MANCHESTER 1824
Manchester University Press

Split flowcharts in teaching digital system design

Binyamin B. Abramov and Ilya Levin
School of Education, Tel Aviv University, Tel Aviv, Israel
ilia1@post.tau.ac.il

Abstract Teaching design of digital systems is usually based on one of two main paradigms, namely either declarative–based on a functional description of system; or procedural–based on constructing a flow chart (algorithm) defining a sequence of steps of the system's operation. The paper proposes an integrative approach for digital systems description and digital systems design teaching, both for computer engineering and electrical engineering students. This approach combines together the main aspects of the above two paradigms. The central concept of the new approach is a split flowchart. The proposed approach enables a parallel representation of the description of digital system functional fragments that in turn can be described procedurally. This paper reports results of testing the effect of different paradigms on the design process and on its results, and especially the efficiency of the proposed approach in comparison with others. Additionally, a case study of teaching an introductory digital design course based on the proposed approach is presented.

Keywords computer engineering education; digital systems design; electrical engineering education; flowchart; split flowchart; teaching design

The present paper belongs to the field of computer and electrical engineering education. Specifically, we deal with teaching digital systems design. Naturally, the most valuable skills acquired during the digital systems design course are theoretical and practical skills for designing systems. The system design process is a highly branched and iterative process, accompanied by a number of dilemmas, for example by accepting/rejecting alternative ideas. Methodological aspects of teaching are of great importance in such cases.

During the last few decades, a number of leading universities have worked intensively in order to restructure the computer engineering curricula for beginners and sophomores, by developing innovative approaches for undergraduate engineering education so as to improve young engineers' design skills.[1–3]

In our paper, we present some aspects of teaching essentials of digital systems design for an undergraduate programme. Specifically, the focus of our study is on the systems' representation and modelling. Any learning activity, including learning of design principles, takes place in a particular context. Given the specific learning activity, its context has to be 'designed' to satisfy its goals.[4] The present study focuses on the context of designing digital systems, to implement its particular predefined goals.

The course 'Introduction to Digital Systems Design' is a general course developed for teaching the principles of design, testing and modelling of digital systems. This field of engineering has traditionally been considered as a part of electrical engineering. The complexity of modern digital systems and mutual penetration of various engineering fields which until recently were clearly distinguishable mean that a knowledge of principles of digital design has become an integral part of computer engineering. In our opinion, the course digital design should become a basic course both for computer engineering and electrical engineering students.

Our study was conducted on problems of the designing of combinational and sequential circuits. We tied together the two sides/approaches of computer engineering: the hardware approach (nonflexible digital circuits, hardwired, and tailored for particular purpose, characterized by parallel operation and high performance), and the program approach (a classical procedural approach, characterized by flexibility and changeability). Our study deals with a benefit of using a so-called split flowchart in the design of digital systems. We believe that this idea will combine the best of the two existing approaches.

Educational background: existing design paradigms and representation tools

During a design process, the designer navigates through an abstract, so-called problem domain and employs various strategies to elaborate the problem's description. In modern complex digital systems design, any design can be presented within three domains as shown in the Gajski and Kuhn Y-chart.[5] Figure 1 shows the Y-chart with its behavioral, structural, and physical axes corresponding to the three domains. In the behavioral domain, the intended behavior is specified (ideally) in a way that is not biased towards any particular implementation. In the structural domain, one deals with the system as a hierarchy of functional units and their interconnection. In the physical domain, the structure of the design is mapped into space, without any reference to its functionality. The abstraction levels used are not fixed, but a typical set[6] can be seen in Fig. 1. Five concentric circles that characterize the

