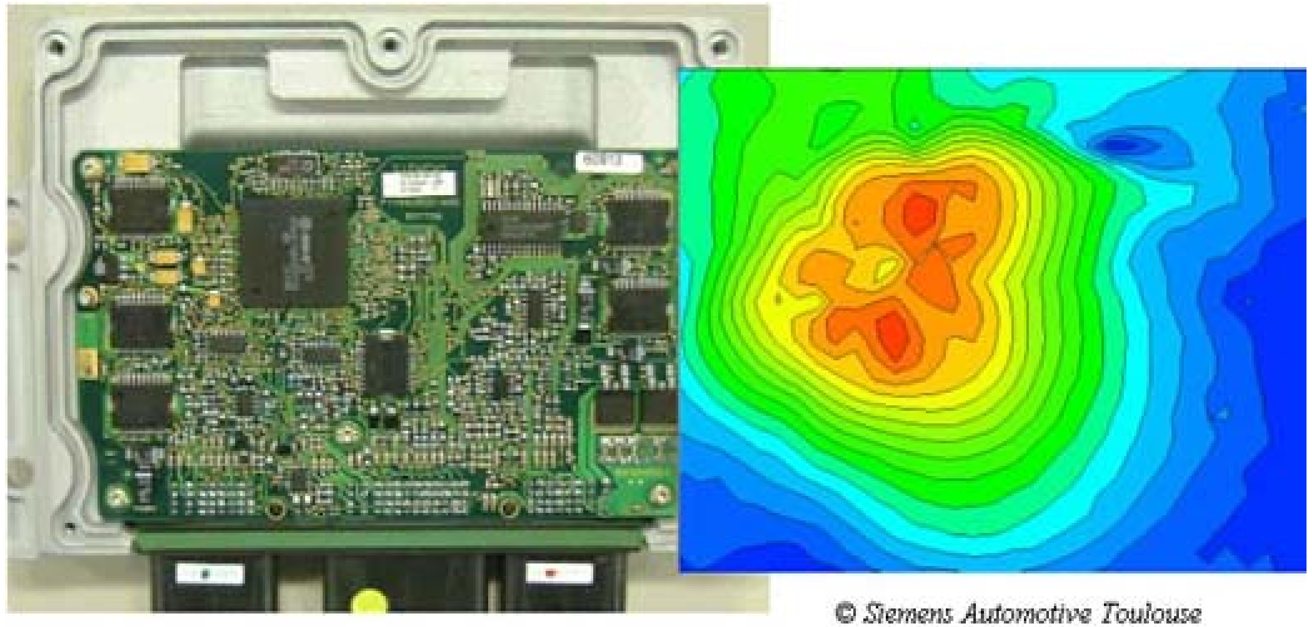


▲ Flomerics image showing the thermal solution without a metal lid.

Source: http://www.flotherm.com/applications/app141/hot_chip.pdf

شبیه‌سازی سازگاری الکترومغناطیسی
EMC simulation

Example: car engine controller


© Siemens Automotive Toulouse

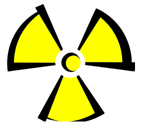
Red: high emission

Validation of EMC properties often done at the end of the design phase.

Source: http://intrage.insa-tlse.fr/ ~etienne/emccourse/what_for.html

محدودیت‌های شبیه‌سازی‌ها
Simulations Limitations

- Typically slower than the actual design.
  ☞ **Violations of timing constraints** likely if simulator is connected to the actual environment
- Simulations in the real environment may be **dangerous**
- There may be huge amounts of data and it may be impossible to simulate enough data in the available time.
- Most actual systems are too complex to allow simulating all possible cases (inputs).
  Simulations can help finding errors in designs, but they cannot guarantee the absence of errors.



