

Odessa National Polytechnic University



Master Course

CO-DESIGN AND TESTING OF SAFETY-CRITICAL EMBEDDED SYSTEMS

Alexander Drozd

drozd@ukr.net

General course information

1. Object of Study:

Concepts of Safety-Critical Embedded Systems (S-CES):
Co-design and Testing.

2. Prerequisites:

Computer Systems and System Analysis; Foundations of Logic Engineering; Probability Theory; Theory of Self-Checking Circuits; Modeling Foundation knowledge.

3. Subject of Study:

Principles, methods and techniques in co-design and testing of S-CES.

4. Aims:

Acquisition of knowledge about methods and techniques in co-design and testing of S-CES and their components.

Teaching and Learning Time Allocation

#	Module	Lectures	Lab Classes	Private Study
1	Co-design foundation of S-CES	2	0	2
2	Dependability of S-CES and their digital components	4	0	2
3	On-line testing for digital components of S-CES	10	14	12
4	Checkability of S-CES digital components	2	4	2
	Total:	18	18	18

MODULE 1.

Co-design foundation of S-CES

#	Topic of lecture	Lectures	Lab Classes	Private Study
1	Traditional ideas of S-CES co-design	2	0	2
	Total:	2	0	2

MODULE 1. Co-Design Foundation of S-CES

Lecture 1. Traditional ideas of S-CES co-design

1.1. Component approach

1.2. Standards regulating legislative of S-CES

1.3. Life-cycle of S-CES

1.1. Component Approach

Component-based technology is information technology based on **component representation** of systems and on use of well-tested software and hardware products.

COTS-approach (Commercial-Off-The-Shelf) – reuse of **commercial** components.

CrOTS-approach (Critical-Off-The-Shelf) – reuse of components in **critical** applications.

Component approach constitutes the use of **library components** developed formerly and commonly employed in commercial and critical applications, including the components of one's own design.

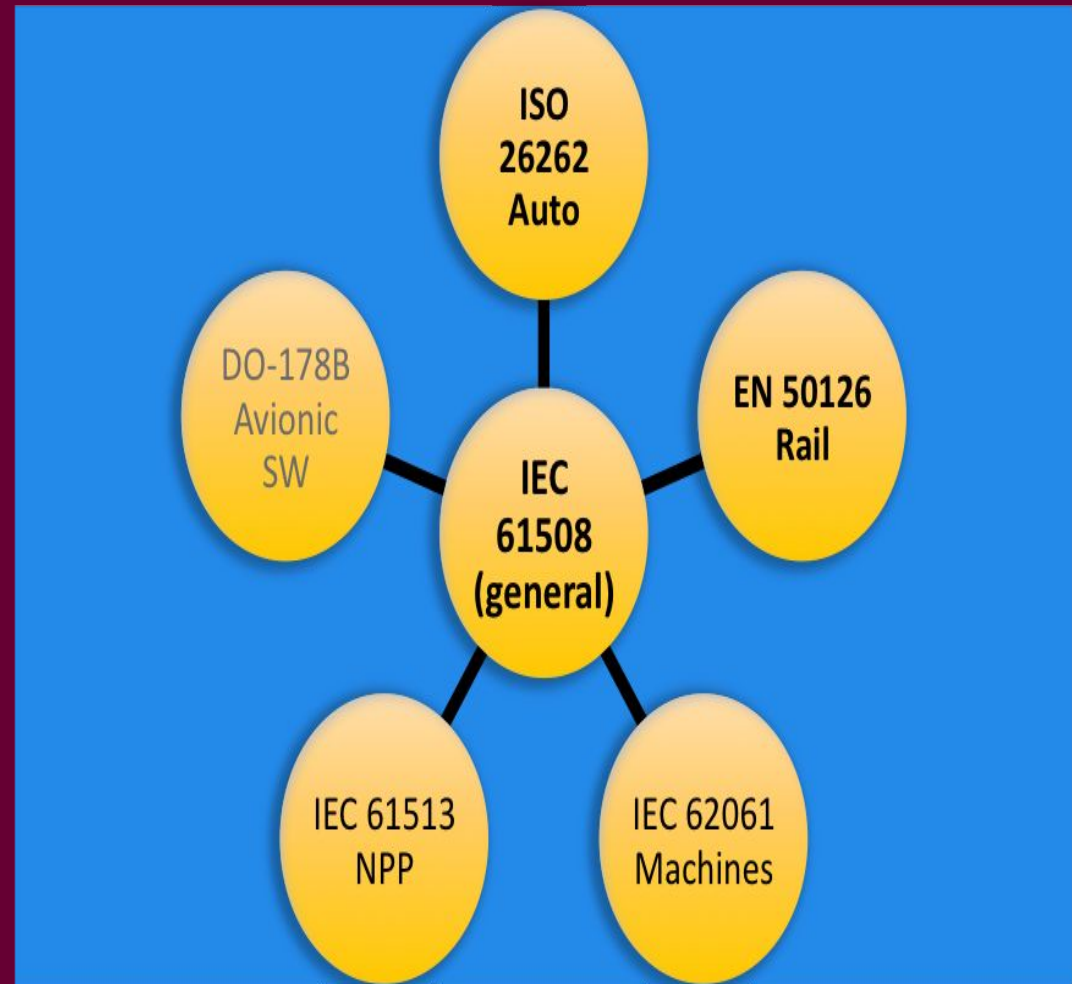
1.2. Standards regulating legislative of S-CES

IEC 61508 (general for electronics & digital)
and
EN 50126 (Railway)

DO 178-B (Avionics)
and
ISO 26262 (Automotive)

IEC 61513
(Nuclear power plants)
and
IEC 62061 (Machines)

IEC – International Electrotechnical Commission



This slide from presentation of
M. Fusani ISTI - CNR, Pisa, Italy

1.2. Standards regulating legislative of S-CES

IEC 61508 – Safety of electrical, electronic and programmable systems important to safety

IEC 61508-1:1998 ‘General requirements’

IEC 61508-2:2000 ‘Requirements to electrical, electronic and programmable systems’

IEC 61508-3:1998 ‘Requirements to software’

IEC 61508-4:1998 ‘Definitions to Abbreviations’

IEC 61508-5:1998 ‘Examples of methods for determining safety integrity levels’

IEC 61508-6:2000 ‘Guide for use of IEC 61508-2 and IEC 61508-3’

IEC 61508-7:2000 ‘Overview of techniques and measures’

1.2. Standards regulating legislative of S-CES

Features of IEC 61508 standard

1. The use of **safety integrity levels** concept – every unit of equipment is developed and analysed with contribution in safety of critical object.
2. Consideration of **full life-cycle of S-CES**
3. Positioning of **software as essential S-CES** component which is source of possible failures influencing on safety of critical object
4. **Flexibility of requirements** for the critical objects. It allows to be foundation for development of standards to specific areas of industry

1.2. Standards regulating legislative of S-CES

IEC 61508 standard as foundation for development of standards to specific areas of industry

ECSS – European Cooperation for Space Standardization

ECSS-E-10 ‘Space Engineering – System Development’

ECSS-E-40A ‘Space Engineering – Software Development’

ECSS-Q-20 ‘Guarantee Production Space Destination – Quality Assurance’

ECSS-Q-80B ‘Guarantee Production Space Destination – Quality Assurance of Software’

1.2. Standards regulating legislative of S-CES

IEC 61508 standard as foundation for development of standards to specific areas of industry

RTCA – Radio Technical Commission for Aeronautics

DO-178B:1992 ‘Consideration of software at certification of on-board systems and equipments’

MIRA – Motor Industry Research Association

MISRA-C:2004 ‘Guide for use of language C++ in critical systems’

CENELEC – European Committee for Electrotechnical Standardization

EN 50126 ‘Objects of railway transport. Requirements and validation of dependability, reliability, maintainability and safety’

1.2. Standards regulating legislative of S-CES

IEC 61508 standard as foundation for development of standards to specific areas of industry

IAEA – International Atomic Energy Agency

IAEA NS-G-1.1 ‘Software and computer-based systems important to safety in nuclear power plants’

IAEA NS-G-1.2 ‘Safety assessment and verification for nuclear power plants’

IAEA NS-G-1.3 ‘Instrumentation and control systems important to safety in nuclear power plants’

1.2. Standards regulating legislative of S-CES

IEC 61508 standard as foundation for development of standards to specific areas of industry

IEC – International Technical Commission

IEC 60780:1998 ‘Nuclear power plants – Electrical equipment of the safety system - Qualification’

IEC 60880:2006 ‘Nuclear power plants – Instrumentation and control systems important to safety – Software aspects for computer-based systems performing category A functions’

IEC 60980:1989 ‘Recommended practices for seismic qualification of electrical equipment of the safety system for nuclear generating stations’

IEC 60987:2007 ‘Nuclear power plants – Instrumentation and control systems important to safety – Hardware design requirements for computer-based systems’

1.2. Standards regulating legislative of S-CES

IEC 61508 standard as foundation for development of standards to specific areas of industry

IEC – International Technical Commission

IEC 61226:2005 ‘Nuclear power plants – Instrumentation and control systems important to safety – Classification of instrumentation and control functions’

IEC 61513:2001 ‘Nuclear power plants – Instrumentation and control systems important to safety – General requirements for systems’

IEC 62138:2004 ‘Nuclear power plants – Instrumentation and control systems important to safety – Software aspects for computer-based systems performing category B or C functions’

IEC 62340:2007 ‘Nuclear power plants – Instrumentation and control systems important to safety – Requirements for coping with common cause failure’

1.3. Life-cycle of S-CES

1. Stages of FPGA-based digital component development

1. Development of signal formation algorithm block-diagram.
2. Development of program models of control algorithms in CASE-tools environment.
3. Integration of signal formation algorithm block-diagram program models in CASE-tools environment.
4. Implementation of integrated digital component program models to FPGA.

CASE – Computer Aided Software / System Engineering

1.3. Life-cycle of S-CES

2. Results of FPGA-based digital component development

1. Block-diagrams according to control algorithms.
2. Program models of control algorithms in CASE-tools environment.
3. Integrated program model of control algorithms in CASE-tools environment.
4. FPGA with implemented integrated program model.

1.3. Life-cycle of S-CES

3. Verification stages of FPGA-based digital component development

1. Verification of block-diagrams according to control algorithms.

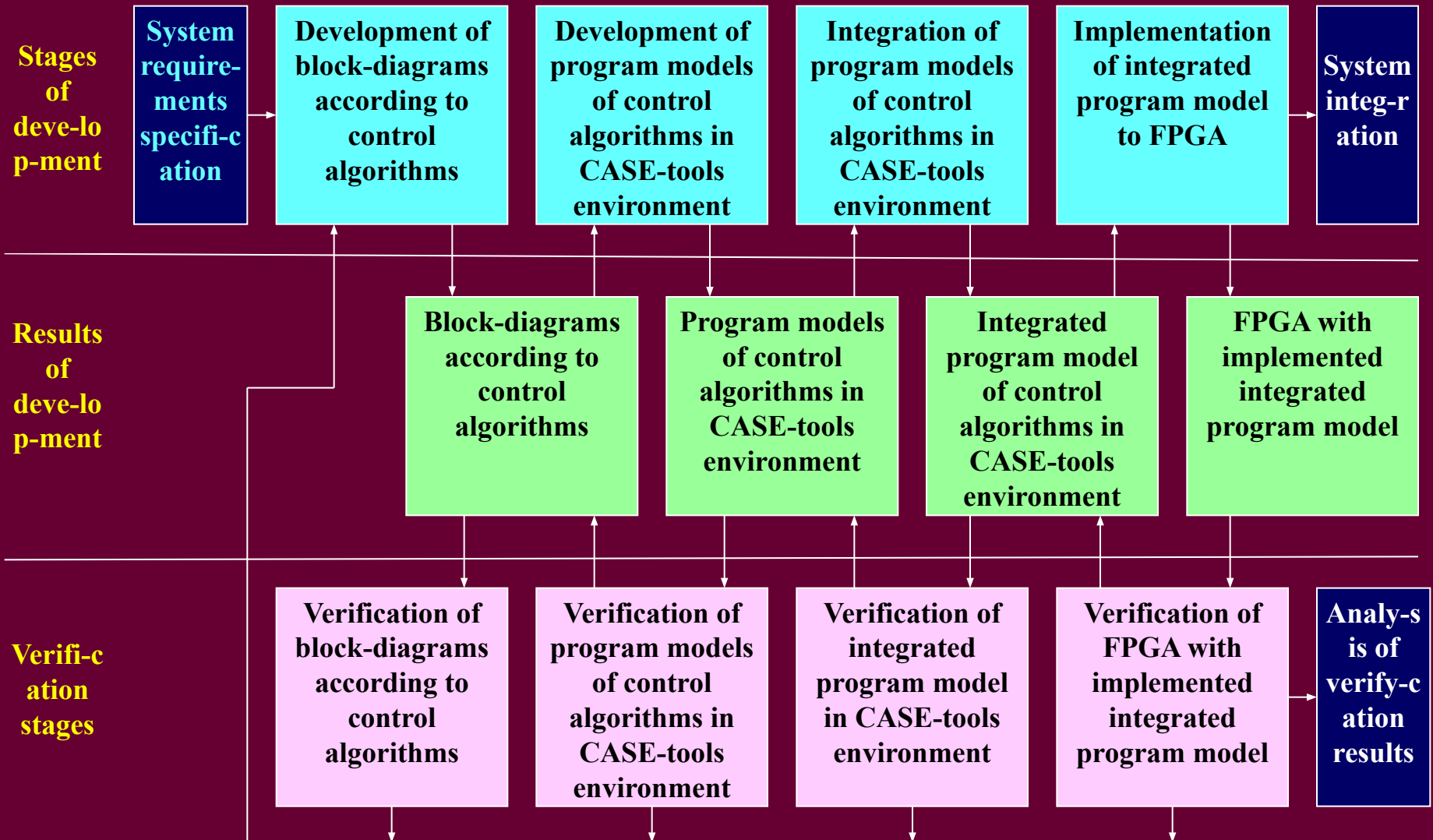
2. Verification of program models of control algorithms in CASE-tools environment.

3. Verification of integrated program model in CASE-tools environment.

4. Verification of FPGA with implemented integrated program model.

1.3. Life-cycle of S-CES

2. A life-cycle of FPGA-based S-CES



Reading List

1. Бахмач Е.С., Герасименко А.Д., Головир В.А. и др. Отказобезопасные информационно-управляющие системы на программируемой логике / Под ред. Харченко В.С. и Скляра В.В. – Национальный аэрокосмический университет «ХАИ», Научно-производственное предприятие «Радий», 2008. – 380 с.

В3 Программные средства и их влияние на надежность и безопасность ИУС, с. 17, 18; 2.1 Обзор нормативных документов в области ИУС критических объектов, с. 55 – 59; 3.3. Жизненный цикл ИУС с программируемой логикой, с. 81 – 86.

2. Kharchenko V.S., Sklyar V.V. FPGA-based NPP Instrumentation and Control Systems: Development and Safety Assessment / Bakhmach E.S., Herasimenko A.D., Golovyr V.A. a.o.. – Research and Production Corporation “Radiy”, National Aerospace University “KhAI”, State Scientific Technical Center on Nuclear and Radiation Safety, 2008. – 188 p.

1.4.1 Problems of ensuring dependability, p. 22, 23; 5.2 Analysis of I&C systems conformity to regulatory safety requirements, p.127 – 133; 2.3.1. Life cycle of FPGA-based Instrumentation and Control Systems, p. 44 – 49.

Conclusion

1. **Co-design of S-CES** is based on traditional ideas such as **Component approach, Standards regulating legislative and Life-cycle of S-CES**
2. **Component approach** constitutes the use of **library components** developed formerly and commonly employed in commercial and critical applications, including the components of one's own design.
3. The main **standard** is **IEC 61508** – Safety of electrical, electronic and programmable systems important to safety.
4. **Life-cycle** of FPGA-based S-CES digital component contains 4 stages of development with verification of results obtained on every stage.

Questions and tasks

1. What is **the S-CES**?
2. What **Traditional ideas** of S-CES co-design do you know?
3. What is **the Component approach**?
4. What **Standards** regulate legislative of S-CES?
5. What **Stages** are contained with **Life-cycle** of FPGA-based S-CES?

MODULE 2.

Dependability of S-CES and their digital components

#	Topic of lecture	Lectures	Lab Classes	Private Study
2	Foundation of Dependability	2	0	1
3	Fault Tolerance of S-CES and their digital components	2	0	1
	Total:	4	0	2

MODULE 2. Dependability of S-CES and their digital components

Lecture 2. Foundation of Dependability

2.1. Introduction into dependability

2.2. Dependability Threats

2.3. Dependability Attributes

2.4. Dependability Measures

2.5. Safety and Reliability

2.6. Forms of Dependability Requirements

2.7. The Means to attain Dependability Techniques

2.1. Introduction into Dependability

2.1.1. Motivation of Dependability Consideration

Increase of requirements to modern computer systems from **Reliability to Dependability.**

Reasons:

Growth of computer system complexity

Expansion of a set of tasks solved with use of computer systems including **critical application areas**

Amplification of interdependence and interaction between hardware and software of computer systems including processes of **co-design S-CES on programmable elements.**

2.1.2. Related Works

Different aspects of **Dependability**, principles of construction and realization of **dependable computer systems** have been studied **for the last two decades**.

1. Avizienis A., Laprie J.-C. Dependable Computing: From Concepts to Application // IEEE Transactions on Computers, 1986. Vol. 74, No. 5. P. 629-638.

Authors formulated the principle of “Dependable Computing” as computation resistant to hardware and software failures (caused by their defects brought in design and not revealed in the course of detected).

2. Dobson I., Randell B. Building Reliable Secure Computing Systems out of Unreliable Insecure components // Proc. of IEEE Conference on Security and Privacy, Oakland, USA. 1986. P. 186-193.

Authors defined “Secure-Fault Tolerance” and proposed a principle of its realization for various types of computer systems.

3. Avizienis A., Laprie J.-C, Randell B., Landwehr C. Basic Concepts and Taxonomy of Dependable and Secure Computing // IEEE Transactions on Dependable and Secure Computing, 2004. Vol. 1. No. 1. P. 11-33.

2.1.3. Definition of Dependability

Dependability is ability to avoid service failures that are more frequent or more severe than is acceptable. When service failures are more frequent or more severe than acceptable: dependability failure.

Attributes - properties expected from the system and according to which assessment of service quality resulting from threats and means opposing to them is conducted.

Means - methods and techniques enabling

- 1) to provide service on which reliance can be placed
- 2) to have confidence in its ability.

Threats - undesired (not unexpected) circumstances causing or resulting from undependability (reliance cannot or will not any longer be placed on the service).

2.2. Dependability Threats

**Dependability Threats - Faults,
Errors,
Failures.**

**Faults: development (design) or operational (phase of creation or occurrence),
internal or external (system boundaries),
hardware or software (domain),
natural or human-made (phenomenological case),
accidental, non-malicious, deliberate or deliberately malicious (intent),
permanent or transient (persistence).**

2.2. Dependability Threats

Faults: Development or Design Faults

Physical Faults

Interaction Faults

Development or Design Faults:

erroneous acts or decisions in system development bring to appearance of a fault in its design which becomes apparent in computer system operation under certain terms and causes an error in computation process, thus leading to a malfunction or failure (non-rendering of service)

- **software flaws,**
- **malicious logics.**

2.2. Dependability Threats

Physical Faults:

due to natural (internal) causes a fault appears bringing to an error in computation process, thus leading to a malfunction or failure.

Interaction Faults:

due to external information, physical or other effects a fault appears bringing to an error in computation process and then a computer system malfunction or failure.

2.2. Dependability Threats

Failures: content, early or late timing, halt or erratic (domain), signaled or unsignaled (detectability), consistent or inconsistent (consistency), minor or catastrophic (consequences).

2.2. Dependability Threats

Fault error failure chain is a way from **correct service** up to **incorrect service**.

Fault

Short-circuit in
memory chip

First written to by program

Fault
activation

Error

Wrong bit value

Read by program, cascade of
erroneous results

Error
propagation

Failure

Erroneous output

This slide from presentation
of Felicita Di Giandomenico
ISTI - CNR, Pisa, Italy

2.3. Dependability Attributes

Readiness for usage – **Availability**.

Continuity of service – **Reliability**.

Absence of catastrophic consequences on the users & env. – **Safety**.

Absence of unauthorized disclosure of inf. – **Confidentiality**.

Absence of improper system alterations – **Integrity**.

Ability to undergo repairs and evolutions – **Maintainability**.

Availability, Confidentiality, Integrity – **Security**.

Absence of unauthorized access to, or handling of, system state.

2.4. Dependability Measures

The alternation of correct-incorrect service delivery is quantified to define the **Measures of Dependability**:

Reliability: a measure of the continuous delivery of correct service – or the time to failure;

Availability: a measure of the delivery of correct service with respect to the alternation of correct and incorrect service;

Maintainability: a measure of the time to service restoration since the last failure occurrence.

2.5. Safety and Reliability

Safety is an extension of **Reliability**:

the state of correct service and the states of incorrect service due to non-catastrophic failure are grouped into **a safe state**:

- **Safety** is a measure of continuous safeness, or equivalently, of the time to catastrophic failure;
- **Safety** is thus **Reliability** with respect to catastrophic failures.

2.6. Forms of Dependability Requirements

Availability: – “The database must be accessible 99% of the time”

Rate of occurrence of failures: – “the probability that a failure of a flight control system will cause an accident with fatalities or loss of aircraft must be less than 10^{-9} per hour of flight”.

Probability of surviving mission: – The probability that the flight and ordnance control system in a fighter plane are still operational at the end of a two hour mission must be more than...

Other forms of requirements:

- **Fault tolerance:** this system must provide uninterrupted service with up to one component failure, and fail safely if two fail;
- **Specific defensive mechanisms:** “these data shall be held in duplicate on two disks.”

2.7. The Means to attain Dependability Techniques

The development of a **Dependable Computing System** calls for the combined utilization of a set of **four techniques**:

- **Fault prevention**: how to prevent the occurrence or introduction of faults;
- **Fault removal**: how to reduce the number or severity of faults;
- **Fault forecasting**: how to estimate the present number, the future incidence and the likely consequences of faults.
- **Fault tolerance**: how to deliver correct service in the presence of faults.

2.7.1. Fault Prevention

Fault Prevention is attained by *quality control techniques* employed during the design and manufacturing of hardware and software:

- They include *structured programming, information hiding, modularization, etc.*, for software, and *rigorous design rules and selection of high-quality, mass-manufactured hardware components* for hardware.
- *Simple design*, possibly at the cost of constraining functionality or increasing cost
- *Formal proof* of important properties of the design
- Provision of *appropriate operating environment* (air conditioning, protection against mechanical damage) intend to prevent operational physical faults, while training, rigorous procedures for *maintenance, 'foolproof' packages*, intend to prevent interaction faults.

2.7.2. Fault Removal

Fault Removal is performed both during the development, and during the operational life of a system.

- During development it consists of three steps: **verification**, **diagnosis**, **correction**.
- **Verification** is the process of checking whether the system adheres to given properties. If it does not, the other two steps follow:
 - After correction, **verification should be repeated** to check that fault removal had no undesired consequences; the verification performed at this stage is usually termed non-regression verification.
 - **Checking** the specification is usually referred to as **validation**.

2.7.2.1. Fault Removal during Development

Verification Techniques can be classified according to whether or not they **exercise** the system.

- Without actual execution is **static verification**:
static analysis (e.g., inspections or walk-through),
model-checking, theorem proving.
- Exercising the system is **dynamic verification**: either with **symbolic inputs** in the case of **symbolic execution**, or **actual inputs** in the case of **testing.**
- Important is the verification of **fault tolerance mechanisms**, especially a) **formal static verification**, and b) **testing** that includes faults or errors in the test patterns: **fault injection.**
- As well as **verifying** that the system cannot do more than what is specified important to **safety** and **security.**

2.7.2.2. Fault Removal during the Operational Life

Fault Removal during the operational life of a system is **corrective** or **preventive maintenance**.

- **Corrective maintenance** is aimed at **removing faults** that have produced one or more errors and have been reported.
- **Preventive maintenance** is aimed to uncover and remove faults before they might cause errors during normal operation.
 - a) **physical faults** that have occurred since the last preventive maintenance actions;
 - b) **design faults** that have led to errors in other similar systems.
- These forms of **maintenance** apply to non-fault-tolerant systems as well as fault-tolerant systems, that can be maintainable **on-line** (without interrupting service delivery) or **off-line** (during service outage).

2.7.3. Fault Forecasting

Fault Forecasting is conducted by performing an evaluation of the system behavior with respect to fault occurrence or activation.

- **Qualitative Evaluation**: aims to identify, classify, rank the failure modes, or the event combinations (component failures or environmental conditions) that would lead to system failures.
- **Qualitative Evaluation** or probabilistic: which aims to evaluate in terms of probabilities the extent to which the relevant attributes of dependability are satisfied.
 - Through either **specific methods** (e.g., FMEA for qualitative evaluation, or Markov chains and stochastic Petri nets for quantitative evaluation).
 - **Methods** applicable to both forms of evaluation (e.g., reliability block diagrams, fault-trees).

Reading List

1. Бахмач Е.С., Герасименко А.Д., Головир В.А. и др. Отказобезопасные информационно-управляющие системы на программируемой логике / Под ред. Харченко В.С. и Скляра В.В. – Национальный аэрокосмический университет «ХАИ», Научно-производственное предприятие «Радий», 2008. – 380 с.
1.2 Гарантоспособность и ее свойства, с. 29 – 36;
1.4.2 Отказоустойчивость и отказобезопасность, с. 42 – 45.
2. Kharchenko V.S., Sklyar V.V. FPGA-based NPP Instrumentation and Control Systems: Development and Safety Assessment / Bakhmach E.S., Herasimenko A.D., Golovyr V.A. a.o.. – Research and Production Corporation “Radiy”, National Aerospace University “KhAI”, State Scientific Technical Center on Nuclear and Radiation Safety, 2008. – 188 p.
1.2 Dependability and its attributes, p. 16 – 34.
3. Avizienis A., Laprie J.-C, Randell B., Landwehr C. Basic Concepts and Taxonomy of Dependable and Secure Computing // IEEE Transactions on Dependable and Secure Computing, 2004. Vol. 1. No. 1. P. 11- 33.

Conclusion

1. **Dependability** integrates a set of attributes, such as **Availability, Reliability, Safety, Confidentiality, Integrity and Maintainability**.
2. **Dependability threats** consist of **Faults, Errors and Failures**.
3. **Measures of Dependability** are defined using **Reliability, Availability and Maintainability**
4. **Safety** can be considered as an extension of **reliability**
5. **Means** to attain Dependability contain 4 Techniques: **Prevention, Removal, Forecasting and Tolerance of Faults**.
6. **Evolution of the Dependability concept: Resilience, Survivability and Trustworthiness (Reliability of Results)**.

Questions and tasks

1. What is **the Dependability**?
2. What **Dependability threats** of S-CES do you know?
3. What kinds of faults do you know?
4. Define essence of **Availability, Reliability, Safety, Confidentiality, Integrity and Maintainability**.
5. What **Components of Security** do you know?
6. What **Measures of Dependability** do you know?
7. What **Techniques** are contained with Means to attain Dependability?
8. Define essence of **Prevention, Removal, Forecasting and Tolerance of Faults**.

MODULE 2. Dependability of S-CES and their digital components

Lecture 3. Fault Tolerance of S-CES and their digital components

3.1. Introduction into Fault Tolerance

3.2. Error Detection

3.3. Recovery

3.4. Dependability Measures

3.5. Fault Tolerant Technologies

3.1. Introduction into Fault Tolerance

3.1.1. Motivation of Fault Tolerance Consideration

Fault Tolerance is a base of any S-CES and their components.

Reasons:

Fault Tolerance is the main mechanism, instrument ensuring **Dependability**

Fault Tolerance ensures **operative resistance to hardware and software failures**

3.1.2. Related Works

1. Dobson I., Randell B. Building Reliable Secure Computing Systems out of Unreliable Insecure components // Proc. of IEEE Conference on Security and Privacy, Oakland, USA. 1986. P. 186-193.

Authors defined “Secure-Fault Tolerance” and proposed a principle of its realization for various types of computer systems.

2. Jean-Claude Laprie, Jean Arlat, Christian Beounes, Karama Kanoun and Catherine Hourtolle, Hardware and Software Fault Tolerance: Denition and Analysis of Architectural Solutions, in Proceedings FTCS 17, 1987

3. Lee P.A. and Anderson T., Fault Tolerance - Principles and Practice, second edition, Springer Verlag/Wien, 1990

3.1.3. Definition of Fault Tolerance

Fault Tolerance is intended to preserve the delivery of **correct service** in the presence of **active faults**.

Fault Tolerance:

- **Error Detection**
- **Recovery**

Effectiveness of Fault Tolerance: the effectiveness of **error** and **fault handling** mechanisms (their coverage) has a strong influence on **Dependability Measures**

3.2. Error Detection

Error Detection defines the presence of an error.

Fault Tolerance is generally implemented by **error detection** and **subsequent system recovery**.

Error detection originates an **error signal** or **message** within the system. An error that is present but not detected is **a latent error**.

There exist two classes of **error detection techniques**:

- **concurrent error detection**, which takes place **during service delivery**,
- **preemptive error detection**, which takes place while **service delivery is suspended**; it checks the system for **latent errors** and **dormant faults**.

3.3. Recovery

System Recovery transforms a system state that contains one or more errors and (possibly) faults into a state without detected errors and faults that can be activated again.

Recovery consists of

- **Error Handling**
- **Fault Handling (Fault treatment).**

3.3.1. Error Handling

Error Handling eliminates errors from the system state.

Error Handling may take three forms:

- **Rollback:** the state transformation consists of returning the system back to a saved state that existed prior to error detection; that saved state is a checkpoint;
- **Compensation:** the erroneous state contains enough redundancy to enable error elimination;
- **Rollforward:** the state without detected errors is a new state.

3.3.2. Fault Handling

Fault Handling prevents located faults from being activated again.

Fault Handling involves four steps:

- **Fault Diagnosis**: identifies and records the cause(s) of error(s), in terms of both location and type;
- **Fault Isolation**: performs physical or logical exclusion of the faulty components from further participation in service delivery, i.e., it makes the fault dormant;
- **System Reconfiguration**: either switches in spare components or reassigns tasks among non-failed components;
- **System Reinitialization**: checks, updates and records the new configuration and updates system tables and records.

3.4. Fault-Tolerant Technologies

Fault-Tolerant Technologies traditionally used in co-design of S-CES:

- Use of **Detecting** and **Correcting** codes.
- **Majority Structures.**
- **Multi-Version Systems.**

Fault-Tolerant Technologies based on various kinds of **Redundancy** and **Reconfiguration.**

Operative nature of the opposition to faults in safety-critical I&CS determines the important role of **the methods and means of On-Line Testing** in maintenance of **Fault Tolerance.**

3.4.1 Use of Detecting and Correcting codes

3.4.1.1. Residue Checking for Error Detection in arithmetic components

Residue check equations:

$KA + KB = KS$ for an operation of addition $A + B = S$

$KA \cdot KB = KV$ for an operation of multiplication $A \cdot B = V$

$KB \cdot KC + KD = KA$ for an operation of division $A/B,$

$C = A \text{ div } B, D = A \text{ mod } B,$

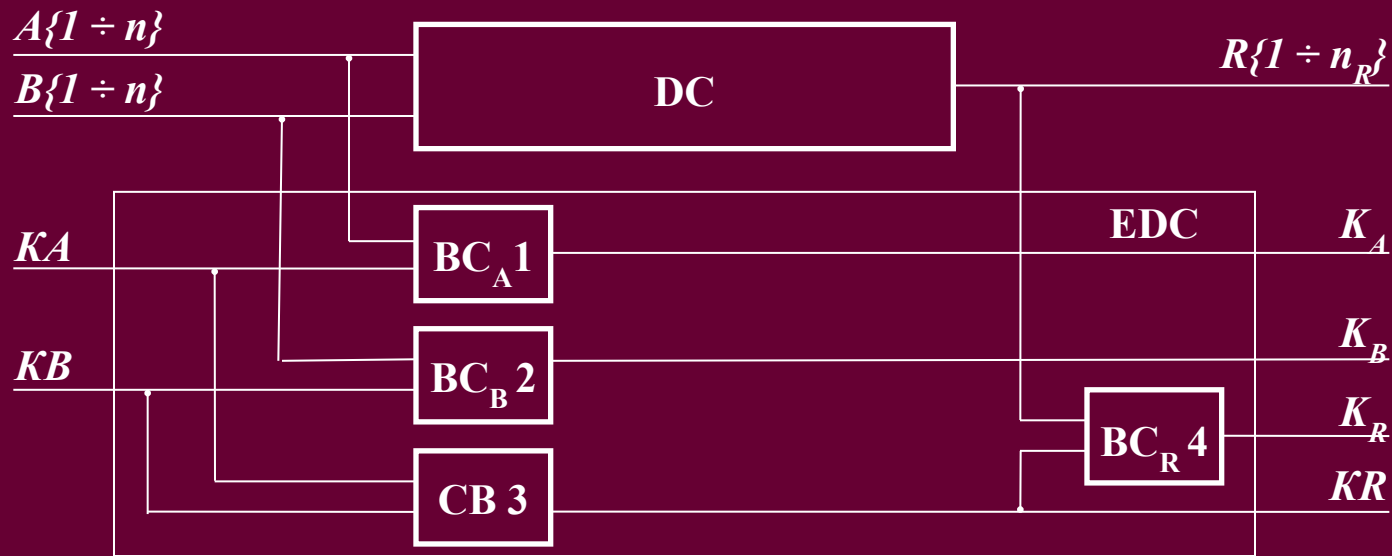
where KA, KB, KS, KV, KC, KD – residue check codes

by modulo $m,$

$KA = A \text{ mod } m, KB = B \text{ mod } m, KS = S \text{ mod } m,$

$KV = V \text{ mod } m, KC = C \text{ mod } m, KD = D \text{ mod } m.$

3.4.1.1. Residue Checking for Error Detection in arithmetic components



Blocks BC_A and BC_B check the operands A and B by computing the check codes KA and KB and also comparing them with the input check codes KA and KB . Results of comparison are the error indication codes K_A and K_B .

Block CB calculates the check code KR of the result R ($R = S$ for addition and $R = V$ for multiplication).

Block BC_R checks the result R comparing its by modulo with the check code KR

3.4.1.2. Hamming Correcting Code for Memory Recover

Generating Matrix of linear code

<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>K3</i>	<i>K2</i>	<i>K1</i>
1							0	0	1
	1						0	1	0
		1					0	1	1
			1				1	0	0
				1			1	0	1
					1		1	1	0
						1	1	1	1

Code *K3 K2 K1* defines number of an erroneous bit *1, 2, 3, 4, 5, 6* or *7*.

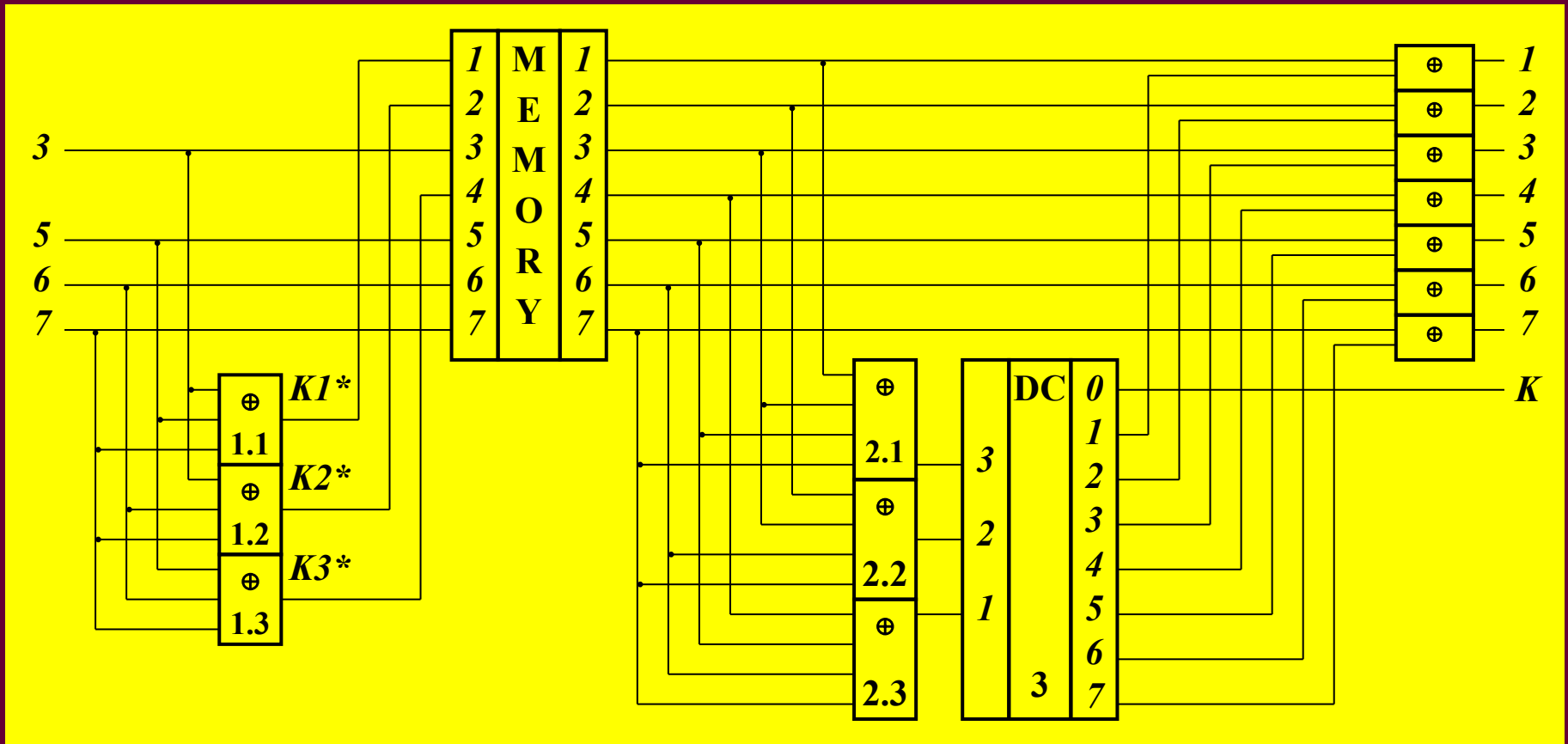
$K1 = 1 \oplus 3 \oplus 5 \oplus 7$ Both the bit *1* and check bit *k1* have number *1*

$K2 = 2 \oplus 3 \oplus 6 \oplus 7$ Both the bit *2* and check bit *k2* have number *2*

$K3 = 4 \oplus 5 \oplus 6 \oplus 7$ Both the bit *4* and check bit *k3* have number *4*

For unique defining a number of the erroneous bit, the bits *1, 2* and *4* are eliminated: $K1^* = 3 \oplus 5 \oplus 7$, $K2^* = 3 \oplus 6 \oplus 7$, $K1^* = 5 \oplus 6 \oplus 7$.

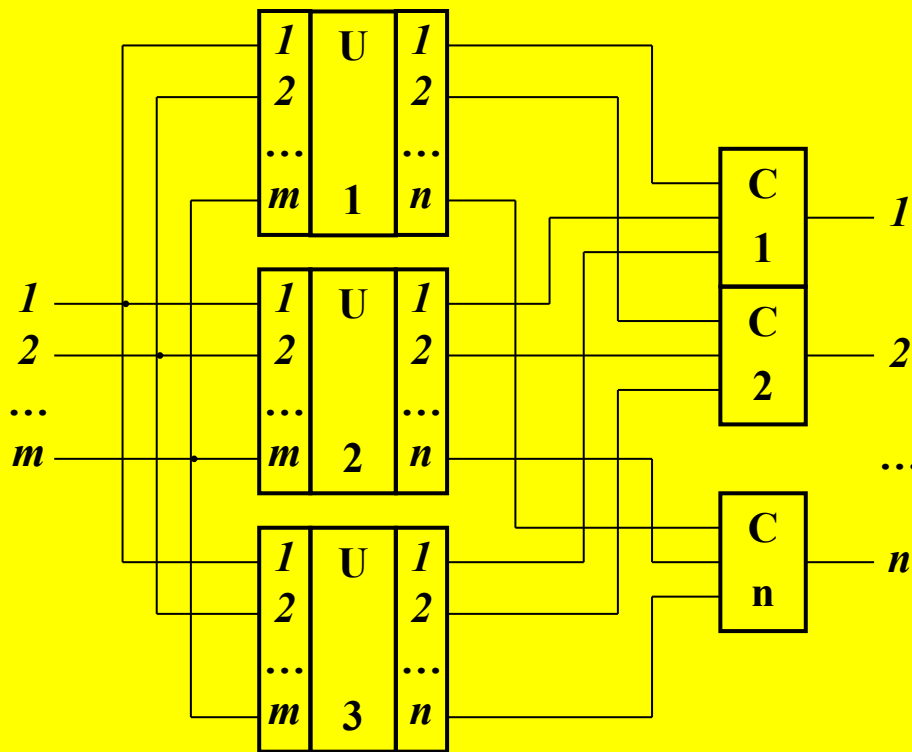
3.4.1.2. Hamming Correcting Code for Memory Recover



Circuit for Memory Recover using Hamming Correcting Code

3.4.2. Majority Structures

Majority structure can be obtained using correcting code



Majority circuit

Generating Matrix of correcting code for Majority Structures

	1				
1	2	...	n	1	2
1				1	
	1				1
		1			
			1		

Majority element calculates carry function of full adder $C = 12 \vee 13 \vee 23$

The errors caused by input faults are not detected

3.4.3. Multi-Version Systems

Multi-Version System (MVS) contains more than one **version** for solving a computing task.

The version is defined as a method of system function realization. For embedded systems it can be hardware means to solve a computing task.

Multi-Version System are aimed to provide protection against **failure due to common reason:**

- Errors of design;
- Physical Defects of Manufactory;
- Faults during Operation.

3.4.3. Multi-Version Systems

Multi-Version System based on Diversity (Multi-Versity or Version Redundancy).

Diversity means **a type of redundancy** based on introduction of two or more versions.