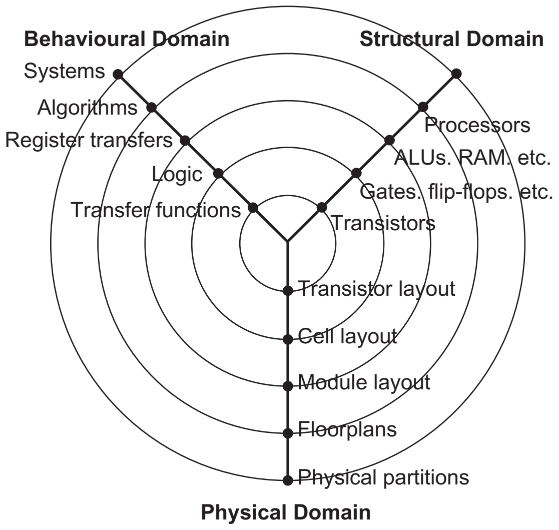

Fig. 1 *Gajski-Kuhn Y-chart.*

hierarchical levels within the design process are: architectural, algorithmic, functional block or register-transfer, logic and circuit levels. The abstraction increases from the inner to the outer circle. Each circle characterizes a model (explained below). The designer's attention shifts from higher level overall views of the problem down, to consideration of lower level details of the problem.

The design process of a digital system can be divided into two phases: the phase of design implementation (the main objective in our course) and the phase of design verification.

Based on the design hierarchy indicated above (Fig. 1), our study deals with two main domains: a behavioral domain that describes temporal and functional behavior of a system, and a structural one, that describes a system as assembled from subsystems. We focus on the behavioral, particularly on the algorithmic level of the abstraction. The design process transforms a set of specifications into implementation. Actually, both the implementation and the specification are forms of description of the system functionality that belong to different levels of abstraction.

Given the behavioral description of the system's (observed or desired) functioning, from the formal model stage onwards, we suggest introducing and using the distinction between two main existing approaches to design: the procedural (algorithmic) approach and the declarative approach.[7]

Within the declarative approach/paradigm, a person (e.g., student, user, designer) defines the structure/composition of the system and focuses on a logical scheme inside the digital system, implemented by means of basic elements or primitives (e.g., logic gates, memory units) (see, e.g., Ref. 8). The declarative approach is characterized by so-called 'what' aspects of the solution. In contrast, the procedural approach focuses on the behavior of the system, on the algorithm and rules of its functioning – so-called 'how' aspects of the solution (see, for example, Ref. 9). A large digital system usually involves complex tasks (algorithms), which can be expressed as a sequence of actions based on the system states and external commands. The design by this approach normally starts with an abstract (usually a graphic) description, such as a flowchart.

The flowchart is a directed, connected graph containing an initial, a final (Fig. 2(a)) and a finite set of conditional (Fig. 2(b)) and assignment (Fig. 2(c)) vertices.

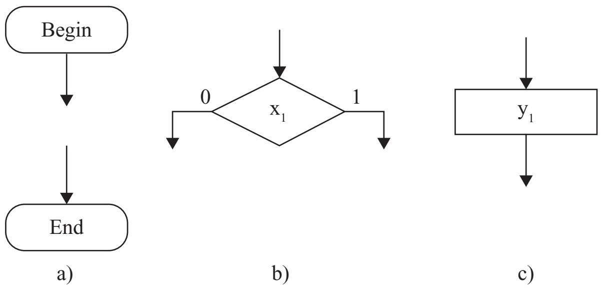

Fig. 2 *Flowchart basic shapes.*

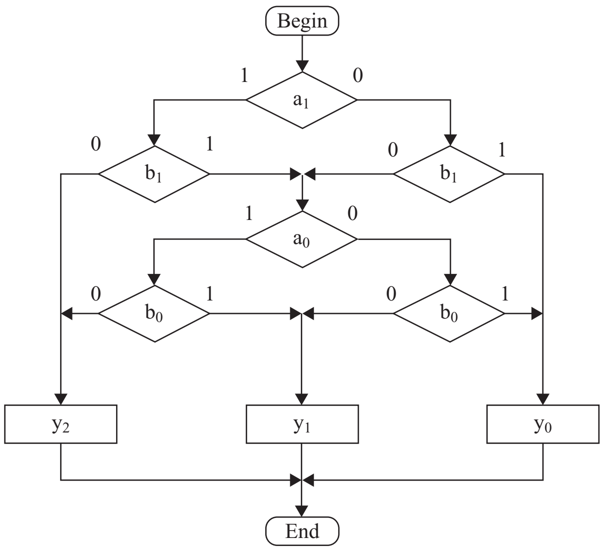

Fig. 3 *Flowchart of 2-bit comparator.*

The conventional flowchart satisfies the following rules:

- Every output is connected with only one input
- Every input is connected with at least one output
- Every vertex is located at least at one of the paths leading from the initial vertex to a final one
- One of the outputs of the conditional vertex can be connected with its input.
- One of the logical conditions of the set X is written in each conditional vertex.

The above notation enables description of both combinational and sequential circuits. Figure 3 shows an example of a simple description of a combinational circuit (2-bit comparator) by using a flowchart.

The split flowchart concept

We consider a specific system description style – *Dichotomic Networks* (DN) – as a class of representations, that corresponds to a particular specification of a digital system. Dichotomic networks comprise so-called *Dichotomic Fragments* (DF).[10–12] We refer to the DF-fragments as to binary trees, while the DN is a system of the interconnected fragments. Some fundamental properties of logical specifications, such as completeness and non-contradiction can easily be checked in the DN. The DFs can be considered as cognitive templates that are preferable to humans when creating digital system representation.

Two important and well-known kinds of the dichotomic representation of a digital system are Binary Decision Trees (BDTs) and Binary Decision Diagrams (BDDs), and in both cases we talk about varieties of flowcharts. We can use a flowchart to specify a complex sequence of events involving commands (inputs) and actions (operations or outputs), which are hallmarks of a complex algorithm. Being a very convenient form for system specification, the flowchart, at the same time, has a significant cognitive contradiction. On the one hand, the flowchart corresponds to the classical von Neumann's architecture that is based on the sequential nature of algorithms. On the other hand, at the implementation level, behavior of the flowchart is principally concurrent. One of the main requirements for a modern systems' specification is its ability to support the concurrency. Additionally, the flowchart description includes a significant redundancy, associated with its tree-like nature.

To overcome the above flowchart disadvantages we introduce a novel type of system specification – a so-called split flowchart. Being an alternative to the conventional *monolithic* flowchart system description of the system, the split flowchart supports concurrency by describing the system as a set of component concurrently functioning flowcharts. Notice that a particular kind of split flowchart notation was efficiently used for specifying the logic control of robots, defined by a set of concurrently functioning sub-flowcharts.[13]

Development of such a flowchart requires designer's thinking in terms of flowchart paths, each corresponding to a specific disjoint cube. The entire flowchart forms a single dichotomic fragment. The flowchart specification is quite natural, logical, and testable; however, it requires bearing in mind all the possible logical paths (situations) and referring to them within the system. While this strict requirement renders the flowchart specification logical, it puts designers in a position where they have to define very sophisticated and even artificial/unexpected states of the system. When the number of variables grows, the process of defining the flowchart specification becomes a difficult and unpractical task.

The split flowchart – a newly introduced concept having all the advantages of a conventional flowchart but being free from the above drawbacks – is the main subject of our study. A digital system can be defined both as a single complex flowchart, and as a split flowchart comprising a set of connected conventional flow-charts of lower complexity. In many cases, the split flowchart is preferable since it allows reduction of the initial description complexity. We define the split flowchart by presenting it as a network of connected conventional flowcharts (component flowcharts), and in one of the following ways:

1 Parallel connection: connecting roots of two or more component flowcharts.
2 Sequential connection: replacing one terminal node of a flowchart with another flowchart.