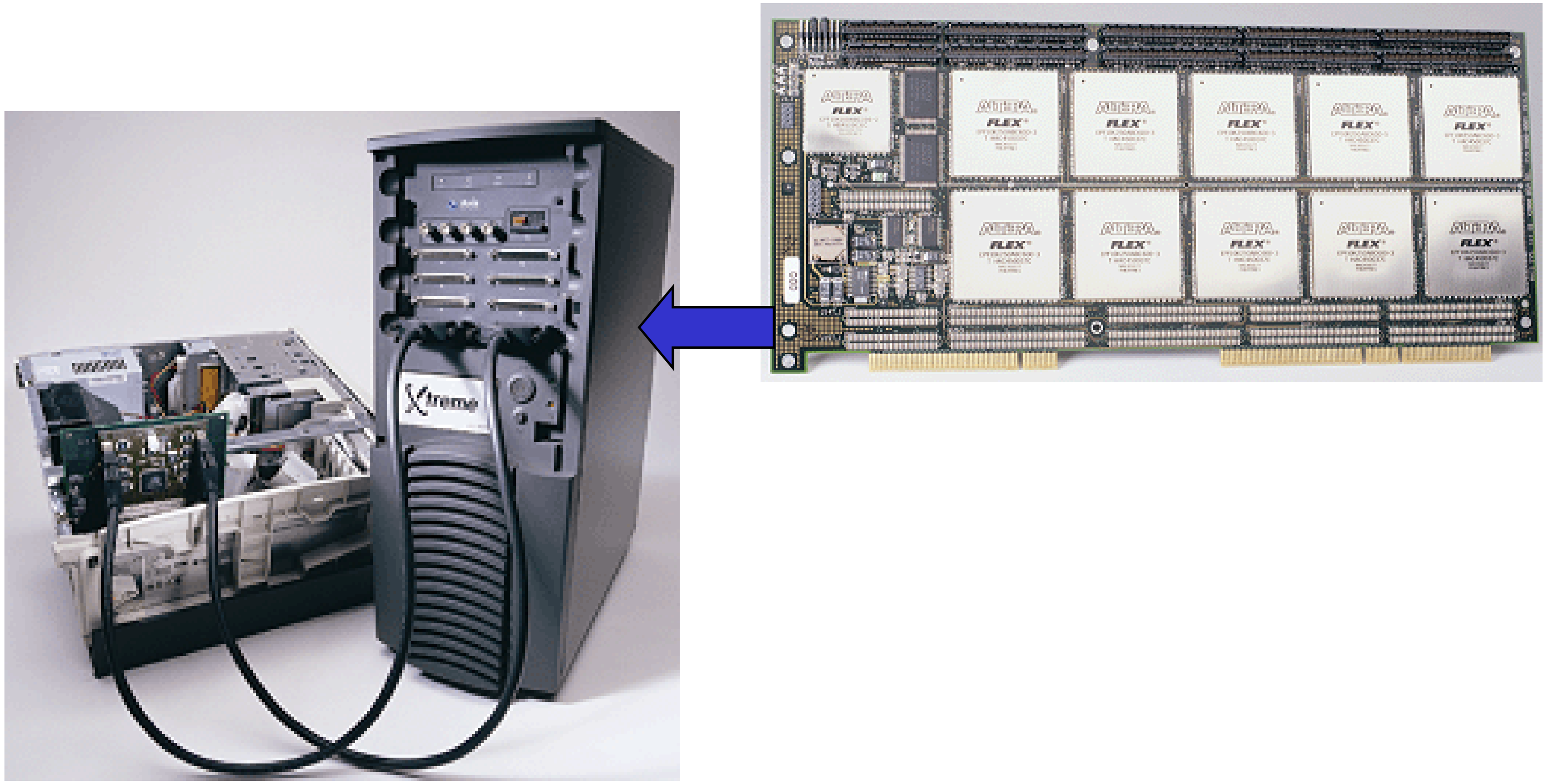
نمونه‌سازی سریع
Rapid prototyping/Emulation

- **Prototype**: Embedded system that can be generated quickly and behaves very similar to the final product.
- May be larger, more power consuming and have other properties that can be accepted in the validation phase
- Can be built, for example, using FPGAs.