In regulatory documents the application of **Version Redundancy** goes under the name of “**Principle of diversity**”

Nuclear engineering uses a class of MVS including two versions in accordance with international standards, such as:

- IEC 61513:2001** ‘Nuclear power plants – Instrumentation and control systems important to safety – General requirements for systems’
- IEC 62340:2007** ‘Nuclear power plants – Instrumentation and control systems important to safety – Requirements for coping with common cause failure’

3.4.3. Multi-Version Systems

A two-version system W is described by quintuple:

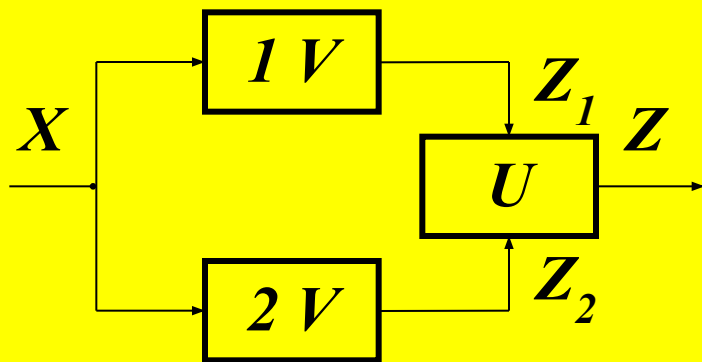
$$W = \{X, F, Z, V, U\},$$

where X and Z – input and output signals;

F – set of functions performed;

V – two-element set of versions v_1, v_2 with outputs U_1, U_2 ;

U – function of version execution results processing
(representations of Z_1, Z_2 in Z).



A Structure of two-version S-CES

Control signal Z (system output) is generated by solver in accordance with outputs of versions Z_1 and Z_2 .

The solver may be realized as OR circuit if faulty version defines its output in 'zero' value.

3.4.3. Multi-Version Systems

A Classification of Diversity Types

Software diversity is the use of different programs designed and implemented by different development groups with different programming languages and tools to accomplish the same safety goals.

Equipment (hardware) diversity is the use of different equipment to perform similar safety functions in which different means sufficiently unlike as to significantly decrease vulnerability to common failure.

Human (life cycle) diversity is the use of different project groups with different key personnel to accomplish the same project goals.

3.4.3. Multi-Version Systems

A Classification of Diversity Types

Design diversity is the use of different approaches including both software and hardware, to solve the same or similar problem.

Functional diversity is the use of different physical functions performing though they may have overlapping safety effects.

Signal diversity is the use of different sensed parameters to initiate protective action, In which any of parameters may independently indicate in abnormal condition, even if the other parameters fail to be sensed correctly.

3.4.3. Multi-Version Systems

Diversity types in FPGA-based S-CES

Diversity type	Way of diversity implementation
Diversity of electronic elements	Diversity of firm developers of electronic elements
	Diversity of technologies of electronic elements producing
	Diversity of electronic elements families
	Diversity of electronic elements from the same family
Diversity of CASE-tools	Diversity of developers of CASE-tools
	Diversity of CASE-tools
	Diversity of configuration of CASE-tools
Diversity of projects development languages	Diversity on the base of graphical language and hardware description language
	Diversity of hardware description languages
Diversity of specification	Diversity of specification languages

3.4.4. Multi-Version Systems

Two-version system is considered as simplest MVS. It has only two independent versions. And requirement of independent versions is used for each two versions of MVS.

That's why complexity of MVS is increased with growing amount of versions. And this complexity is the main limitation of multi-version technology development.

We offer a new set of MVS with strongly connected versions (SVS), which protects against failure due common reason having maximal common part of versions.

We revise requirement to undependability of versions and show that only common part of all versions should be absent for protecting against failure due common reason:

$$A_1 \cap \dots \cap A_i \cap \dots \cap A_N = \emptyset. \quad (1)$$

3.4.5. Computer Systems with Strongly Connected Versions

Computer Systems with Strongly Connected Versions is **MVS** for which exception of means for performance of any one version excludes opportunities of performance of any other version.

Let's designate addition to version A_i as

$$\square A_i = A \setminus A_i.$$

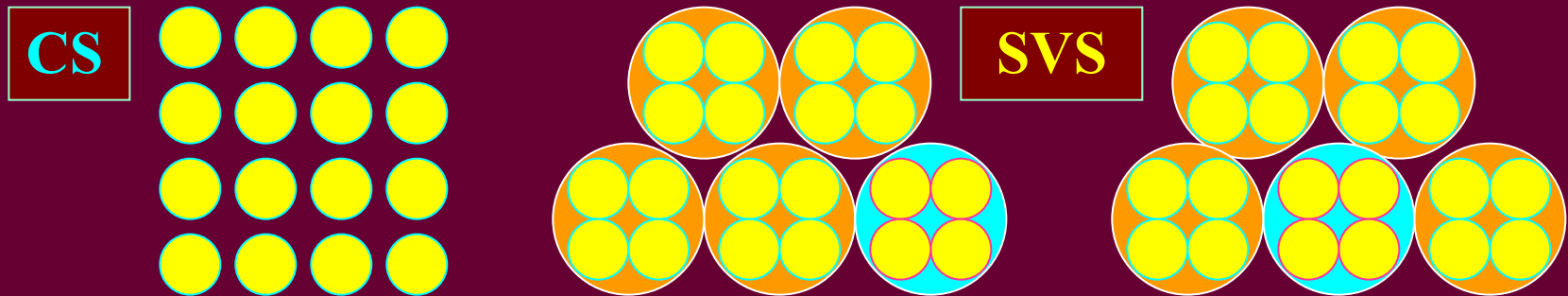
Then the determining attribute of **SVS** is that additions to versions do not include versions,

i.e. for $i = 1 \div N$ and $j = 1 \div N$
is carried out $\square A_i \not\subset A_j$ (2)

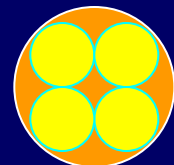
3.4.5. Computer Systems with Strongly Connected Versions

Structure of SVS

Basis for SVS creation are CS that have a modular structure using sets of **identical elements** ●



Identical elements of initial CS are united in **identical sections**

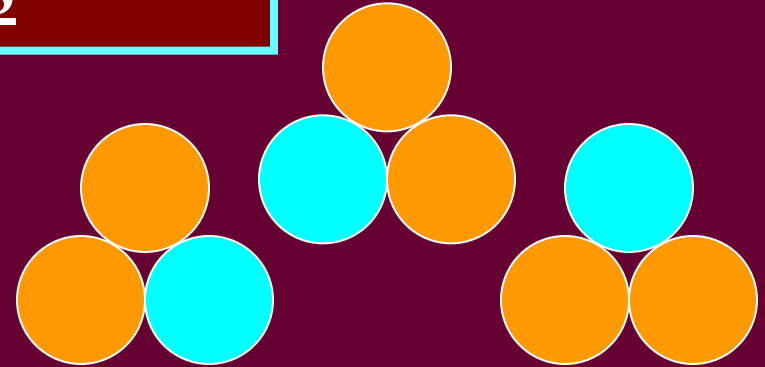


The amount of additional sections in SVS is less than the amount of sections in a version.

3.4.5. Computer Systems with Strongly Connected Versions

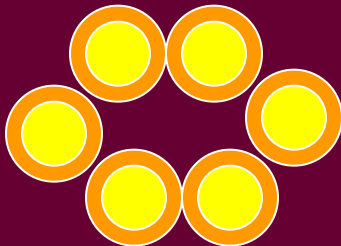
Structure of SVS

A minimum quantity of versions in a SVS is **three**

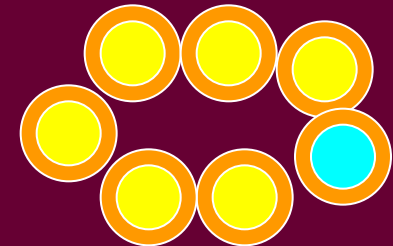


A maximum quantity of versions in a SVS is achieved in case **the section has one element:**

CS



SVS



SVS is **simplified** with **increase** of versions quantity

3.4.5. Computer Systems with Strongly Connected Versions

Protection from Failure due to the Common Reason

The SVS becomes protected from failure due to the common reason using two components:

- the multitude of versions, that contains at least one **true version**;
- means of a choice of the **true version**.

3.4.5. Computer Systems with Strongly Connected Versions

Complexity of SVS

Complexity of SVS

$$Q_{SVS} = Q_{IE} + Q_{CM},$$

where Q_{IE} – complexity of identical elements;
 Q_{CM} – complexity of choice means.

$$Q_{IE} = R + R / K,$$

$$Q_{CM} = (K + 1) \lambda,$$

where R – quantity of identical elements in CS;
 K – quantity of identical elements in CS; λ –
coefficient of proportionality.

$$Q_{SVS \text{ MIN}} = R (1 + 1/K)^2,$$

$$Q_{DC \text{ MIN}} = 2R (1 + 1/K^2).$$

$$K = \sqrt{R/\lambda}$$

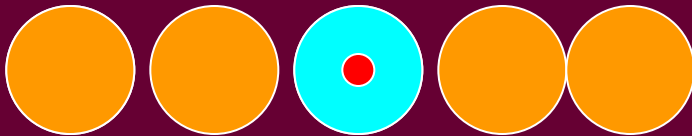
$$Q_{DC \text{ MIN}} / Q_{SVS \text{ MIN}} = 2(1 - 2K / (K + 1)^2).$$

3.4.5. Computer Systems with Strongly Connected Versions

Choice of the True Version

The SVS can be realized with:

- a parallel choice of the true version;



- a consecutive choice of the true version.



3.4.5. Computer Systems with Strongly Connected Versions

Choice of the True Version

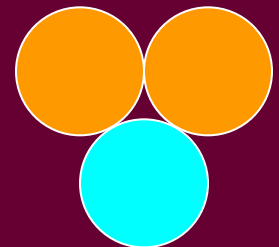
Choice of the true version is executed by the **on-line testing methods** using means of hardware check

The version can be checked up using two approaches.

- **external**, i.e. check of total system;
- **internal**, i.e. check of each version by its own means.

The check of the version can be:

- **direct**, which estimates the version itself;
- **indirect**, which estimates its addition.



3.4.5. Computer Systems with Strongly Connected Versions

Choice of the True Version

A **parallel choice** of the true version is realized by the **internal** check of versions.

Direct check puts the true version into operation

Indirect check disconnects the incorrect addition of the true version.

A **consecutive choice** of the true version is based on **external check** of versions.

Change of versions is carried out before detection of the true version.

Reading List

1. Бахмач Е.С., Герасименко А.Д., Головир В.А. и др. Отказобезопасные информационно-управляющие системы на программируемой логике / Под ред. Харченко В.С. и Скляра В.В. – Национальный аэрокосмический университет «ХАИ», Научно-производственное предприятие «Радий», 2008. – 380 с.
1.4.3 Принцип диверсности (многоверсионности), с. 45 – 47;
8.5 Жизненный цикл многоверсионных ИУС, с. 119 – 224.
2. Kharchenko V.S., Sklyar V.V. FPGA-based NPP Instrumentation and Control Systems: Development and Safety Assessment / Bakhmach E.S., Herasimenko A.D., Golovyr V.A. a.o.. – Research and Production Corporation “Rادیy”, National Aerospace University “KhAI”, State Scientific Technical Center on Nuclear and Radiation Safety, 2008. – 188 p.
4.1 General concepts of multi-version system theory, p. 70 – 71.
4.1 Diversity types in FPGA-based I&C systems, p. 71 – 74.
3. Monographs of System Dependability. Dependability of Networks. – Wrocław, Poland. – 2010. – 210 p.
3. Multi-version computer systems with use of strongly connected versions, p. 39 – 50.

Conclusion

1. **Fault Tolerance** is a base of any S-CES and their components ensuring **Dependability**.
2. **Fault Tolerance** of S-CES is executed by **Error Detection and Recovery**.
3. **Recovery** consists of **Error Handling** (rollback, compensation, rollforward) and **Fault Treatment** (Fault diagnosis and isolation, System reconfiguration and reinitialization).
4. **Fault-Tolerant Technologies** based on various kinds of **Redundancy** and **Reconfiguration** using the methods and means of **On-Line Testing**.
5. **Multi-Version System** ensures **resistance to failure due to common reason**.
6. **Computer Systems with Strongly Connected Versions** is **simplified with increase of versions quantity**.

Questions and tasks

1. What is **the Fault Tolerance**?
2. What kinds of **the Fault Tolerance** do you know?
3. Recite the **Error detection techniques**.
4. What forms of **Error Handling** and **Fault Treatment** do you know?
5. What property of **On-Line Testing** is essential for **Fault-Tolerant Technologies**?
6. What is it “**Principle of diversity**”?
7. What types of **Diversity** do you know?
8. Define essence of **Computer Systems with Strongly Connected Versions**.

MODULE 3.

On-line testing for digital component of S-CES

#	Topic of lecture	Lectures	Lab Classes	Private Study
4	Processing and checking of exact data	2	0	2
5	Approximate data processing	2	0	2
6	Reliability of on-line testing methods	2	4	4
7	Increase of on-line testing methods reliability	2	2	2
8	Checking by logarithm, inequalities, segments	2	8	2
	Total:	10	14	12

MODULE 3. On-line testing for digital components of S-CES

Lecture 4. Processing and checking of exact data

4.1. Introduction into on-line testing

4.2. Stages of on-line testing development

4.3. Self-checking circuits

4.4. Purpose of on-line testing

4.5. Model of exact data

4.6. Processing of exact and approximate data

4.7. Component on-line testing

4.1. Introduction into On-Line Testing

4.1.1. Motivation of On-Line Testing Consideration

On-Line Testing is a base of any S-CES and their components.

Reasons:

On-Line Testing is aimed to ensure reliability of the calculated results

On-Line Testing ensures **first response** to hardware and software failures

4.1.2. Related Works

1. Metra C., Favalli M. and Ricco B. Concurrent Checking of clock signal correctness // IEEE Design & Test October 1998, P. 42 – 48.

2. Toubia N. A. and McCluskey E. J. Logic synthesis techniques for reduced area implementation of multilevel circuits with concurrent error detection // Proc. IEEE Inf. Conf. on Computer Aided Design. – 1994. – P. 651 – 654.

3. Metra C., Schiano L., Favalli M and Ricco B. Self-checking scheme for the on-line testing of power supply noise. – Proc. Design, Automation and Test in Europe Conf. Paris (France). – 2002. – P. 832 – 836.

4. Nicolaidis M. and Zorian Y. On-line testing for VLSI – a compendium of approaches // Electronic Testing: Theory and Application (JETTA). – 1998. – V. 12. – P. 7 – 20.

5. Горяшко А. П. Синтез диагностируемых схем вычислительных устройств. – М.: Наука, 1987. – 288 с.

4.1.3. Definition of On-Line Testing

On-line testing is considered to be the check of digital circuit operation correctness over working influences.

It has many names:

- *concurrent checking* [1], *concurrent error detection* [2], executing an error detection simultaneously with work of the digital circuit (DC);
- *on-line testing* operatively estimating a technical condition of DC [3];
- *hardware check* in accordance with its hardware realization as against to **program** one [4];
- *built-in check* as opposed to the **remote check** taking into account inseparable connection with circuit [5].

4.2. Stages of On-Line Testing Development

In development of on-line testing it is possible to select three stages:

- the initial stage;
- stage of becoming – the development stage of self-checking circuits which expand the on-line testing for own means within the framework of the exact data processing;
- the present stage expanding the on-line testing for processing of the approximate data.

4.2.1. Initial Stage of On-Line Testing Development

- Data transmission on distance

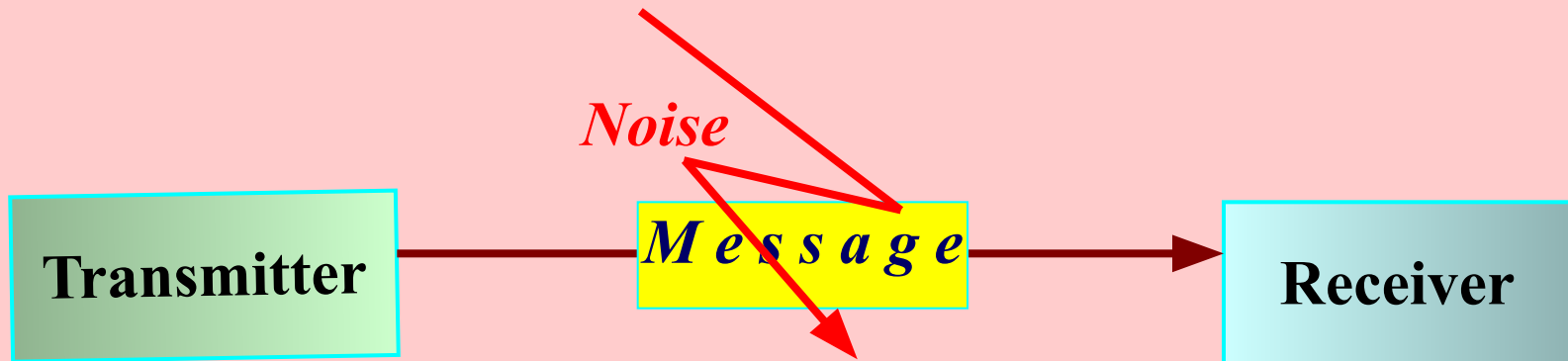
The basis of the theory and practice of on-line testing of computer systems was made with achievements in the field of noiseless data transmission on distance.



4.2.1. Initial Stage of On-Line Testing Development

- Data transmission on distance

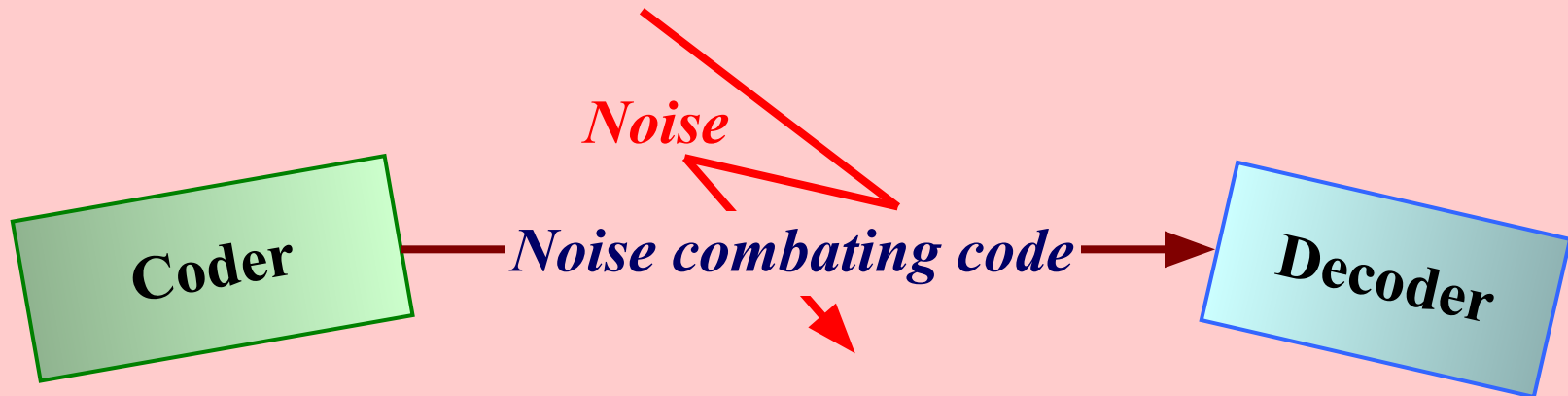
The noises on air deformed transmitted messages.



4.2.1. Initial Stage of On-Line Testing Development

- Data transmission on distance

To transfer correct message the redundant coding the data with help of correcting or detecting codes was used.

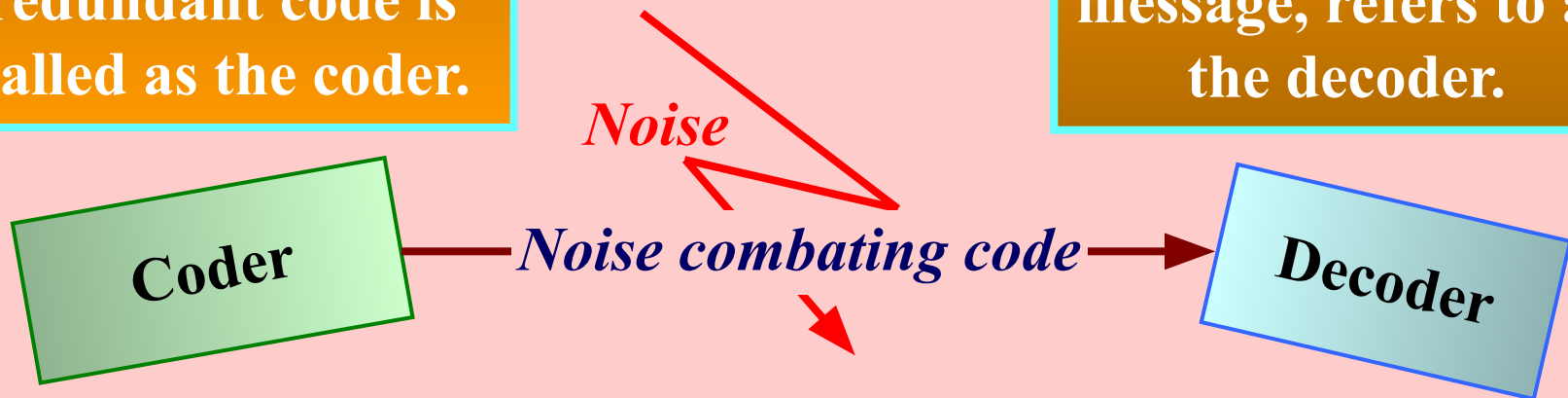


4.2.1. Initial Stage of On-Line Testing Development

To transfer correct message the redundant coding the data with help of correcting or detecting codes was used.

The device which will transform the initial message to a redundant code is called as the coder.

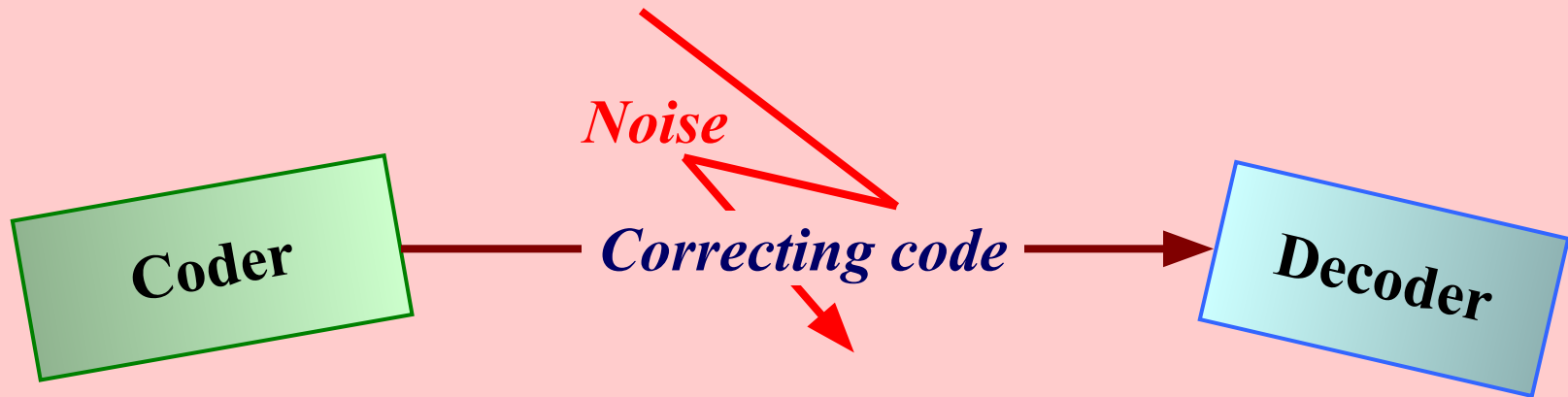
The device that is checking or restoring received message, refers to as the decoder.



4.2.1. Initial Stage of On-Line Testing Development

To transfer correct message the redundant coding the data with help of correcting or detecting codes was used.

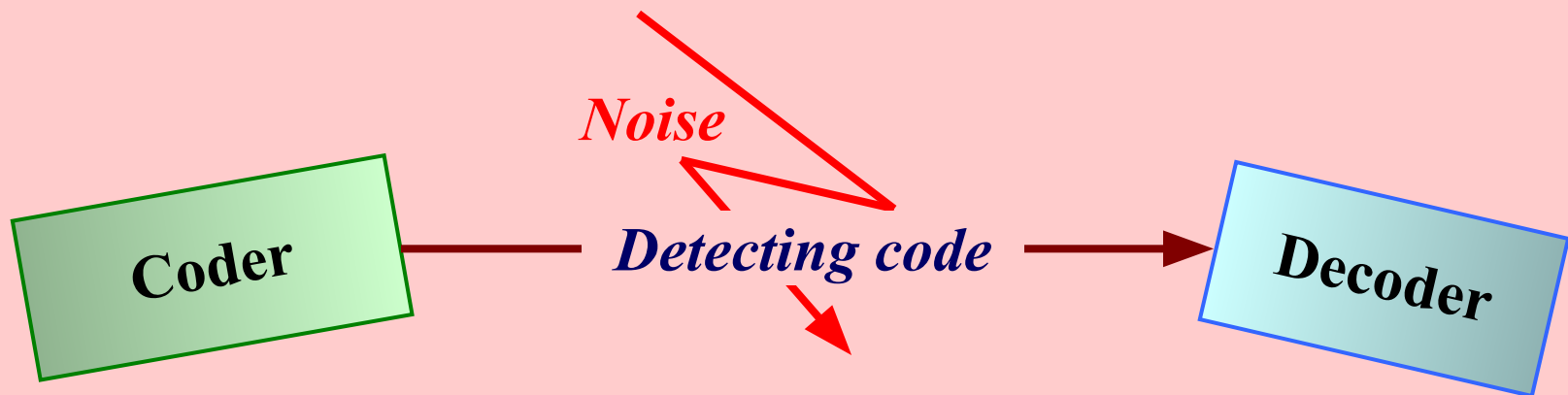
Correcting codes allow to correct errors restoring the message.



4.2.1. Initial Stage of On-Line Testing Development

To transfer correct message the redundant coding the data with help of correcting or detecting codes was used.

Detecting codes allow to check up correctness of the transmitted data. In case of error detection the message will be transferred again.



4.2.1. Initial Stage of On-Line Testing Development

For example,

the elements of the transmitted message are coded by numbers from 000_2 up to 111_2 .

1	1 2 3
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

4.2.1. Initial Stage of On-Line Testing Development

The coder transforms them into words of the group code, which can be defined by the generating array 2 with linear - independent words 1, 2 and 4.

2	1 2 3	4 5 6
1	0 0 1	1 1 0
2	0 1 0	1 0 1
4	1 0 0	0 1 1

1	1 2 3
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

4.2.1. Initial Stage of On-Line Testing Development

The decoder detects an error if it is non-code word. The code words are checked using the linear equation that defines check bits 4, 5 and 6 as the modulo 2 sum of the information bits 1, 2 and 3.

2	1 2 3	4 5 6
1	0 0 1	1 1 0
2	0 1 0	1 0 1
4	1 0 0	0 1 1

$$4 = 2 \oplus 3$$

3	1 2 3	4 5 6
0	0 0 0	0 0 0
1	0 0 1	1 1 0
2	0 1 0	1 0 1
3	0 1 1	0 1 1
4	1 0 0	0 1 1
5	1 0 1	1 0 1
6	1 1 0	1 1 0
7	1 1 1	0 0 0

For example, bit 4 is equal to the modulo 2 sum of the bits 2 and 3.

4	2 \oplus 3	4
1	0 1	1
2	1 0	1
4	0 0	0

4.2.1. Initial Stage of On-Line Testing Development

In case the all equations are true, it is codeword, i.e. correct, and otherwise it is non-codeword and it contains an error.

2	1 2 3	4 5 6
1	0 0 1	1 1 0
2	0 1 0	1 0 1
4	1 0 0	0 1 1

$$4 = 2 \oplus 3$$

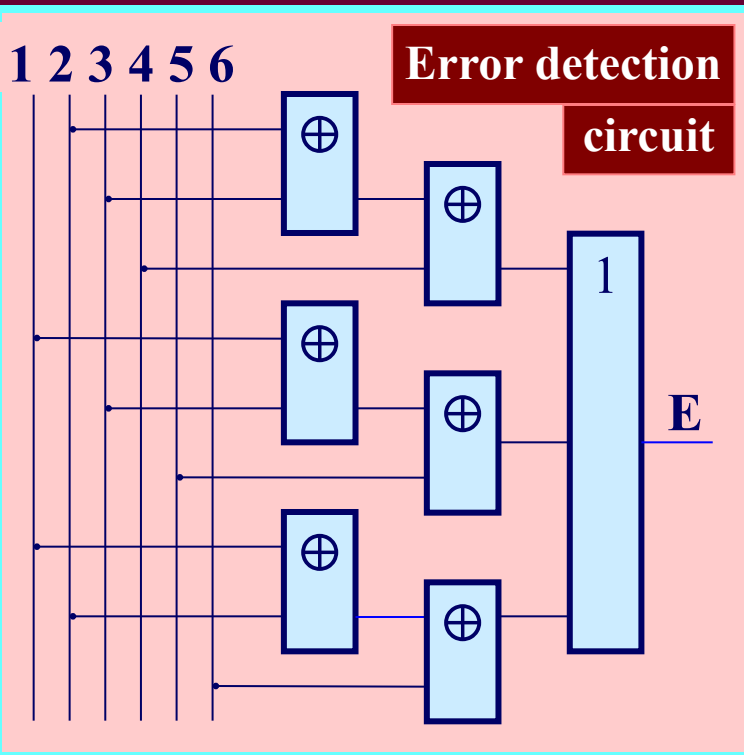
$$5 = 1 \oplus 3$$

$$6 = 1 \oplus 2$$

3	1 2 3	4 5 6
0	0 0 0	0 0 0
1	0 0 1	1 1 0
2	0 1 0	1 0 1
3	0 1 1	0 1 1
4	1 0 0	0 1 1
5	1 0 1	1 0 1
6	1 1 0	1 1 0
7	1 1 1	0 0 0

4.2.1. Initial Stage of On-Line Testing Development

The equations defines the error detection circuit. If the circuit detects an error, its output $E = 1$, otherwise $E = 0$.



$$4 = 2 \oplus 3$$

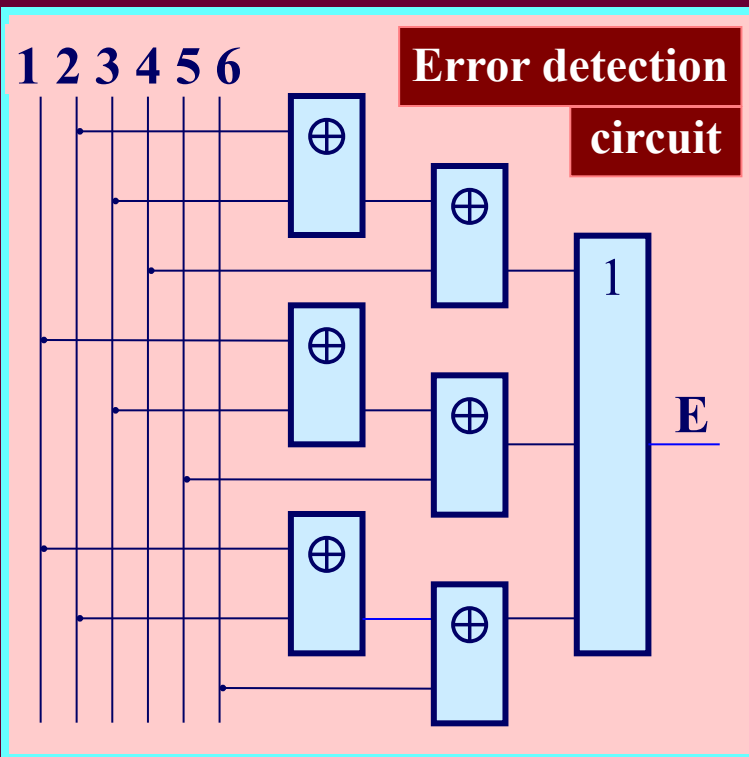
$$5 = 1 \oplus 3$$

$$6 = 1 \oplus 2$$

3	1 2 3	4 5 6
0	0 0 0	0 0 0
1	0 0 1	1 1 0
2	0 1 0	1 0 1
3	0 1 1	0 1 1
4	1 0 0	0 1 1
5	1 0 1	1 0 1
6	1 1 0	1 1 0
7	1 1 1	0 0 0

4.2.1. Initial Stage of On-Line Testing Development

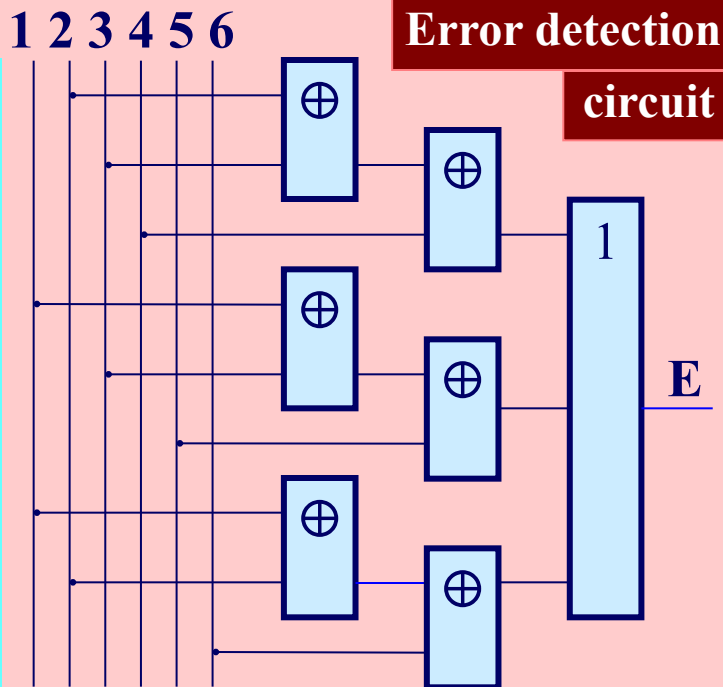
Coders and decoders were considered absolutely reliable during message transfer and consequently were checked only by test in pauses of work.



It has been inherited by on-line testing, where the error detection circuits were used without checking while operation.

4.3. Self-Checking Circuits

In 1968 on the congress in Edinburgh Carter and Schneider for the first time have paid attention to necessity to check the error detection circuit during its work.



To achieve this purpose, they have suggested to build the self-checking circuits.

It was the important step in development of on-line testing, which for the first time has been expanded on his error detection circuits.

4.3. Self-Checking Circuits

- Definitions

A circuit is **fault-secure** for a set of faults F if for every fault in F the circuit never produces an incorrect codeword at the output for an input codeword.

A circuit is **self-testing** for a set of faults F if for every fault in F the circuit produces a non-codeword at the output for at least an input codeword.

If the circuit is both **fault-secure** and **self-testing** it is said to be **totally self-checking**.

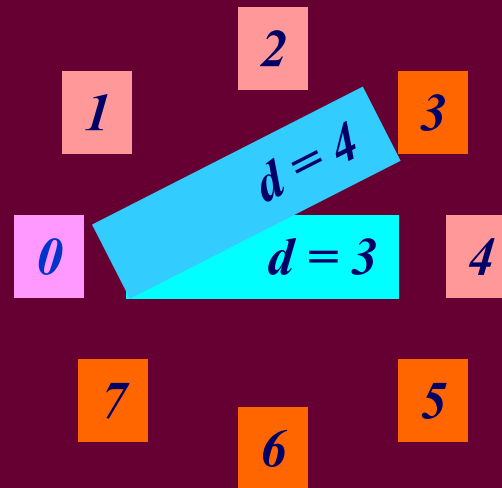
4.3. Self-Checking Circuits

- **Fault-secure circuit**

A circuit is fault-secure for a set of faults F if for every fault in F the circuit never produces an incorrect codeword at the output for an input codeword.

A code distance d between codewords of the pair is an amount of their bits with the differ value.

If fault generates the error in t bits and $t < d$ then the circuit is fault-secure because it produces non-codeword that can not be incorrect codeword.



3	1 2 3	4 5 6
0	0 0 0	0 0 0
1	0 0 1	1 1 0
2	0 1 0	1 0 1
3	0 1 1	0 1 1
4	1 0 0	0 1 1
5	1 0 1	1 0 1
6	1 1 0	1 1 0
7	1 1 1	0 0 0

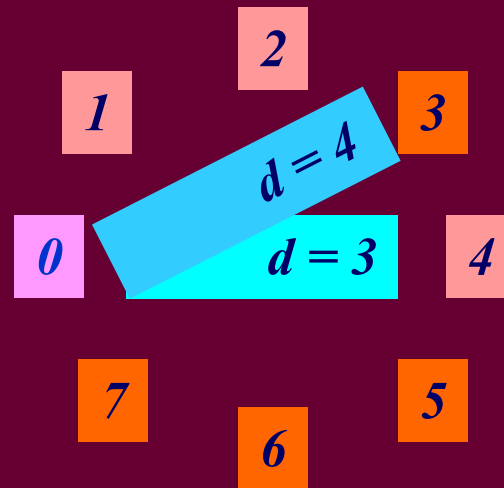
4.3. Self-Checking Circuits

- **Fault-secure circuit**

A circuit is fault-secure for a set of faults F if for every fault in F the circuit never produces an incorrect codeword at the output for an input codeword.

A code distance d between codewords of the pair is an amount of their bits with the differ value.

If fault generates the error in t bits and $t < d$ then the circuit is **fault-secure** because it produces non-codeword that can not be incorrect codeword.



Definition of fault-secure circuit

determines how much **information redundancy** is needed to detect one fault.

4.3. Self-Checking Circuits

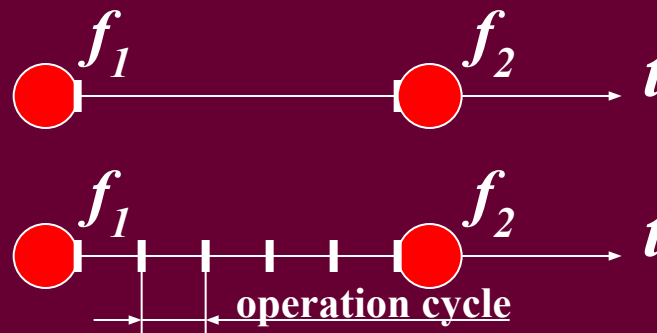
- **Self-Testing circuit**

A circuit is self-testing for a set of faults F if for every fault in F the circuit produces a non-codeword at the output for at least an input codeword.

The self-testing property is aimed to create a condition at which the first fault f_1 should be detected prior to the second fault f_2 of F has occurred.

This condition means that all input codewords should be obtained during the time-interval between faults f_1 and f_2 .

It is satisfied due to **rare occurrence of faults.**



4.3. Self-Checking Circuits

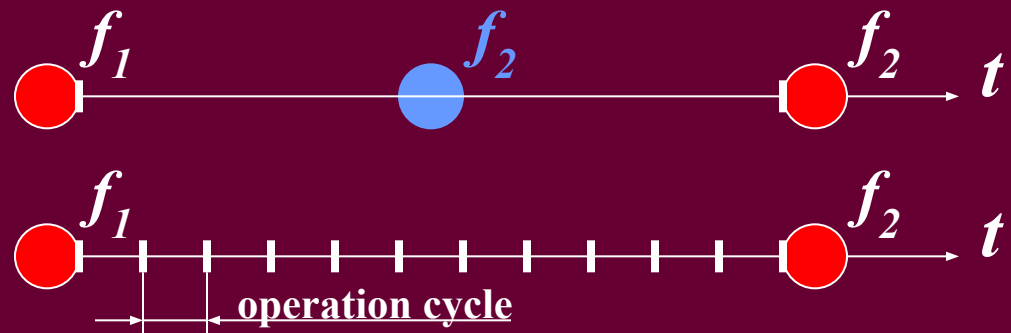
- **Self-Testing circuit**

A circuit is self-testing for a set of faults F if for every fault in F the circuit produces a non-codeword at the output for at least an input codeword.

The self-testing property is aimed to create a condition at which the first fault f_1 should be detected prior to the second fault f_2 of F has occurred.

This condition means that all input codewords should be obtained during the time-interval between faults f_1 and f_2 .

It is satisfied due to **rare occurrence of faults.**



4.3. Self-Checking Circuits

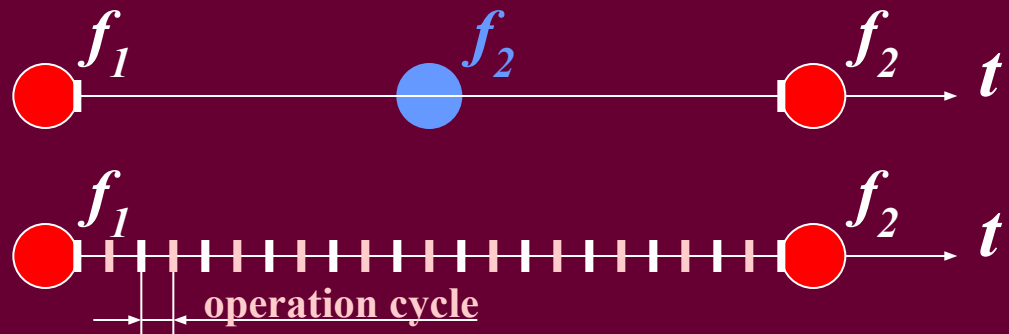
- **Self-Testing circuit**

A circuit is self-testing for a set of faults F if for every fault in F the circuit produces a non-codeword at the output for at least an input codeword.

The self-testing property is aimed to create a condition at which the first fault f_1 should be detected prior to the second fault f_2 of F has occurred.

This condition means that all input codewords should be obtained during the time-interval between faults f_1 and f_2 .

It is satisfied due to **rare occurrence of faults** and **high-frequency operations** of the computing circuits.