The mentioned component flowcharts (sub-graphs of the split flowchart) correspond to our dichotomic fragments. Output vectors of each of the fragments are logically summed. For a graphical representation of a split flowchart, we use the same figures as for the conventional flowchart (Fig. 2) with only one difference: if in the conventional flowchart only conditional vertices are able to have more than one output and only two sub-branches, the split flowchart might have unconditional branching, and the number of branches may be more than two. One example of the flowchart representation of a simple digital system that controls a two-axis manipulator is presented in Fig. 4: a conventional flowchart description C (on the left) and a split flowchart S (on the right).

This example is very illustrative, since there is a wide range of tasks characterized by independent and simultaneously occurring processes; such processes are the main targets for the split flowchart representation. As can be noted, the initial flowchart may be split into component flowcharts almost anywhere; moreover, each of the component flowcharts may also be divided in turn. The next example (Fig. 5) shows this property.

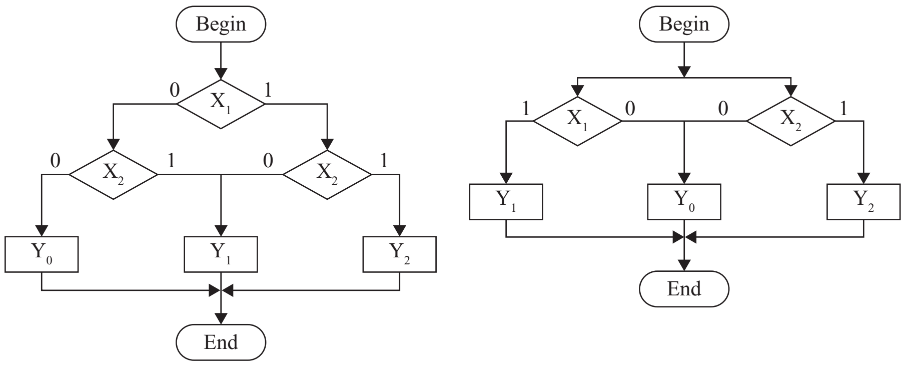

Fig. 4 *Conventional flowchart (left, C) and split flowchart (right, S).*

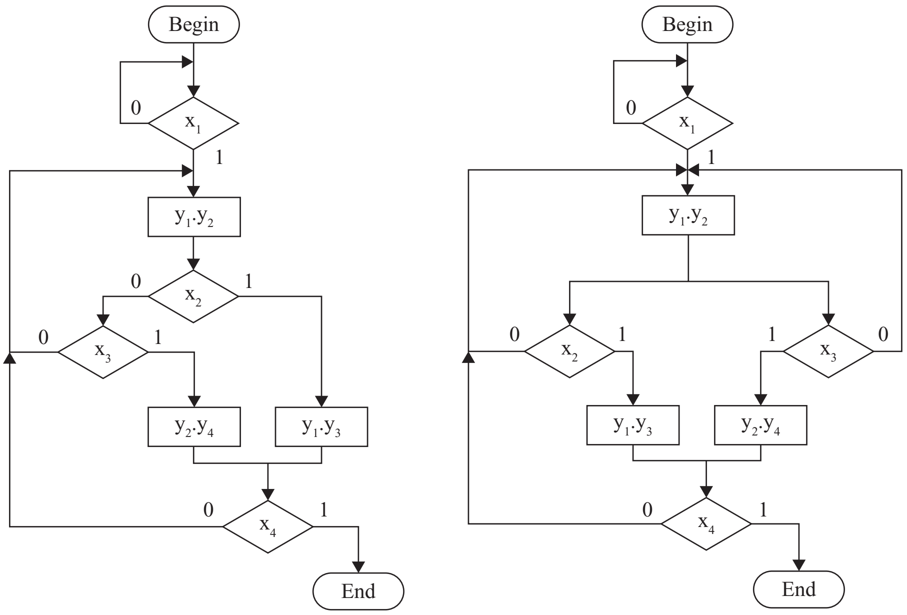

Fig. 5 *Two flowcharts of the same control algorithm of mobile robot direct movement. Conventional flowchart (left, C) and split flowchart (on the right, S).*

We further describe an approach that is based on the hypothesis proven in Refs 14 and 15, about the limited character of the human *working memory* capacity. The working memory is tasked with the burden of processing incoming information, transferring the information to human's long-term memory and retrieval of the information from the long-term memory. Cognitive learning theory[16] provides a number of fundamental principles explaining human learning:

- Human memory has two channels for processing information: visual and auditory. Human memory has a limited capacity for processing information
- Learning occurs by active processing in the memory system.