Example: Quickturn Cobalt System (1997), ~0.5M$ for 500kgate entry level system (no photo of more recent system)

Source & ©: http://www.eedesign.com/editorial/1997/toolsandtech9703.html

مثالی از یک نمونه‌ساز تازه‌تر
Example of a more recent commercial emulator



[www.verisity.com/images/products/xtremep{1|3}.gif]

آزمون : اهداف
Test: Goals

1. **Production test**
2. Is there any way of using test patterns for production test already during the design*?
3. **Test for failures after delivery to customer**

* Workshop focusing on the integration of production testing and design validation: HLDVT IEEE International High Level Design Validation and Test Workshop

آزمون: حوزه‌ی دید
Test: Scope

Testing includes

- the application of test patterns to the inputs of the device under test (DUT) and
- the observation of the results.

More precisely, testing requires the following steps:

1. test pattern generation,
2. test pattern application,
3. response observation, and
4. result comparison.

تولید الگوی آزمون
Test pattern generation

Test pattern generation typically

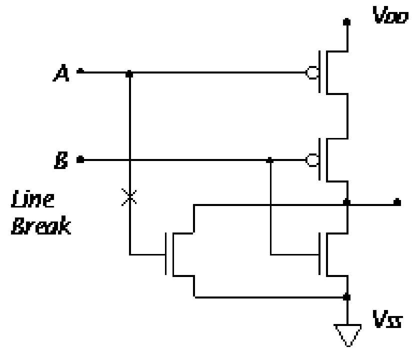
- considers certain fault models and
- generates patterns that enable a distinction between the faulty and the fault-free case.
- Examples:
  - Boolean differences
  - D-Algorithm

مدل‌های نقص
Fault models

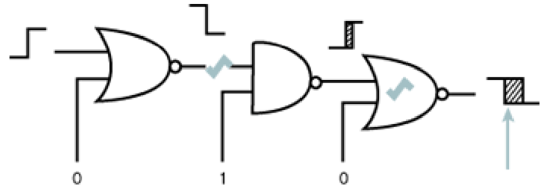
Hardware fault models include:

- **stuck-at fault model** (net permanently connected to ground or V_{dd})
- **stuck-open faults**: for CMOS, open transistors can behave like memories
- **delay faults**: circuit is functionally correct, but the delay is not.

www.cedcc.psu.edu/ee497f/rassp_43/sld022.htm

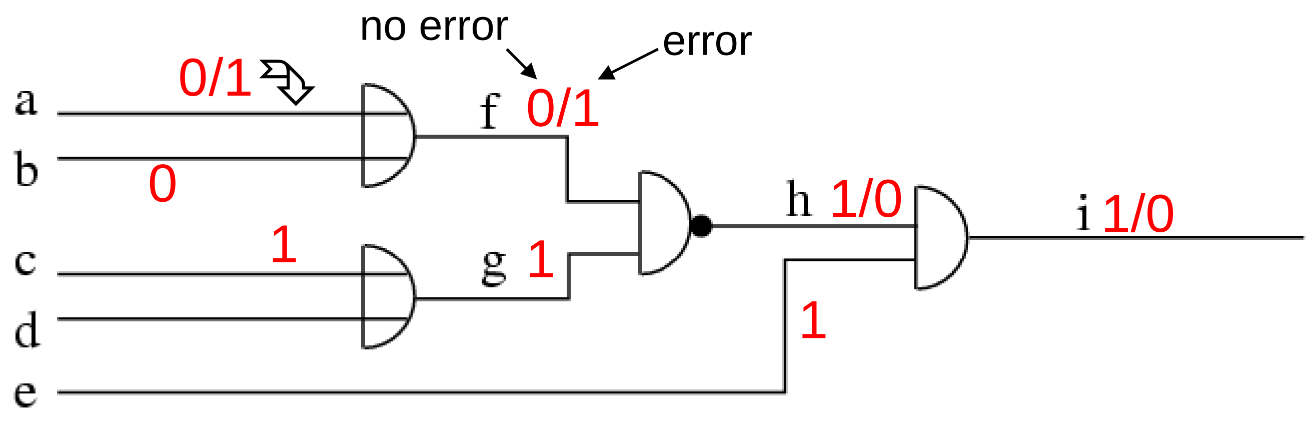


- Break above results in a "memory-effect" in the behavior of the circuit
- With AB=10, there is not path from either VDD or VSS to the output
  - F retains the previous value for some undetermined discharge time