4.3. Self-Checking Circuits

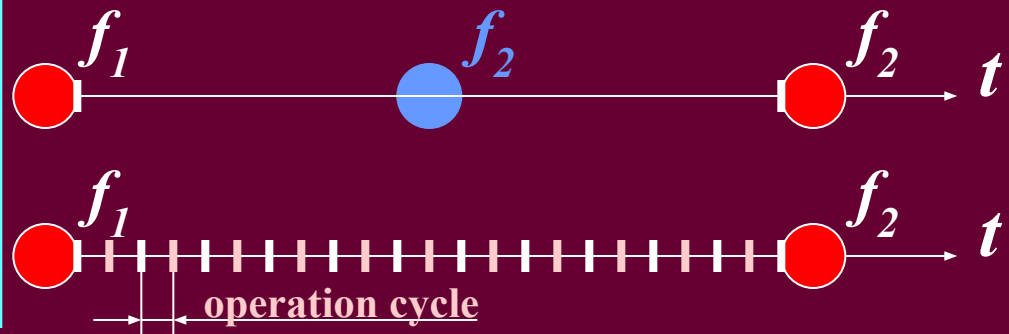
- **Self-Testing circuit**

A circuit is self-testing for a set of faults F if for every fault in F the circuit produces a non-codeword at the output for at least an input codeword.

The self-testing property is aimed to create a condition at which the first fault f_1 should be detected prior to the second fault f_2 of F has occurred.

This condition means that all input codewords should be obtained during the time-interval between faults f_1 and f_2 .

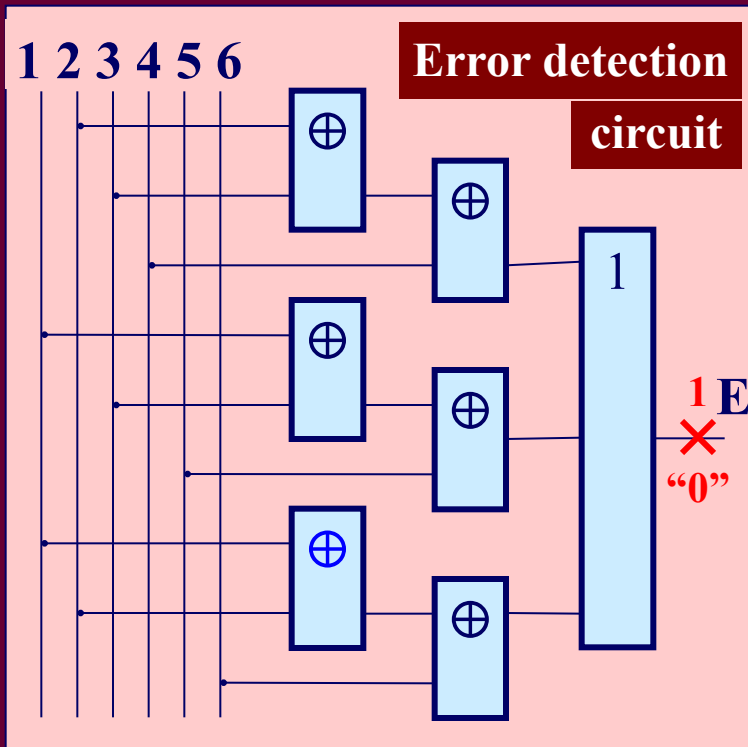
The self-testing property is based on a high level of **reliability** and **productivity** of modern computing circuits.



4.3. Self-Checking Circuits

- **Non-Self-Testing circuit**

By these definitions the designed circuit is not self-checking in a set of stuck-at faults.



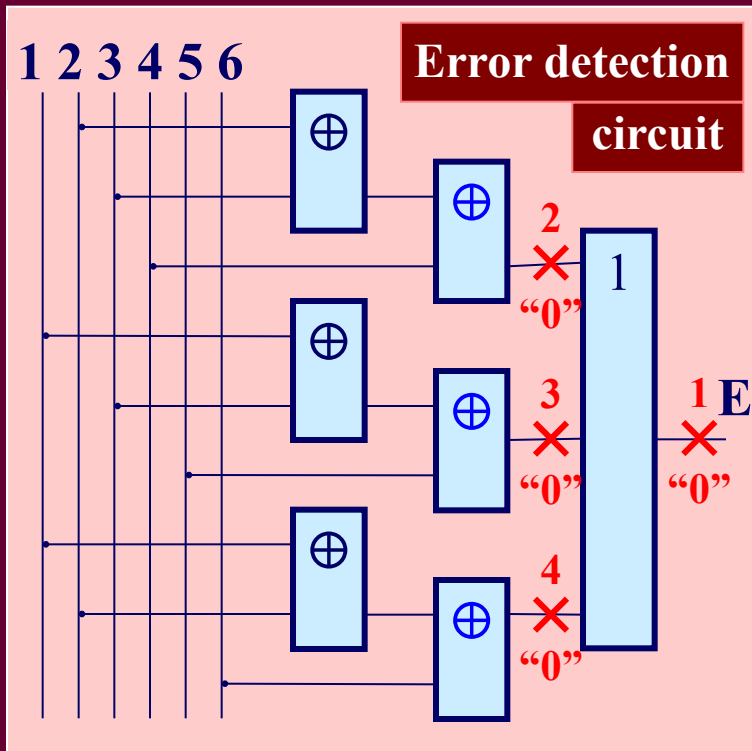
Really, stuck-at «0» fault in a point **1** defines a **codeword** **0** at the output of the circuit on all input code words.

Such circuit is not self-testing and not self-checking in set of the stuck-at faults.

4.3. Self-Checking Circuits

- **Non-Self-Testing circuit**

According to these definitions the designed circuit is not self-checking in a set of stuck-at faults.



Really, stuck-at «0» fault in a point **1** defines a **codeword 0** at the output of the circuit on all input code words.

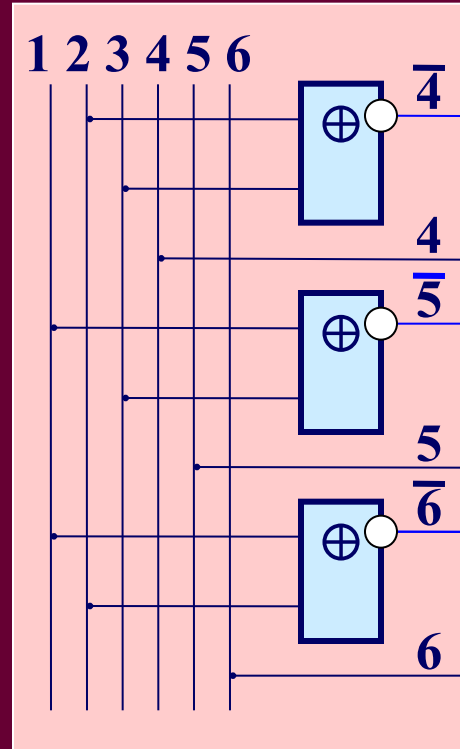
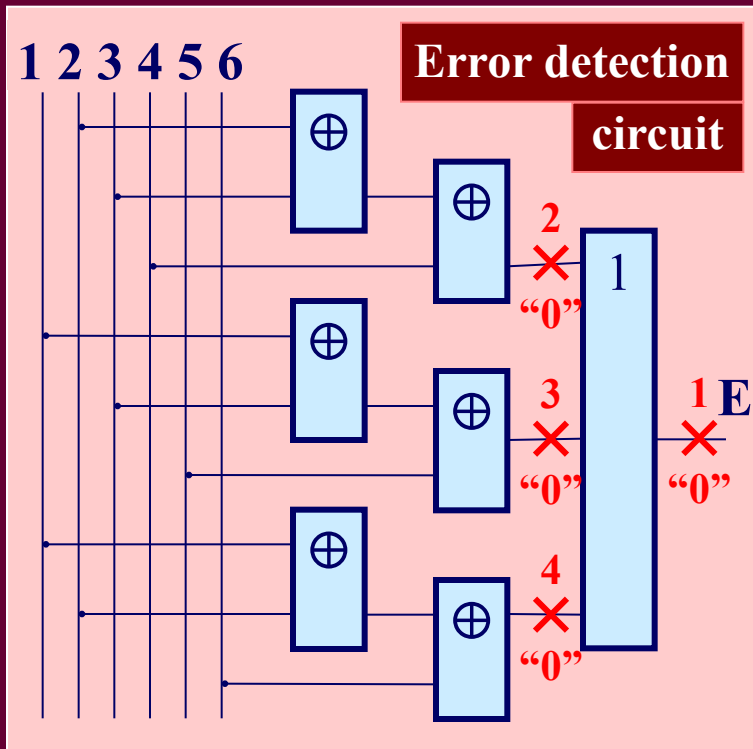
Such circuit is not self-testing and not self-checking in set of the stuck-at faults.

Stuck-at «0» fault in the points **2**, **3** or **4** makes the error detection circuit also not self-checking.

4.3. Self-Checking Circuits

- Design of Self-Checking circuit

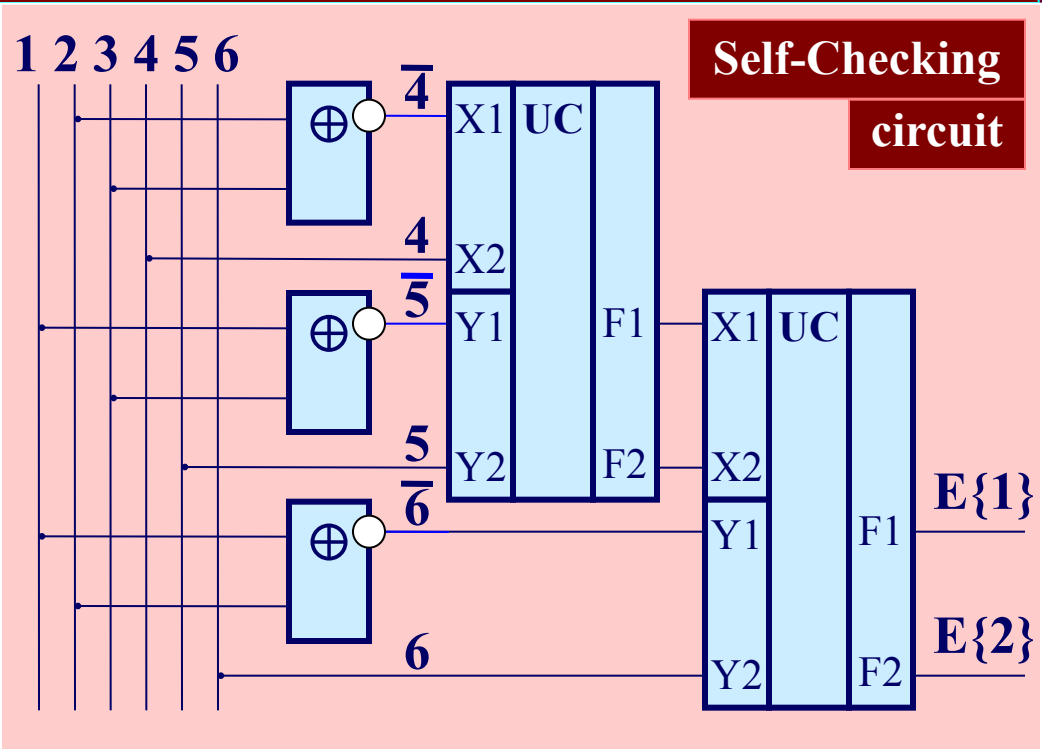
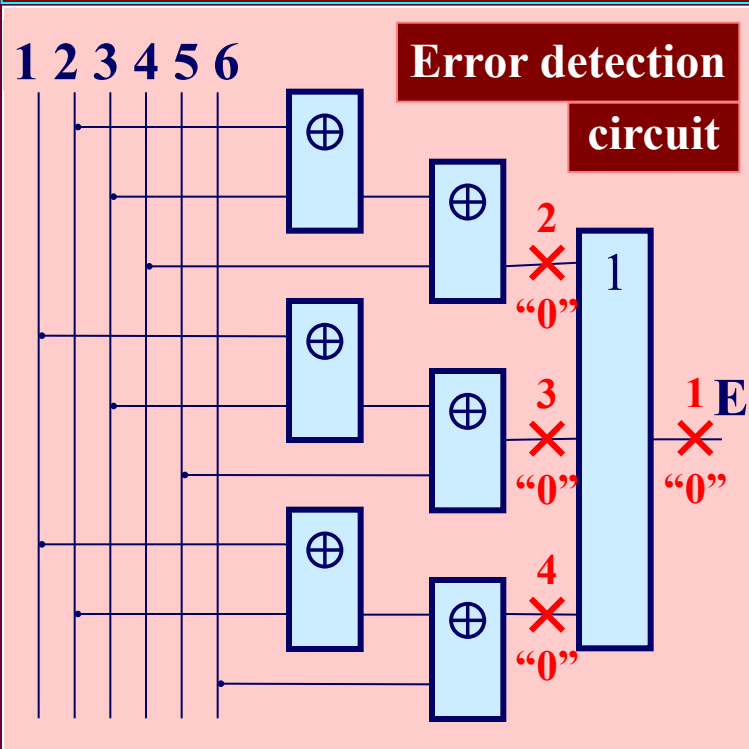
self-checking circuit the bits 4, 5 and 6 are complemented with their inverse bits $\bar{4}$, $\bar{5}$ and $\bar{6}$.



4.3. Self-Checking Circuits

- **Design of Self-Checking circuit** retains Carter's unit (UC), which will transform two pairs of inverse bits $X1=\neg X2$ and $Y1=\neg Y2$ to one pair of inverse bits $F1=\neg F2$.

If even one input pair contains equal bits the output pair will contain equal bits too.



4.3. Self-Checking Circuits

- **Design of Self-Checking circuit** retains Carter's unit (UC), which will transform two pairs of inverse bits $X1 = \neg X2$ and $Y1 = \neg Y2$ to one pair of inverse bits $F1 = \neg F2$.

If even one input pair contains equal bits the output pair will contain equal bits too.

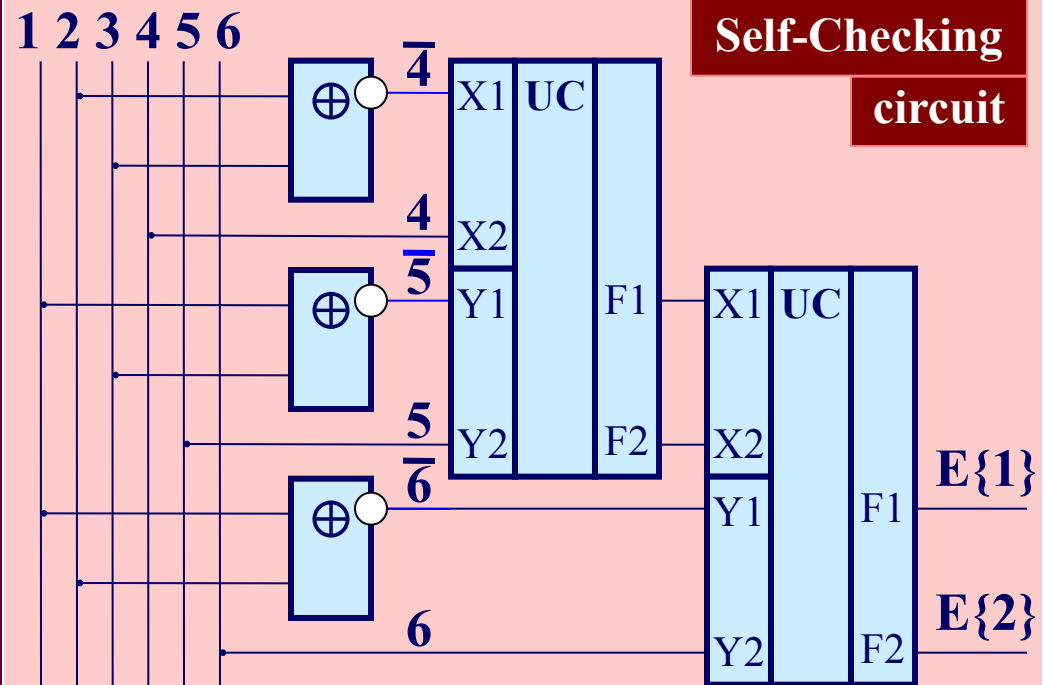
The self-checking circuit has two bits output $E\{1,2\}$.

In case of error detection

$$E\{1\} = E\{2\}$$

and otherwise

$$E\{1\} = \overline{E\{2\}}.$$



4.3. Self-Checking Circuits

- Design of Self-Checking circuit

The next decades **on-line testing** has received wide development in a part of **the self-checking circuit**.

Using parity, residue and other methods of checking, the self-checking circuits were designed:

- self-checking combinational circuits;
- self-checking asynchronous and synchronous sequential machines;
- self-checking Adders and ALUS, Multiply and Divide Arrays.

4.3. Self-Checking Circuits

- Value of Self-Checking circuit

The definitions of self-checking circuit have executed an important role in on-line testing development.

There were determined:

- **conditions to detect faults** using resources required **for one error**;
- **requirements to on-line testing methods** to detect a fault using **the first error** produced in computed result;
- **high level reliability and productivity** of modern computing circuits.

4.4. Purpose of On-Line Testing

• Dogmas of Self-Checking Circuit Theory

However, the definitions of self-checking circuit have also negative influence on on-line testing development.

They have fixed the following dogmas:

- The correct circuit calculates a reliable result, and non-reliable result is computed only on faulty circuit.
- Purpose of on-line testing is to detect a fault of the circuit.
- On-line testing methods have to detect a fault using the first error produced in computed result.

4.4. Purpose of On-Line Testing

- **Dogmas of Self-Checking Circuit Theory**

Is this truth?

The correct circuit calculates a reliable result, and non-reliable result is computed only on faulty circuit.

The truth is that

the correct circuit is necessary only to calculate reliable result, and in itself is not meaningful.

4.4. Purpose of On-Line Testing

- **Dogmas of Self-Checking Circuit Theory**

What is a purpose of on-line testing?

Today the purpose of on-line testing comes from definitions of self-checking circuits.

Purpose of on-line testing is

• to detect a fault of the circuit

• to estimate reliability of the circuit

• to answer a question “Is the circuit correct or not?”

**during the main operations
using actual data.**

**O
r**

4.4. Purpose of On-Line Testing

- **Dogmas of Self-Checking Circuit Theory**

What is a purpose of on-line testing?

Today the purpose of on-line testing comes from definitions of self-checking circuits.

This presentation will show that declared purpose

**a
n
d**

- **defies common sense**

- **contradicts actual on-line testing application**

- **is not achievable for self-checking circuits**

**during the main operations
using actual data.**

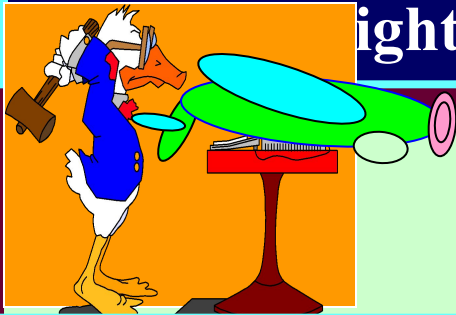
4.4. Purpose of On-Line Testing

Purpose of on-line testing is **to detect a circuit fault** during the main operations using actual data.

Declared purpose defies common sense.

Let's consider computational process as a plane flight.

Detection of the plane faults should be carried out before flight start.



Search for faults during the flight would extremely surprise the passengers.

Creation of the critical conditions is the best way to detect a fault!

The fault can be much more efficiently detected using the off-line testing methods during pauses of the operations.

4.4. Purpose of On-Line Testing

Purpose of on-line testing is **to detect a circuit fault** during the main operations using actual data.

Declared purpose defies common sense.

Faulty circuit can be considered as a mine field.

Circuit fault is a mine.

Test input words are minesweepers that detect mines before the main operations.

Actual data is a farmer working in the field.

Search of faults during computations defies common sense as detection of mines using farmers (actual data).



4.4. Purpose of On-Line Testing

Purpose of on-line testing is **to detect a circuit fault** during the main operations using actual data.

Declared purpose contradicts actual application.

The errors are produced by transient and permanent faults.

Transient faults occur much more often than permanent faults.

Therefore, as a rule, the first detected error is produced by transient fault.

Transient faults are valid for a short period of time.

Therefore, after this period a circuit will be correct again.

That's why on-line testing is not used for circuit fault detection.

4.4. Purpose of On-Line Testing

**Purpose of on-line testing is to answer a question
“Is the circuit correct or not?”**

Declared purpose is not achievable for self-checking circuits

**The first detected error can be produced
by either transient or permanent faults.**

**In case of transient fault
the conclusion that the circuit
is faulty will not be true after
a short period of time.**

**The first detect is not
enough to identify the
permanent fault. It requires
to detect many errors.**

**Therefore, the first detected error cannot answer
a question "Is the circuit faulty or not?"**

4.4. Purpose of On-Line Testing

Actual purpose of on-line testing can be derived from the practice of its application.

Actual purpose of on-line testing is

**O
r**

- **to detect an error, which reduces reliability of the calculated result**
- **to estimate reliability of the calculated result**
- **to answer a question “Is the result reliable or not?”**

during the main operations using actual data.

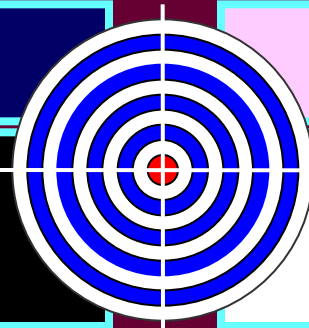
The correct circuit is only necessary to get a reliable result from actual data. **That is why reliability of the circuit by itself should not be the subject of estimation during the main operations.**

4.4. Purpose of On-Line Testing

- Declared vs. Actual purpose

Declared purpose

is to estimate
reliability of a circuit

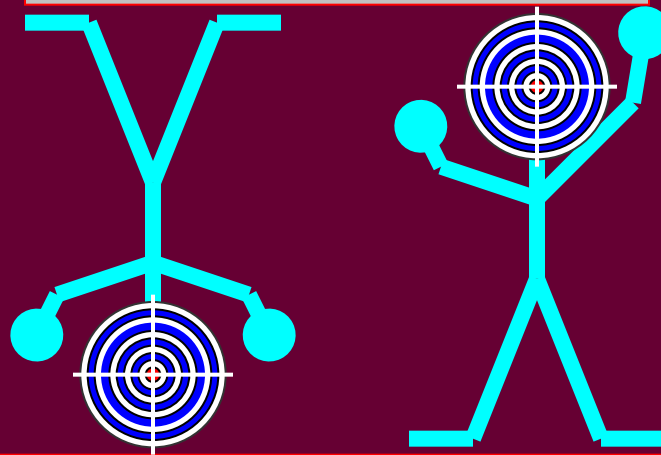


Actual purpose

is to estimate
reliability of a result

PURPOSE

The result
is checked
to answer
a question “Is
a circuit
correct or
faulty”



Means to achieve purpose

Correct circuit
is
only
required to get
a reliable result
from actual
data

4.5. Model of Exact Data

- **What is the reason to declare incorrect purpose?**

This reason is the Model of Exact Data

**This model means that
all numbers
irrespectively of their true nature
are considered as
exact data.**

4.5. Model of Exact Data

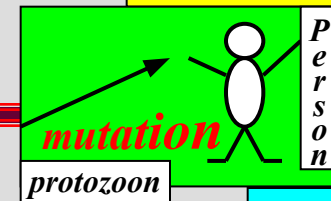
The universe of the approximated data

The universe outside of an error does not exist, does not develop, cannot be studied.

The error is a difference between absolute and relative trues, i.e. the universe is learnt by means of an error.

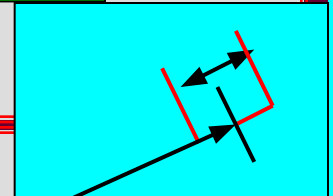


Development of the universe is carried out by a trial and error method.



All exists within the limits of admissions.

The right to make an error is the right to exist.



Quantitative estimations of all things in the universe are numbers with admissions, which are their vital space.

These numbers are the approximated data.

4.5. Model of Exact Data

- What is **Exact Data**?

The **Exact Data** enumerates elements of a set, i.e., it includes only “**integers by nature**”.

All values of codeword can be mapped to the respective **ordinal numbers**. They are integers by nature and belong to **Exact Data**. Everything that can be written down in a field of a computer format is the exact data as well as it can be numbered.

For example, 4-bits codeword has the following values and their **ordinal numbers**:

0 0 1 1

3

4.5. Model of Exact Data

**The exact data model means that all numbers
irrespectively of their true nature
are considered as exact data.**

**Many concepts
first of all connected to a computer,
are under influence of model of the exact data**

4.5. Model of Exact Data

- **On-line testing is based on the Model of Exact Data**

Nobody declared this model

Foundation for

- **self-checking circuit** techniques to obtain reliable results on correct circuit only;

This logic is based on assumption that the correct circuit calculates a reliable result always, and non-reliable result is received only on faulty circuit.

**It is true only
in case of exact data.**

4.5. Model of Exact Data

- On-line testing is based on the Model of Exact Data

Nobody declared this model

Foundation for

- the declared on-line testing purpose to estimate reliability of a circuit through detection of its fault;

All errors are **essential** for reliability of an exact result.

A detected error **concurrently** shows that the calculated **result is non-reliable** and **the circuit has a fault**.

This identifies the declared and actual purposes
for the case of exact data.

4.5. Model of Exact Data

- **On-line testing is based on the Model of Exact Data**

Nobody declared this model

Foundation for

- **the main requirement to on-line testing methods: detect the first error produced by the circuit fault;**

Every error in exact result makes it non-reliable and the computing task terminates abnormally.

The first error detection allows to recalculate this result as soon as it is possible **in case of exact data.**

The first error detection is the fastest way to receive reliable results **in case of exact data.**

4.5. Model of Exact Data

- **On-line testing is based on the Model of Exact Data**

Nobody declared this model

Foundation for

- **self-checking circuit** techniques to obtain reliable results on correct circuit only;
- **the declared on-line testing** purpose to estimate reliability of a circuit through detection of its fault;
- **the main requirement to on-line testing methods:** detect the first error produced by the circuit fault;
- **the on-line testing development within the framework of the exact data processing only.**

Reading List

1. Дрозд А. Этапы развития рабочего диагностирования вычислительных устройств / А. Дрозд // Компьютерные науки и технологии. – Варна (Болгария), 2009. – № 1. – С. 44 – 50.
2. Пархоменко П. П., Согомоян Е. С. и др. Основы технической диагностики. – М.: Энергия, 1981. – 320 с.
3. Согомоян Е. С., Слабаков Е. В. Самопроверяемые вычислительные устройства и системы (обзор) // Автоматика и телемеханика. – 1981. – № 11. – С. 147 – 167.
4. Согомоян Е. С., Слабаков Е. В. Самопроверяемые устройства и отказоустойчивые системы. – М.: Радио и связь, 1989. – 208 с.
5. Дрозд А.В. Нетрадиционный взгляд на рабочее диагностирование вычислительных устройств // Проблемы управления. – 2008. – № 2. – С. 48 – 56. с.
6. Дрозд А.В. Нетрадиционный взгляд на рабочее диагностирование вычислительных устройств / А.В. Дрозд // Автоматизированные системы управления и приборы автоматики. – 2009. – Вып. 147. – С. 15 – 24.

Conclusion

1. **On-line testing** is a base of any S-CES and their components ensuring **reliability of calculated results**.
2. In development of **on-line testing** it is possible to select three stages: **the initial stage**, stage of becoming – **self-checking circuits development** expanding the on-line testing for own means within the framework of the exact data processing, the present stage of on-line testing development **for processing of the approximate data**.
3. **Totally self-checking circuits** detect the faults **using the first error** of the calculated results
4. Self-checking circuits theory defines a **purpose of on-line testing** as **estimation of the circuit reliability**, however the actual purpose is **checking the result reliability**.
5. **Model of exact data** defines development of **on-line testing** within the framework of the exact data processing

Questions and tasks

1. What names of **on-line testing** do you know?
2. Recite the stages of **on-line testing**.
3. Describe the **initial stage** of **on-line testing development**.
4. What **conditions** of **self-checking circuits** do you know?
5. What does **fault security** and **self-testing** mean?
6. What **purpose** of **on-line testing** follows from definitions of **a self-checking circuit**?
7. What is **actual purpose** of **on-line testing**?
8. What is **Exact Data**?
9. What is the **Model of Exact Data**?
10. Describe the role which the **Model of Exact Data** plays in **on-line testing development**.

MODULE 3. On-line testing for digital components of S-CES

Lecture 5. Approximate Data Processing

5.1. Introduction into Approximate Data Processing

5.2. Floating-point Formats and Arithmetic

5.3. Complete and Truncated Operations

5.4. Features of Approximate Data Processing

5.5. Probability of an essential error

5.1. Introduction into Approximate Data Processing

5.1.1. Motivation of Approximate Data Processing Consideration

The majority of processed numbers is approximate data and their volume only increase.

Reasons:

Our Universe is approximate and all in it are structured under its realities including computer Processing

That's why Universe generates approximate data

5.1.2. Related Works

1. Гук М. Процессоры Intel: от 8086 до Pentium II / Гук М. – СПб: Питер, 1997. – 224 с.
2. ANSI/IEEE Std 754-1985. IEEE Standard for Binary Floating-Point Arithmetic. IEEE, New York, USA, 1985. – 18 с.
3. Рабинович З. Л., Раманаускас В. А. Типовые операции в вычислительных машинах. – Киев: Техника, 1980. – 264 с.
4. Савельев А. Я. Прикладная теория цифровых автоматов. – М.: Высш. шк., 1987. – 272 с.
5. Drozd A. On-line testing of computing circuits at approximate data processing / A. Drozd // Радиоелектроніка та інформатика. 2003. № 3. – С. 113 – 116.
6. Демидович Б.П., Марон И.А. Основы вычислительной математики. – М.: Физматгиз, 1966. – 664 с.

5.1.3. Data processed in the S-CES

Two kinds of the S-CES:

1. Like reactor-trip systems for nuclear power plants.



2. Like special dedicated computing systems.



R_M , R_E and R_A – are the results of measurements, exact and approximate data processing accordingly

Processor of the first kind of S-CES operates with exact data

Processor of the second kind of S-CES operates with approximate data

5.1.3. Approximate Data Processing

- **Approximate data**

Approximate data contain results of measurements and are processed in **floating-point format**.

A significance of **approximate data processing** rapidly increases with the computers development.

For example, Intel processors 286 and 386 are complemented in PC by **outside coprocessors 287 and 387** operating with **floating-point formats**.

Starting from processor Intel 486DX the **inside coprocessors** are used for operating with **floating-point formats**.

Pentium-processors have **pipeline inside coprocessors**.

5.2. Floating-point Formats and Arithmetic

- Normal form of data representation

Let a computer works with **8-bit codeword** in range from **$0000\ 0000_2 \div 1111\ 1111_2$** or **$0 \div 255$** .

However it is necessary to solve a computing task in range **$0 \div 1000$** .

For example, it needs to calculate **$800 + 100$** .

This problem was decided using **scale index** **$k_M \geq 1000 / 255$**

Initial data transforms from **range of the computing task** into **range of the codeword**:

$k_M \equiv 4$: **$800 / 4 = 200$** ; **$100 / 4 = 25$** ; **$200 + 25 = 225$** ;

Restoring range of the computing task: **$225 \times 4 = 900$** .

5.2. Floating-point Formats and Arithmetic

- Normal form of data representation

So, Normal form of data representation using two components have discovered:

where m is mantissa or significant;
 $k_M = B^E$ - scale index;
 B - base of numerical system; E - exponent;

The exact data are represented in true form using one component because volume of range and accuracy strongly connected between themselves by size of the codeword.

Approximate data are represented in normal form using two components by reason of significantly different requirements advanced to volume of range and accuracy.

Size of mantissa determines accuracy and exponent size – range.

5.2. Floating-point Formats and Arithmetic

- Normal form of data representation

Normal form $m \times B^E$ represents data using operation of multiplication in a record of floating-point numbers.

That's why

- multiplication is presented in all operations executed with mantissas;
- operations with mantissas and their results inherits the properties and features of a multiplication and a product accordingly

For example,

- an addition of mantissas is executed by matching the exponents shifting one of the mantissas, where shift is special case of multiplication.
- a results of two-place operation has double size

5.2. Floating-point Formats and Arithmetic

- **Standard IEEE-754 (1985)**

Base Formats

- **Single Formats**



- **Double Formats**



Extended Formats:

Single and Double

5.2. Floating-point Formats and Arithmetic

- **Standard IEEE-754 (1985)**

Types of Data	Sign	Bias exponent	Mantissa
Normalized number	\pm	$1 \div 11\dots10$	Any value
Non-normalized number	\pm	0	$\neq 0$
Zero	\pm	0	0
Infinity	\pm	$11\dots11$	0
NaN –No number	\pm	$11\dots11$	$\neq 0$

5.2. Floating-point Formats and Arithmetic

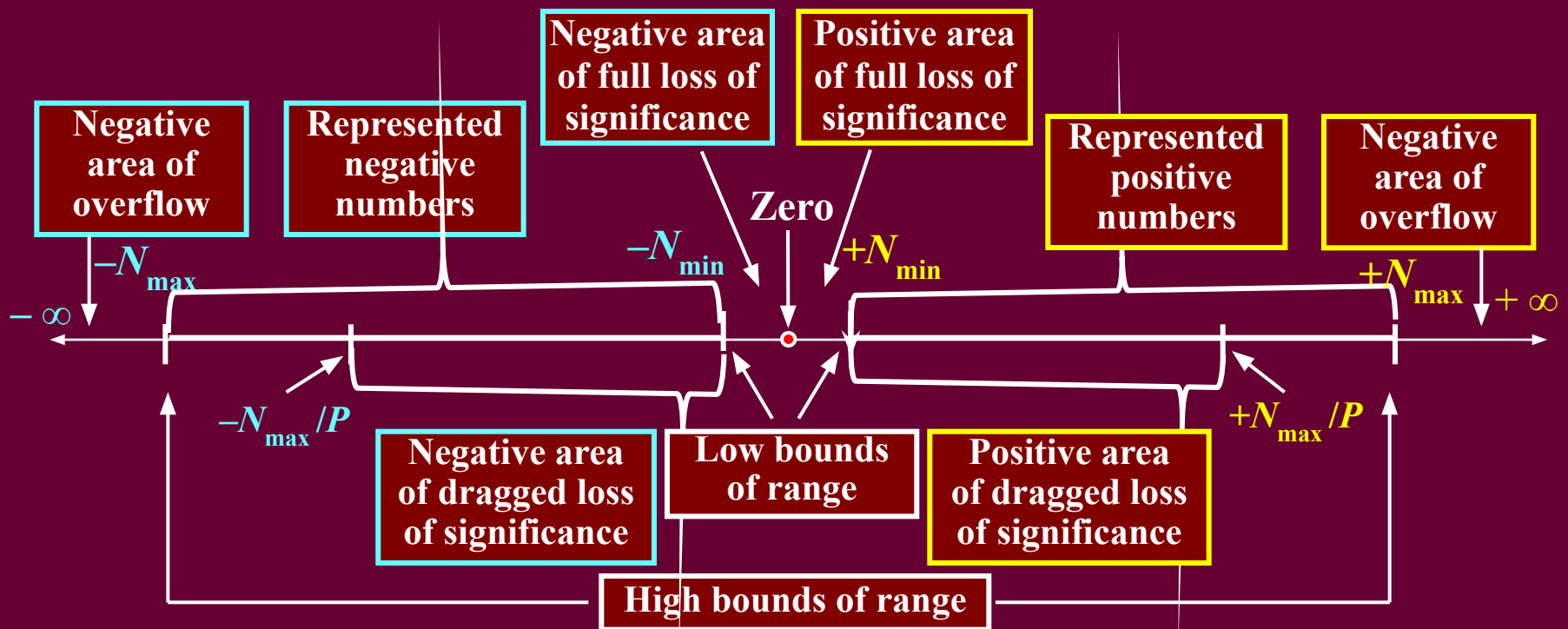
- **Standard IEEE-754 (1985)**

Parameter \ Formats	Single	Double	Double extended
Size of mantissa (in bits)	23	52	≤ 64
Bias exponent	$-126 \div 127$	$-1022 \div 1023$	$-16382 \div 16383$
Bias	127	1023	No regulate
Size of exponent (in bits)	8	11	≤ 15
Size of format (in bits)	32	64	≤ 79
Range of numbers	$10^{-38} \div 10^{38}$	$10^{-308} \div 10^{308}$	No regulate
Amount of exponent values	254	2046	No regulate
Amount of mantissa values	2^{23}	2^{52}	No regulate
Amount of different values	$1,98 \times 2^{23}$	$1,98 \times 2^{63}$	No regulate

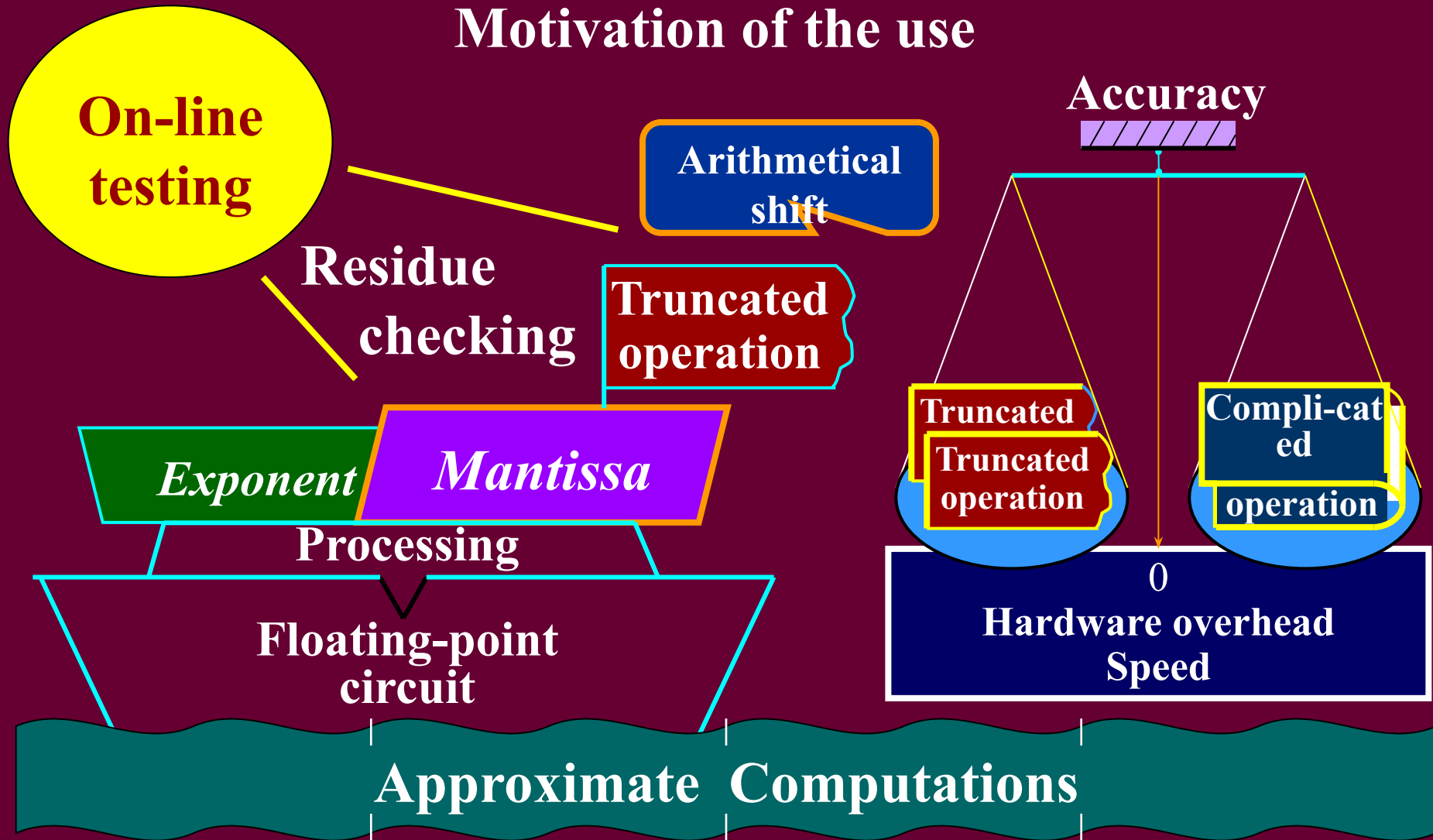
5.2. Floating-point Formats and Arithmetic

- Standard IEEE-754 (1985)