Our above-mentioned approach, with the split flowchart as the main model of this approach, corresponds well to the cognitive principles of design. Indeed:

- The split flowchart uses the graphical language (visualization)
- Each component flowchart includes a smaller number of variables than the corresponding conventional flowchart (complexity reduction)
- Reducing the number of variables in component flowcharts supports using the short-term memory.

Research aims and methodology

The central hypothesis of our study suggests that the above-mentioned properties of the split flowchart – namely, combination of the parallel and the sequential presentation of digital systems in a single model, with the ability to select the desired ratio between these two approaches – will allow participants in the experimental group to achieve better results in comparison with a control group. This advantage would manifest itself both in the construction of a model of the system, and in its implementation (i.e., in synthesis in a given technological basis). Success would be reflected in real achievements (grades assignments of the course), which were expected to be better than when using other representations.

The study was carried out in two engineering colleges. The students group included first-year students from the two specializations: 31 computer engineering students and 34 electrical engineering students. The students were divided into two groups: group A (34 students) that studied a conventional 'Introduction to Digital Systems Design' course. This group served as the control group. The second, experimental group B (31 students) studied the experimental course including the new split flowchart concept. As the control and the experimental groups included students of both specialties, teaching in each group was carried out without separating the students according to their specialties. It is well known that the most fundamental students' activity in the education of engineers is performing a novel *design project*. Design as a discipline is still young. It is neither science nor art, but like any other discipline, it has its own purposes, values, measures and procedures. Actually, designing is a process of creation of a digital system with predefined functionality. This system can be a small unit performing a number of functions according to a detailed technical specification. The notion *novel* is added to the design to emphasize the fact that it is expected from the students that they develop new concept/methodologies/principles etc. in their projects. The emphasis on *design* rather that *research* and/or *development* has a principle character; all activities of a computer and electrical engineer are usually performed in the field of a specific application. A clear comprehension of this application requires creativity that finds its roots in the focus on a specific human interest. The *design* activity involves the whole spectrum of creative functions initiated by that interest: methodology, ability to analyze, recognize/define/perform research activities and ability to perform these tasks.

The course 'Introduction to Digital Systems Design' is a fundamental course, including the principles of digital design both for computer and electrical & electronic engineering students. During the course, we analyze the following issues:

- How to go about designing really complex systems
- How to build a behavioral model of the system
- How to interface between the control unit and the remaining part of system
- How to describe the control unit algorithm
- How do describe digital building blocks of operational units (memories, processing elements, arithmetic units etc.) work
- How to use some of the modern CAD tools to help with the design
- How to deal with testing of such systems.

In this paper, we focus on the following questions:

- Whether the grades in the experimental group, among students who used the split flowcharts are higher than scores of the learners who did not use the split flowcharts?
- What is the use of the split flowcharts in the experimental group, for which tasks and for what level of complexity?
- Into how many component flowcharts (sub-graphs) do the students divide an initial flowchart, and what is the relationship between the number of component flowcharts, the kind of task and its complexity?

We have developed a list of tasks (Table 1) that were offered to students in various ways: as a classroom work, as an assignment, and as an exam.