www.synopsys.com/products/test/tetramax_ds.html

مثال ساده
Simple example



Could we check for a stuck at one error at a (s-a-1(a)) ?
Solution (just guessing):

- f='1' if there is an error
- ☞ a='0', b='0' in order to have f='0' if there is no error
- g='1' in order to propagate error
- c='1' in order to have g='1' (or set d='1')
- e='1' in order to propagate error
- i='1' if there is no error & i='0' if there is

Variable D

Getting rid of 0/1 notation:
☞ Definition:

$$D = \begin{cases} \text{'1' if there is no error} \\ \text{'0' if there is an error} \end{cases}$$

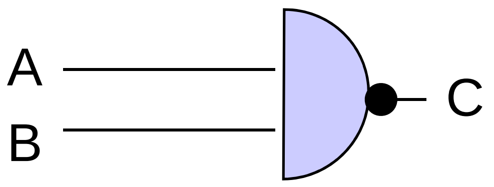
$$\overline{D} = \begin{cases} \text{'0' if there is no error} \\ \text{'1' if there is an error} \end{cases}$$

This is adequate for modeling a **single** error.
Multiple errors would require several variables.

مدل کردن گیت‌ها با مکعب‌های اولیه
Modeling gates with primitive cubes

Definition: Let a function *f* and its complement be represented by implicants. Each entry in a table of implicants and outputs is called a **primitive cube** (pc).

Example: 2-input NAND gate



	fault-free			with fault	s-a-1(A)			s-a-0(A)			s-a-1(B)			s-a-0(B)		
	A	B	C		A	B	C	A	B	C	A	B	C	A	B	C
β_1	0	X	1	α_1	X	0	1	X	X	1	0	X	1	X	X	1
	X	0	1													
β_0	1	1	0	α_0	X	1	0	-	-	-	1	X	0	-	-	-

Primitive cube

مدل کردن گیت‌های نقص‌دار با مکعب‌های D یک نقص
Modeling faulty gates with D-cubes of a fault

Primitive D-cubes of a fault (pdcf's) are cubes which model a condition under which a fault does show up.
Input values generate an output of D (resp. $\overline{D}$) if they are contained in cubes β_1 and α_0 (resp. β_0 and α_1).
Hence, we define the intersection of cubes as follows:
X $\cap$ '0' = '0', X $\cap$ '1'='1', '1' $\cap$ '0' = $\varnothing$ (empty), with X: don't care

pc

fault free				with fault	s-a-1(A)			s-a-0(A)			s-a-1(B)			s-a-0(B)		
	A	B	C		A	B	C	A	B	C	A	B	C	A	B	C
β_1	0	X	1	α_1	X	0	1	X	X	1	0	X	1	X	X	1
	X	0	1													
β_0	1	1	0	α_0	X	1	0	-	-	-	1	X	0	-	-	-

pdcf

	s-a-1(A)			s-a-0(A)			s-a-1(B)			s-a-0(B)			s-a-1(C)+			s-a-0(C)++		
	A	B	C	A₀	B	C	A	B	C	A	B	C	A	B	C	A	B	C
$\beta_0 \cap \alpha_1$		$\emptyset$		D	1	$\overline{D}$		$\emptyset$		1	D	$\overline{D}$	1	1	$\overline{D}$	0	X	D
$\beta_1 \cap \alpha_0$	0*	1	D		$\emptyset$		1	$\overline{D}$	D		$\emptyset$					X	0	D

مدل کردن انتشار با مکعب‌های انتشار
Modeling propagation with propagation cubes (1)

Propagation D-cubes are cubes that model requirements for propagating errors to the output.