Real number in true form

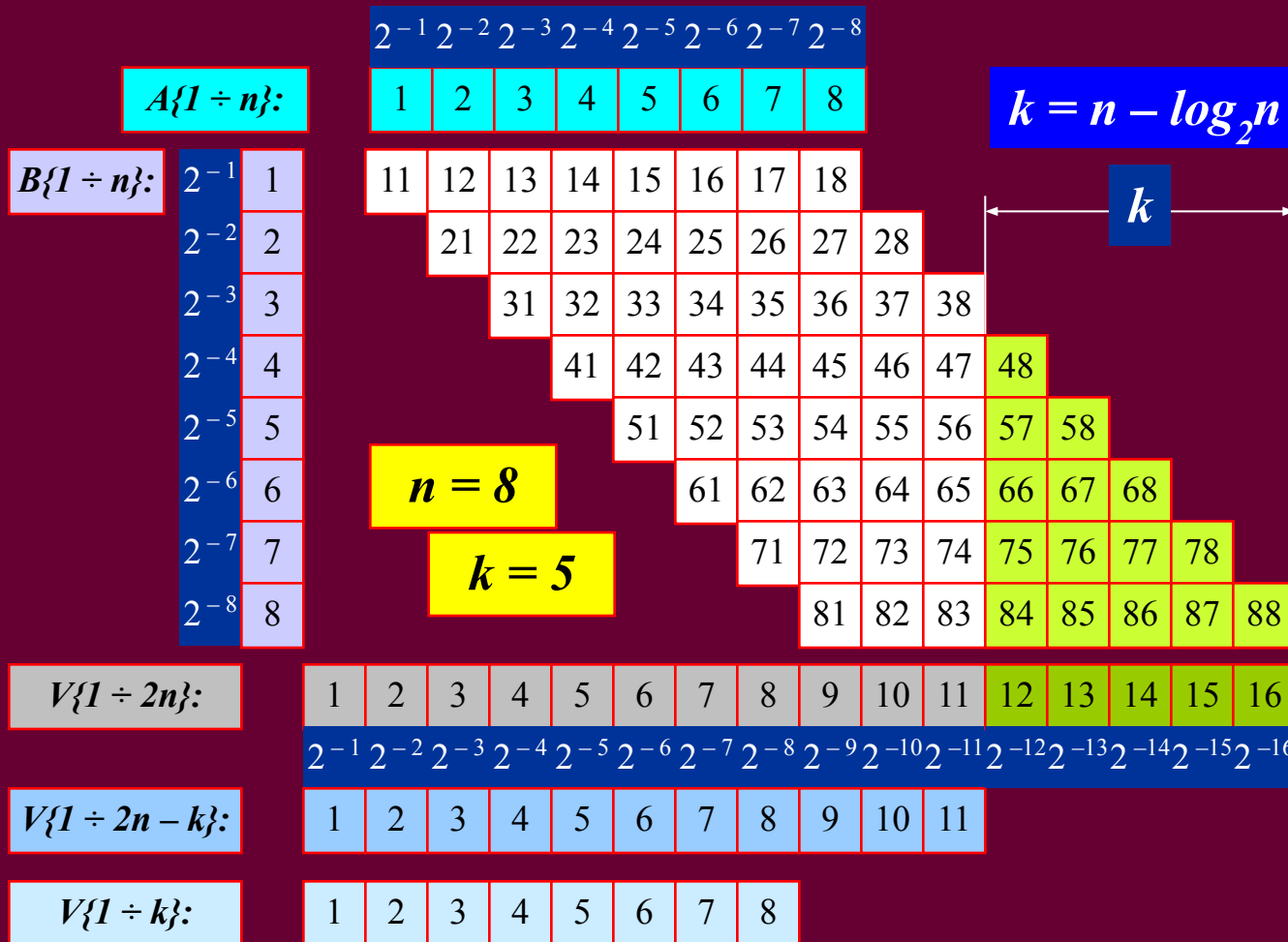


5.3. Complete and Truncated Operations



5.3. Complete and Truncated Operations

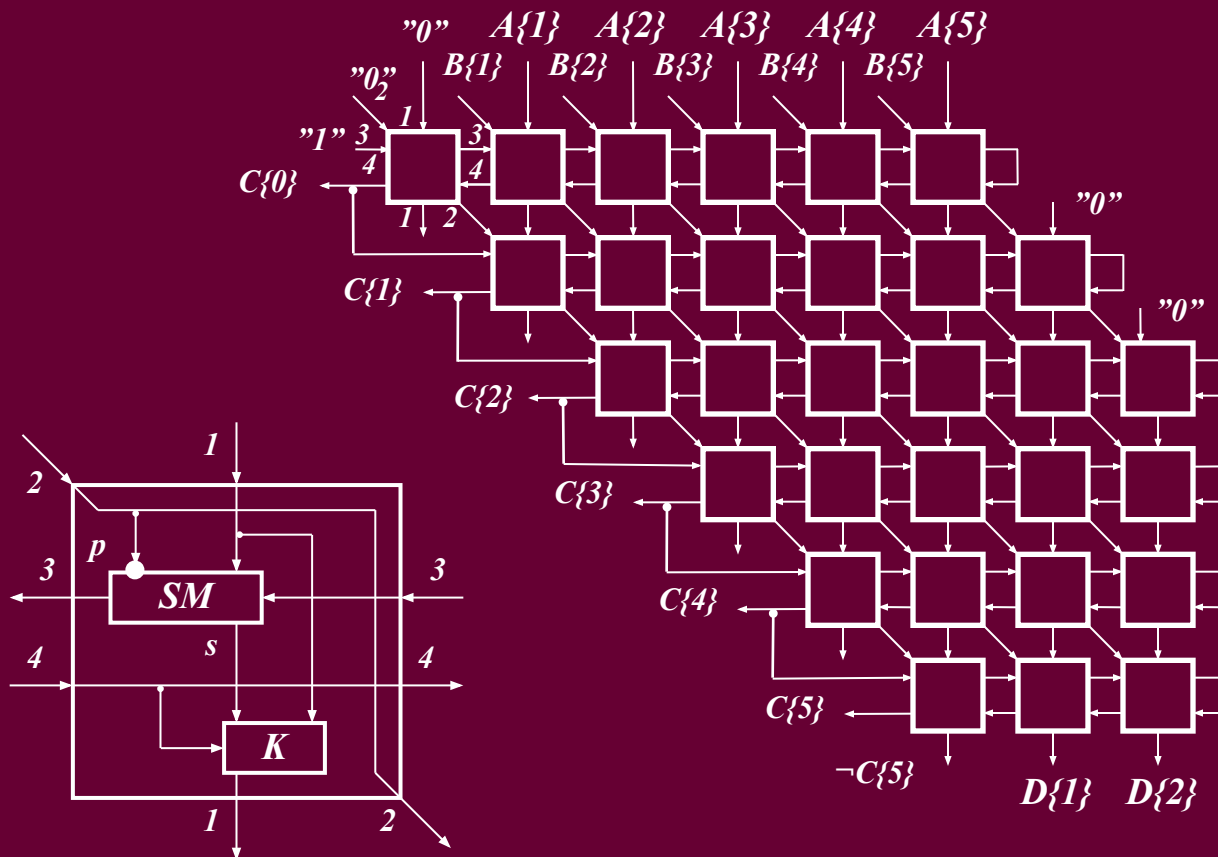
• Truncated multiplication



Truncated multiplication with mantissas reduces almost twice hardware overhead and time operation without lowering an accuracy

5.3. Complete and Truncated Operations

- Truncated non-restoring division



Truncated non-restoring division with mantissas reduces almost twice hardware overhead and time operation without lowering an accuracy

5.3. Complete and Truncated Operations

- **Truncated operation of shift in mantissa addition**

**Truncated
operation
of mantissas
shift
twice reduces
hardware
overhead
without
lowering
an accuracy**

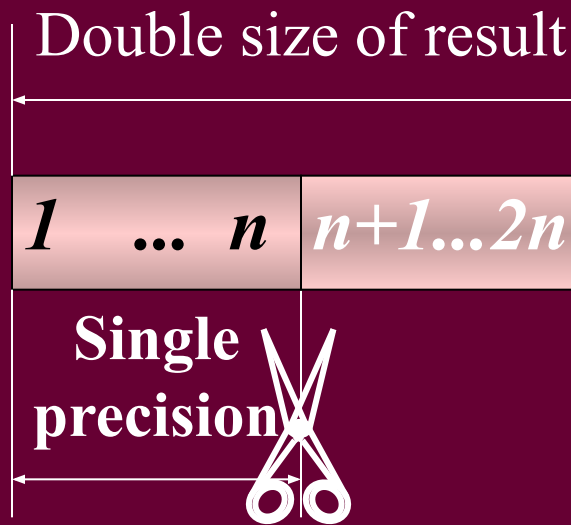
5.4. Features of approximate data processing

1. Deleting of low bits of the calculated result

An approximate number A is represented as a **product**. For example in floating-point format

$$A = m B^E$$

where m is mantissa;
 B is a base of notation;
 E is an exponent.



According to the error theory, a number of exact bits in a result does not exceed a number of exact bits in the operand.

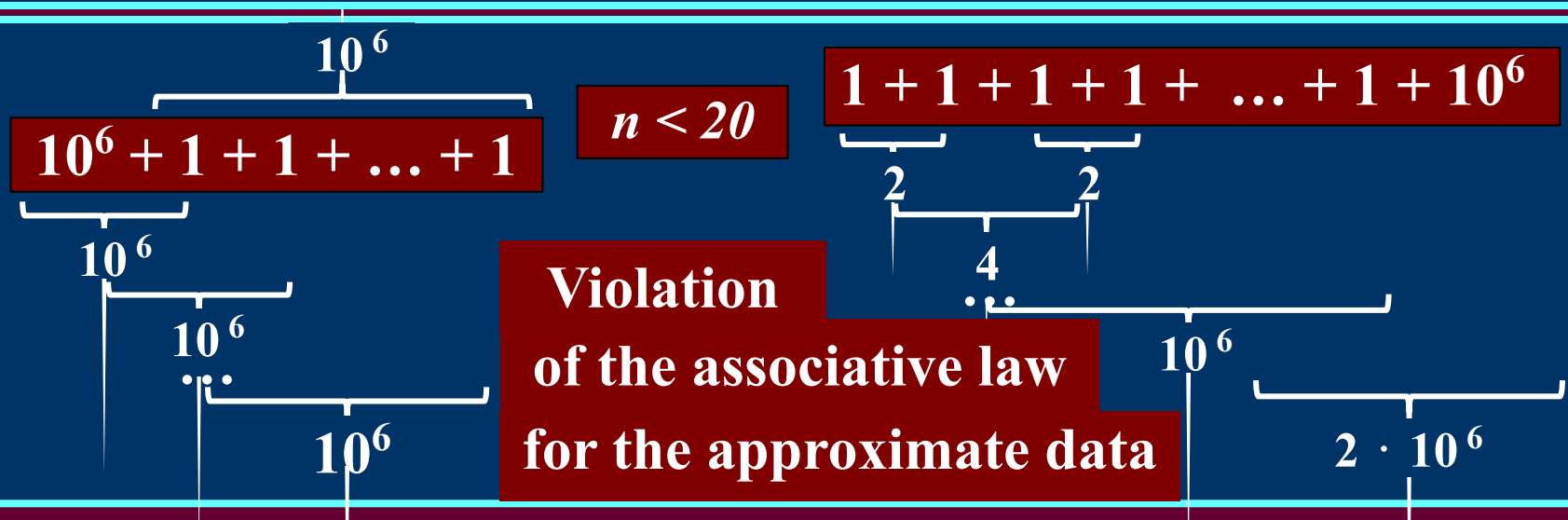
A **product** of two operands doubles a size of the result.

Therefore, the main floating-point formats have a **single precision**.

5.4. Features of approximate data processing

2. Data processing in extended formats

Addition of one million with one million of units by implementing the binary operations with codeword size $n < 20$



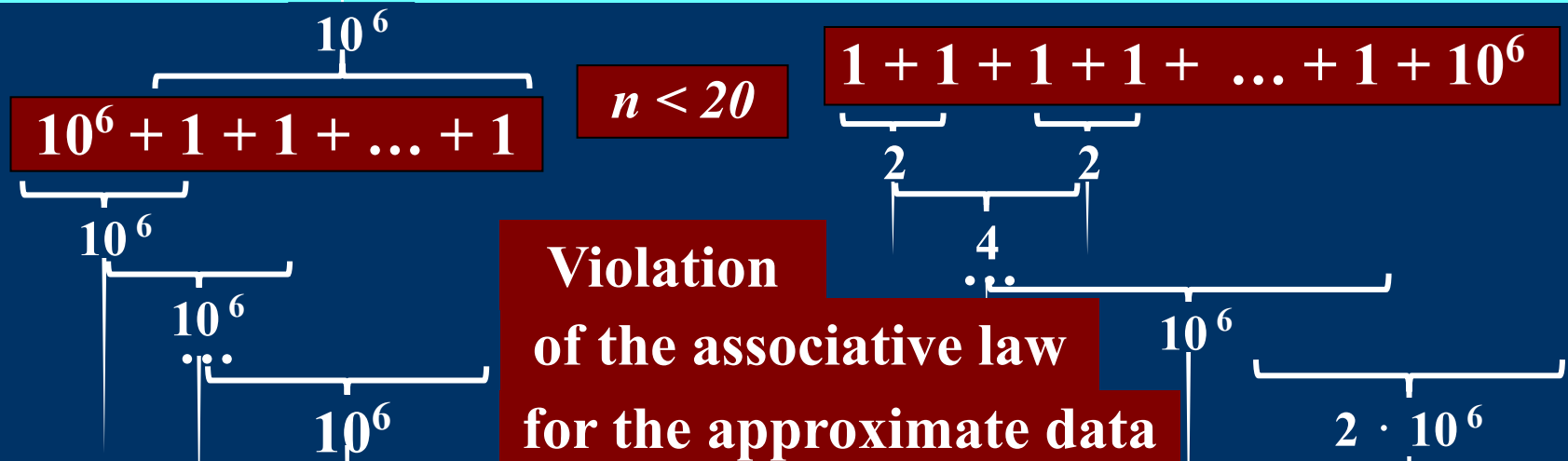
Addition of one million to a unit renders the result of one million because the unit is lost during the exponents matching.

One million of such operations also renders the result equal to the first number, which is one million.

5.4. Features of approximate data processing

2. Data processing in extended formats

Addition of one million with one million of units by implementing the binary operations with codeword size $n < 20$



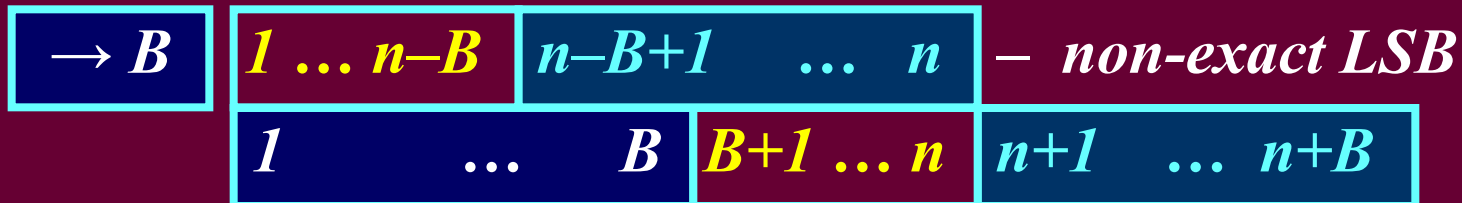
To restore the associative law, the size of the codeword should be increased.

The correct circuit can calculate non-reliable result.

5.4. Features of approximate data processing

3.1. Denormalization of an operand mantissa at the matching the exponents

This action is frequently executed in such operations as **addition, subtraction and matching operands.**



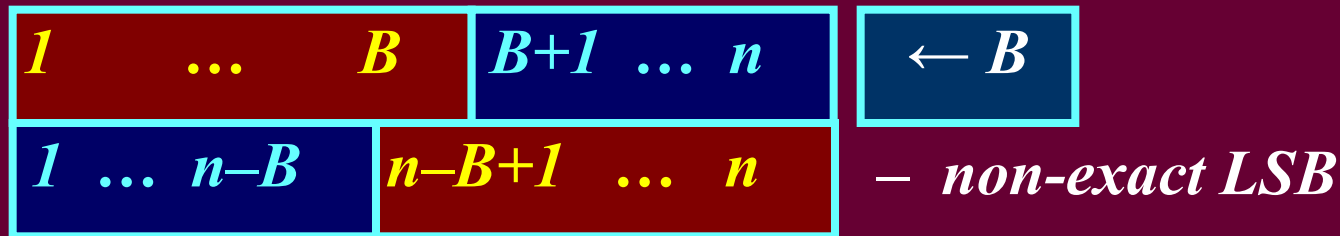
Mantissa of the number with the smaller exponent is shifted down **with loss of least significant bits (LSB).**

Then, **the LSB in the result of all previous operations are eliminated from further calculations.**

5.4. Features of approximate data processing

3.2. Normalization of the result mantissa

This action is executed with **results** in such operations as **addition, subtraction and multiplication**.



Mantissa of the result is cyclic shifted to the left **with filling the low position by LSB**.

Then, the result of **all following operations contain the additional LSB**.

5.5. Probability of an Essential Error

- **Essential and Inessential Errors**

An approximate result has **exact most significant bits (MSB)** and **non-exact LSB**:

exact bits ... non-exact bits

essential ... inessential
ERRORS

Definition:

The error produced by a fault of the computing circuit considered as **essential** error if it reduces the number of exact bits in final result. Otherwise it is considered as **inessential**.

5.5. Probability of an Essential Error

- The factors lowering a probability of essential error

1. Error elimination with discarded bits of the result

Eliminated errors are inessential.

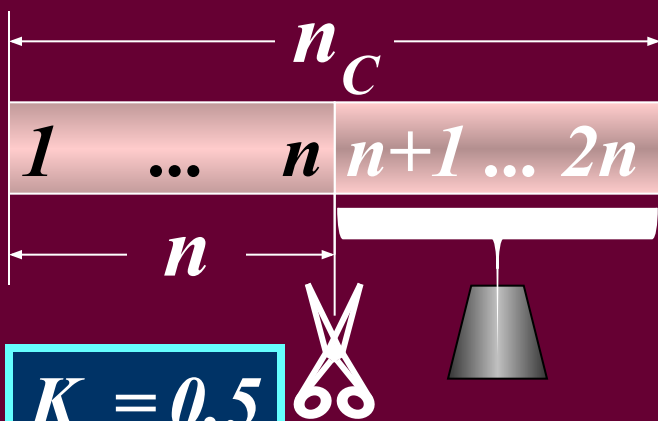
Factor K_1 defines a share of errors remained after elimination of LSB.

$$K_1 = n / n_c$$

n and n_c are numbers of kept and total calculated bits.

A half of all errors is inessential.

The faulty circuit can calculate the reliable result in case of inessential errors.

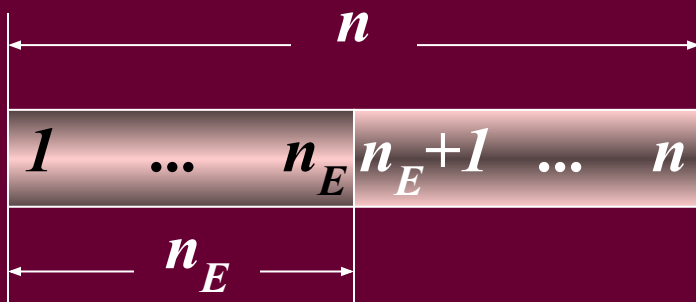


$$K_1 = 0.5$$

5.5. Probability of an Essential Error

- The factors lowering a probability of essential error

2. Increase of a share of inessential errors with use of the extended formats



Factor K_2 defines a share of essential errors in extended format.

$$K_2 = n_E / n$$

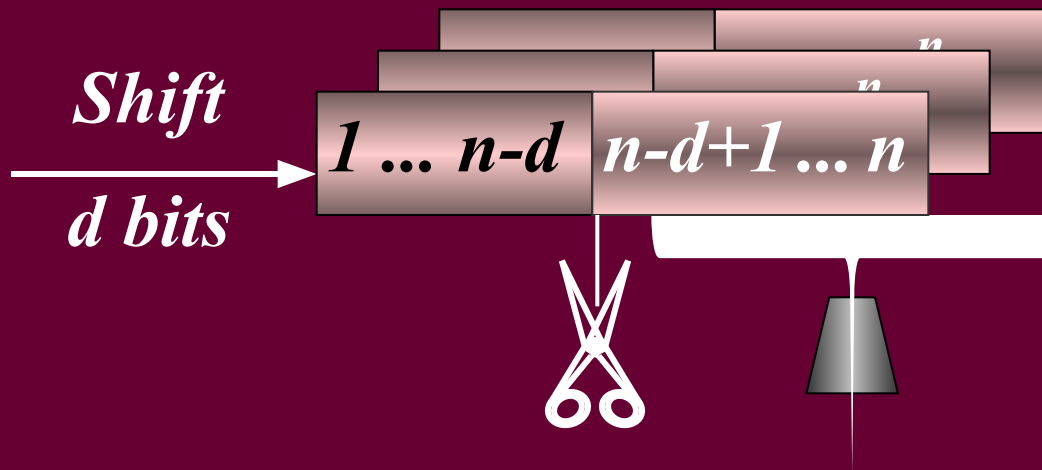
In the formats for floating-point arithmetic on PC size of mantissa increases 2.7 times from 24 bits in a single format up to 64 bits in a double extended format.

n_E and n are the number of exact bits and total number of bits in enlarged mantissa of the extended format.

5.5. Probability of an Essential Error

- The factors lowering a probability of essential error

3.2. Elimination of errors in results of all previous operations



$$K_{3.1} = 1 - \frac{O_S d}{O_C n}$$

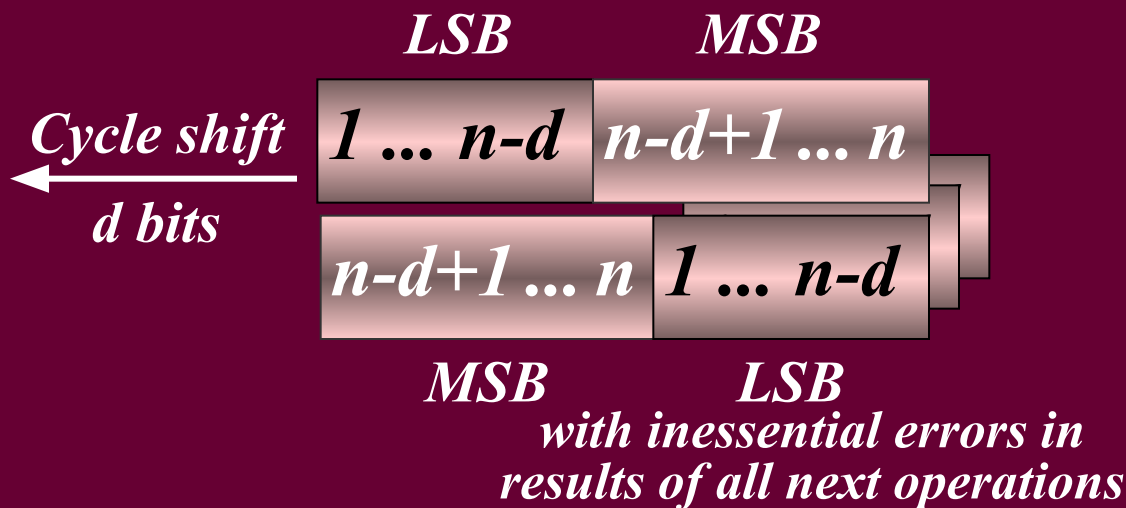
For series of denormalization, K_3 is defined as a product of the factors $K_{3.1}$ calculated for each of these operations.

O_S and O_C are the hardware overhead of computing circuits preceding a shifter and total number of computing circuits.

5.5. Probability of an Essential Error

- The factors lowering a probability of essential error

3.2. Reducing the essential errors amount in results of operations following after normalization



$$K_{3.2} = 1 - \frac{O_s d}{O_c n}$$

For series of normalization, K_3 is defined as a product of the factors $K_{3.2}$ calculated for each of these operations.

O_s and O_c are the hardware overhead of computing circuits following after a shifter and total number of computing circuits.

5.5. Probability of an Essential Error

- The factors lowering a probability of essential error

Probability that the occurred error is essential

$$P_E = K_1 K_2 K_3$$

$$P_E \ll 1$$

For approximate data processing
the majority of errors produced by the circuit
faults belongs to inessential errors.

Reading List

1. Полин Е. Л. Арифметика ЭВМ . Часть 2 / Одеськ. нац. політехніч. ун.-т. – Одеса: АО Бахва, 2002. – 150 с.
7.1.3. Свойства формата с плавающей точкой, с. 115 – 122.
7.2. Стандарт IEEE 754, с. 123 – 131.
2. Дрозд О.В. Контроль за модулем обчислювальних пристроїв. Навч. посібн. для студ. спеціальності 7.091501 – «Комп'ютерні та інтелектуальні системи та мережі» / Одеськ. нац. політехніч. ун.-т. – Одеса: АО Бахва, 2002. – 144 с.
3.1. Скорочення обчислень у ОП, с. 51 – 74.
3. Дрозд А. Этапы развития рабочего диагностирования вычислительных устройств / А. Дрозд // Компьютерные науки и технологии. – Варна (Болгария), 2009. – № 1. – С. 44 – 50.

Conclusion

1. The majority of processed numbers is **approximate data** and **their volume only increase**.
2. **Approximate data** contain results of measurements and are processed in **normal form** using the **floating-point formats**, such as **Standard IEEE 754 formats**.
3. **Approximate data** are represented using **two components** by reason of significantly different requirements advanced to **volume of range** and **accuracy**: size of **mantissa** determines **accuracy** and exponent size – **range**.
4. The truncated operations are the main methods for processing mantissas in **floating-point formats**.
5. The errors produced by the circuit faults in **MSB** and **LSB** of **approximated results** are **essential** and **inessential** accordingly
6. Features of **approximate data** processing determine **factors** significantly lowering a probability of an essential error which is the **general parameter of on-line testing objects**.

Questions and tasks

1. What role do the **approximate data** play in computer processing?
2. What kind of the **approximate data** do you know?
3. Describe the **issues** of **Standard IEEE 754**.
4. Why **approximate data** are represented using two components?
5. What role do the **truncated operations** play in mantissa processing?
6. What are the **essential** and **inessential errors**?
7. What features of **approximate data processing** do the **factors** lowering a probability of an essential error determine?
8. What role do the **probability of an essential error** play in **on-line testing**?

MODULE 3. On-line testing for digital components of S-CES

Lecture 6. Reliability of on-line testing methods

6.1. Reliability of traditional on-line testing methods

6.2. The ways for increasing on-line testing reliability

6.3. The first way for increasing on-line testing reliability

6.4. Residue checking a truncated multiplication

6.5. Residue checking a truncated division of mantissas

6.6. Residue checking a truncated operation of shift

6.1. Reliability of traditional on-line testing methods

6.1.1. Motivation of traditional on-line testing methods reliability consideration

Estimation in reliability of traditional on-line testing methods should be revised.

Reasons:

Our universe is **approximate** and all in it are structured under its realities including **on-line testing methods**

Traditional on-line testing methods have been developed for **exact data** processing and was estimated within framework of **Exact Data Model**.

6.1.2. Related Works

1. Журавлев Ю. П., Котелюк Л. А., Циклинский Н. И. Надежность и контроль ЭВМ. – М.: Советское радио, 1978. – 416 с.
2. Щербаков Н. С. Достоверность работы цифровых устройств. – М.: Машиностроение, 1989. – 224 с.
3. Согомонян Е. С., Слабаков Е. В. Самопроверяемые устройства и отказоустойчивые системы. – М.: Радио и связь, 1989. – 208 с.
4. Рабинович З. Л., Раманаускас В. А. Типовые операции в вычислительных машинах. – Киев: Техника, 1980. – 264 с.
5. Савельев А. Я. Прикладная теория цифровых автоматов. – М.: Высш. шк., 1987. – 272 с.
6. Граф Ш., Гессель М. Схемы поиска неисправностей. – М.: Энергоатомиздат, 1989. – 144 с.

6.1.3. What is reliability of on-line testing methods?

Traditionally, **reliability of on-line testing method** is estimated and considered as **probability of error detection**

Such view on **reliability of on-line testing method** does not take into account features of **on-line testing objects**:

Reliability of on-line testing method should be considered using two parameters:

- **probability of error detection** characterizing **an on-line testing method**;
- **probability of essential error** characterizing **an on-line testing object**.

6.1.3. What is reliability of on-line testing methods?

Reliability of on-line testing method can be considered using unit-side square.

	P_E	P_N
P_D	1 P_{DE}	2 P_{DN}
P_S	3 P_{SE}	4 P_{SN}

P_D is a probability of error detection

P_S is a probability of error skipping

$$P_S = 1 - P_D$$

P_E is a probability of an essential error

P_N is a probability of an inessential error

$$P_N = 1 - P_E$$

P_{DE} is a probability of essential error detection.

P_{DN} is a probability of inessential error detection.

P_{SE} is a probability of essential error skipping.

P_{SN} is a probability of inessential error skipping.

$$\begin{aligned}
 &P_{DE} + \\
 &+ P_{DN} + \\
 &+ P_{SE} + \\
 &+ P_{SN} = 1
 \end{aligned}$$

6.1.3. What is reliability of on-line testing methods?

Reliability of on-line testing methods is defined on dependence of the purpose of on-line testing

	P_E	P_N
P_D	1 P_{DE}	2 P_{DN}
P_S	3 P_{SE}	4 P_{SN}

Estimation of on-line testing method Reliability as a Probability of error detection ignoring a Probability of essential error follows from the Model of Exact Data.

According to declared purpose of on-line testing a method is reliable if the circuit fault is detected irrespectively of error type (essential or inessential).

$$R_{DR} = P_{DE} + P_{DN} = P_D$$

6.1.3. What is reliability of on-line testing methods?

Reliability of on-line testing methods is defined on dependence of the purpose of on-line testing

	P_E	P_N
P_D	1 P_{DE}	2 P_{DN}
P_S	3 P_{SE}	4 P_{SN}

An on-line testing method **defines a result as non-reliable by the error detection.** However an actual tag of non-reliable result is essential error occurrence.

it states the truth about the result: detects the essential errors in case of non-reliable result and **skip inessential ones otherwise.**

According to **actual purpose of on-line testing a method is reliable if correctly estimates a calculated result as reliable or non-reliable.**

$$R_{AR} = P_{DE} + P_{SN} = P_D P_E + (1 - P_D) (1 - P_E)$$

Reliability of on-line testing method is consist of the checking the results

6.1.4. Reliability of on-line testing methods for exact data

$$R_{AR} = P_{DE} + P_{SN} = P_D P_E + (1 - P_D) (1 - P_E)$$

	P_E
P_D	1 P_{DE}
P_S	3 P_{SE}

Exact results have probability $P_E = 1$.

Traditional on-line testing methods based on totally self-checking circuit theory have high detection probability

$$P_D \gg P_S$$

$$R_{AR} = P_D$$

$$R_{AR} \rightarrow 1.$$

Traditional on-line testing methods demonstrate **high reliability** in checking the **exact** results.

6.1.5. Low reliability of traditional on-line testing methods

$$R_{AR} = P_{DE} + P_{SN} = P_D P_E + (1 - P_D) (1 - P_E)$$

1. **Traditional on-line testing methods based on self-checking circuit theory** within framework of the Model of Exact Data have high probability of error detection P_D .

2. **Approximate results** have low probability of essential error P_E

	P_E	P_N
P_D	1 P_{DE}	2 P_{DN}
P_S	P_{SE}	4 P_{SN}

Reliability of traditional on-line testing methods contains low parts 1 and 4 of unit-side square: $R_{AR} \rightarrow 0$.

6.1.5. Low reliability of traditional on-line testing methods

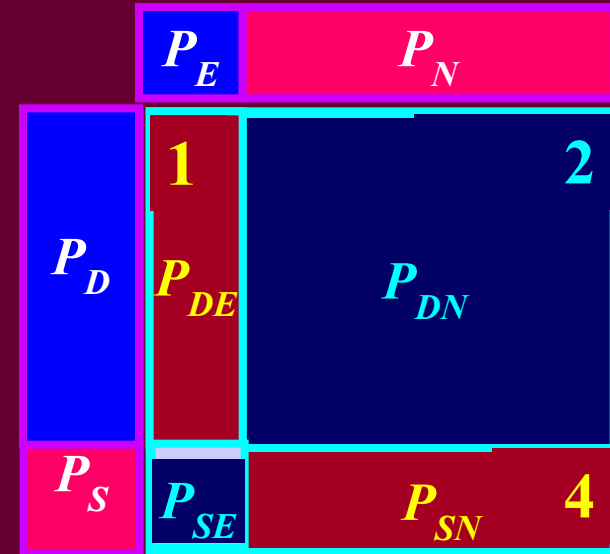
New property of on-line testing methods

1. A difference between **declared** and **actual purpose** of on-line testing is defined by **the part 2** describing a probability of inessential error.

2. **This part 2 is largest** in unit-side square and its area is close to unit: $P_{DN} \rightarrow 1$

3. **The part 2 demonstrates a new property of an on-line testing method to eject reliable results.** For exact data **ejection of reliable results** can be only in case of fault in error detection circuit.

An on-line testing method becomes approximate as our Universe.



6.1.5. Low reliability of traditional on-line testing methods

COMPARISON

CURRENT VIEW

1. Existing on-line testing is applicable to any type of data.
2. A purpose of on-line testing is to estimate reliability of computing circuit.
3. All processed numbers are considered as the exact data.
4. All errors are essential for reliability of computed result.
5. Traditional on-line testing methods have high reliability: detect almost all errors and faults.

NEW VIEW

1. Existing on-line testing is applicable to the exact data only.
2. A purpose of on-line testing is to estimate reliability of computation result.
3. Processed numbers are in most cases approximate data.
4. Basically, the errors are inessential.
5. Traditional on-line testing methods have low reliability of result checking: mainly detect inessential errors.

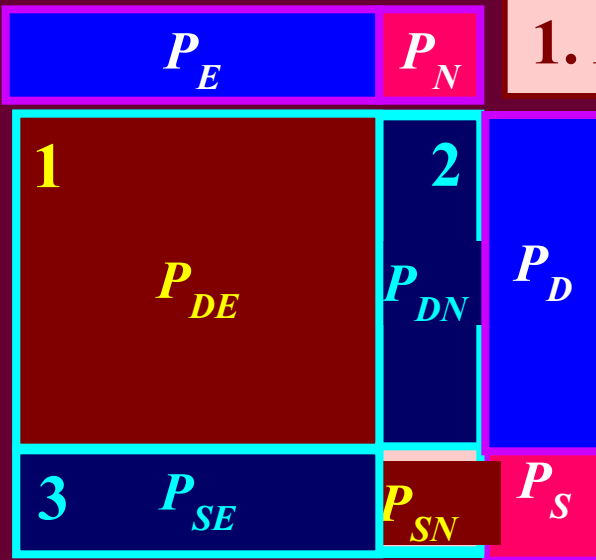
6.2. The ways for increasing on-line testing reliability

$$D = P_D P_E + (1 - P_D)(1 - P_E)$$

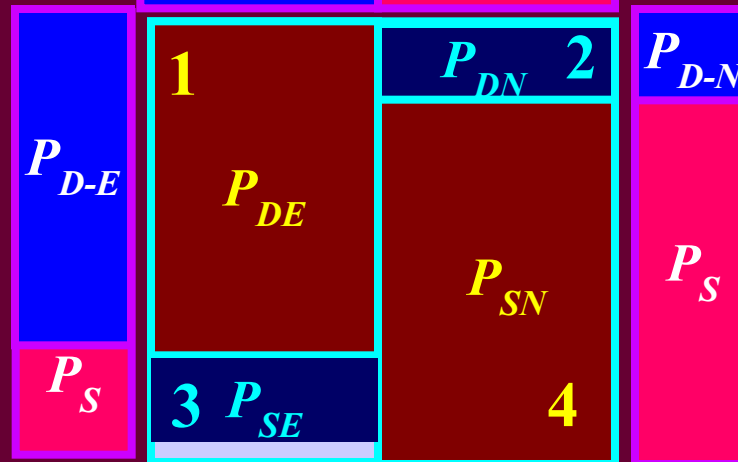
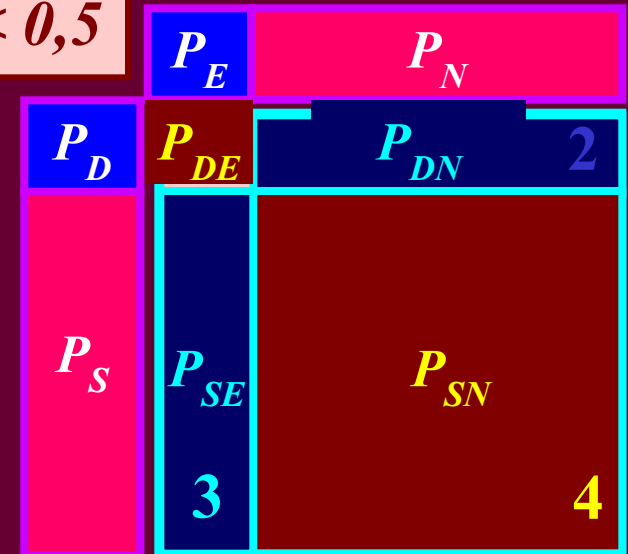
$$D \uparrow = P_D \uparrow P_E \uparrow \text{ или } P_S \downarrow P_N \downarrow$$

$$1. P_E > 0,5$$

$$2. P_E < 0,5$$



$$3. P_{D-E} > P_{D-N}$$



6.2. The ways for increasing on-line testing reliability

$$D = P_D P_E + (1 - P_D)(1 - P_E)$$

$$D \uparrow = P_D \uparrow P_E \uparrow \text{ or } P_S \downarrow P_N \downarrow$$

On-Line Testing Methods

1. $\frac{P_E}{P_D} > 0,5$
 $P_D > 0,5$

Residue checking of truncated operations

2. $\frac{P_E}{P_D} < 0,5$
 $P_D < 0,5$

1. *Checking with natural inf. redundancy.*

2. *Checking by simplified operation.*

3. $P_{D-E} > P_{D-N}$

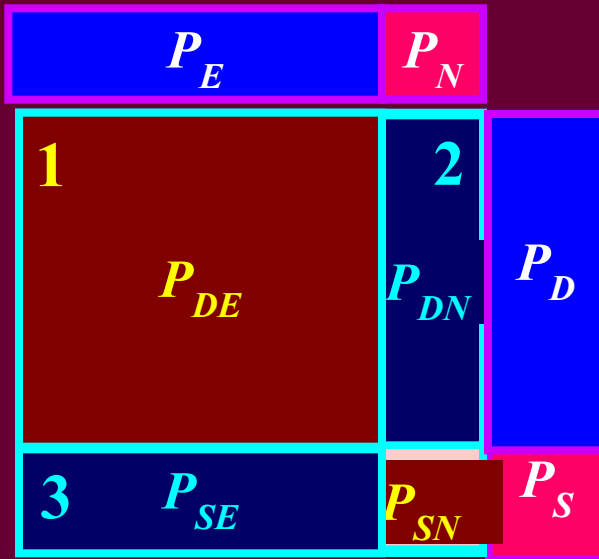
1. *Logarithm checking*

2. *Checking by inequalities*

3. *Checking by segments*

6.3. The first way for increasing on-line testing reliability

$$D \uparrow = P_D \uparrow P_E \uparrow$$



$$(P_E > 0,5) \ \& \ (P_D > 0,5)$$

1. The first way is increasing the part 1 of unit-side square raising a probability of essential error

2. The first way allows to develop the on-line testing methods with traditionally high probability of error detection

3. This way provides the high probability of essential error detection

6.3. The first way for increasing on-line testing reliability

1. Residue checking of truncated operations

High probability of essential error

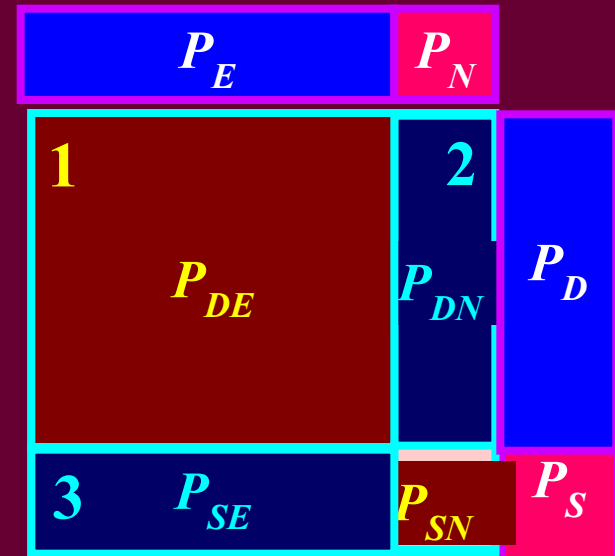
$$P_E > 0,5$$

can be achieved only for truncated operations

Residue checking is the main on-line testing method for arithmetic of complete operations

That's why residue checking is rationally to extend on truncated operations

$$D \uparrow = P_D \uparrow P_E \uparrow$$



$$(P_E > 0,5) \ \& \ (P_D > 0,5)$$

6.4. Residue checking a truncated multiplication

The method is based on a decomposition of high part of the product conjunction array (PCA) into fragments.

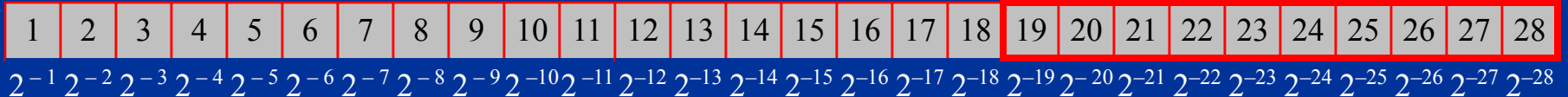
A fragment is defined as a part of PCA described with a product $V_i = \pm A_i B_i$, where A_i and B_i are operands A and B or their parts.

The method uses definition of a fragment and representation of a truncated product in check codes:

$$KV_i = \pm KA_i KB_i$$

$$KV_T = \sum_{i=1}^{k+1} KV_i$$

$V\{1 \div 2n\}$:



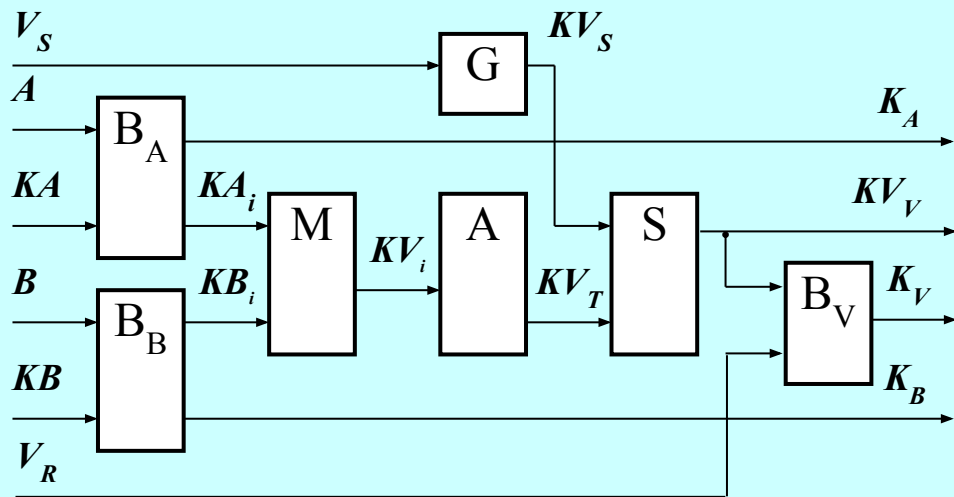
The method compares the check codes of truncated product calculated by two ways:

- using truncated product;
- using operands.

High part of the PCA can be represented as a sum of fragments:

$$V_T = \sum_{i=1}^{k+1} V_i$$

6.4. Residue checking a truncated multiplication



Error detection circuit

Blocks B_A and B_B check the operands A and B by computing the check codes KA and KB and comparing them with the input check codes KA and KB . Results of comparison are the error indication codes K_A and K_B .

The check codes KA_i and KB_i are composed of operand bits or computed during the generation of the check codes KA and KB .

Block M computes the check codes KV_i , $i=1 \div k-1$, of the fragments by the formula (1). Block A calculates the check code KV_T of the truncated product by the formula (2).

The block G generates the check code KV_S of the excluded bits V_S . Block S computes the check code of the result KV_V .