TABLE 1 *The course tasks*

No.	Task	Kind of system	Complexity	Type of work
1	1 out of N code detector	combinatorial	4–10	homework
2	CSA adder	combinatorial	7–9	class work
3	Decimal digits detector	combinatorial	7	exam
4	Interrupt controller	combinatorial	5	exam
5	Sequential multiplier	sequential	5	class work
6	Distance sensor controller	sequential	7	homework
7	Modulo 6 counter	sequential	8	homework
8	Square root calculator	sequential	6	exam
9	Autonomous robot controller	sequential	+10	class work, homework

The assessment of the task complexity in this study is reflected by the number of input variables for the case of designing combinational systems, and by the number of input variables together with the number of internal states for the case of sequential systems designing. Some of the tasks (1 and 2 in Table 1) were set with multiple levels of complexity. For example, task 1 comprises seven separate design tasks of designing a 1-out-of-N code detector with the number of input variables from 4 and up to 10.

Case study

As was noted, our approach is especially appropriate to problems/tasks having so-called parallel nature, i.e., comprising a number of independent processes. In a set of test problems (Table 1), there are several tasks having such a nature. The most prominent representative of them is the task of designing a unit detecting words of 1-out-of-*N* code. In information theory, a constant-weight code (*M*-out-of-*N* code) is an error detection/correction code where all codewords share the same Hamming weight. A special case of *M*-out-of-*N* codes, are the 1-out-of-*N* codes that encode $\log_2 N$ bits for codewords of *N* bits. In addition to its use as an error detection/correction code, the mentioned code is widely used to encode Finite State Machines (FSM) in order to obtain the high performance, highly reliable FSM. This task is relatively simple from the algorithmic point of view, but the complexity of its solution heavily depends on the size of the input word (the number of input variables). Figure 6 illustrates a possible solution of the 1-out-of-*N* code detection problem, for $N = 8$. This solution was submitted by one of the students from the experimental group B; the solution was highly graded. The solution explicitly utilizes a combination of the two main approaches of digital systems representation: declarative and procedural. The flowchart starts from the vertex Begin and is divided into four independent sub-charts or sub-graphs (denoted P1 in Fig. 6). Each of the four sub-graphs gets, from the set of input variables, two variables belonging only to that specific sub-graph, and checks them for membership in the 1-out-of *N* code. Outputs of each of the four sub-charts P1 are checked in a sub-chart labeled P2 in Fig. 6, so that each sub-chart P2 receives the results from two sub-charts P1. As a result, we have the test results of each half word of the input word, which feed the next sub-chart P2 to produce the final result.

This solution is flexible since each of the sub-charts of the two types (type P1, type) shown in Fig. 6 allows representing it as an algorithmic routine (in accordance with the procedural approach). The letter P in the sub-charts corresponds to – Procedure, or to a basic component in the architecture of the system (according to the declarative approach). The letter P can also be considered as a Primitive (a basic system component). Moreover, these different interpretations of the sub-charts allow the students to choose representations that are appropriate to their preferences and skills. The given task can be divided into sub-charts with different number of input variables, followed by the detailed implementation of each of them (as shown in the dashed box Fig. 6), and once realizing it at the hardware level, with subsequent usage as a black box with a known functionality. Comparison of the above solution with