An error D ($\overline{D}$) at input r gets propagated to the output as f=D ($\overline{D}$) iff r='0' implies f='0' and r='1' implies f='1' (non-inverting)

An error D ($\overline{D}$) at input r gets propagated to the output as f= $\overline{D}$(D) iff r='0' implies f='1' and r='1' implies f='0' (inverting).

Hence, consider intersection of β_1 and β_0 while ignoring input r.

مدل کردن انتشار با مکعب‌های انتشار
Modeling propagation with propagation cubes (2)

Hence, consider intersection of β_1 and β_0 while ignoring input *r*.
Example: 2-input NAND gate

pc

fault free				with fault	s-a-1(A)			s-a-0(A)			s-a-1(B)			s-a-0(B)		
	A	B	C		A	B	C	A	B	C	A	B	C	A	B	C
β_1	0	X	1	α_1	X	0	1	X	X	1	0	X	1	X	X	1
	X	0	1													
β_0	1	1	0	α_0	X	1	0	-	-	-	1	X	0	-	-	-

pdcf

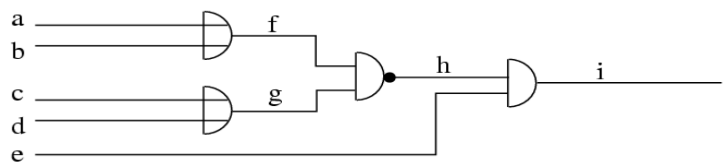
	A	B	C		A	B	C		A	B	C		A	B	C
$\beta_{1/A=1}$	1	0	1	$\beta_{1/A=0}$	0	X	1	$\beta_{1/B=1}$	0	1	1	$\beta_{1/B=0}$	X	0	1
$\beta_{0/A=0}$		$\emptyset$		$\beta_{0/A=1}$	1	1	0	$\beta_{0/B=0}$		$\emptyset$		$\beta_{0/B=1}$	1	1	0
$\bigcap_r$		$\emptyset$			$\overline{D}$	1	D			$\emptyset$			1	D	$\overline{D}$
					D	1	$\overline{D}$						1	$\overline{D}$	D

D-Algorithm (1)

1. Select *D*-cube for the error under consideration.
2. **Implication**: Imply signals whose value results unambiguously from the preceding selection. Based on the intersection between the "test cube" (set of known signals) and primitive cubes of gates reached by the test cube. Return to last step if intersection is empty (backtracking).
3. **D-drive:** *D*-frontier = all gates whose outputs are unspecified and whose inputs carry a value of *D* or $\overline{D}$. Select gate $\in$ *D*-frontier. Propagate signal to output by intersecting test cube with pdcf of that gate.
   Return to last step if no non-empty intersection exists.
4. Iterate steps 2 and 3 until some signal has reached output

D-Algorithm (2)

5. **Line justification:** Unspecified inputs will be adjusted by intersecting the test cube and primitive cubes of the gates. Backtracking if required.



Example

Typ. Runtime: O((# of gates)²)
1 pattern per error

Primitive cubes for NAND

fehlerfrei				fehlerhaft	s-a-1(A)			s-a-0(A)			s-a-1(B)			s-a-0(B)		
	A	B	C		A	B	C	A	B	C	A	B	C	A	B	C
β_1	0	X	1	α_1	X	0	1	X	X	1	0	X	1	X	X	1
	X	0	1													
β_0	1	1	0	α_0	X	1	0	-	-	-	1	X	0	-	-	-

Pdcfs for NAND

	s-a-1(A)			s-a-0(A)			s-a-1(B)			s-a-0(B)			s-a-1(C)+			s-a-0(C)++		
	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C
$\beta_0 \cap \alpha_1$		$\emptyset$		D	1	$\overline{D}$		$\emptyset$		1	D	$\overline{D}$	1	1	$\overline{D}$	0	X	D
$\beta_1 \cap \alpha_0$	0*	1	D		$\emptyset$		1	$\overline{D}$	D		$\emptyset$					X	0	D