Block B_V checks the result V_R by comparing it with the check code KV_V . Result of comparison is the error indication code K_V .

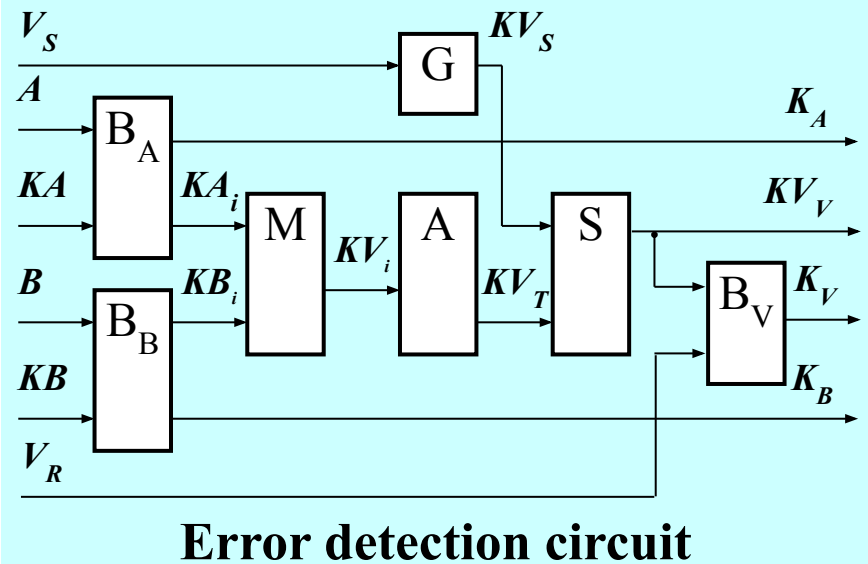
$$KV_T = \sum_{i=1}^{k+1} KV_i \quad (1)$$

$$KV_i = \pm KA_i KB_i \quad (2)$$

6.4. Residue checking a truncated multiplication

The method of residue checking a truncated multiplication defines the following steps:

- Choice of the PCA decomposition into fragments;
- Description of fragments;
- Description of the check codes KA_i and KB_i , composed of operands bits;
- Definition of formulas for calculated check codes KA_i and KB_i ;
- Design of the blocks B_A and B_B in accordance with obtained formulas;
- Design of the blocks M and A taking into account the descriptions of fragments and check codes KA_i , KB_i ;
- Design of the blocks G and S using values of n and k ;
- Design of the block B_V as a block B_A for the following error detection circuit where result is used as operand.



$$KV_T = \sum_{i=1}^{k+1} KV_i \quad (1)$$

$$KV_i = \pm KA_i KB_i \quad (2)$$

6.4. Residue checking a truncated multiplication

Choice of the PCA decomposition into fragments should be aimed to design a high quality error detection circuit.

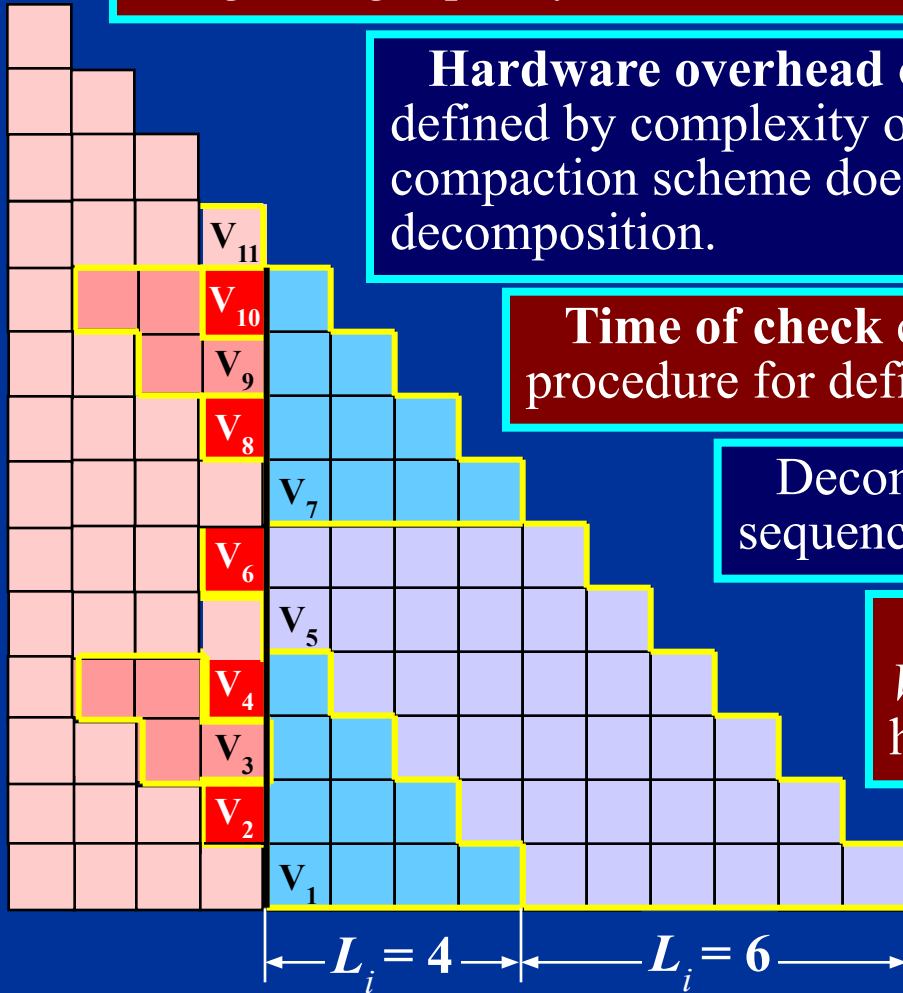
Hardware overhead of the error detection circuit is mainly defined by complexity of the blocks B_A and B_B which as compaction scheme does not depend in complexity on the PCA decomposition.

Time of check can be reduced using the following procedure for defining the PCA decomposition.

Decomposition is defined specifying a sequence of central - symmetric fragments.

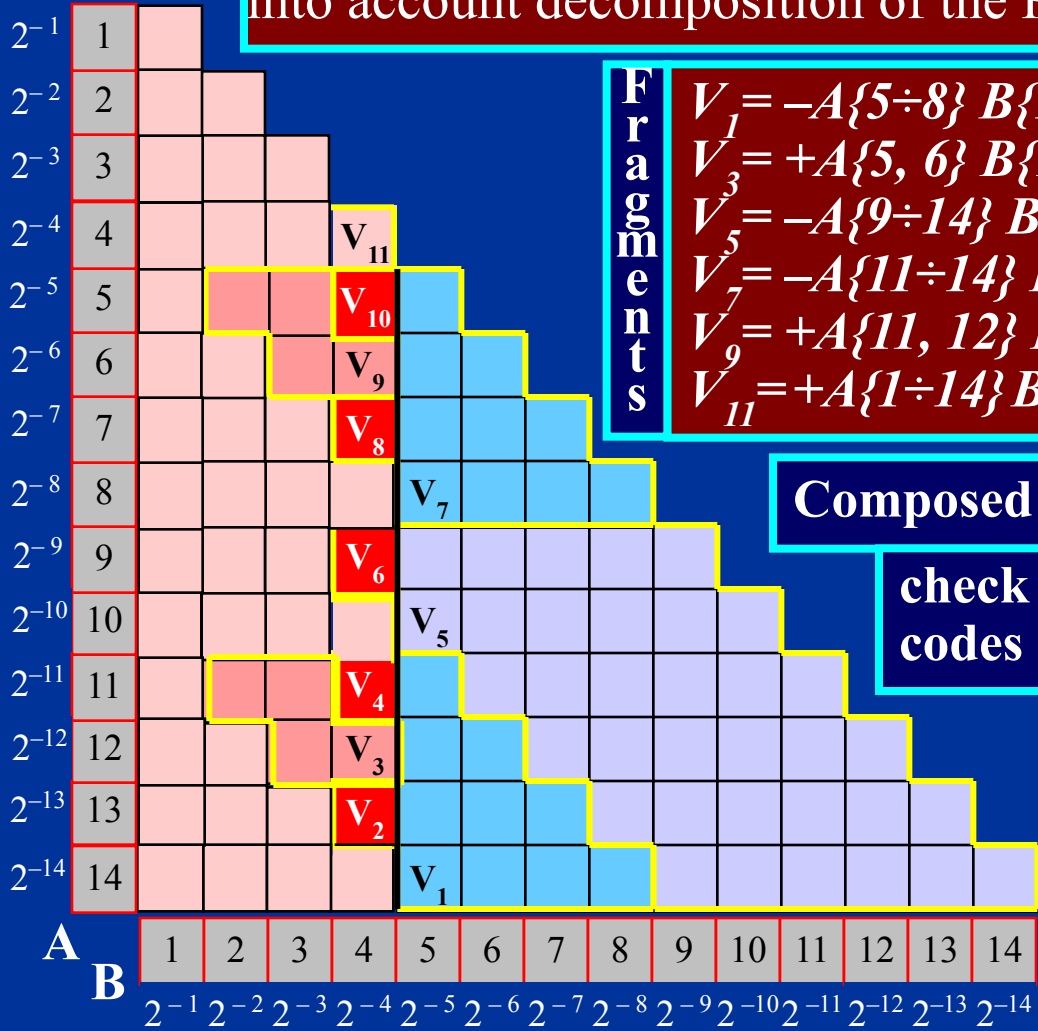
The first central - symmetric fragment $V_i = -A\{n-Li+1 \div n\} B\{n-Li+1 \div n\} 2^{-2n}$ has size $L_i = 2 E(k/4+1)$.

It defines high and low parts like the PCA high part with $k = k - L_i$. Process is following before $k > 1$.



6.4. Residue checking a truncated multiplication

Blocks of the error detection circuit are developed taking into account decomposition of the PCA into fragments.



Fragments

$$\begin{aligned}
 V_1 &= -A\{5 \div 8\} B\{11 \div 14\} 2^{-22} \\
 V_3 &= +A\{5, 6\} B\{11, 12\} 2^{-18} \\
 V_5 &= -A\{9 \div 14\} B\{9 \div 14\} 2^{-28} \\
 V_7 &= -A\{11 \div 14\} B\{5 \div 8\} 2^{-22} \\
 V_9 &= +A\{11, 12\} B\{5, 6\} 2^{-18} \\
 V_{11} &= +A\{1 \div 14\} B\{1 \div 14\} 2^{-28}
 \end{aligned}$$

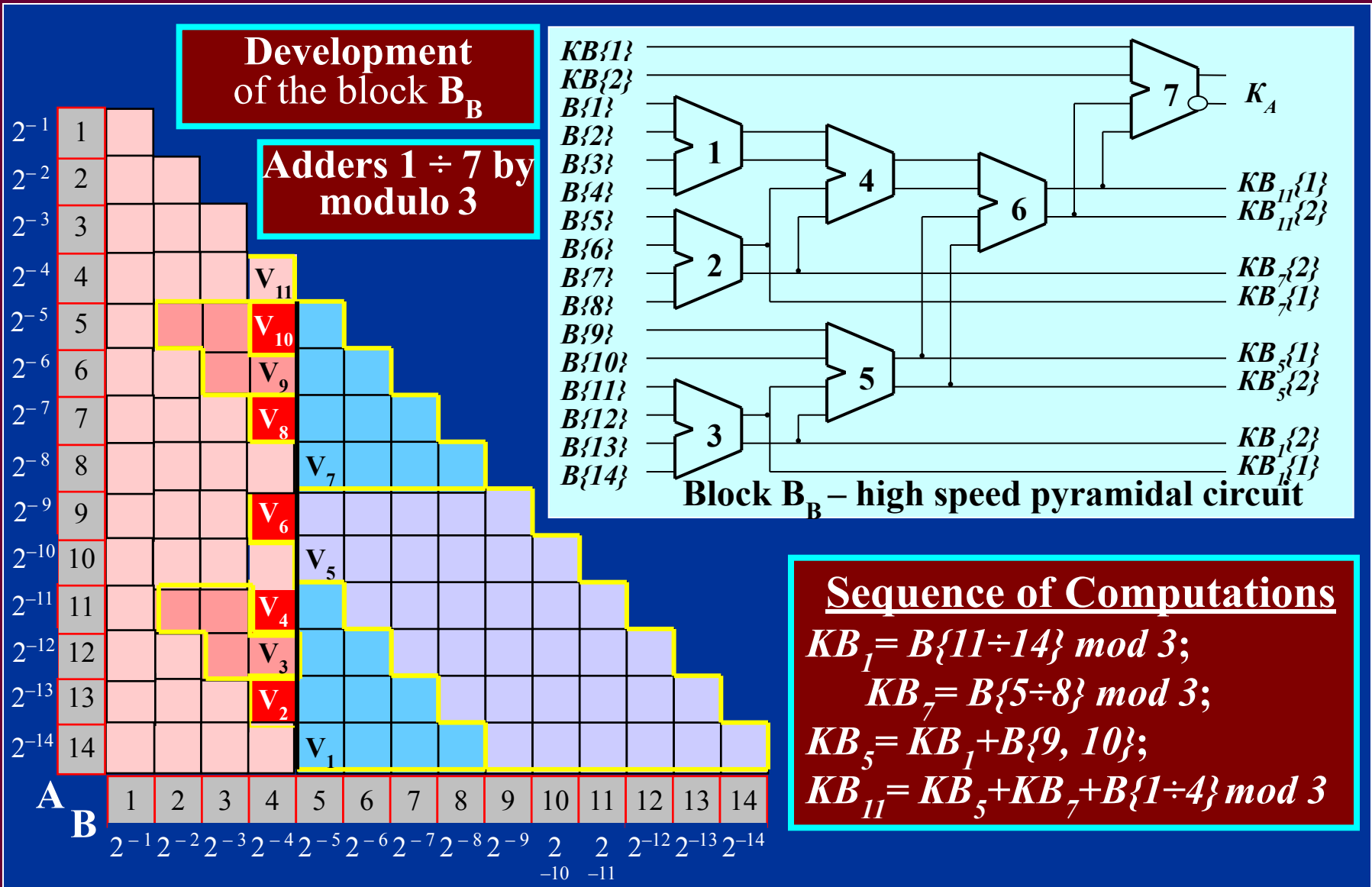
$$\begin{aligned}
 V_2 &= +A\{5\} B\{13\} 2^{-18} \\
 V_4 &= +A\{7\} B\{11\} 2^{-18} \\
 V_6 &= +A\{9\} B\{9\} 2^{-18} \\
 V_8 &= +A\{11\} B\{7\} 2^{-18} \\
 V_{10} &= +A\{13\} B\{5\} 2^{-18}
 \end{aligned}$$

Composed
check codes

$$\begin{aligned}
 KA_2 &= (A\{5\} 2^{-18}) \bmod 3 = -A\{5\}; \\
 KA_3 &= (A\{5, 6\}) \bmod 3 = A\{5, 6\}; \\
 KA_4 &= -A\{7\}; KA_6 = -A\{9\}; \\
 KA_8 &= -A\{11\}; KA_9 = A\{11, 12\}; \\
 KA_{10} &= -A\{13\};
 \end{aligned}$$

$$\begin{aligned}
 KB_2 &= -B\{13\}; KB_3 = B\{11, 12\}; \\
 KB_4 &= -B\{11\}; KB_6 = -B\{9\}; \\
 KB_8 &= -B\{7\}; KB_9 = B\{5, 6\}; \\
 KB_{10} &= -B\{5\};
 \end{aligned}$$

6.4. Residue checking a truncated multiplication



6.4. Residue checking a truncated multiplication

Hardware overhead

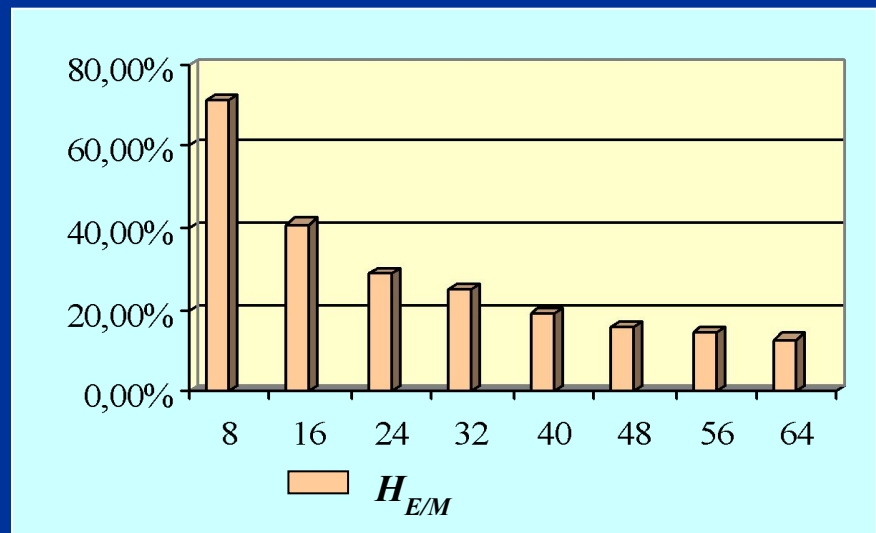
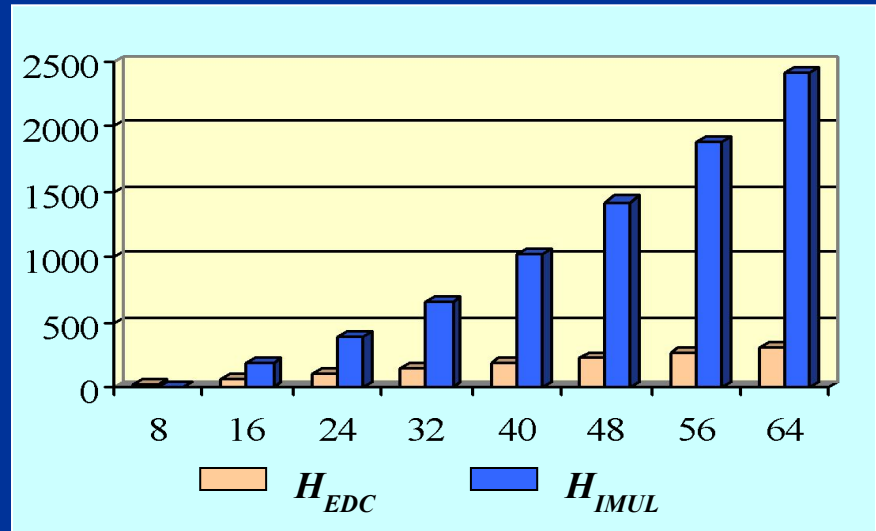
- of Error Detection Circuit:
 $H_{EDC} = 4n + k$ (in FA – full adder)

- of Multiplier:

$$H_{MUL} = n^2 - k^2 / 2 \text{ (in FA)}$$

- Relative

$$H_{E/M} = (8n + 2k) / (2n^2 - k^2)$$



6.5. Residue checking a truncated division of mantissas

Correlation of truncated multiplication and division


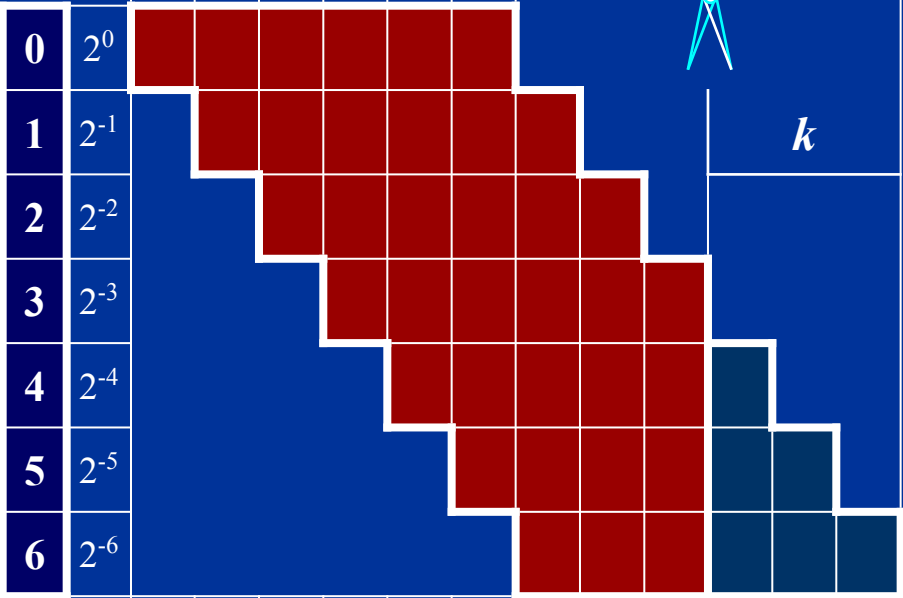
A truncated non-restoring division is an inverse operation for truncated multiplication of the binary *divisor* on *quotient* represented in notation $1, \square 1$.

Truncated multiplication of *divisor* $D = d\{1 \div n\} \cdot 2^{-n}$ on *quotient* $Q = q\{0 \div n\} \cdot 2^{-n}$ determines left part 1 of Conjunctions Array (CA).

Truncated $(2n - k)$ -bits product

$V_{TR} = V\{1 \div 2n - k\} \cdot 2^{-(2n-k)}$,
 is calculated on this part as
 $V_{TR} = A - R_{TR}$ where
 $A = a\{1 \div n\} \cdot 2^{-n}$ is *dividend*;
 $R_{TR} = r\{1 \div n-k\} \cdot 2^{-(n-k)}$ is *truncated remainder*.

CA for product of divisor on quotient

Quotient $Q\{0 \div n\}$		1	2	3	4	5	6	Divisor $D\{1 \div n\}$											
		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}												
0	2^0																		
1	2^{-1}													k					
2	2^{-2}																		
3	2^{-3}																		
4	2^{-4}																		
5	2^{-5}																		
6	2^{-6}																		
		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}						

6.5. Residue checking a truncated division of mantissas

Decomposition of the CA left part on $k+1$ fragments

$$V_i = D_i \cdot Q_i, i = 1 \div k+1 \quad (k=3, i = 1 \div 4)$$

Quotient $Q\{0 \div n\}$	1	2	3	4	5	6	Divisor $D\{1 \div n\}$			
	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}				
0	2^0									
1	2^{-1}									
2	2^{-2}									
3	2^{-3}								V_4	
4	2^{-4}								V_3	
5	2^{-5}								V_2	
6	2^{-6}								V_1	
		2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}
Dividend $A\{1 \div n\}$	1	2	3	4	5	6				

$$\begin{aligned}
 V_1 &= D\{1 \div 3\} \cdot Q\{6\} \cdot 2^{-9}; \\
 V_2 &= D\{1 \div 4\} \cdot Q\{5\} \cdot 2^{-9}; \\
 V_3 &= D\{1 \div 5\} \cdot Q\{4\} \cdot 2^{-9}; \\
 V_4 &= D\{1 \div 6\} \cdot Q\{0 \div 3\} \cdot 2^{-9}.
 \end{aligned}$$

$$\begin{aligned}
 KD_1 &= -D\{1 \div 3\} \bmod 3; \\
 KD_2 &= (KD_1 + D\{4\}) \bmod 3; \\
 KD_3 &= (KD_2 - D\{5\}) \bmod 3; \\
 KD_4 &= (KD_3 + D\{6\}) \bmod 3;
 \end{aligned}$$

$$\begin{aligned}
 KQ_1 &= Q\{6\}; \\
 KQ_2 &= -Q\{5\}; \\
 KQ_3 &= Q\{6\}; \\
 KQ_4 &= -Q\{0 \div 3\} \bmod 3;
 \end{aligned}$$

6.5. Residue checking a truncated division of mantissas

Error Detection circuit

$$KV_{TR} = \sum_{i=1}^{k+1} KV_i$$

$$KV_{TR}^* = KA - KR_{TR}$$

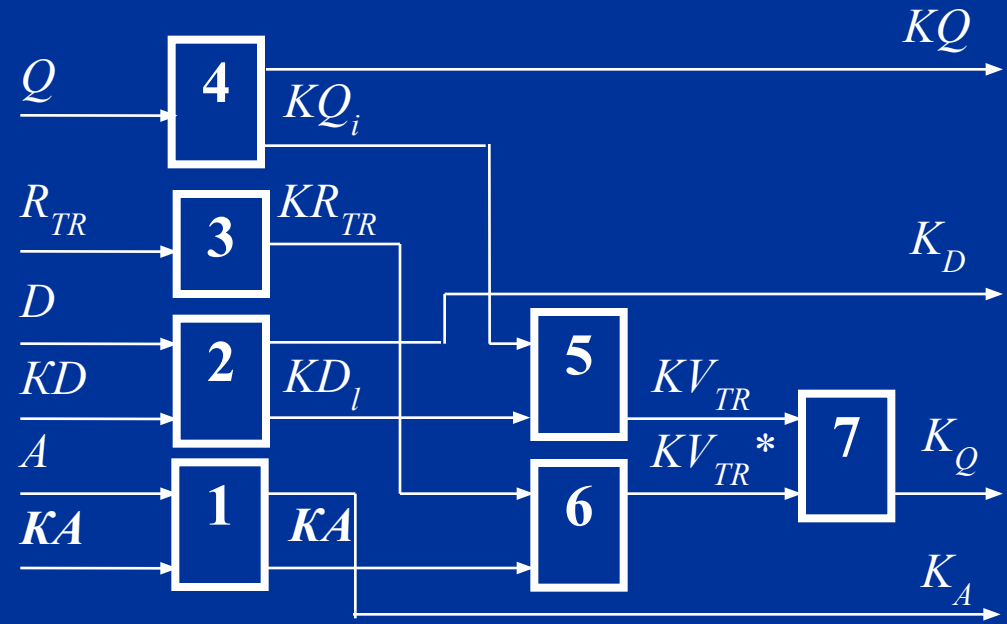
where $KA = A \bmod m$;

$$KR_{TR} = R_{TR} \bmod m;$$

$$KV_i = KD_i \cdot KQ_i;$$

$$KD_i = D_i \bmod m;$$

$$KQ_i = Q_i \bmod m.$$



Blocks 1 and 2 check the input numbers: dividend A and divisor D .
 Blocks 3 and 4 generate check codes KQ and KR of quotient Q and residue R .
 Blocks 5 and 6 calculate check codes KV_{TR} and KV_{TR}^* .
 Block 7 compares check codes KV_{TR} , KV_{TR}^* and calculates indicate code K_Q .

6.6. Residue checking a truncated operation of shift

Truncated shift is executed in floating-point addition

1. Definition of operation $C=A+B$,
where $A=a_1 \cdot 2^{a_2}$; $B=b_1 \cdot 2^{b_2}$;

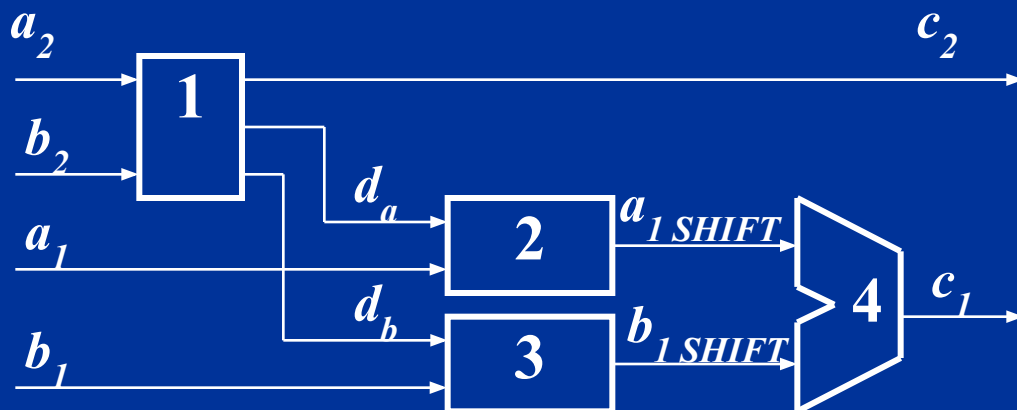
2. Execution of operation

2.1. Processing the exponents

$$c_2 = \max(a_2, b_2);$$
$$d_a = c_2 - a_2; \quad d_b = c_2 - b_2.$$

2.2. Processing the mantissas

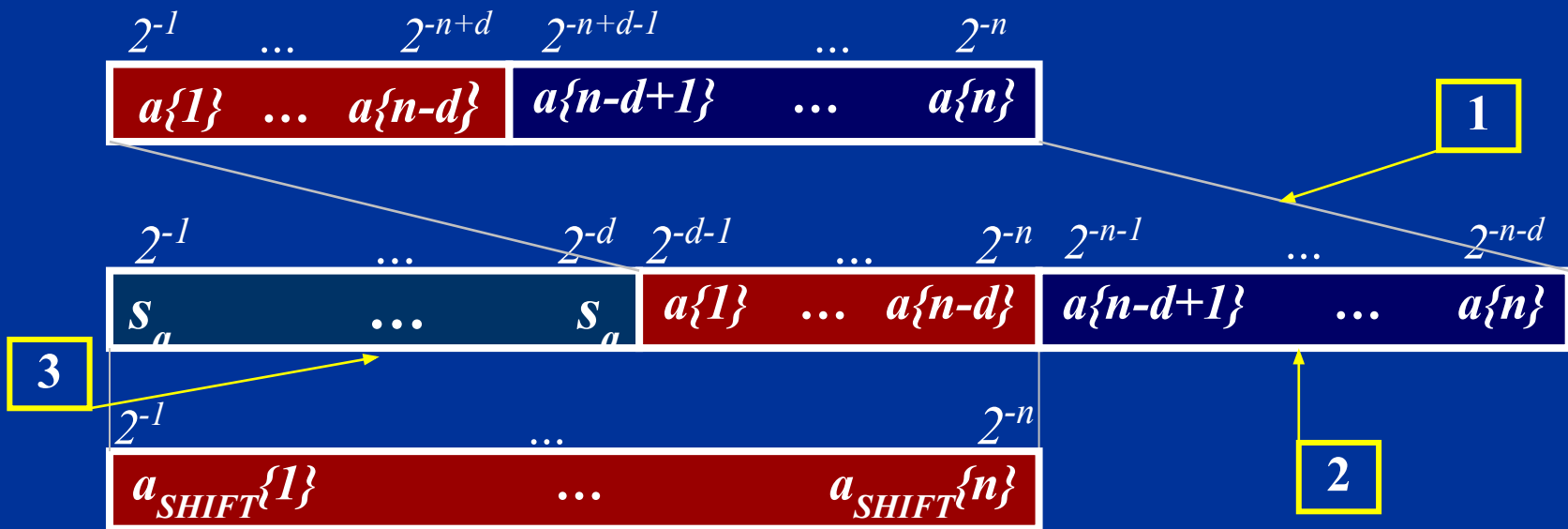
$$a_{1 \text{ SHIFT}} = a_1 \cdot 2^{-d_a};$$
$$b_{1 \text{ SHIFT}} = b_1 \cdot 2^{-d_b};$$
$$c_1 = a_{1 \text{ SHIFT}} + b_{1 \text{ SHIFT}}$$



3. The floating-point adder consists of the block 1 for the exponent processing, barrel-shifters 2 and 3, adder 4.

6.6. Residue checking a truncated operation of shift

Arithmetic shift of a mantissa

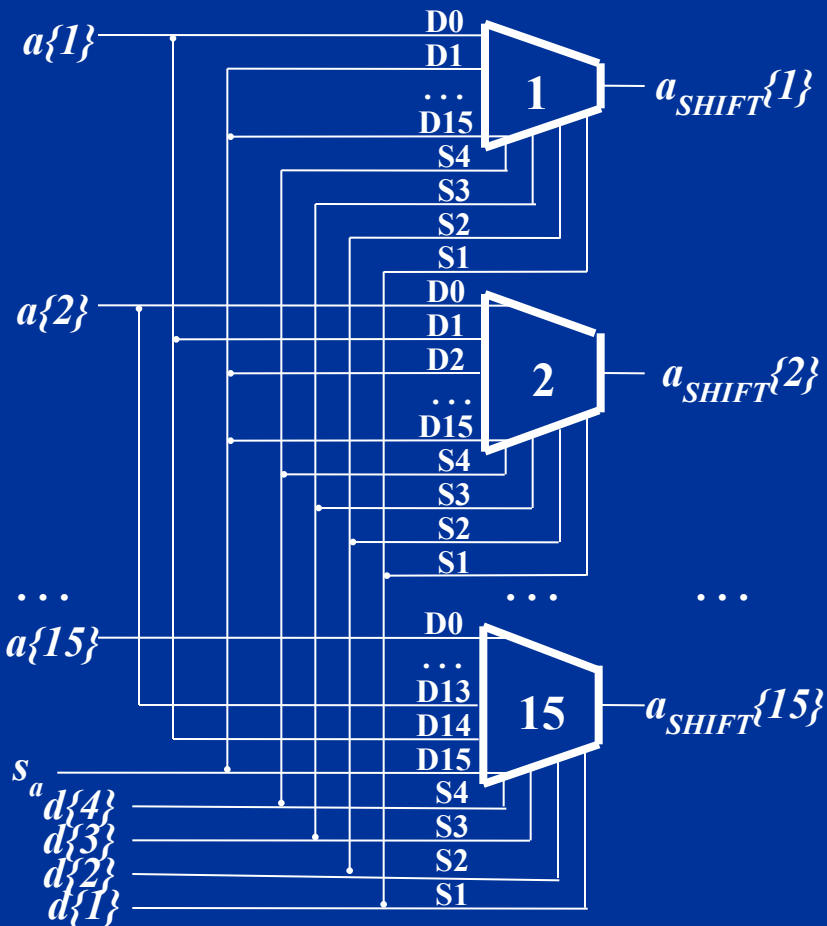


An operation of arithmetic shift contains three actions: $a_{SHIFT} = a \cdot 2^{-d} - a_0 + a_s$.

1. The reduction of the bit weights for the mantissa a in 2^d times.
2. The truncation of the d low bits of the mantissa a (the code $a_0 = a\{n-d+1:n\}$).
3. The sign bit padding in the position with bit weights $2^{-1} \div 2^{-d}$ for complement code of the mantissa a . Sign bits $s_a \dots s_a$ compose the code a_s .

6.6. Residue checking a truncated operation of shift

Arithmetic shift is executed using the Barrel-shifter



The Barrel-shifter contains n of n -to-1 multiplexers.

The multiplexer hardware overhead q is proportional to the operand size n .

The barrel-shifter hardware overhead $Q_{SHIFT} = nq$ is proportional to the square of the operand size n and makes the main hardware overhead of the floating-point adder.

Barrel-shifter executes a **truncated** operation, which reduces twice the hardware overhead in comparison with the long shifter computing complete $2n$ -bit result $a_C = a_{SHIFT\{1 \div 2n\}} 2^{-2n}$.

6.6. Residue checking a truncated operation of shift

Conversion a_0 into $a_{01} = a_0 \cdot 2^d$

d				$i=1 \div n$																			
4	3	2	1	1	2	3	4	...	12	13	14	15											
2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	...	2^{-12}	2^{-13}	2^{-14}	2^{-15}											
0	0	0	0																				
0	0	0	1									15	15										
0	0	1	0								14	15	14	15									
0	0	1	1							13	14	15	13	14	15								
0	1	0	0						12	13	14	15	12	13	14	15							
...																	
1	1	0	0				4	...	12	13	14	15	4	...	12	13	14	15					
1	1	0	1			3	4	...	12	13	14	15	3	4	...	12	13	14	15				

6.6. Residue checking a truncated operation of shift

Conversion a_{01} into a_{02} with keeping the bit weights by mod 3

d				$f_p, i=1 \div n$											$F_p, j=1 \div 2^r$															
4	3	2	1	1	2	3	4	5	6	7	8	9	...	14	15	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	...	2^{-14}	2^{-15}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}
				1	2	1	2	1	2	1	2	1	...	2	1	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
0	0	0	0																											
0	0	0	1												15	15														
0	0	1	0											14	15		14	15												
0	0	1	1										...	14	15	13	14	15												
0	1	0	0										...	14	15					12	13	14	15							
0	1	0	1										...	14	15	11				12	13	14	15							
0	1	1	0										...	14	15		10	11	12	13	14	15								
0	1	1	1								9	...		14	15	9	10	11	12	13	14	15								
1	0	0	0							8	9	...		14	15								8	9	10	11	12	13	14	15
1	0	0	1							7	8	...		14	15	7							8	9	10	11	12	13	14	15

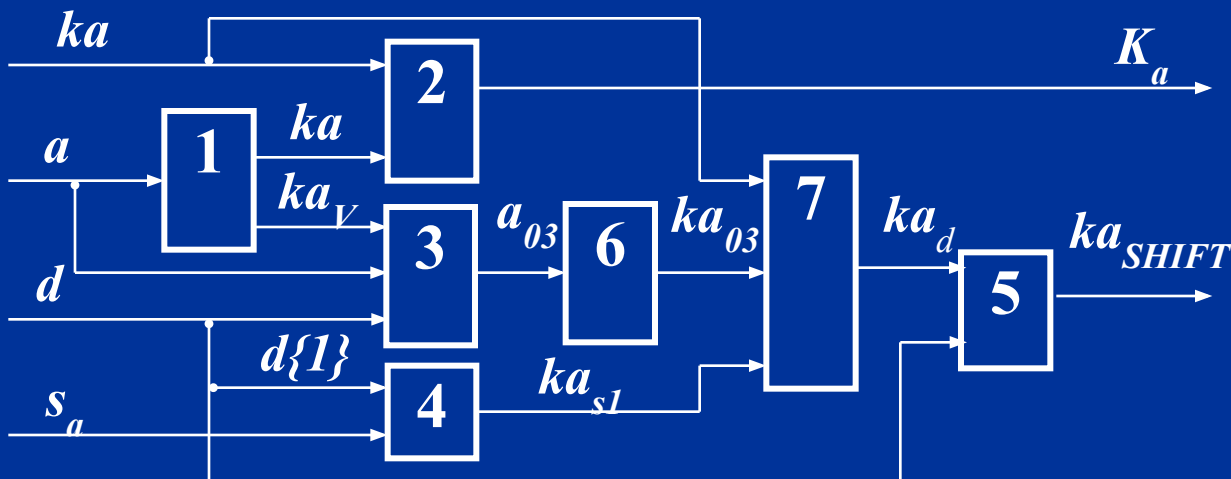
6.6. Residue checking a truncated operation of shift

Conversion a_{01} into a_{02} with calculating the check codes

d				$F_j, j=1 \div 2^r$												$V_p, l=1 \div 2r-1$										
4	3	2	1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	3	4	5	6	7	
2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}								
				1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	1	2	1	2	1	2	1	
0	0	0	0																							
0	0	0	1	15				$ka_{12 \div 15}\{2,1\} = a\{12 \div 15\} \bmod 3$										15								
0	0	1	0		14	15													14	15						
0	0	1	1	13	14	15												13	14	15						
0	1	0	0				12	13	14	15		$ka_{8 \div 15}\{2,1\} = (a\{8 \div 11\} + ka_{12 \div 15}\{2,1\}) \bmod 3$											$ka_{12 \div 15}\{2,1\}$			
0	1	0	1	11			12	13	14	15								11				$ka_{12 \div 15}\{2,1\}$				
0	1	1	0		10	11	12	$ka_{4 \div 7}\{2,1\} = a\{4 \div 7\} \bmod 3$											10	11	$ka_{12 \div 15}\{2,1\}$					
0	1	1	1	9	10	11	12											9	10	11	$ka_{12 \div 15}\{2,1\}$					
1	0	0	0								8	9	10	11	12	13	14	15							$ka_{8 \div 15}\{2,1\}$	

6.6. Residue checking a truncated operation of shift

Simplification of the checking computation



1. Conversion of the restricted bits a_0 in the code a_{01} simplifies the unit 3 in $\sigma_{01} = 1.5$ times.

2. Conversion of the code a_{01} in a_{02} simplifies the unit 3 in $\sigma_{02} = 2n/r$ times. For $n=15$ $\sigma_{02} = 7,5$.

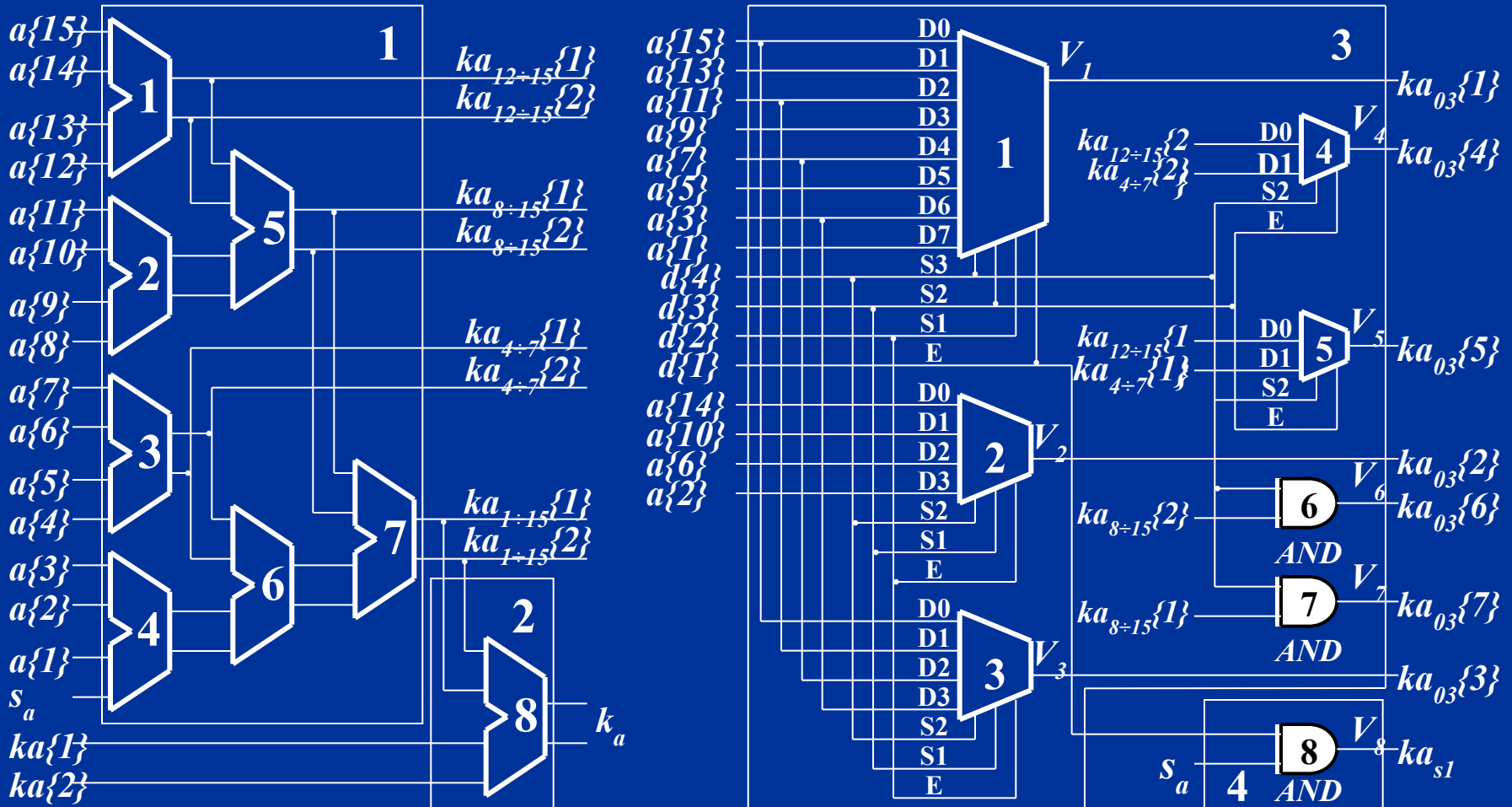
3. Conversion of the code a_{02} in a_{03} simplifies the unit 3 in $\sigma_{03} = 2n/3$ times and the unit 6 in $\sigma = n/(2r-1)$ times. For $n=15$ $\sigma_{03} = 10$, $\sigma = 2.1$.