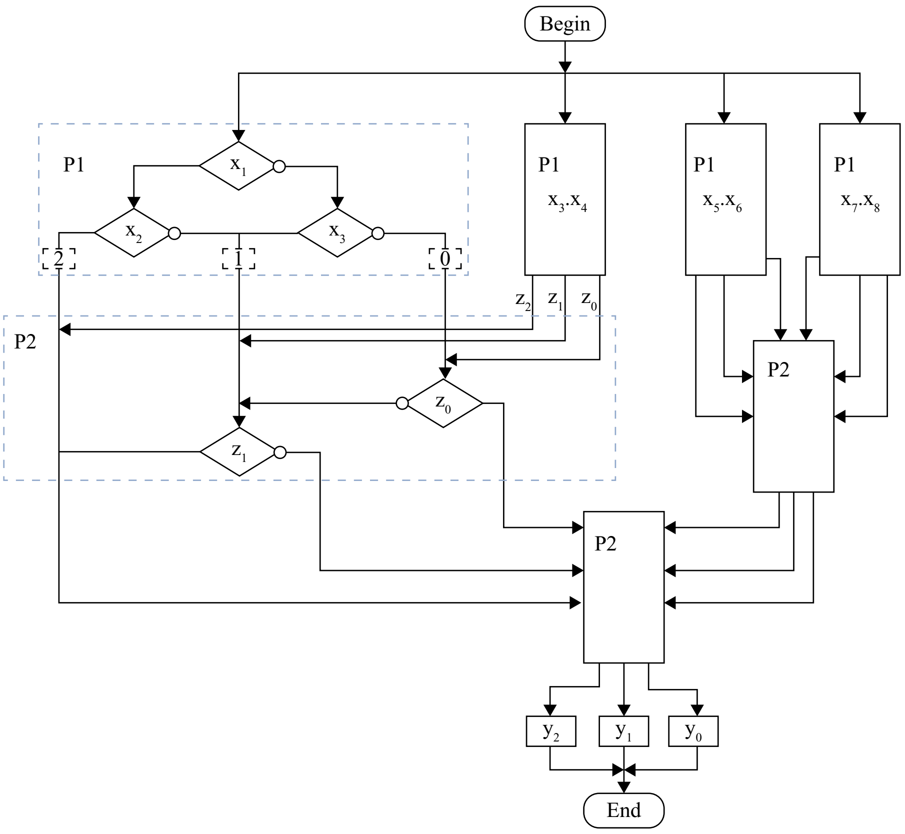

Fig. 6 *1-out-of-*N *code detection unit described by split flowchart.*

the conventional flowchart solution shows a clear advantage of the proposed split approach. This advantage is reflected in the simplicity of presentation, the number of vertices in each of the component flowcharts (much less than in the conventional flowchart), its readability and the ability to satisfy the students' preferences and potentials.

Results

Figure 7 shows the relationship between the split flowchart prevalence and the task complexity. These results were obtained in the analysis of solutions of the 1-out-of-*N* code detection problem with different complexity (from 4 to 10).

The main reasons for not choosing a split flow chart as a means of implementation, and their distribution are shown in Fig. 8.

The main results of our study are as follows:

- The experimental group students, who used the split flowchart, achieved higher scores than the control group students who used conventional flowcharts for

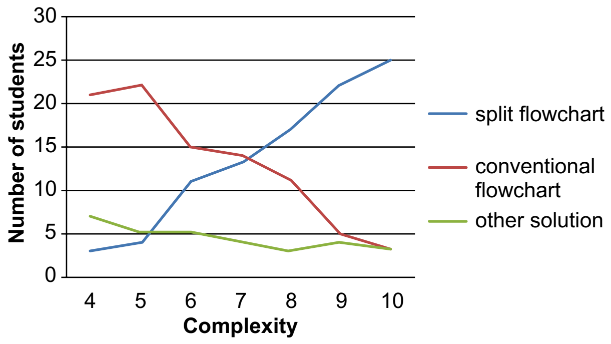

Fig. 7 *1-out-of-*N *code detection by split flowchart.*

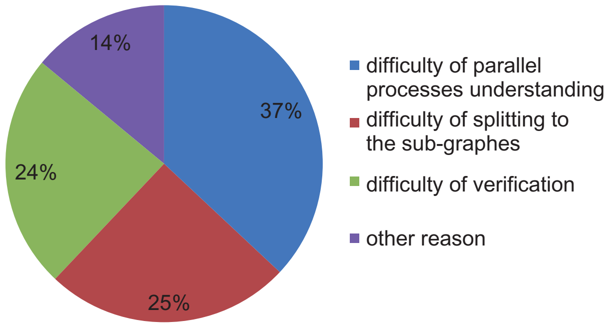

Fig. 8 *Main reasons for not choosing a split flowchart.*

solution of the same task. It was especially clear in tasks of combinatorial circuits design. For sequential systems design, students also achieved better results when using split flowcharts, although the advantages were less noticeable than in the case of combinatorial circuit design.