Propagation D-cubes for NAND

	A	B	C		A	B	C		A	B	C		A	B	C
$\beta_{1/A=1}$	1	0	1	$\beta_{1/A=0}$	0	X	1	$\beta_{1/B=1}$	0	1	1	$\beta_{1/B=0}$	X	0	1
$\beta_{0/A=0}$		$\emptyset$		$\beta_{0/A=1}$	1	1	0	$\beta_{0/B=0}$		$\emptyset$		$\beta_{0/B=1}$	1	1	0
$\bigcap_r$		$\emptyset$			$\overline{D}$	1	D			$\emptyset$			1	D	$\overline{D}$
					D	1	$\overline{D}$						1	$\overline{D}$	D

a	b	c	d	e	f	g	h	i	
X	1				1				
1	X				1				
0	0				0				
		X	1			1			
		1	X			1			
		0	0			0			
					0	X	1		
					X	0	1		
					1	1	0		
				0			X	0	
				X			0	0	
				1			1	1	
					0	1	D		pdcf s-a-1(f)
0	0				0	1	D		Implikation
0	0			1	0	1	D	D	D-drive
0	0	1	X	1	0	1	D	D	line justificati

پوشش نقص
Fault coverage

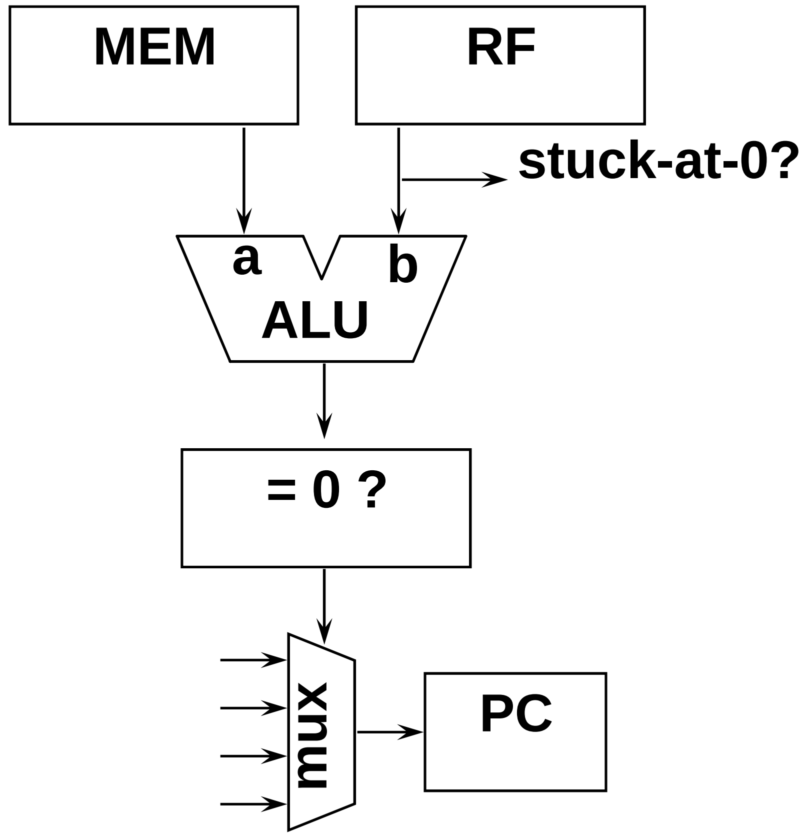
A certain set of test patterns will not always detect all faults that are possible within a fault model

☞

$$coverage = \frac{\text{Number of detectable faults for a given test pattern set}}{\text{Number of faults possible due to the fault model}}$$