The checking hardware overhead reduces from square dependence on the operand size to linear one.

6.6. Residue checking a truncated operation of shift

Unit 1: modulo-3 generator
 Unit 2: modulo-3 comparator

Unit 3: generator of the check code ka_{03}
 Unit 4: generator of the check code ka_{s1}



Reading List

1. Дрозд А.В. Нетрадиционный взгляд на рабочее диагностирование вычислительных устройств / А.В. Дрозд // Автоматизированные системы управления и приборы автоматики. – 2009. – Вып. 147. – С. 15 – 24.
2. Дрозд О.В. Контроль за модулем обчислювальних пристроїв. Навч. посібн. для студ. спеціальності 7.091501 – «Комп'ютерні та інтелектуальні системи та мережі» / Одеськ. нац. політехніч. ун.-т. – Одеса: АО Бахва, 2002. – 144 с.
3. Контроль ОП зі скороченим виконанням операцій, с. 74 – 135.
- 3 Drozd A. V., Lobachev M. V. Efficient On-line Testing Method for Floating-Point Adder. – Proc.. Design, Automation and Test in Europe. Conference and Exhibition 2001 (DATE 2001). Munich, Germany, 13 – 16 March 2001. – P. 307 – 311.
- 4 Drozd A. V., Lobachev M.V., Drozd J. V. Efficient On-line Testing Method for a Floating-Point Iterative Array Divider. – Proc. Design, Automation and Test in Europe. Conference and Exhibition 2002 (DATE 2002). Paris, France, 4 – 8 March 2002. – P. 1127.

Conclusion

1. **Traditional on-line testing methods** have **low reliability of approximated result checking**: mainly detect inessential errors.
2. **On-line testing reliability** can be **increased** by **three ways**: **increasing** a probability of essential error; **reducing** a probability of error detection and also **detecting essential and inessential errors** with **different probabilities**.
3. The **first way** can be realized using **truncated operations only** because only these operations can have the high probability of essential error.
4. The **first way** allows to develop **the on-line testing methods** with **traditionally high probability of error detection**
5. **The truncated multiplication** can be checked by modulo using decomposition of product conjunction array into **fragments**.
6. **The another truncated operations** can be checked using fragment approach as well as they **inherit the properties of multiplication**.

Questions and tasks

1. What is a **reliability of the on-line testing methods**?
2. What **reliability** do **the traditional on-line testing methods** demonstrate in **approximate data processing**?
3. Describe the **ways** to increase **reliability of the traditional on-line testing methods** for **approximate data processing**.
4. What **conditions** does **the first way** use for increasing the **reliability of the on-line testing methods**?
5. What role do the **truncated arithmetic operations** play in mantissa checking?
6. What approach does the residue checking method use for truncated operations?

MODULE 3. On-line testing for digital components of S-CES

Lecture 7. Increase of on-line testing methods reliability

7.1. The second way for increasing on-line testing reliability

7.2. Checking with use of natural information redundancy

7.3. The use of product information redundancy

7.4. Checking of a squarer

7.5. Checking by simplified operation

7.6. The models of operation simplification

7.7. Execution of check calculations

7.1. The second way for increasing on-line testing reliability

7.1.1. Motivation of increasing an on-line testing reliability by the second way

Second way answers a common case of on-line testing objects.

Reasons:

The second way increases on-line testing reliability using a low probability of essential error.

On-line testing objects, as a rule, have a low probability of essential error.

7.1.2. Related Works

1. Савченко Ю. Г. Цифровые устройства, нечувствительные к неисправностям элементов. – М.: Советское радио, 1977. – 176 с.
2. Сушкевич А. К. Теория чисел. – Харьков: Изд. ХГУ, 1956.
3. Селлерс Ф. Методы обнаружения ошибок в работе ЭЦВМ. – М.: Мир, 1972. – 310 с.
4. Граф Ш., Гессель М. Схемы поиска неисправностей. – М.: Энергоатомиздат, 1989. – 144 с.

7.1. The second way for increasing on-line testing reliability

7.1.3. Features of the second way

In case of a low probability of essential error the increase of on-line testing reliability can be achieved only reducing a probability of error detection.

Reduction requirements to error detection promote simplification of the check circuits.

Earlier reduction of an error detection probability has been aimed at simplification of the on-line testing means.

However now the goal is **increase of reliability of the on-line testing methods**. This goal can be achieved with **simplification of the check circuits**.

7.1. The second way for increasing on-line testing reliability

7.1.3. Features of the second way

The main requirement to reduction of an error detection probability is **to keep a set of detected faults.**

Every probable fault should be detected **at least an input codeword.**

The probable fault distorts a result at the output of single-step arithmetic circuits **on the weight of any one bit.**

The error looks like $\pm 2^r$, where r is number of the result bit.

The set of faults **detected by residue checking (modulo three)** can be used as **the comparison templet** of set of the probable faults.

7.2. Checking with use of natural information redundancy

7.2.1. Natural information redundancy

The code containing **the forbidden words** is characterized by its **information redundancy**.

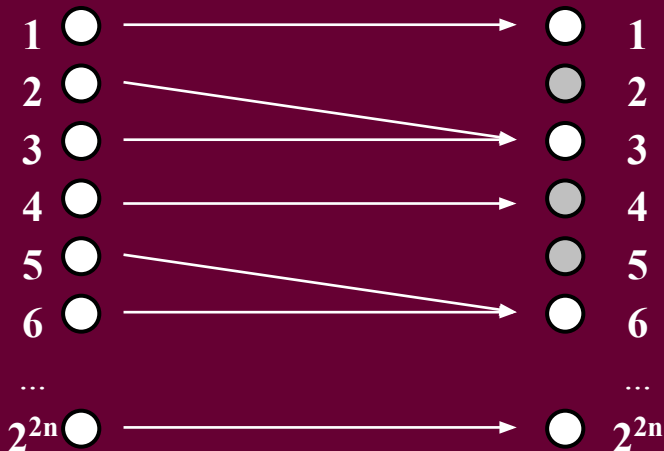
Natural information redundancy is alternative to **information redundancy created by expansion of a code** introducing the additional bits.

Considered checking methods use **natural information redundancy** of the arithmetic operation results.

7.3. The use of product information redundancy

A product of complete operation has **natural information redundancy**.

Really **the product** contains **the forbidden words**.
This follows from execution of **the commutative law** or **multiplication to zero**



Both sets of input and output words of multiplication have the same capacity 2^{2n} , where n is size of operands.

However the same output word can correspond to several input words.

7.3. The use of product information redundancy

Checking the products using prime numbers

Fermat (1601-1665) supposition: the number $C = 2^n + 1$, $n=2^x$ (x is natural number) are **prime**.

x	0	1	2	3	4
n	1	2	4	8	16
C	3	5	17	257	65537

Euler (1707-1783) refuted of Fermat statement for $x = 5$, but the statement are true for $x < 5$ including the cases of wide-spread word size $n = 8$ and $n = 16$.

A prime number $C = 2^n + 1$ cannot be a product of two n -bit binary factors.

Bits of product for $n = 8$															
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

7.3. The use of product information redundancy

Checking the products using prime numbers

A prime number $C = 2n+1$ and numbers which is multiply to C are forbidden words for a product of two n -bit binary factors.

These words compose double code $G(n, n)$ without zero-word.

n high bits of a product							n low bits of a product							Forbidden words			
$2n$		$n+1$	n			1	(2^n+1)	×	k
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	(2^8+1)	×	1
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	(2^8+1)	×	2
0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	(2^8+1)	×	3
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	(2^8+1)	×	4
			(2^8+1)	×	...
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	(2^8+1)	×	(2^8-1)

7.3. The use of product information redundancy

Checking the products using prime numbers

The checking method verifies that:

- multipliers $A\{1 \div n\}$ and $B\{1 \div n\}$ are not zero
- product $V\{1 \div 2n\}$ is forbidden word $k(2^n + 1)$.

Error is detected, if only one of two conditions performs:

$$(A\{1 \div n\} \neq 0) \ \& \ (B\{1 \div n\} \neq 0);$$
$$V\{1 \div n\} = V\{n + 1 \div 2n\}.$$

Every probable fault of iterative array multiplier is detected **at least on one input word**: $A\{1 \div n\} B\{1 \div n\} \pm 2^r = k(2^n + 1)$.

It is proved by factorization of the formula $k(2^n + 1) \pm 2^r$ on multipliers $A\{1 \div n\}$ and $B\{1 \div n\}$ **at least for one value k** .

7.3. The use of product information redundancy

Checking the products using prime numbers

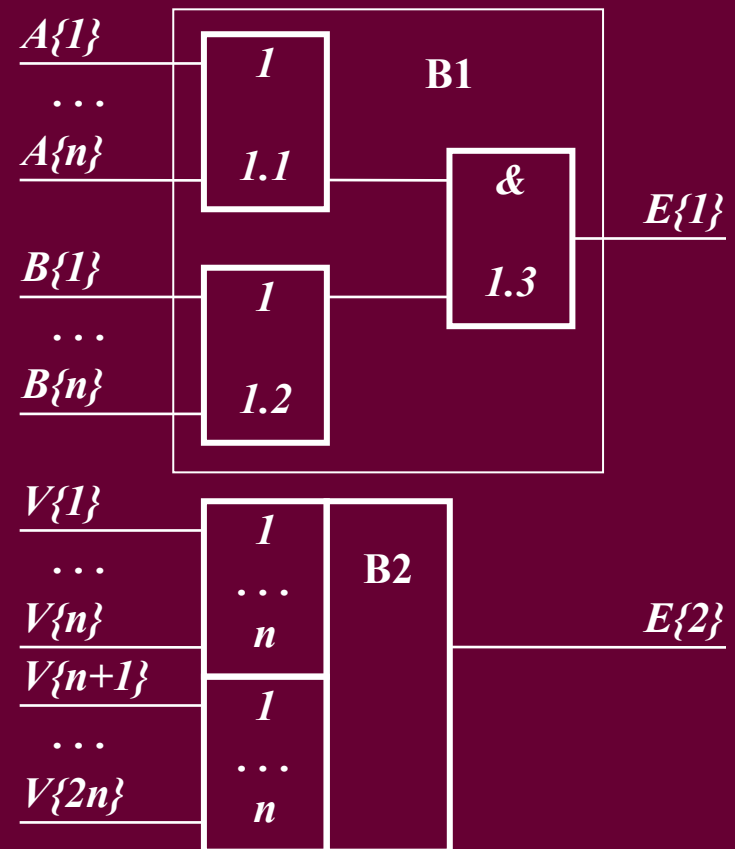
The code $E\{1, 2\} = 00_2$, if at least one of factors is zero and the product is not zero: the low and high parts of product are different.

The code $E\{1, 2\} = 11_2$, if both of the factors are not zero and the product assumes forbidden word: the low and high bits of product are equal.

The code $E\{1, 2\} = 01_2$, if at least one of the factors is zero and the low and high bits of product are equal: $V\{1 \div 2n\} = 0$.

The code $E\{1, 2\} = 10_2$, if both of the factors are not zero and the low and high parts of non-zero product are different.

If $E\{1, 2\} = 00_2$ or 11_2 then **fault is detected**;
If **work** is **correct** then $E\{1, 2\} = 01$ or 10 .



7.3. The use of product information redundancy

Checking the products using prime numbers

This checking method can be extended on **mantissa processing** taking into account a range of the normalized mantissa codeword: $2^{n-1} \div 2^n - 1$.

Such range excludes **zero** as a value of **a product**.

This peculiarity eliminates a check of factors to be equal to zero and eliminates the block B1 of the checker.

The checker contains only the comparator (Block B2) which can be designed on Carter's units.

7.3. The use of product information redundancy

Checking the products using prime numbers

A probability of error detection $P_D = 3 \cdot 2^{-n}$,
 $P_{D n=8} = 0,012$; $P_{D n=16} = 4,6 \cdot 10^{-5}$.

A reliability of the checking method $R = 1 - P_E$,
 $R = 0,9$ for $P_E = 0,1$.

Time of permanent fault detection $T = \ln 2 / P_D$,
 $T_{n=8} = 59$; $T_{n=16} = 15142$ (clock units);

The checker based on use of prime numbers is simplest for multipliers. It is simpler of the residue checker more than 5,3 times.

7.3. The use of product information redundancy

Checking the products using prime numbers

The described checking method has such lack as limited application: only for two size of word – $n = 8$ and $n = 16$.

This checking method can be extended on another size of word using prime number $C^* = 2^n - 1$.

n	3	5	7	13	17	19	31
C^*	7	31	127	8191	131071	524287	2147483647

A prime number $C^* = 2^n - 1$ can be a product of two n -bit binary factors only in case the factor is equal to C^* .

Bits of product for $n = 7$													
14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1

7.3. The use of product information redundancy

Checking the products using prime numbers

A prime number $C^* = 2^n - 1$ and numbers which is multiply to C^* can be a product of two n -bit binary factors.

These words compose double code $G(n, \neg n)$ with inverse part without words which are equal to C^* in their high part.

n high bits of a product							n low bits of a product							C^*		
$2n$			$n+1$	n			1	(2^n-1)	×	k
0	0	0	0	0	0	0	1	1	1	1	1	1	1	(2^7-1)	×	1
0	0	0	0	0	0	1	1	1	1	1	1	1	0	(2^7-1)	×	2
0	0	0	0	0	1	0	1	1	1	1	1	0	1	(2^7-1)	×	3
0	0	0	0	0	1	1	1	1	1	1	1	0	0	(2^7-1)	×	4
				(2^7-1)	×	...
1	1	1	1	1	1	0	0	0	0	0	0	0	1	(2^7-1)	×	(2^7-1)

7.3. The use of product information redundancy

Checking the products using prime numbers

The checking method verifies that:

- multipliers $A\{1 \div n\}$ and $B\{1 \div n\}$ are not C^* and not zero
- product $V\{1 \div 2n\}$ is word $k (2^n - 1)$.

Error is detected, if only one of two conditions performs:
 $(A\{1 \div n\} \neq C^*) \& (B\{1 \div n\} \neq C^*)$ for $A\{1 \div n\}$, $(B\{1 \div n\} \neq 0$
 $V\{1 \div n\} = \neg V\{n + 1 \div 2n\}$.

Every probable fault of iterative array multiplier is detected **at least on one input word**: $A\{1 \div n\} B\{1 \div n\} \pm 2^r = k (2^n - 1)$.

It is proved by factorization of the formula $k (2^n - 1) \pm 2^r$ on multipliers $A\{1 \div n\}$ and $B\{1 \div n\}$ **at least for one value k** .

7.3. The use of product information redundancy

Checking the products using prime numbers

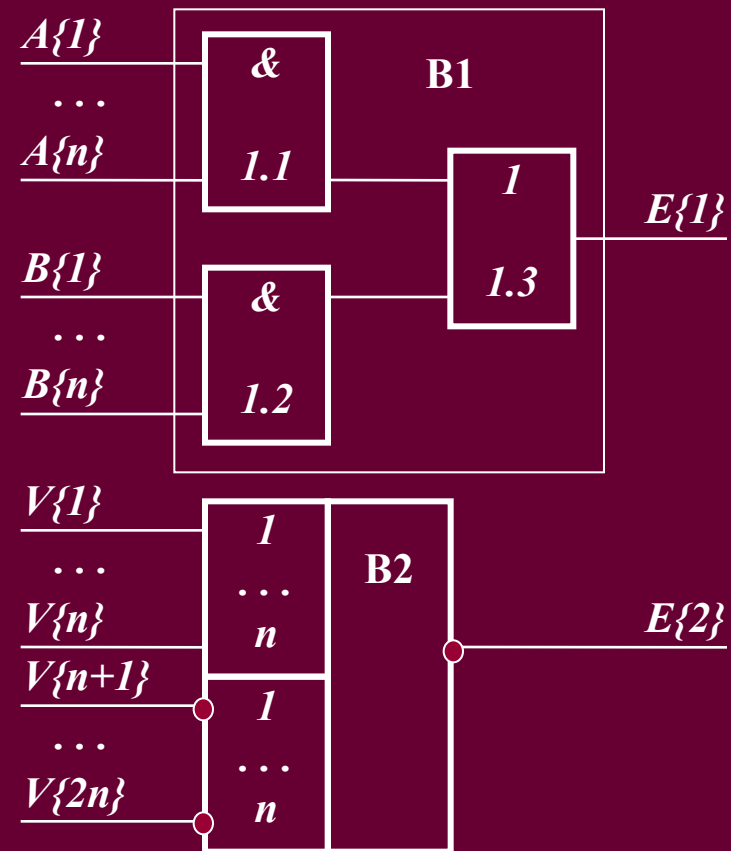
The code $E\{1, 2\} = 11_2$, if at least one of factors is C^* and the low and high parts of product are not inverse.

The code $E\{1, 2\} = 00_2$, if both of the factors are not equal to C^* and the low and high bits of product are inverse.

The code $E\{1, 2\} = 01_2$, if at least one of the factors is C^* and the low and high bits of product are inverse.

The code $E\{1, 2\} = 10_2$, if both of the factors are not equal to C^* and the low and high parts of non-zero product are not inverse.

If $E\{1, 2\} = 00_2$ or 11_2 then **fault is detected**;
If **work is correct** then $E\{1, 2\} = 01$ or 10 .



7.3. The use of product information redundancy

Checking the products using prime numbers

The checking method is not correct in case at least one of factors is equal to zero. This case should be identified in checker additionally for codeword in range $0 \div 2^n - 1$.

Both the checking method and **checker** are quite correct for **mantissa processing** taking into account a range of the normalized mantissa codeword: $2^{n-1} \div 2^n - 1$.

7.3. The use of product information redundancy

Checking the products using prime numbers

A probability of error detection $P_D = 3 \cdot 2^{-n}$,
 $P_{D n=7} = 0,023$; $P_{D n=17} = 2,3 \cdot 10^{-5}$.

A reliability of the checking method $R = 1 - P_E$,
 $R = 0,9$ for $P_E = 0,1$.

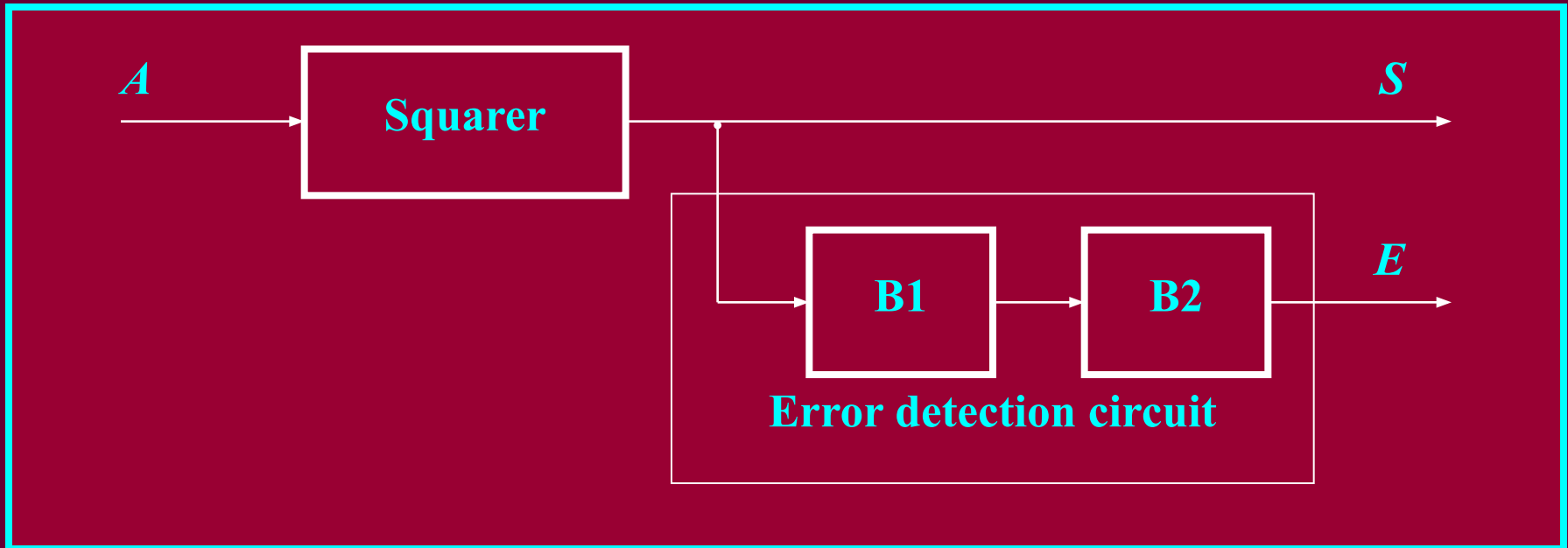
Time of permanent fault detection $T = \ln 2 / P_D$,
 $T_{n=7} = 30$; $T_{n=8} = 30284$ (clock units);

The checker based on use of prime numbers is simplest for multipliers. It is simpler of the residue checker more than 5,3 times.

7.4. Checking of a squarer

- Error detection circuit of squarer

Way 2.
Decrease of P_D



Block B1 calculates residue R by modulo m of result $S = A^2$.

Block B2 calculates check code E which identifies the forbidden values of residue R .

7.4. Checking of a squarer

- Estimation of error detection probability

$$m = 15$$

1. Calculation of square $S = A^2$ and residue $R = S \bmod m$ for values of an operand on the half of period $A = 0 \div (m - 1) / 2$.

A	0	1	2	3	4	5	6	7
S	0	1	4	9	16	25	36	49
R	0	1	4	9	1	10	6	4

2. Creation of a set X of the allowed values x for the residue R and an index F of their occurrences for values of an operand on the period $A = 0 \div m - 1$.

X	0	1	4	6	9	10
F	1	4	4	2	2	2

3. Creation of a set Z of the forbidden values z ;

Z	2	3	5	7	8	11	12	13	14
-----	---	---	---	---	---	----	----	----	----

7.4. Checking of a squarer

- Estimation of error detection probability

4. Creation of a set Y of the typical error $y = \pm 2^r$ by modulo m , where r is number of a bit in result, $r = 0 \div 2n - 1$.

4.1 A set Y of the typical error $y = \pm 2^r$ by modulo m is finite: positive errors not more m and negative errors not more m .

4.2 The typical error $y = \pm 2^r$ by modulo m can be obtained duplicating value of the error by modulo m from 1 before 1 or -1 .

4.3 This process can be considered in detail on example $m = 13$.

$$m = 15$$

Y	1	2	4	8	-1	-2	-4	-8
-----	---	---	---	---	----	----	----	----

$$2^0=1, \quad 1 \times 2=2, \quad 2 \times 2=4, \quad 4 \times 2=8, \\ 8 \times 2=16: 16 \bmod 15 = 1.$$

$$m = 13$$

$$2^0=1, \quad 1 \times 2=2, \quad 2 \times 2=4, \quad 4 \times 2=8, \\ 8 \times 2=16: \\ \quad \underline{-13} \\ \quad \quad 3, 3 \times 2=6, 6 \times 2=12 \\ \quad \quad \quad \underline{-13} \\ \quad \quad \quad \quad -1$$

$$Y \{1, 2, 4, 8, 3, 6\}.$$

7.4. Checking of a squarer

• Estimation of error detection probability

5. Creation of the error detection table using occurrences of allowed values x from condition $z = (x + y) \bmod m$;

X	0	1	4	6	9	10
F	1	4	4	2	2	2

6. Calculation of maximal P_H and minimal P_L error detection probabilities:

$$P_H = \text{Sum}_{MAX} / (m Y^*);$$

$$P_L = \text{Sum}_{MIN} / (m Y^*);$$

where Sum_{MAX} is the sum of all elements of the table;

Sum_{MIN} is the least sum of lines which elements cover all columns;

Y^* is amount of elements in set Y .

$m = 15$

z/y	1	2	4	8	-1	-2	-4	-8	Sum
2	4	1		2		4	2	2	15
3		4		2	4				10
5	4		4		2		2		12
7	2					2		1	5
8		2	4	1	2	2		4	15
11	2	2					1	4	9
12		2		4			4		10
13			2			1		2	5
14			2	2	1	4			9

$$Y^* = 8$$

$$P_{DH} = 0,75$$

$$\text{Sum}_{MAX} = 90$$

$$P_{DL} = 0,15$$

$$\text{Sum}_{MIN} = 18 \text{ for } z = 11 \text{ and } z = 14$$

7.4. Checking of a squarer

- Estimation of the checking method reliability

$$R = P_D P_E + (1 - P_D) (1 - P_E)$$

1. Case of exact data: $P_E = 1$

$$P_D = P_{DH} = 0,75$$

$$R = 0,75$$

	P_E
P_D	P_{DE}
P_S	P_{SE}

	P_E	P_N
P_D	P_{DE}	P_{DN}
P_S	P_{SE}	P_{SN}

2. Case of approximate data: $P_E = 0,1$

$$P_D = P_{DH} = 0,75$$

$$R = 0,30$$

$$P_D = P_{DH} = 0,15$$

$$R = 0,78$$

	P_E	P_N
P_D	P_{DE}	P_{DN}
P_S	P_{SE}	P_{SN}

7.5. Checking by simplified operation

Simplification of operation

The checking method is based on operation simplification limiting of **a set of the input words** down to **the set of check words**.

For example, a multiplier can be checked as **squarer** on input words composed of **equal factors**.

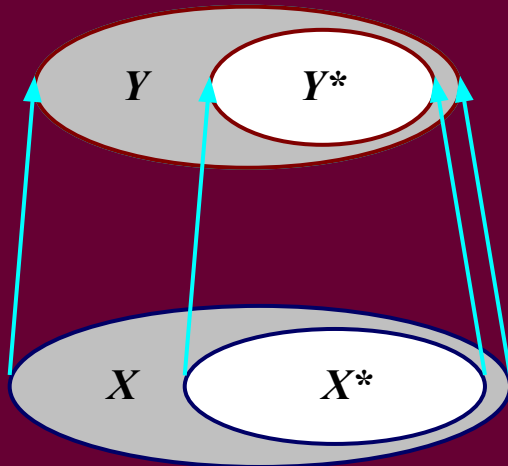
Such solution is **not correct**: **the probable faults** – shorts between the same bits of the factors – **are do not detected**.

This solution can be improved using the factors which are **equal by modulo 3**.

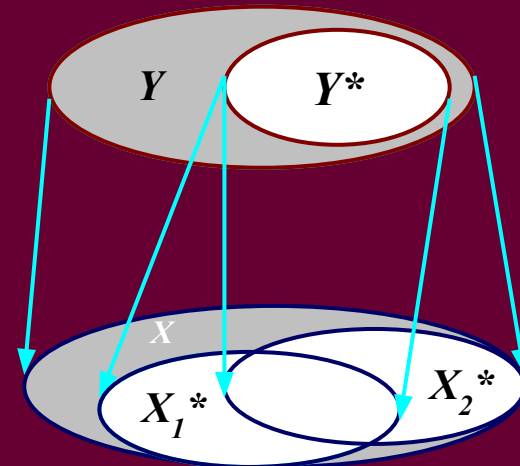
7.5. Checking by simplified operation

Limiting conditions

The method defines limiting conditions for operands and results.



Simplification bottom-up:
limiting conditions imposed upon operands determine limiting condition for the result.



Simplification top-down:
limiting condition imposed upon result determine limiting conditions for the operands.

7.5. Checking by simplified operation

The models of the Operation Simplification

A model of simplification of the computing operation contains limiting conditions (LC) and logic operation executed with their.

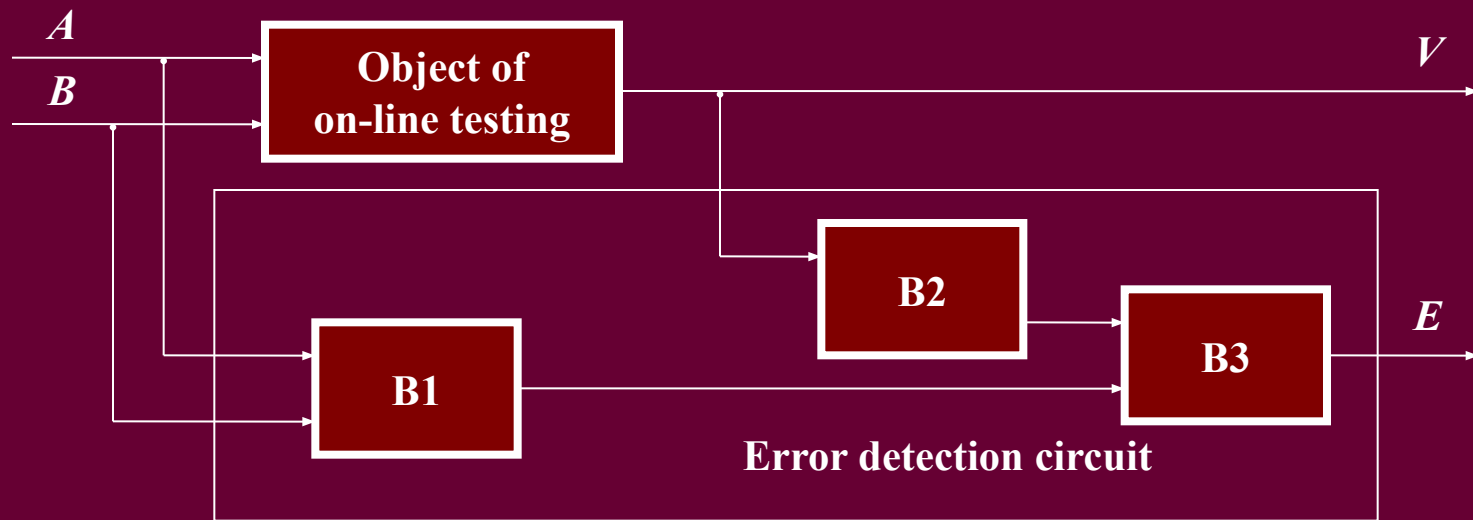
Composite LC is LC for operands composed of some LC.

The LC for operands can be dependent or independent determining equal or different LC for the result accordingly.

In order to keep a set of the detected fault the **dependent** LC should be processed only using logic operations **OR** or **XOR**;
the **independent** LC should be processed only using logic operations **AND**.

7.5. Checking by simplified operation

Structure of the Error detection circuit



Block B1 uses LC for operands identifying the input words, on which the operation can be transformed to simplified form.

Block B2 checks LC for results of the operation considered in simplified form.

Block B3 forms an error indication code, which detects an error only in case of the input word identification in block B1 and detection of this error in block B2.

7.5. Checking by simplified operation

The models of execution of the check calculations

Two kinds of the check calculations are used:

- forming the codes of LC for the operands and the result;
- execution of logic operations with the codes of LC.

The codes of LC are formed by modulo 3 keeping a set of the faults detected if the residue checking.

The codes of LC can take allowed values 01_2 or 10_2 and forbidden values 00_2 or 11_2 .

Both the logic operation OR with allowed values and AND with forbidden values of the LC codes are executed **on a Carter's unit**.

The logic operation NOT transforms the allowed values to forbidden one's or on the contrary inverting one of code bits by NOT-unit.

The Carter's and NOT units allow to execute any logic operation as well as OR, AND, NOT compose functionally complete basis.

7.5. Checking by simplified operation

Design of the Checker

Initial data for checker design is a required probability P_D of error detection. It is used for determining the LC for operands.

For example, the LC for multiplier checker (complete operation) with low $P_D = 0,07$ can be determined as follows.

$$A \times B = V;$$

$$V = R \cdot V_1 \cdot V_2;$$

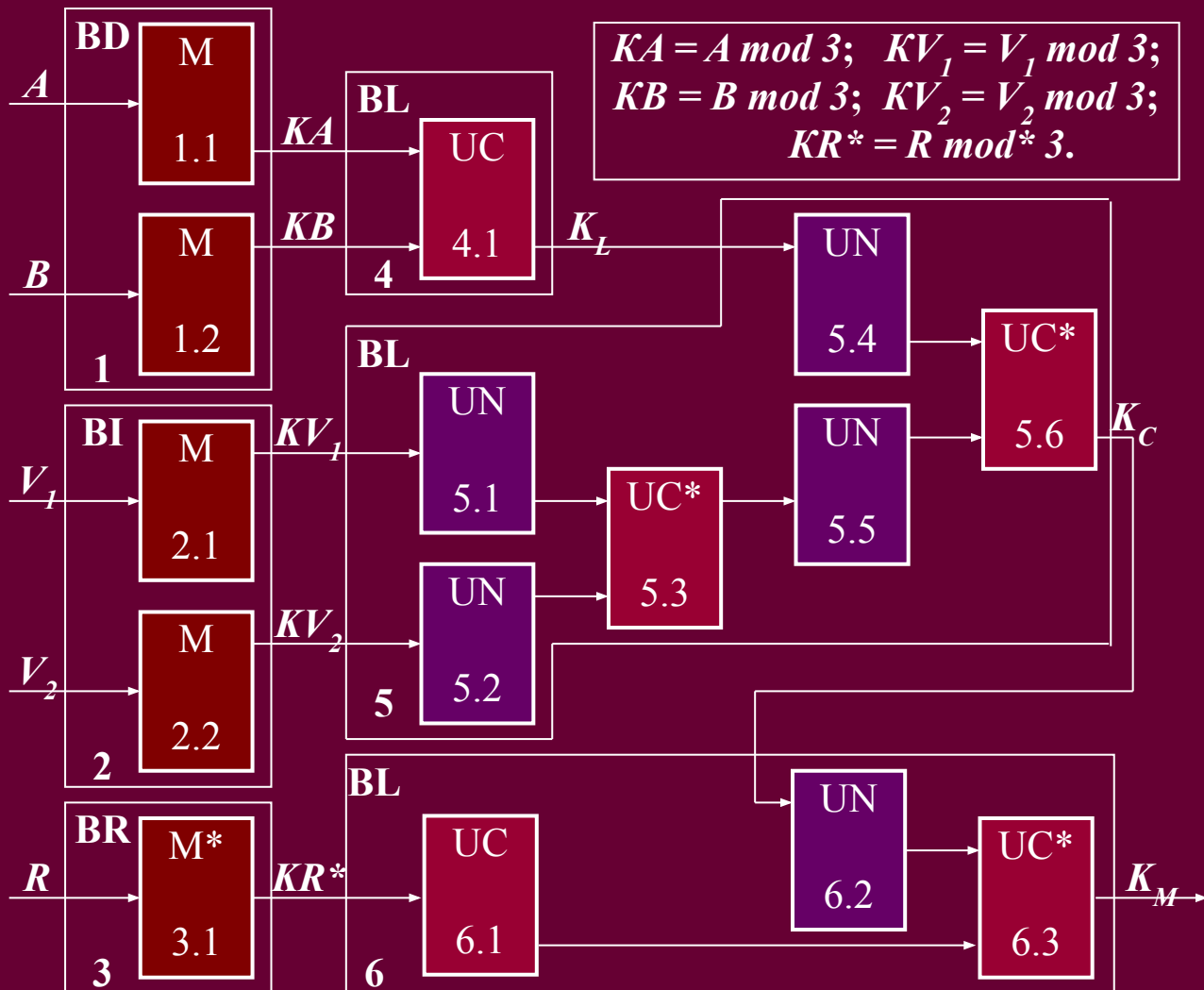
 $V:$
 R
 V_1
 V_2

LC	Type of LC	Set of check words	Logic operation		Set of check words	P_D
$A \bmod 3 = 0$	D	$0,33 G$	OR		$0,56 G$	$0,06$
$B \bmod 3 = 0$	D	$0,33 G$				
$V_1 \bmod 3 = 0$	I	$0,33 G$	AND		$0,06 G$	
$V_2 \bmod 3 = 0$	I	$0,33 G$				
$R \bmod 3 = 0$	R	D – dependent LC, I – independent LC, R – LC for result				

G is a set of total inputs word

7.5. Checking by simplified operation

Design of the Checker

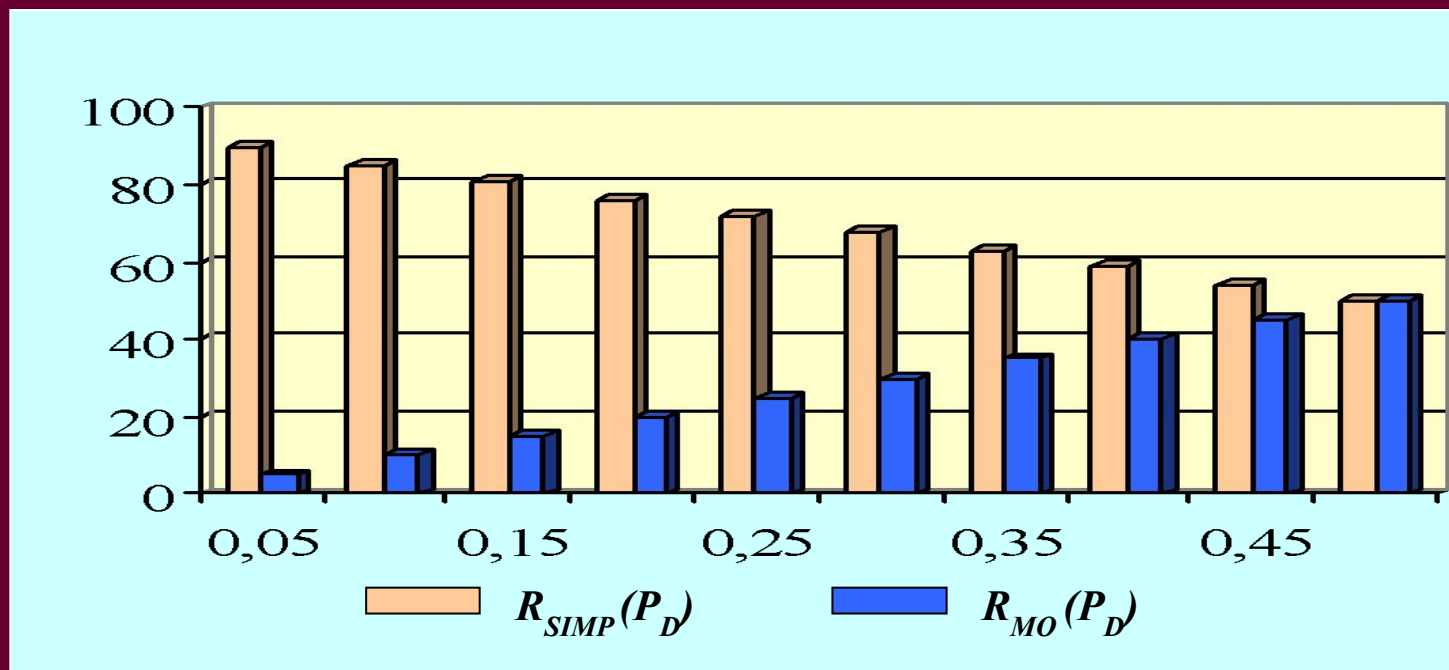


M – the generator of residue code;
UC – the Carter's unit;
UN – the NOT-unit;
 • – the inverse output;
BD – the block forming the dependent LC;
BI – the block forming the independent LC;
BL – the block executing the logic operation with the codes of LC;
 K_L – the composite code of dependent LC;
 K_C – the composite code of independent LC;
 K_M – the code of error indication

7.5. Checking by simplified operation

Estimation of the method

Reliability of the checking by simplified operation
in comparison with the residue checking method



Reading List

1. Drozd A. V. Efficient Method of Failure Detection in Iterative Array Multiplier. – Proc. Design, Automation and Test in Europe. Conference and Exhibition 2000 (DATE 2000). Paris, France, 27 – 30 March 2000. – P. 764.
2. Said Mouafak Montaha M. New On-Line Testing Method to Increase the Reliability of Checking Approximated Results / M. Said Mouafak Montaha, M.V. Lobachev, O.V. Drozd // 4-th international Conference “Advanced Computer Systems and Networks: Design and Application”. Lviv, Ukraine, 17-19 December, P. 166-168, 2009.

Conclusion

- 1. The second way can be realized using natural information redundancy of results of the arithmetic operations or simplifying a calculating operation in check.**
- 2. The natural information redundancy of a complete product can be realized using the prime numbers.**
- 3. The use of the prime numbers allows to design the simplest checkers for on-line testing of the iterative array multiplier.**
- 4. The squarer can be effectively checked using the forbidden values of a residue by modulo.**
- 5. The checking by simplified operation determines and forms by modulo the limiting conditions for operands and result and also executes the logic operation with these conditions.**
- 6. The second way for increasing a reliability of on-line testing methods reduces a probability of error detection without truncating a set of the detected faults.**

Questions and tasks

1. What is the second method for increasing a reliability **of the on-line testing methods**?
2. What **the methods** are by the second way realized?
3. Describe the **use of the prime numbers** for on-line testing the complete product of mantissas.
4. Describe the **procedure of the error detection probability assessment in the method of the squarer on-line testing** ?
5. What the models are in the **checking method by simplified operation** used?
6. What the main requirement does upon the methods by the second way impose?