- In the experimental group, the split flowchart was the most common solution for the tasks of combinatorial circuit design with complexity = 7 and above (Fig. 7).
- In the experimental group, for the tasks of combinatorial circuit design, with complexity 7 and above, the split flowchart included more than two sub graphs.
- Starting from the task complexity = 8, the students' grades went down in both groups, while the experimental group students who used the split flowchart kept reasonable grades in comparison with the grades in the control group.
- There was no significant difference between the students of computer and electrical engineering.
- Despite the discussed advantages of the split flow chart in comparison with other approaches, this approach is still not generally accepted. Among the reasons for not choosing the split flowchart as a means of implementation, the main reason is difficulty of parallel processes understanding (Fig. 8).

Conclusions

Our paper describes a new method for digital system representations and using this method in teaching an Introductory Digital Design course. The method is based on using split flowcharts. The proposed approach allows, depending on the personal preference of a particular student, to combine two alternative styles of thinking about a digital system – concurrent and procedural. Experimental results demonstrate the efficiency of the proposed approach of teaching digital design in comparison with existing approaches. The main advantage of the split flowcharts approach is the possibility of designing systems of high complexity and, consequently, it opens a way for teaching sophisticated design in undergraduate classes. Moreover, the experimental results show that a course that is based on models of high-level abstractions can be successful in teaching both electrical engineering and computer engineering students.

References

1 P. W. Samaras, 'Integrating the first two years', in *Proc. ASEE* (1991), pp.16–19.

2 D. Lee, E. Trauth and D. Farwell, 'Critical skills and knowledge requirements of IS professionals: A joint academic/industry investigation', MIS Quarterly. Special Issue on IS Curricula and Pedagogy, **19**(3) (1995), 313–340.

3 J. Rice, T. M. Bayles, G. Russ and J. Ross, 'Preparing freshmen for future energy issues', *Proc. ASEE* (2007).

4 Y. Reich, E. Kolberg and I. Levin, 'Designing contexts for learning design', *Int. J. Eng. Educ.*, **22**(3) (2006), pp. 489–495.

5 D. D. Gajski, *Silicon Compilation* (Addison-Wesley, New York, 1988).

6 D. D. Gajski, *Principles of Digital Design* (Prentice Hall, Upper Saddle River, NJ, 1997).

7 I. Levin and D. Mioduser, 'A multiple-constructs framework for teaching control concepts', *IEEE Trans. Educ.*, **39**(4) (1996), 488–496.

8 I. Levin, 'Matrix model of logical simulator within spreadsheet'. *Int. J. Elect. Enging Educ.*, **30**(3) (1993), 216–223.

9 I. Levin, 'Behavioral simulation of an arithmetic unit using the spreadsheet', *Int. J. Elect. Enging Educ.*, **31** (1994), 334–341.

10 I. Levin and O. Keren, 'Split multi-terminal binary decision diagrams', in *Proc. 8th International Workshop on Boolean Problems (2008), 161–167.*

11 B. Abramov, O. Keren and I. Levin, 'Teaching cognitive-inspired design of sequential circuits', in *Proc. 8th European Workshop on Microelectronics Education*, Darmstadt, Germany, (2010).

12 I. Levin and O. Keren 'A generalized if-then-else operator for the representation of multi-output functions', *Mathematical Problems in Engineering*, vol. 2013, Article ID 401616, 13 pp.

13 I. Levin and V. Levit, 'Controlware for learning with mobile robots', *Computer Sci. Educ.*, **8**(3) (1998), 181–196.

14 G. Miller, 'The magical number seven, plus or minus two: Some limits of our capacity for processing information', *The Psychological Review*, **63**(2) (1956), 81–97.

15 P. Lindsay and D. Norman, *Human information processing (an introduction to psychology)* (Academic Press, New York, 1977).

16 R. C. Atkinson and R. M Shiffrin, 'Human memory: a proposed system and its control processes', in *The psychology of learning and motivation* (Academic Press, London, 1968).