For actual designs, the coverage should be at least in the order of 98 to 99%

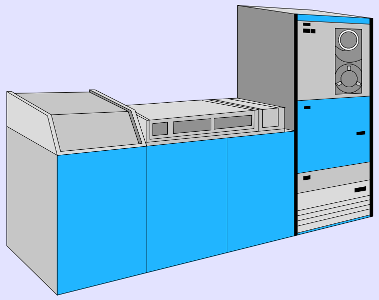
تولید برنامه‌ی خود آزمون
Generation of Self-Test Program Generation - Key concept -



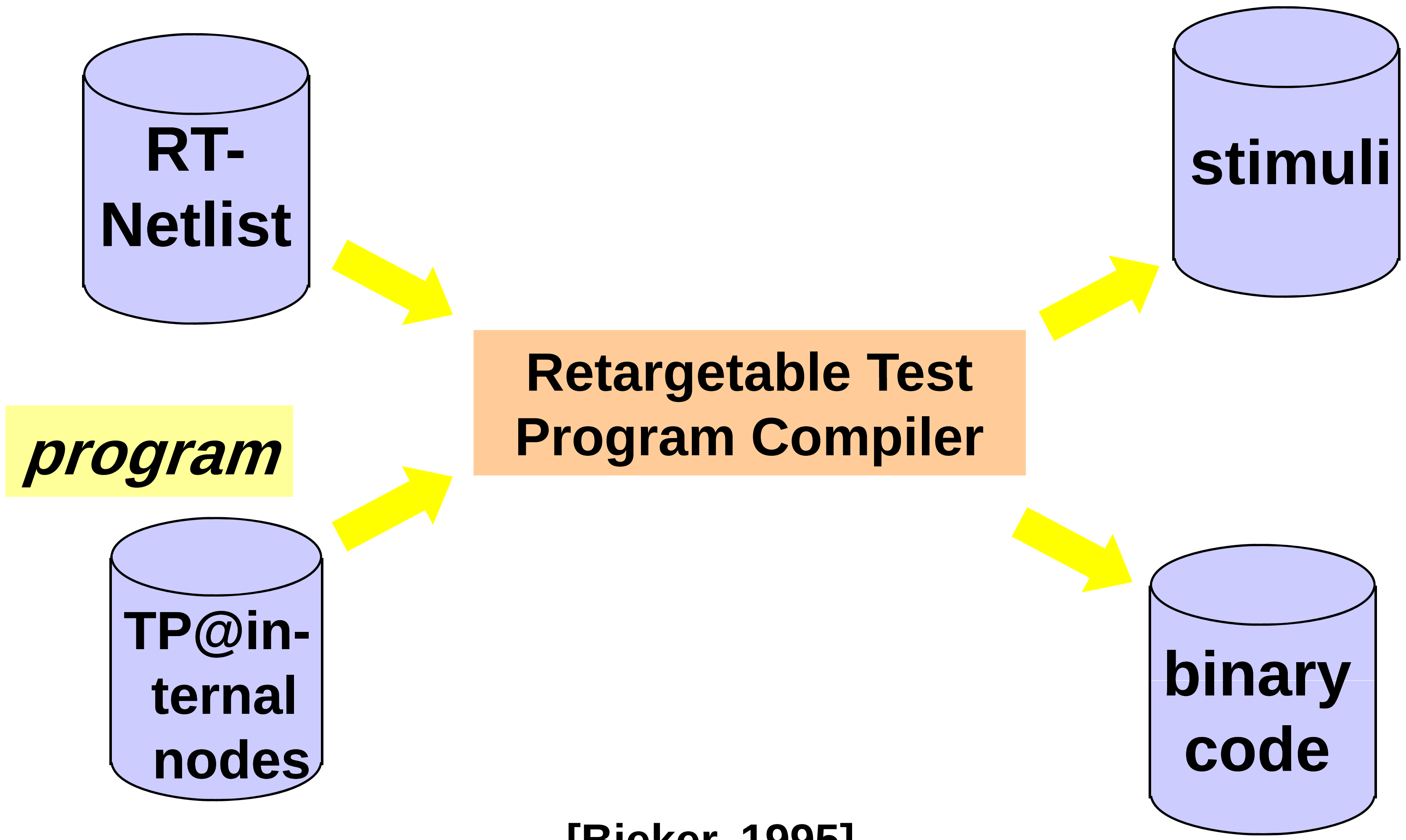
```
RF(0)     := "11...1";
MEM(0) := "11...1";
IF MEM(0) - R(0) <>"00...0"
        THEN Error;
```