MODULE 3. On-line testing for digital components of S-CES

Lecture 8. Checking by logarithm, inequalities, segments

8.1. The third way for increasing on-line testing reliability

8.2. The logarithm checking

8.3. The checking by inequalities

8.4. The checking by segments

8.1. The third way for increasing on-line testing reliability

8.1.1. Motivation of increasing an on-line testing reliability by the third way

The third way allows to obtain the most effective solutions.

Reasons:

The third way is directly aimed at distinction of essential and inessential errors taking into account a size of the error.

8.1.2. Related Works

1. Селлерс Ф. Методы обнаружения ошибок в работе ЭЦВМ. – М.: Мир, 1972. – 310 с.
2. Журавлев Ю. П., Котелюк Л. А., Циклинский Н. И. Надежность и контроль ЭВМ. – М.: Советское радио, 1978. – 416 с.
3. Моллов В. К. Структурно-функциональные методы оперативного контроля и диагностики цифровых устройств управляющих систем: Автореф. дис. . . канд. техн. наук: 05.13.13 / Киевск. политехн. ин-т – Киев, 1989. – 16 с.
4. Тоценко В. Г., Киселев И.М. Метод повышения эффективности диагностирования дискретных устройств с регулярной структурой // Управляющие системы и машины. – 1977. – № 5. – С. 97 – 102.
5. Байда Н. П., Кузьмин И., Шпилевой В. Микропроцессорные системы поэлементного диагностирования РЭА. – М.: Радио и связь, 1987. – 256 с.

8.1. The third way for increasing on-line testing reliability

8.1.3. Features of the third way

The main feature of a third way is use of the different probabilities of detection for essential and inessential errors.

The third way increases on-line testing reliability estimating a size of the result and its error.

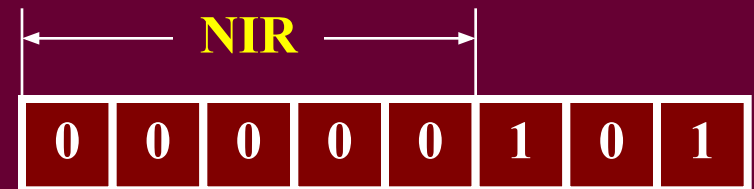
The methods of a third way difference the essential and inessential errors as well as well detect an error in high and low bits of the result.

8.2. The logarithm checking

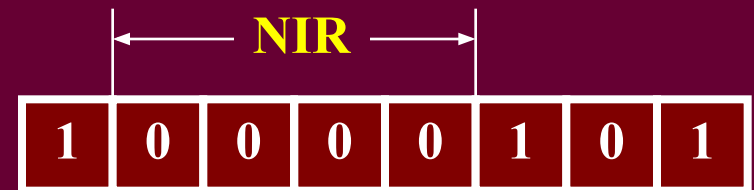
8.2.1. The use of the Natural Information Redundancy

The **logarithm checking** is based on the use of the **Natural Information Redundancy (NIR)** of data **formats** in form of not quite use of the codeword **high positions**.

1. Fixed-point format



2. Floating-point format



8.2. The logarithm checking

8.2.2. Definition of the check code of a number or mantissa

Check code KA of fixed-point number A is equal to amount of bits of a significant part of this number.

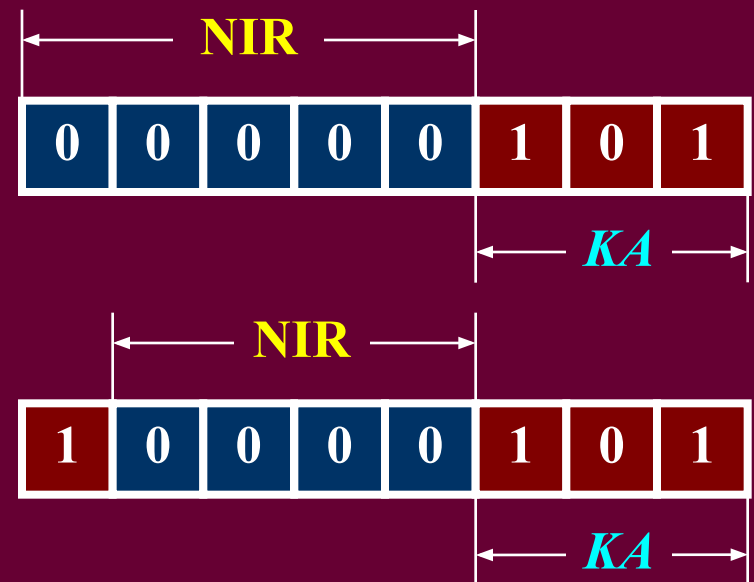
Check code KA of mantissa A is equal to amount of bits of a check part of this mantissa.

1. Fixed-point format

$KA = \text{Int}(\log_2 A)$ for $A > 0$;
 $KA = 0$ for $A = 0$.

2. Floating-point format

$KA = \text{Int}(\log_2 (A-1))$ for $A > 0$.



8.2. The logarithm checking

8.2.3. Calculation of the check code of a number or a mantissa

The check code is calculated using the truth form of a number or a mantissa by two steps:

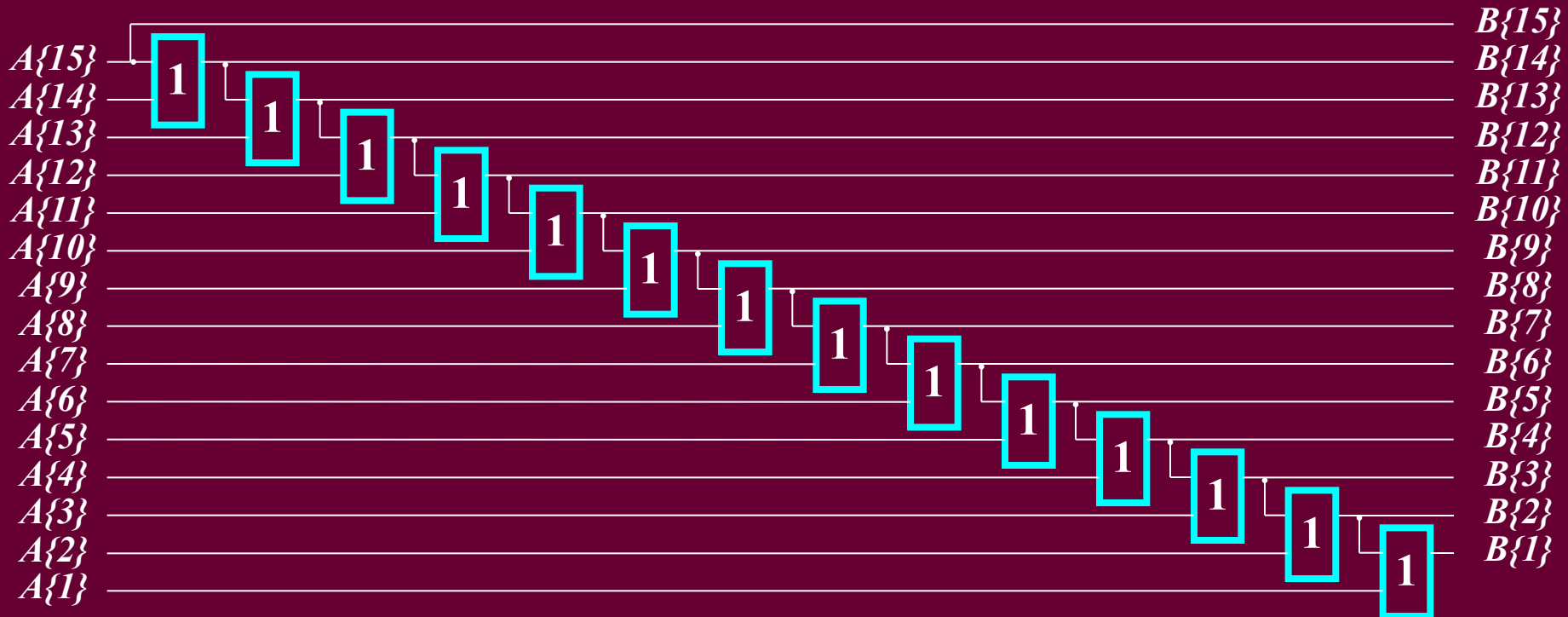
1. Filling the most significant (check) part by the units;

2. Calculation of units amount.

8.2. The logarithm checking

8.2.3.1. Filling the most significant (check) part by the units

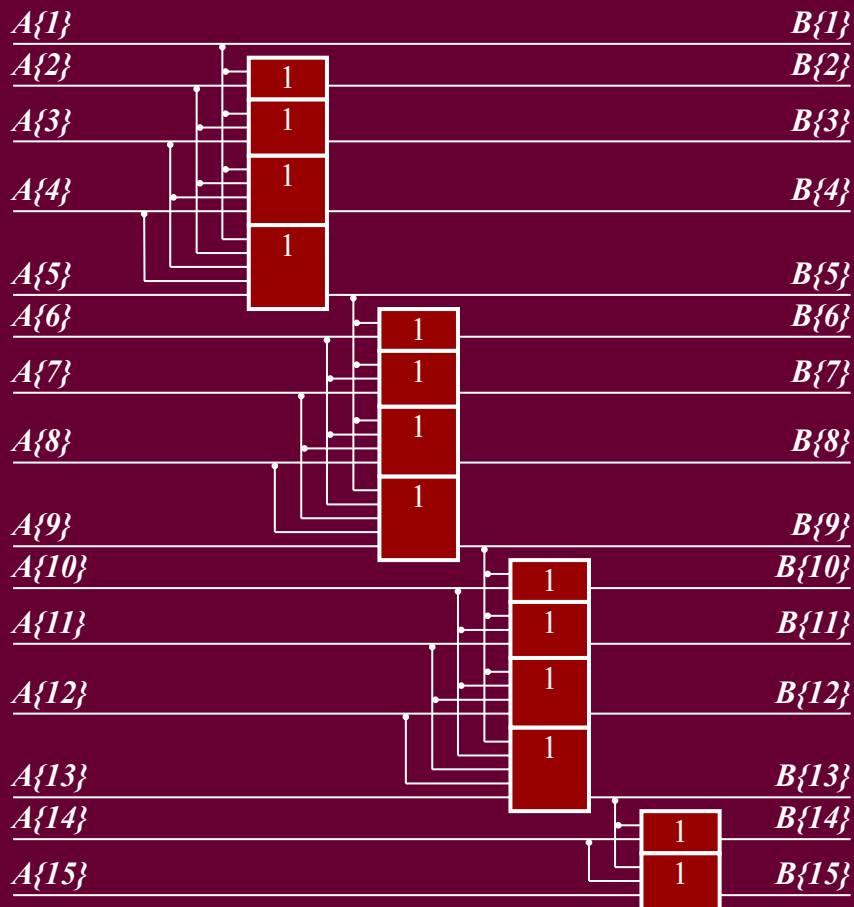
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
A	0	0	0	0	1	0	1	1	0	1	1	1	0	0	1
B	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1



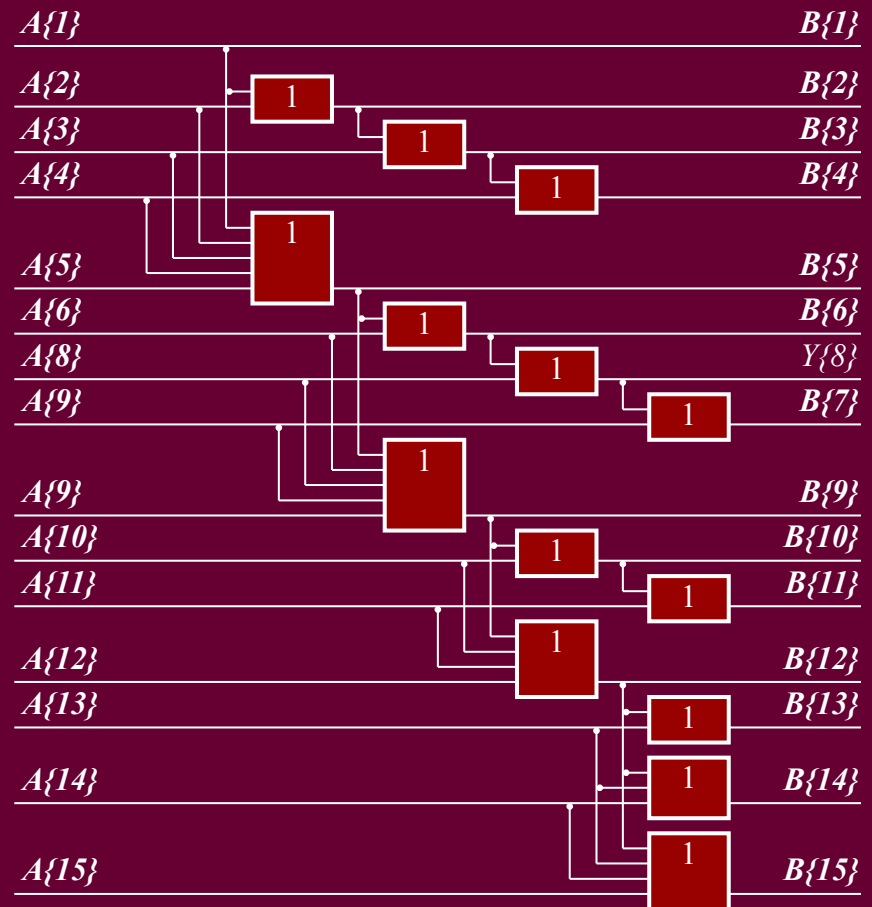
8.2. The logarithm checking

8.2.3.1. Filling the most significant (check) part by the units

A circuit with a serial-group calculation of the code B

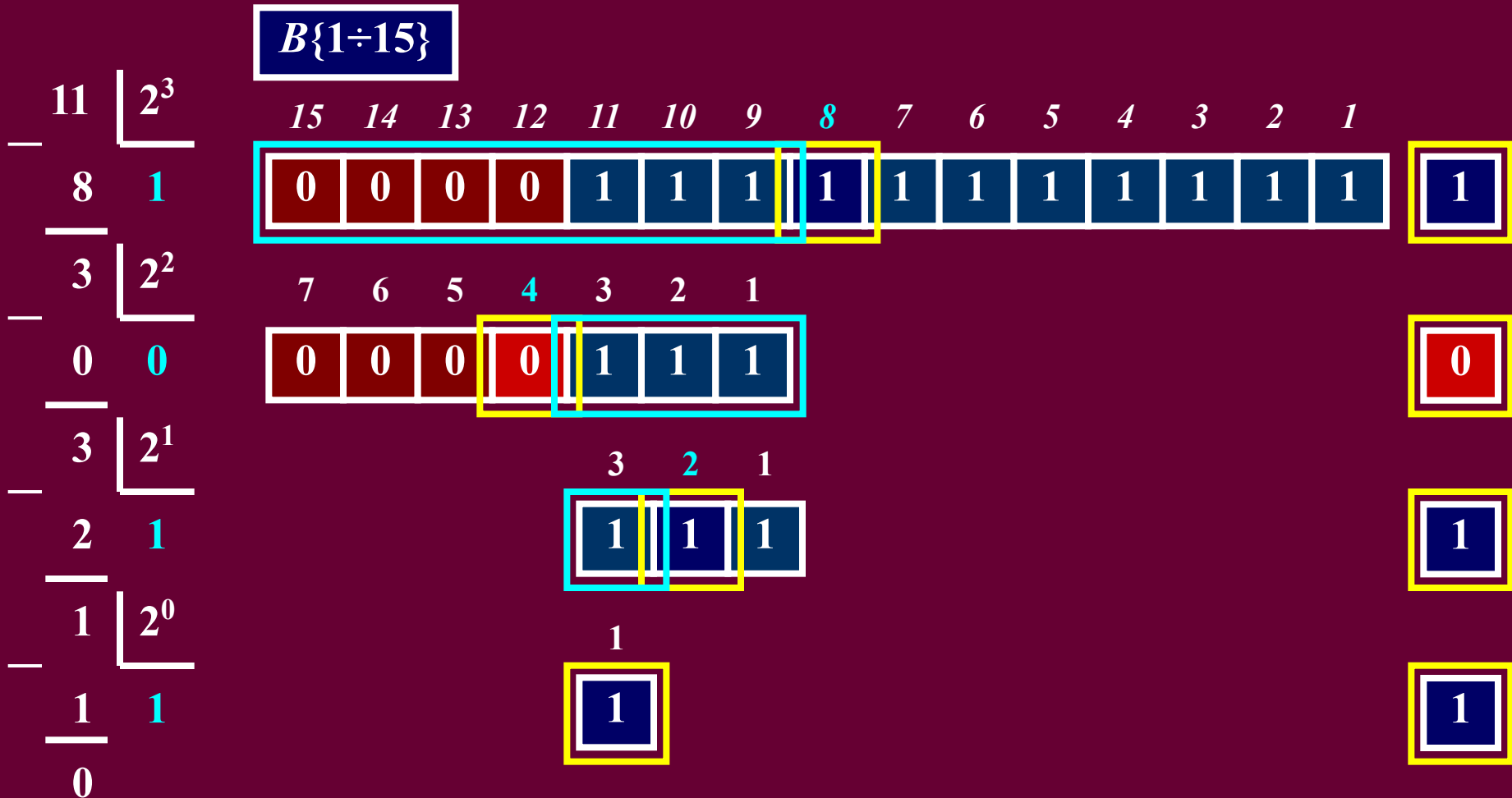


A circuit with a serial calculation of the bits in groups of the code B



8.2. The logarithm checking

8.2.3.2. Calculation of units amount



8.2. The logarithm checking

8.2.4. The check equations for the arithmetic operations

The check codes of operands allow predict the check code of arithmetic operation result with difference $\alpha \leq 1$

• **For addition $S = A + B$, $A \geq 0$ and $B \geq 0$: $KS = KS^* + \alpha$, where $KS^* = \max(KA, KB)$; $\alpha = 0$ or $\alpha = 1$.**

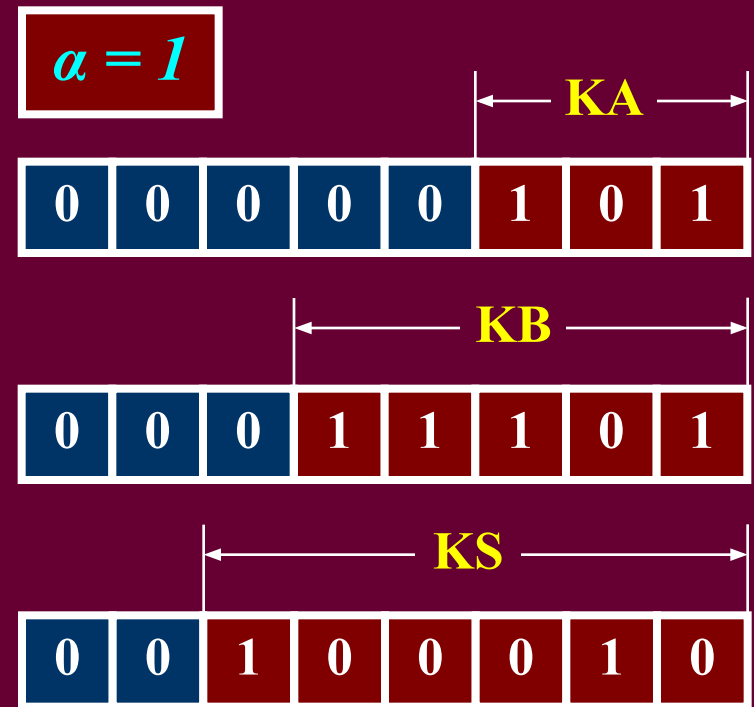
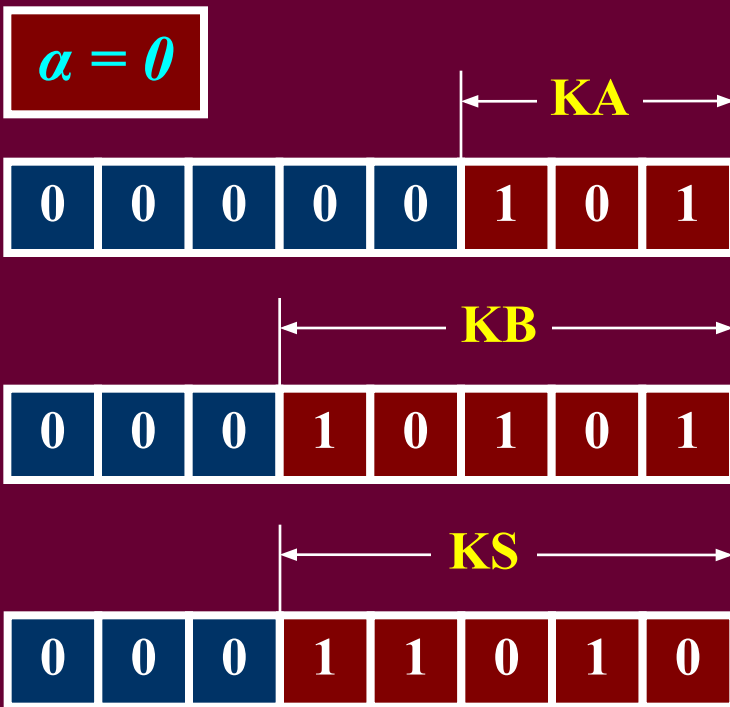
• **For multiplication $P = A \cdot B$, $A > 0$ and $B > 0$: $KP = KP^* - \alpha$, where $KP^* = KA + KB$; $\alpha = 0$ or $\alpha = 1$.**

• **For division $Q = A / B$, $A > 0$ and $B > 0$: $KQ = KQ^* + \alpha$, where $KQ^* = KA - KB$; $\alpha = 0$ or $\alpha = 1$.**

8.2. The logarithm checking

8.2.4. The check equations for the arithmetic operations

- **For addition $S = A + B$, $A \geq 0$ and $B \geq 0$: $KS = KS^* + \alpha$, where $KS^* = \max(KA, KB)$; $\alpha = 0$ or $\alpha = 1$.**



8.2. The logarithm checking

8.2.4. The check equations for the arithmetic operations

- **For addition $S = A + B$: $KS = KS^* + \alpha$,**
where $KS^* = \max(KA_R, KB_R)$; $\alpha = 0$ or $\alpha = 1$.

Sign S	Sign B	Sign A	Addition		KA_R	KB_R	KS_R
			initial	transformed			
0	0	0	$A + B = S$	$A + B = S$	KA	KB	KS
0	0	1	$- A + B = S$	$ A + S = B$	KA	KS	KB
0	1	0	$A - B = S$	$ B + S = A$	KB	KS	KA
1	0	1	$- A + B = - S $	$B + S = A $	KB	KS	KA
1	1	0	$A - B = - S $	$A + S = B $	KA	KS	KB
1	1	1	$- A - B = - S $	$ A + B = S $	KA	KB	KS

$$KA_R = KA \wedge \neg U_1 \vee KB \wedge U_1; \quad KB_R = KB \wedge \neg U_2 \vee KS \wedge U_2;$$

$$KS_R = KA \wedge U_1 \vee KS \wedge \neg U_2 \vee KB \wedge U_3,$$

where $U_1 = \text{Sign } A \oplus \text{Sign } S$, $U_2 = \text{Sign } A \oplus \text{Sign } B$, $U_3 = \text{Sign } A \oplus \text{Sign } S$.

8.2. The logarithm checking

8.2.4. The check equations for the arithmetic operations

• **For multiplication:** $P = A B$, $A > 0$ and $B > 0$, $KP = KP^* - \alpha$, where $KP^* = KA + KB$; $\alpha = 0$ or $\alpha = 1$.

$$2^{KA-1} \leq A < 2^{KA}$$

For $KA =$ $2^{KB-1} \leq B < 2^{KB}$

$$2^{KP-1} \leq P < 2^{KP}$$

$$KP - 1 = (KA - 1) + (KB - 1)$$

$$KP = KA + KB - 1$$

$$KP = KA + KB$$

8.2. The logarithm checking

8.2.4. The check equations for the arithmetic operations

- For multiplication: $P = A \cdot B$, $A \geq 0$ and $B \geq 0$,

- $KP = KP^* - \alpha$;

- $KP^* = KA \cdot Z_B + KB \cdot Z_A$;

- where $\alpha = 0$ or $\alpha = 1$;

- Z_A – tag of zero for A ;

- $Z_A = 0$ if $A = 0$ and $Z_A = 1$ if $A \neq 0$;

- Z_B – tag of zero for B ;

- $Z_B = 0$ if $B = 0$ and $Z_B = 1$ if $B \neq 0$.

8.2. The logarithm checking

8.2.4. The check equations for the arithmetic operations

• **For division:** $Q = A / B$, $A > 0$ and $B > 0$, $KQ = KQ^* + \alpha$, where $KQ^* = KA - KB$; $\alpha = 0$ or $\alpha = 1$.

$$2^{KA-1} \leq A < 2^{KA}$$

$$2^{KB-1} \leq B < 2^{KB}$$

$$2^{KQ-1} \leq Q < 2^{KQ}$$

$$KQ - 1 = (KA - 1) - KB$$

$$KQ = KA - KB$$

$$KQ = KA - (KB - 1)$$

$$KQ = KA - KB + 1$$

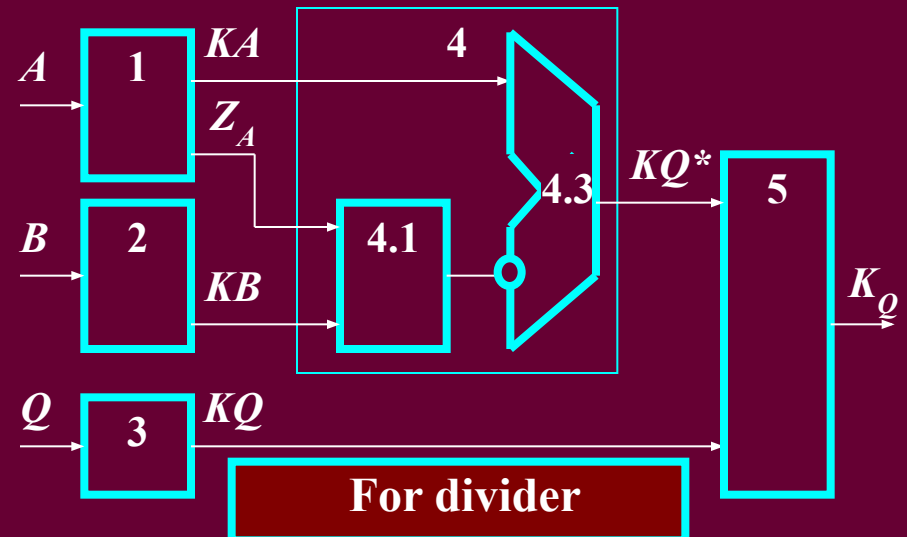
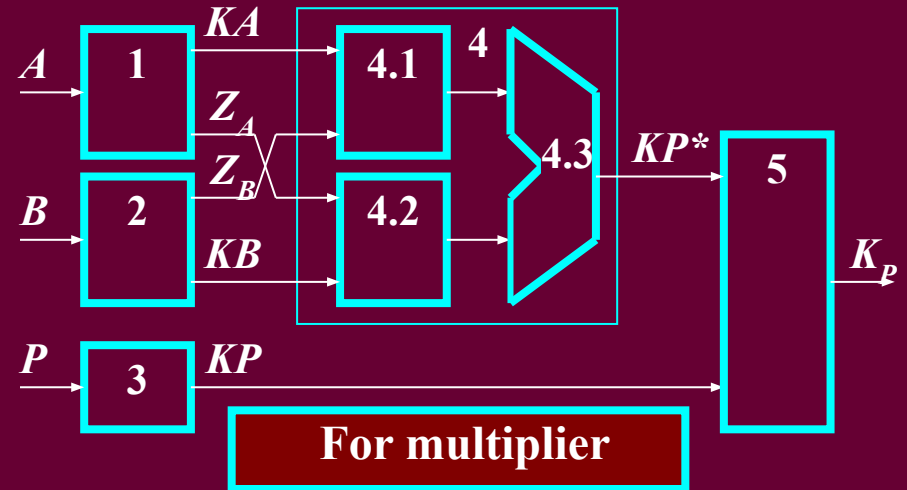
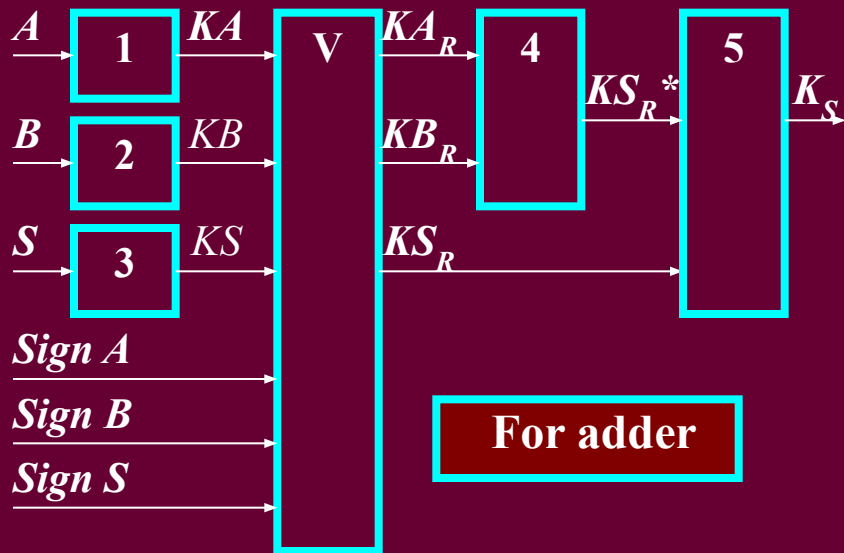
8.2. The logarithm checking

8.2.4. The check equations for the arithmetic operations

- For division: $Q = A / B$, $A \geq 0$ and $B > 0$,
 - $KQ = KQ^* + \alpha$;
 - $KQ^* = KA - KB$;
- where $\alpha = 0$ or $\alpha = 1$;
- Z_A – tag of zero for A ;
 - $Z_A = 0$ if $A = 0$ and $Z_A = 1$ if $A \neq 0$;

8.2. The logarithm checking

8.2.5. Circuits of the check

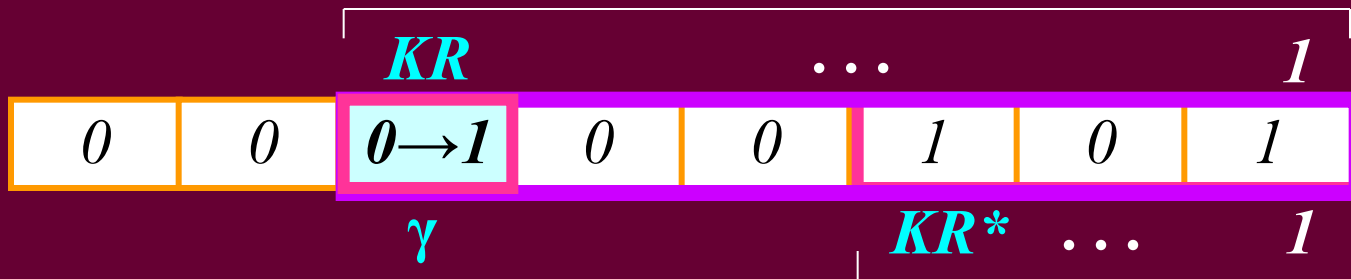


1, 2, 3 – formers of check codes
 V – unit of check codes rename
 4 – checking block
 4.1, 4.2 – gates AND
 4.3 – adder
 5 – comparator

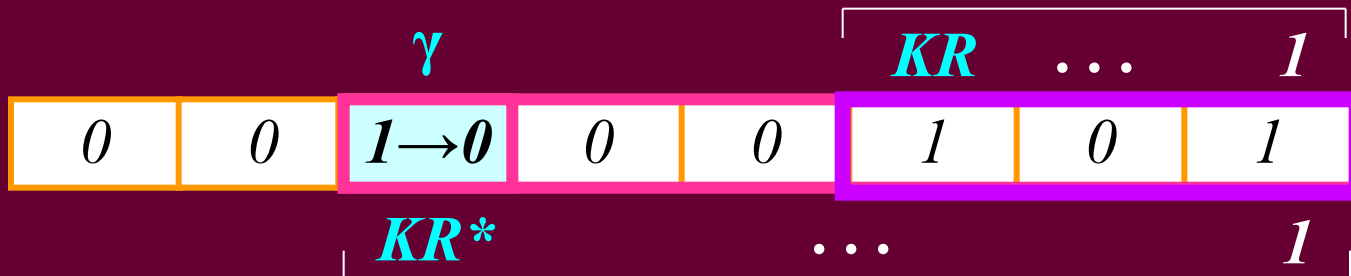
8.2. The logarithm checking

8.2.6. Error detection

1. The error $0 \rightarrow 1$ in the bit γ



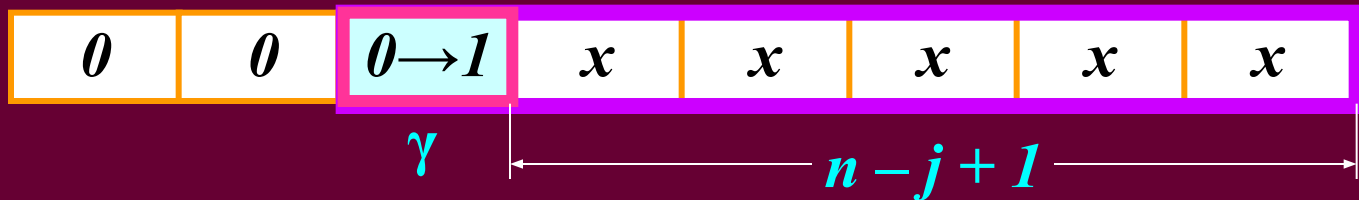
2. The error $1 \rightarrow 0$ in the bit γ



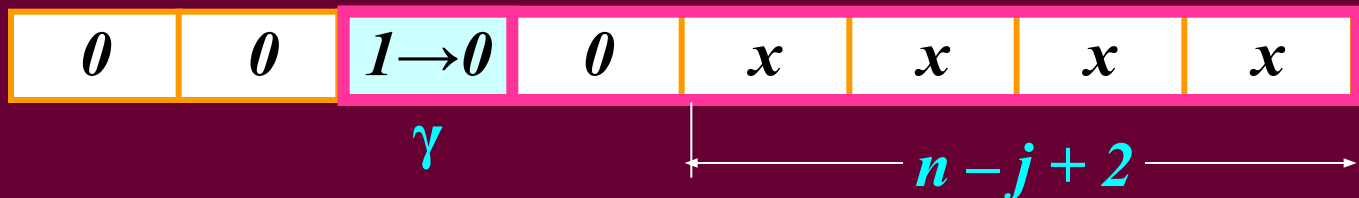
8.2. The logarithm checking

8.2.6. Error detection

1. The error $0 \rightarrow 1$ in the bit γ is detected with $P_D = 2^{-n+j-1}$



2. The error $1 \rightarrow 0$ in the bit γ is detected with $P_D = 2^{-n+j-2}$



The error detection probability is proportional to value 2^{-j} of an error in the bit γ .

8.3. Checking by inequalities

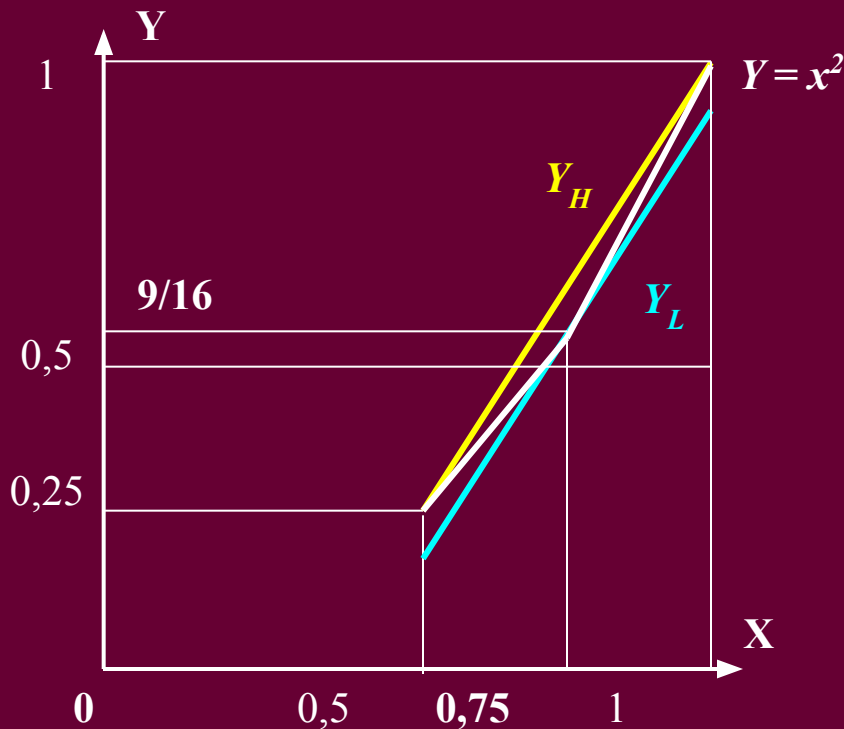
A method of the checking by inequalities includes:

1. Definition and calculation of high and low boards of the result

2. Comparison of the result with its high and low boards

8.3. Checking by inequalities

8.3.1. Definition of the result boards for a mantissa squarer



$$0.5 \leq x < 1$$

1. The high board Y_H connects boundary points $(0.5, 0.25)$ and $(1, 1)$ of the result graph.

$$Y_H = 3/2 x - 1/2$$

1. The low board Y_L is tangent to the high bound passing the point $(0.75, 9/16)$ of the result graph.

$$Y_L = 3/2 x - 9/16$$

8.3. Checking by inequalities

8.3.2. Error detection estimation

$$\Delta Y_H = Y_H - Y$$

Positive error $a = \Delta Y_H$

$$a = 3/2 x - 1/2 - x^2,$$

$$P_{N-DH} = 2 (x_1 - x_2),$$

$$P_{DH} = \sqrt{1-16a}, \quad a < 1/16.$$

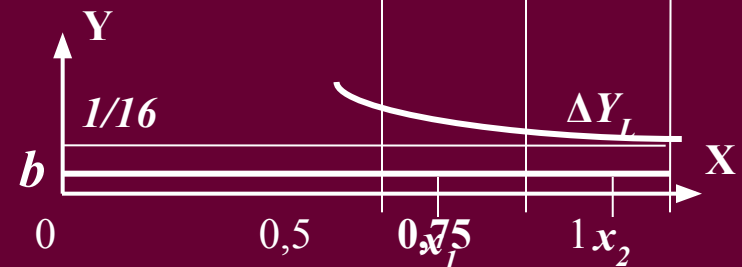
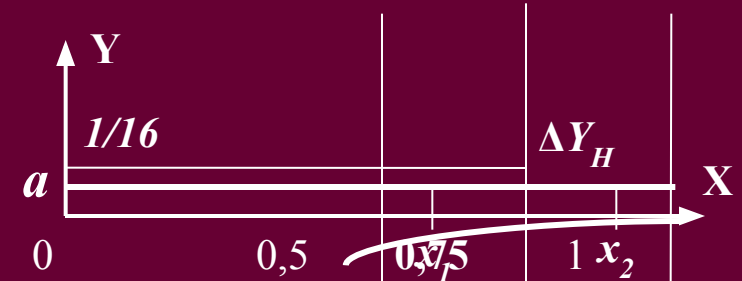
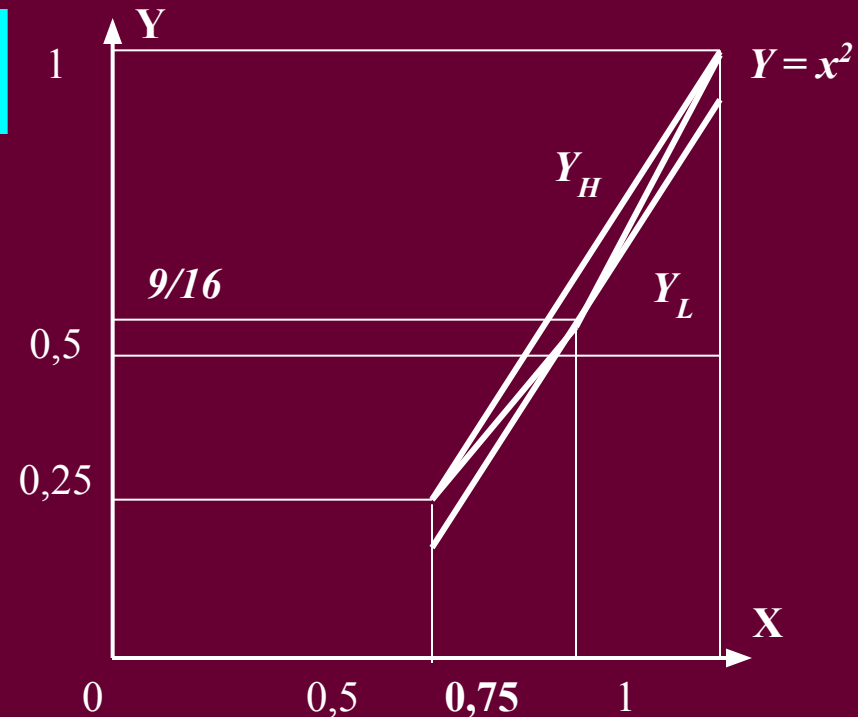
$$\Delta Y_L = Y - Y_L$$

Negative error $b = \Delta Y_L$

$$b = x^2 - 3/2 x + 9/16,$$

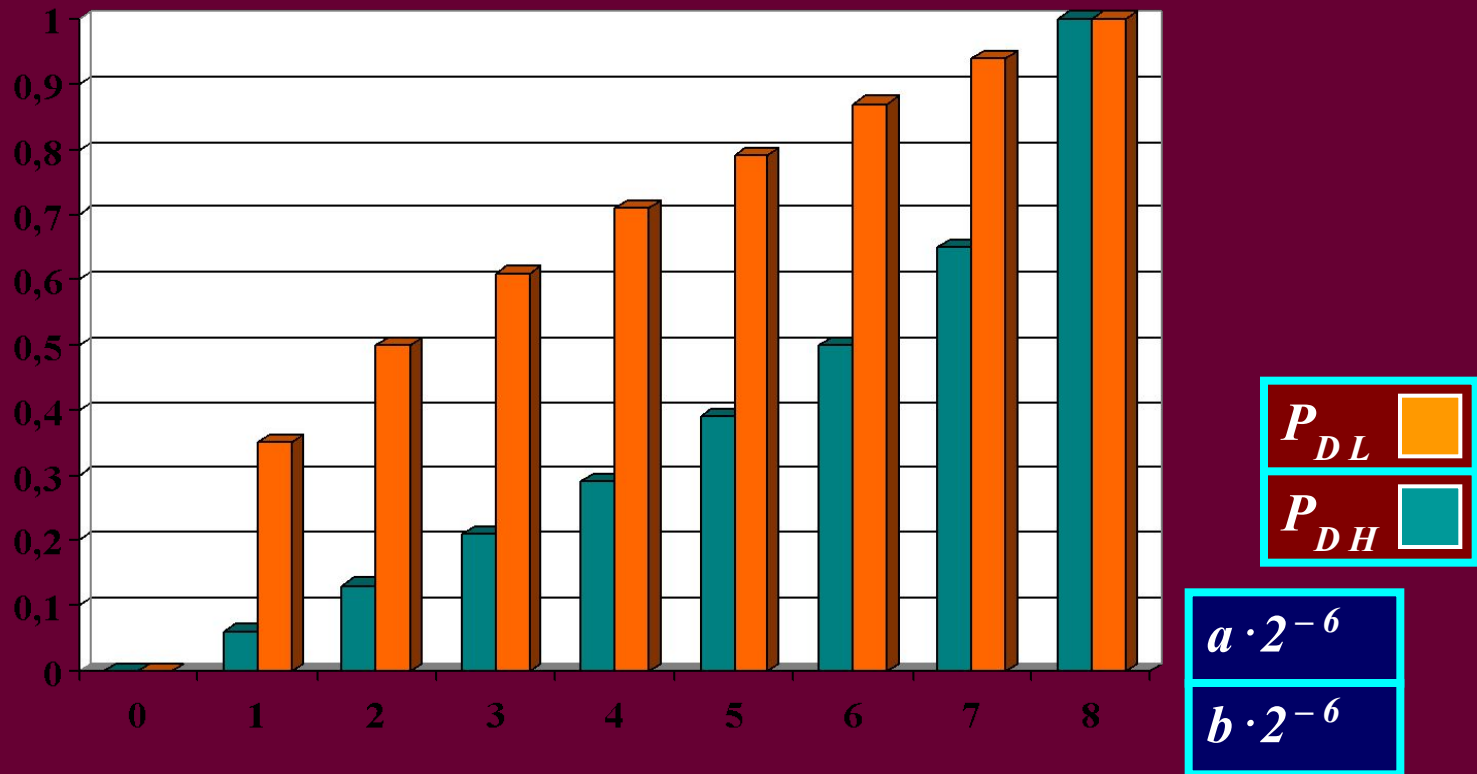
$$P_{N-DL} = 1 + 2 (x_1 - x_2),$$

$$P_{DL} = 1 - 4\sqrt{b}, \quad b < 1/16$$



8.3. Checking by inequalities

8.3.2. Error detection estimation



The error detection probability is increased with growing an error.

8.4. The checking by segments

The method of checking by segments decomposes the result into segments of bits and provides for them the required probabilities of error detection

$$P_1 \geq \dots \geq P_i \geq \dots \geq P_Z,$$

where $i = 1 \div Z$;

Z – an amount of segments.

The method is based on use of the natural time redundancy in form of the **Passive Stock of Checking Time (PSCT)**.

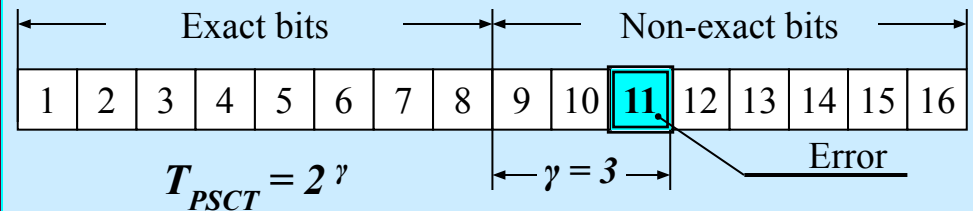
The **PSCT** allows detecting an error during some time T that is called interval of the PSCT.

8.4. The checking by segments

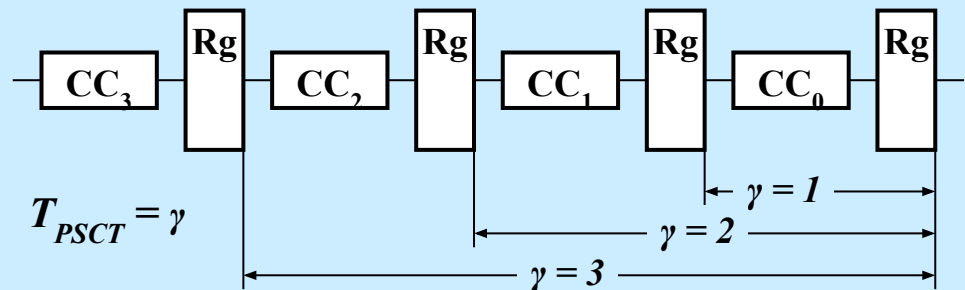
8.4.1. Natural Time Redundancy

Examples of the PSCT components

1. Time during which the result remains reliable despite of action of fault in circuit



2. Time during which the unreliable result is not dangerous

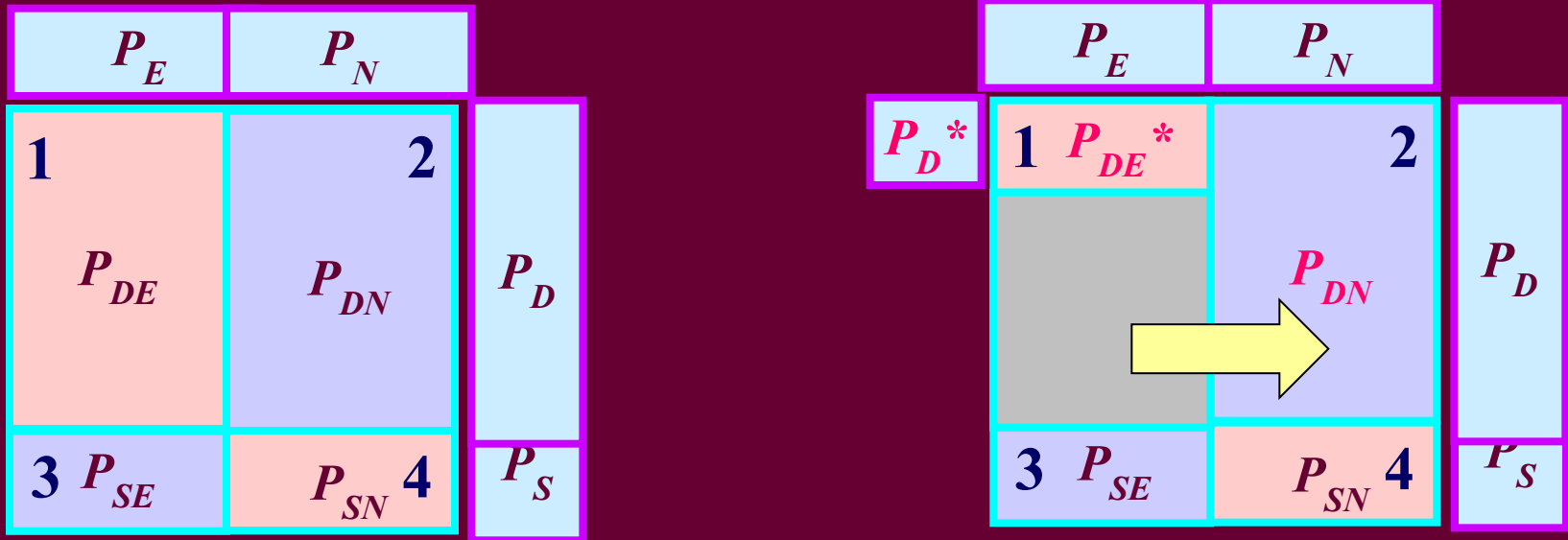


Probability of error detection in a segment of the result

$$P_D^* = \ln 2 / T_{PSCT}$$

8.4. The checking by segments

8.4.2. Reliability of the checking by segments



Estimation of reliability in checking the result

without consideration of PSCT

$$D = P_{DE} + P_{SN}$$

with consideration of PSCT

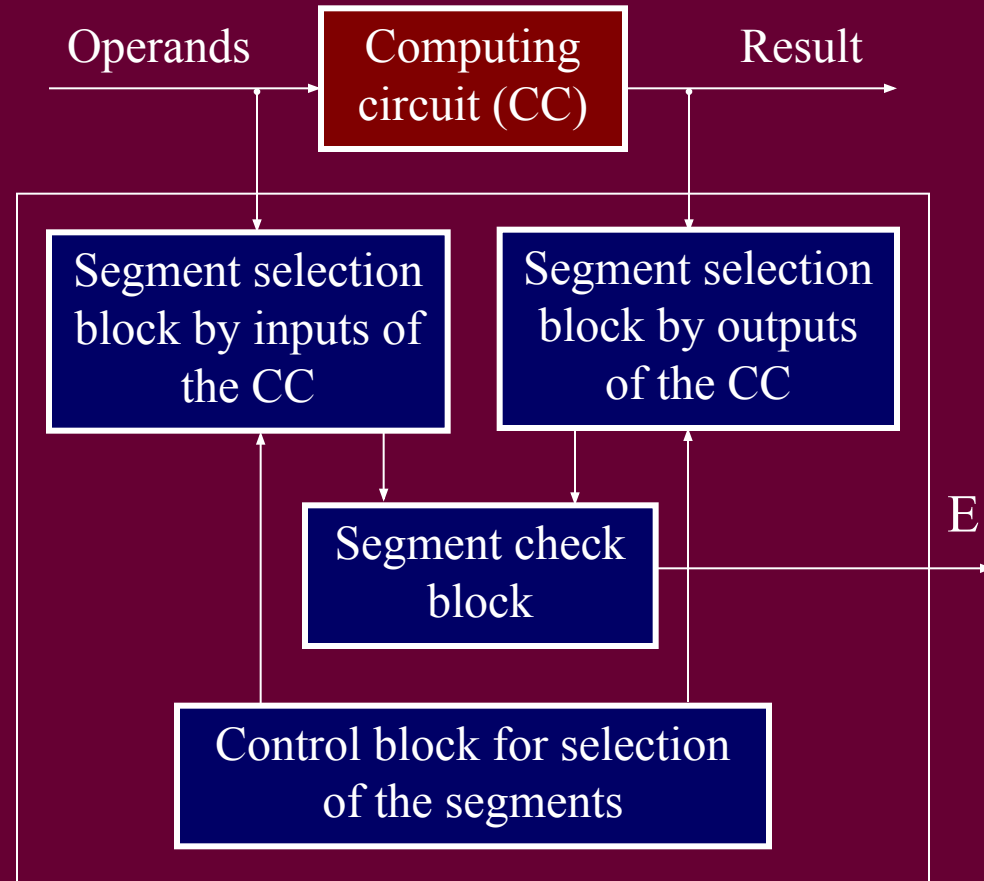
$$D_{PSCT} = P_{DE}^* + P_{SN}$$

8.4. The checking by segments

8.4.3. Segment-serial checking method

1. Division of a result on segments of the bits
2. Serial checking the segments
3. Setting the frequency distribution of a checking the result segments.

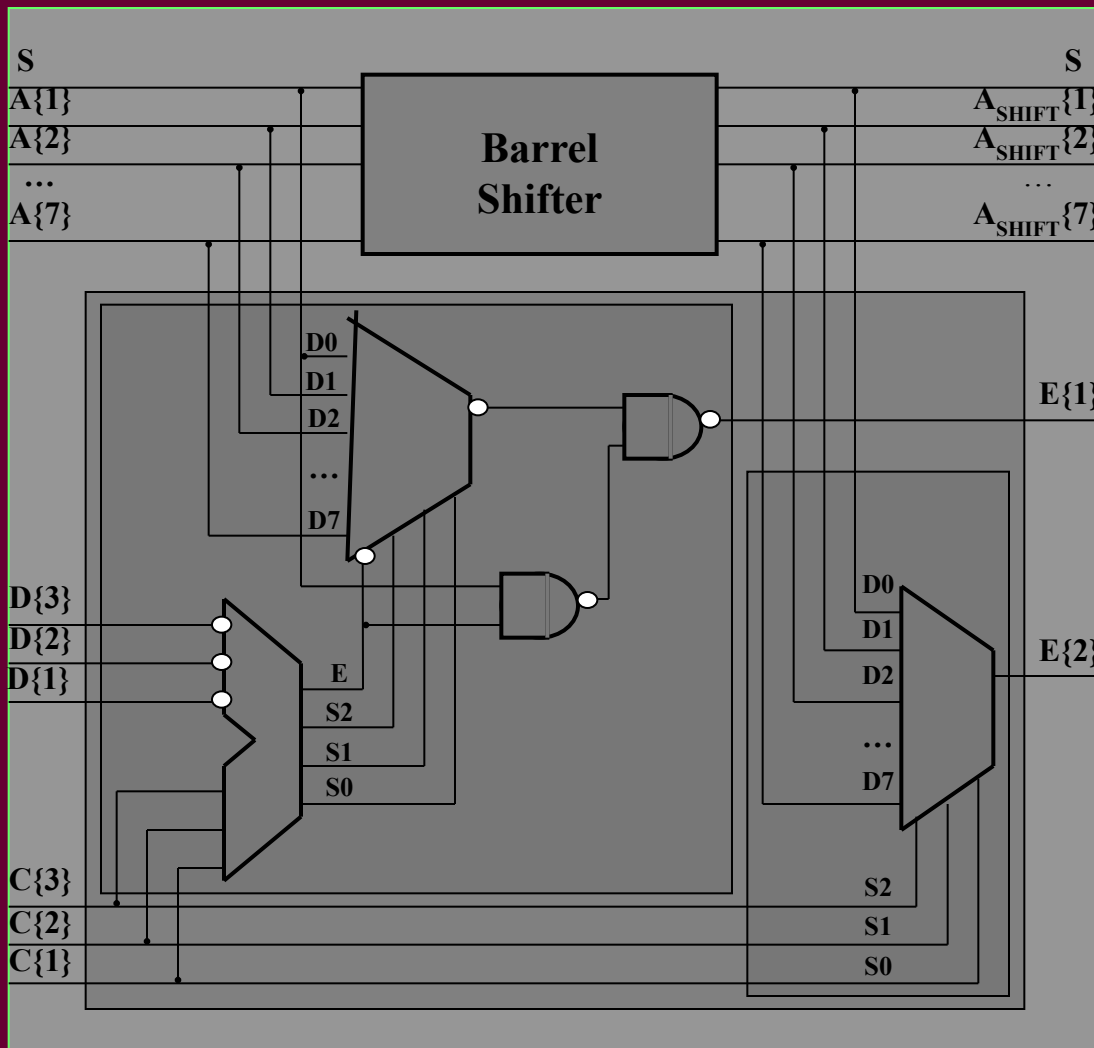
The segment-serial checking allows to raise check frequency of the high true bits of the result and probability of essential error detection



Error detection scheme

8.4. The checking by segments

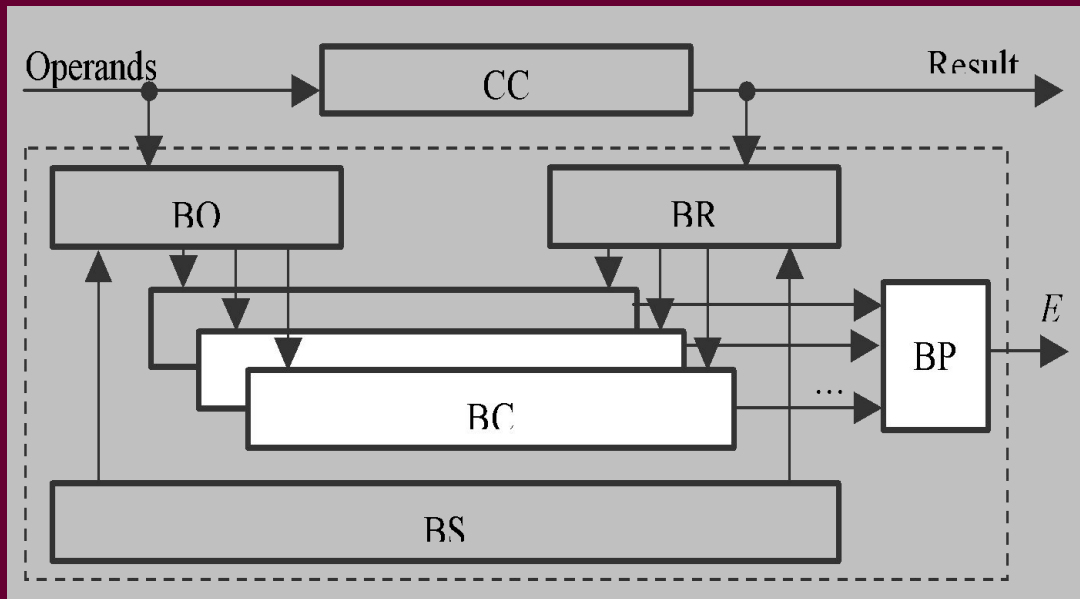
8.4.4. Segment-serial checking of the Barrel Shifter



$P_D = 1/n$					
$h_D = P_{DE} / P_{DN}, h_D > 1$					
$h_N = n_E / n_N$					
$P_{DE} = \frac{P_D h_D (h_N + 1)}{h_D h_N + 1}$					
$P_{DN} = \frac{P_D (h_N + 1)}{h_D h_N + 1}$					
h_D	h_N	K_T	n	P_D	D_C
4	1	0.2	16	0.6	0.2
P_{DE}		P_{DE}		P_{SKIP}	P_{REJECT}
0.1		0.025		0.18	0,02

8.4. The checking by segments

8.4.5. Error Detection Circuit with some check blocks



BO – operand block **BR** – result block
BS – control block
BC – check blocks
BP – pack block

An amount of the BC
$$N_T = \lceil P_{SUM} / P_D \rceil$$
, where
$$P_{SUM} = \sum_{i=1}^Z P_i$$

The block **BO** connects inputs of the circuit elements, which calculate the selected segments, to blocks **BC**.

The block **BR** connects outputs of the circuit elements.

The block **BS** sets sequence of a choice of segments groups.

The blocks **BC** check the selected segments and calculate check codes, which specify correctness of result in these segments.

The block **BP** compresses the check codes up to code **E** of result correctness.

8.4. The checking by segments

8.4.6. Choice of check points

Array P of bits P_{ij}
in binary codes of
probabilities P_i

Segments	Probabilities	Bits $j=1..m, m=4$			
$i=1..Z, Z=5$	P_i	4	3	2	1
1	$0.1101_2 = 13/16$	1	1	0	1
2	$0.1011_2 = 11/16$	1	0	1	1
3	$0.1001_2 = 9/16$	1	0	0	1
4	$0.0110_2 = 6/16$	0	1	1	0
5	$0.0100_2 = 4/16$	0	1	0	0

Sequences of segment checks

Functions	Clock cycles of interval T															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M_{40}	s_1	s_1	s_1	s_1	s_1	s_1	s_1	s_1	s_2	s_2	s_2	s_2	s_2	s_2	s_2	s_2
M_{30}	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_3	s_1	s_1	s_1	s_1	s_4	s_4	s_4	s_4
M_1	s_5	s_5	s_5	s_5	s_2	s_2	s_2	0	0	0	s_4	s_4	s_1	s_3	0	0

8.4. The checking by segments

8.4.7. Increase in reliability

Reliability of the checking the result in a segment i :

$$D_i = P_i P_E + (1 - P_i) (1 - P_E).$$

The size of increase in reliability for segment i :

$$\delta D_i = (P_D - P_i) (1 - P_E), P_D \gg P_i$$

For example, for $P_D = 0.5$, $P_i = 0.1$, $P_E = 0.1$, the size of increase in reliability $\delta D_i = 0,36$.

The size of increase in reliability:

$$\delta D = \sum_{i=1}^Z (\delta E_i \delta D_i),$$

where $\delta E_i = E_i / E_{CC}$;

E_i is complexity of segment calculation;

E_{CC} is complexity of computing circuit.

Reading List

1. Дрозд А. В. Использование логарифмического контроля для обнаружения отказов арифметических устройств // Вісн. НТУУ «КПІ». Інф., упр. та обчисл. техніка. – К., 1998. – Вип. 31. – С. 224 – 231.
2. Дрозд А. В., Зуда М., Лобачев М. В. Использование логарифмических оценок в функциональном диагностировании вычислительных устройств с плавающей точкой // Тр. Одес. политехн. ун-та. – Одесса, 2001. – Вып. 1 (13). – С. 93 – 96.
3. Drozd A. , Al-Azzeh R., Drozd J., Lobachev M. The logarithmic checking method for on-line testing of computing circuits for processing of the approximated data. – Proc. of Euromicro Symposium on Digital System Design, Rennes, France, pp. 416 – 423, 2004.
4. Дрозд А. В. Контроль вычислительных устройств по неравенствам // Ученые записки Симферопольского гос. ун-та. – Винница-Симферополь, 1998. – Спецвып. – С. 237 – 240.
5. Drozd A., Lobachev M., Reza Kolahi. “Effectiveness of on-line testing methods in approximate data processing,” in *Proc. IEEE East-West Design & Test Conference*, Odessa, Ukraine, 15 –19 Sept., pp. 62 – 65, 2005.

Conclusion

1. The third way is directly aimed at distinction of essential and inessential errors taking into account a size of the error.
2. The logarithm checking, the checking by inequalities and the checking by segments increase a reliability of on-line testing methods using the third way.
3. The logarithm checking is based on the use of the **Natural Information Redundancy** of data **formats** in form of not quite use of the codeword **high positions**.
4. The checking by inequalities estimates a result as reliable in case this result is allocated within its high and low bounds.
5. The checking by segments is based on use of the natural time redundancy in form of the **Passive Stock of Checking Time**
6. The methods developed by the third way show high effectiveness using the natural time and information redundancy.

Questions and tasks

1. What feature of the third way for increasing a reliability of **the on-line testing methods do you know?**
2. What **the methods** are by the third way realized?
3. Describe the **use of the natural information redundancy of the data format in the logarithm checking.**
4. What tag does the reliable result in the checking by inequalities determine?
5. Describe the **use of the natural time redundancy in the checking by segments.**
6. What does the high effectiveness of the third way methods ensure?

MODULE 4.

Checkability of S-CES digital components

#	Topic of lecture	Lectures	Lab Classes	Private Study
9	Checkability of S-CES digital components: a problem, assessment, solutions	2	4	2
	Total:	2	4	2

MODULE 4. Checkability of S-CES digital components

Lecture 9. Checkability of S-CES digital components: a problem, assessment, solutions

9.1. Introduction into checkability

9.2. **The model** of a digital component in view of the on-line testing for S-CES

9.3. **The method** for estimating a checkability of S-CES digital components

9.4. **The ways** to increase a checkability of S-CES digital components

9.1. Introduction into checkability

9.1.1. Motivation of the checkability consideration for digital components of the S-CES

Reasons:

1. High requirements in safety impose upon the digital components of S-CES.
2. A Fault-Tolerant Technology is traditional solution of a safety problem for the digital components.
3. The Fault-Tolerant Technology can not solve the problem of digital component safety in case of S-CES.

9.1.2. Related Works

1. Yastrebenetsky M.A. (edit.). NPP I&Cs: Problems of Safety / M.A. Yastrebenetsky. – Ukraine, Kyiv: Technika, 2004.
2. Локазюк В.Н., Остроумов С.Б., Поморова О.В. и др. Отказоустойчивые встроенные системы на программируемой логике. Лекционный материал / Под ред. Харченко В.С. – Министерство образования и науки Украины. Национальный аэрокосмический университет «ХАИ», 2008. – 264 с.
3. Kharchenko V.S., Sklyar V.V. FPGA-based NPP Instrumentation and Control Systems: Development and Safety Assessment / Bakhmach E.S., Herasimenko A.D., Golovyv V.A. a.o.. – Research and Production Corporation “Radyi”, National Aerospace University “KhAI”, State Scientific Technical Center on Nuclear and Radiation Safety, 2008. – 188 p.
4. Щербаков Н. С. Достоверность работы цифровых устройств. – М.: Машиностроение, 1989. – 224 с.
5. Беннеттс Р.Дж. Проектирование тестопригодных логических схем. М.: Радио и связь, 1995. – 180 с.

9.1.3 Peculiarities of the S-CES

1. Two main operational modes, i.e. normal and emergency ones of S-CES and heir components.

2. Some certain degree of inertia of the controlled objects in comparison with that of high-rate digital components.

For most of operating time, the S-CES run in the normal mode. The emergency one, i.e. for which the S-CES are designed, is a rare event as a rule and at best may never occur.

First peculiarity generates a problem of maintaining the functionality of the components in the emergency mode by taking advantage of the normal mode provisions.

Second peculiarity provides a resource of time which may be used to resolve the problem.

9.1.4. A problem of maintaining the functionality of the S-CES components in the emergency mode

Both in the **normal** and **emergency** modes, the S-CES components operate with **different sets of input data**.

In the **normal** mode, the input **data vary within small ranges**.

On such a **limited** set of the input words the digital circuit of the component takes **constant values in many its points**.

This fact generates the conditions for **latent accumulation of constant faults** which may appear at the input words in the **emergency mode** and **counteract** the component to perform its functions.

9.1.5. Purpose of on-line testing for the S-CES components in the emergency mode

On-line testing is aimed at the **checking the reliability of the results** calculated by a digital component during basic operations performance **on operating sequences of input words.**

It is **correct** for the digital components operating in a **single** i.e. **only normal mode.**

For S-CES this **purpose should be expanded** adding the **checking of the availability** of the digital component to calculate **reliable results** in the **emergency mode.**

9.2. The model of a digital component in view of the on-line testing for S-CES

9.2.1 The initial model

$$M(S_N, S_C, S),$$

where: S_N is a component description characterizing its functioning in the **normal mode** – a limited set I_N of input words in the normal mode of operation;

S_C is a component description characterizing its functioning in the **emergency mode** – a limited set I_C of input words used for identifying the emergency mode;

S is a component description common both for **normal and emergency modes** (description D of the digital circuit of the tested component and the set F of its typical faults).

9.2. The model of a digital component in view of the on-line testing for S-CES

9.2.1 The initial model

Description D of the digital circuit should be illustrated by **the specific elements**.

For instance, the description of the digital circuit on **FPGA** should contain **the list of points** of two types:

- **internal points**, i.e. bits of memory **LUT**;
- **external points** which include all other points like bits of LUT address or its output.

External points can be **input** and **output (check points)**.

Besides, the description should contain the **functions** which define the **dependences** of ones **external points** upon others (from **input** points up to **output** points).

9.2. The model of a digital component in view of the on-line testing for S-CES

9.2.2. Controllable points of the digital component

1. **An internal point** of the digital circuit is *a controllable* one if the limited set of input words contains at least one word, on which this point is chosen in its LUT. Otherwise, the internal point is *a non-controllable* one.

2. **An external point** of the digital circuit is *a partially controllable* one (*0 or 1-controllable* point) if this point takes only a value '0' or only a value '1' on the limited set of input words. Otherwise, the external point is *a controllable* one.

9.2. The model of a digital component in view of the on-line testing for S-CES

9.2.3. *Observable* points of the digital component:

1. A point of the digital circuit is *a partially observable* one (0 or 1- *observable* point) if a path from this point up to a check point is activated on the limited set of input words only for one value '0' or '1'.

2. In case the path is activated for both values '0' and '1' the point is *observable* one.

3. Otherwise the point is a *non-observable* one.

The path is activated if a change of value of the given point is transferred to a check point.

9.2. The model of a digital component in view of the on-line testing for S-CES

9.2.4. Properties of the controllable and **observable points**

Statement 1. The observable internal point is also a **controllable**.

Statement 2. For the assigned input word **the result** is determined only by the values of points of the circuit, which are **observable ones**.

9.2. The model of a digital component in view of the on-line testing for S-CES

9.2.5. Controllability and observability of the points

- **Controllability C** can accept 3 values: 0, 1, 2 or 1, 2, 3 for an **internal** and **external** point, accordingly.

Values 0, 1, 2 and 3 distinguish cases of *non-controlled*, *1-controlled*, *0-controlled* and *controlled* point, accordingly.

- **Observability O** of **an external** point can accept 4 values: 0, 1, 2 and 3 in cases of *non-observable*, *1-observable*, *0-observable* and *observable* point, accordingly.

Observability of **an internal** point can accept only values 0, 1 and 2.

9.2. The model of a digital component in view of the on-line testing for S-CES

9.2.6 The resulting model

$$M(C_N, O_N, C_C, O_C),$$

where: C_N and O_N are the *controllability C* and *observability O* for every points of the S-CES digital component in a normal mode;

C_C and O_C are the *controllability C* and *observability O* for every points of the S-CES digital component in an emergency mode.

9.3. The method for estimating a checkability of S-CES digital components

9.3.1. The dangerous points of the S-CES digital components

A checkability of the digital component is in break in the considered point under coincidence of two events:

- possibility of the latent fault occurrence in the normal mode;**
- possibility of this fault appearance in the emergency mode.**

Such point is dangerous for the S-CES digital component.

9.3. The method for estimating a checkability of S-CES digital components

9.3.2. Possibilities of the latent fault accumulation in a normal mode

- The point is *a non-controllable* one and *a value* in it coincides with a value defined by the stuck-at fault
- The point is a non-observable one.

9.3. The method for estimating a checkability of S-CES digital components

9.3.3. Possibilities of activity of the accumulated fault in the emergency mode

- The point is *an observable* and *non-controllable* and its value as a value of the non-controllable point is distinct from the value defined by the stuck-at fault;
- The point is *a controllable* and *an observable* one.

9.3. The method for estimating a checkability of S-CES digital components

9.3.4. Conditions of dangerous points detection

The external point is dangerous to an emergency mode under the following condition:

$$((C_N + C_E = 3) \text{ or } (O_N + C_E = 3) \text{ or } (O_N = 0)) \text{ and } (O_E > 0).$$

The internal point is dangerous to an emergency mode under the following condition:

$$(O_N = 0) \text{ and } (O_E > 0).$$

9.3. The method for estimating a checkability of S-CES digital components

9.3.5. Checkability of a digital component

Checkability of a digital component can be appreciated by the following formula:

$$K = 1 - N_E / N_T,$$

where N_E – amount of dangerous points;
 N_T – total of the circuit points.

9.4. The ways to increase a checkability of S-CES digital components

9.4.1. Research of the digital component checkability

Form1

Exit A 245 C Multiplier word size 8 Dangerous points 2

B 245 60025 Start Mantissa Controllability N

Input Normal Emergency A [0-1] B [0-1]

Base Value 128 245 [18 0] [17 2] [16 2] [15 0] [14 0] [13 0] [12 0]

Range of Data * 1 [28 0] [27 1] [26 2] [25 0] [24 0] [23 0] [22 0]

From	Step	Up to	[38 1]	[37 4]	[36 3]	[35 0]	[34 0]	[33 0]	[32 0]
10	10	80	[48 0]	[47 1]	[46 1]	[45 0]	[44 0]	[43 0]	[42 0]
			[58 2]	[57 4]	[56 3]	[55 0]	[54 0]	[53 0]	[52 0]
			[68 6]	[67 5]	[66 4]	[65 2]	[64 2]	[63 3]	[62]
			[78 6]	[77 7]	[76 5]	[75 4]	[74 3]	[73 3]	[72 1]
			[88 3]	[87 6]	[86 4]	[85 0]	[84 0]	[83 0]	[82 0]

[a=3 b=2] [P=2 SM a1=2 S=3 c b1=3]

Change of R D N	1	2	3	4	5	6	7	8
Quan-ty Danger.p.	97	71	62	36	33	31	2	0
CHECKABILITY	65,3%	74,6%	77,8%	87,1%	88,2%	88,9%	99,3%	100,0%

Iterative array multiplier of 8-bits mantissas

The base value of the factors in a normal mode is 128. The threshold is 245.

The range of the factors in a normal mode is changed from 10 by step 10 up to 80.

An amount of the dangerous points reduces from 97 down to 0

The multiplier checkability increases from 65% up to 100%

9.4. The ways to increase a checkability of S-CES digital components

9.4.1. Research of the digital component checkability

Form1

Exit A 175 C Multiplier word size 8 Dangerous points 3

B 175 30625 Start Mantissa Value

Input Normal Emergency A B

Base Value 128 * 18 0 17 2 16 2 15 0 14 0 13 0 12 0

Range of Data 10 1 28 0 27 1 26 3 25 0 24 0 23 0 22 0

From Step Up to 38 0 37 1 36 1 35 2 34 0 33 0 32 0

245 -10 175 48 0 47 1 46 1 45 0 44 1 43 0 42 0

58 0 57 1 56 1 55 0 54 0 53 0 52 0

68 3 67 4 66 3 65 1 64 1 63 1 62

78 4 77 7 76 5 75 1 74 1 73 1 72 1

88 3 87 4 86 4 85 0 84 0 83 0 82 0

Diagram: a 2x2 grid with labels a, b, P, SM, S, c, a1, b1

Change of B V E	1	2	3	4	5	6	7	8
Quan-ty Danger.p.	97	72	52	71	42	28	67	48
CHECKABILITY	65,3%	74,2%	81,4%	74,6%	85,0%	90,0%	76,0%	82,8%

Iterative array multiplier of 8-bits mantissas

In a normal mode the base value is 128. The range of factors is 10.

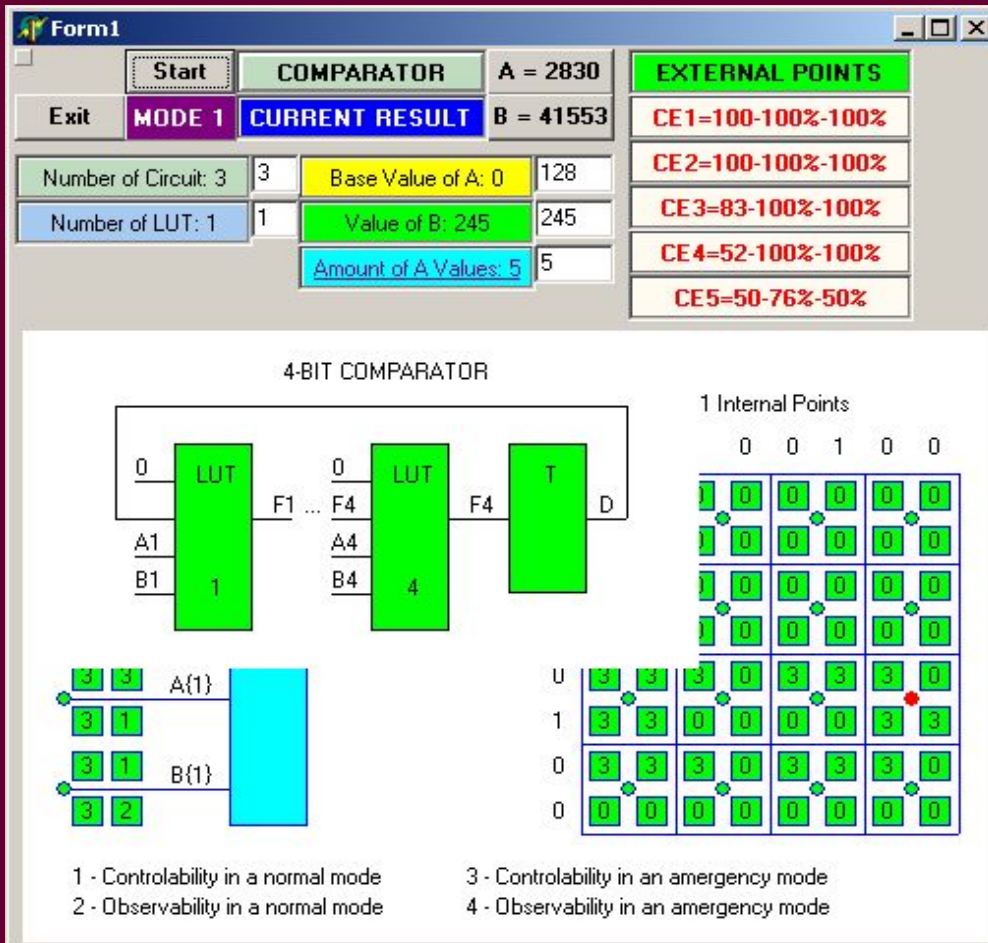
The threshold is reduced from 245 by step -10 down to 175.

An amount of the dangerous points reduces from 97 down to 48.

The multiplier checkability increases from 65.3% up to 82.8%

9.4. The ways to increase a checkability of S-CES digital components

9.4.1. Research of the digital component checkability



Serial-parallel comparator of 16-bits codewords

1 bit 16 clock unit comparator, 2 bit 8 clock unit comparator, 4 bit 4 clock unit comparator, 8 bit 2 clock unit comparator, 16 bit 1 clock unit comparator,

The threshold is 245. Range of input word A in a normal mode is 5

The comparator checkability increases from 50% up to 100%

9.4. The ways to increase a checkability of S-CES digital components

9.4.2. Reasons of low checkability of the S-CES digital components

Particularities of the S-CES digital components:

1. High level of **the input data consistency** in a normal mode.
2. High value of ratio of the threshold per **noise**.
3. High level of **the circuit parallelism**.

There are results of use of the high technology

9.4. The ways to increase a checkability of S-CES digital components

9.4.2. Reasons of low checkability of the S-CES digital components

Particularities of the S-CES digital components:

1. High level of **the input data consistency** in a normal mode.

2. High value of ratio of the threshold per **noise**.

3. High level of **the circuit parallelism**.

Aftermath:

1. **The limited change of input data in the normal mode.**

2. **The limited percent of input data in the normal mode.**

3. **Processing of input data in a parallel code using the simultaneous circuits.**

9.4. The ways to increase a checkability of S-CES digital components

9.4.3. Conditions to overcome a low checkability

- 1. Change of input data alternating a normal mode with a simulated one**
- 2. Reducing the threshold accuracy**
- 3. Reuse of the circuit points during data processing in a serial code.**

9.4. The ways to increase a checkability of S-CES digital components

9.4.3.1. Change of input data alternating a normal mode with a simulated one

1. **Simulated mode** is aimed at testing of the digital components on **input words of an emergency mode**.

2. Transition of the digital component in a simulated mode is associated with **risks** of its total exclusion from operation in a normal or simulated mode and creation of emergency mode.

3. Reduction of these risks demands to check application of the simulated mode using the **on-line testing** methods and means.

9.4. The ways to increase a checkability of S-CES digital components

9.4.3.2. Reducing the threshold accuracy

1. **The threshold accuracy** can be as high as to difference a normal and an emergency modes in both directions:

- from a normal mode to an emergency one;
- from an emergency mode to a normal one.

9.4. The ways to increase a checkability of S-CES digital components

9.4.3.3. Reuse of the circuit points during data processing in a serial code

1. **Frequency of data processing** can be reduced taking into account some certain degree of inertia of the controlled objects, sensors and analog-to-digital converters in comparison with that of high-rate digital components.

2. **Frequency of serial data processing** can be increased using

- **high frequency** of the bits processing in a serial code;
- **possibilities to parallel the serial code** processing, without essential lowering of the S-CES component checkability.

9.4. The ways to increase a checkability of S-CES digital components

9.4.4. Processing input data in a serial code using the clocked circuits

9.4.4.1. Influence of the serial code processing on controllability and observability of the circuit points.

1. **Reuse of circuit points** can change the values of them. This increases **controllability** of the circuit points.

2. **The serial code processing** shortens ways from circuit points up to check points. This can increase **observability** of the circuit points.

9.4. The ways to increase a checkability of S-CES digital components

9.4.4.2. Influence of the serial code processing on a checkability of the S-CES components.

1. Increase of **controlability** and **observability** in a normal mode leads to **reducing an amount of the dangerous points**.

2. Increase of **controlability** and **observability** in an emergency mode results in **increase of an amount of the dangerous points**.

3. **A checkability** of the S-CES components can be increased or reduced by **the serial code processing**.

9.4. The ways to increase a checkability of S-CES digital components

9.4.4.3. Dominant role of a checkability of the points in a normal mode.

1. In case the circuit point is checkable (controlable and observable) in a normal mode it is **not dangerous** one irrespectively of an emergency mode.

2. That's why **increase of a checkability of the circuit points** in both normal and emergency modes should **increase a checkability of the S-CES components**.

Reading List

1. Drozd A. On-line testing of safety-critical I&C systems in normal and emergency modes: Problems and solutions / A. Drozd, V. Kharchenko, S. Antoshchuk, M. Drozd // First International Workshop “Critical Infrastructure Safety and Security“ (CrISS-DESSERT’11). – Kirovograd, Ukraine, 11 – 13 May, P. 139 – 147, 2011.
2. Drozd A. Checkability of safety-critical I&Cc system components in normal and emergency modes / A.Drozd, V.Kharchenko, S.Antoshchuk, M.Drozd // Journal of Information, Control and Management Systems. – 2011. – Vol. 1, No.1.
3. Drozd A. Checkability of the digital components in safety-critical systems: problems and solutions / A. Drozd, V. Kharchenko, S. Antoshchuk, J. Sulima, M. Drozd // Proc. IEEE East-West Design & Test Symposium. – Sevastopol, Ukraine. – 9-12 Sept., 2011. – P. 411 – 416.

Conclusion

1. The fault tolerant technology does not solve a problem of safety **for the S-CES**.
2. The reason of this follows from **peculiarities of the S-CES** like two-modes systems and consists of low checkability of the digital components.
3. This conclusion is confirmed by using the method for checkability estimation. The method is based on analysis of controllability and observability of the digital component points in both a normal and an emergency mode.
4. The reasons of the low digital component checkability follow from use of **the high technologies**, such as high level of **the input data consistency** in a normal mode, high value of **ratio of the threshold per noise**, high level of **the circuit parallelism**.
5. The ways to increase checkability are based on rational use of **the high technologies**.

Questions and tasks

1. Why the fault tolerant technology does not allow to solve a problem of safety **for the S-CES**?
2. What is **the reason** of low checkability of the **S-CES** digital components?
3. Describe the **main issue** of the method for the checkability estimation.
4. What ways to increase the checkability of the **S-CES** digital components **do you know**?