تولید برنامه‌ی آزمون
Test Program Generation (2)

- Programs running on the processors to be tested
- Well-known concept (diagnostics @ mainframes)
- Very poor tool support
- Mostly ad-hoc process:
  Initial ad-hoc program;
  Extended until sufficient coverage achieved;
  Extended if undetected errors are reported by field service



Self-Test Programs generated by Retargetable Test Compiler



[Bieker, 1995]

شبیه‌سازی نقص (۱)
Fault simulation (1)

Coverage can be computed with **fault simulation**:

- ∀ faults ∈ fault model: check if distinction between faulty and the fault-free case can be made:

```
Simulate fault-free system;
 ∀ faults ∈ fault model DO
  ∀ test patterns DO
   Simulate faulty system;
   Can the fault be observed for ≥1 pattern?
```

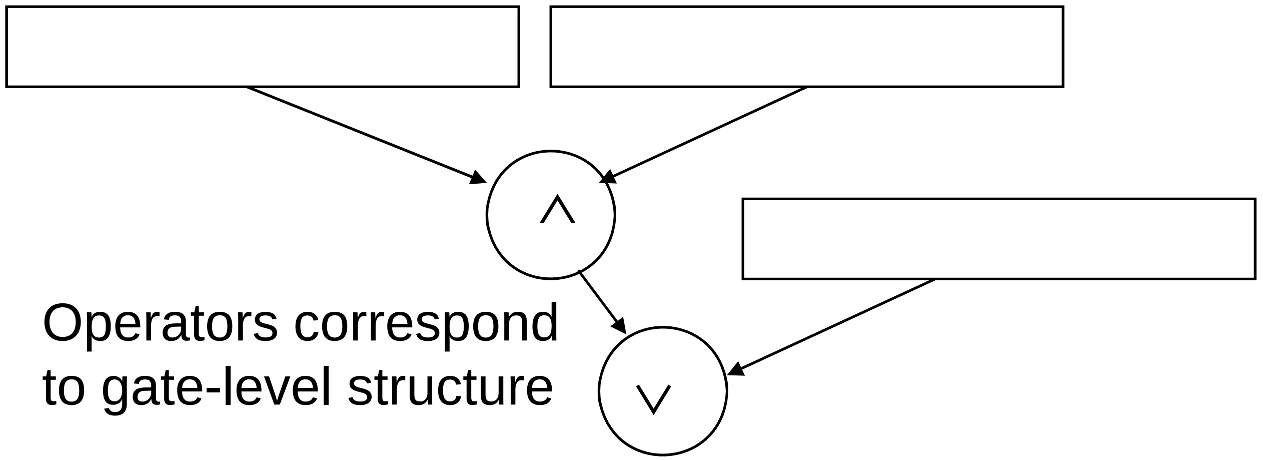
  Faults are called **redundant** if they do not affect the observable behavior of the system,

- Fault simulation checks whether mechanisms for improving fault tolerance actually help.

شبیه‌سازی نقص (۲)
Fault simulation (2)

High computational requirements.
Parallel fault-simulation at the gate level:
Each bit in a word represents a different input pattern.
E.g.: 32 input patterns simulated at the same time.

Each bit corresponds to one test pattern



Operators correspond to gate-level structure

خلاصه
Summary

Validation is the process of checking whether or not a certain (possibly partial) design is appropriate for its purpose, meets all constraints and will perform as expected.

Techniques

- Simulation (used at various steps)
- Test
  - TPG (D-Algorithm, generation of assembly prog., ..)
  - Application of test patterns
  - Checking the results
  - Fault simulation for computing coverage
- Emulation