FORMALIZING THE LOGIC DESIGN OF REGISTER COMPONENTS IN DISCRETE SYSTEMS

A. A. Mishchenko and A. T. Mishchenko

Discrete hardware design involves at a certain stage the logic design of three types of components: functional components (combinational logic networks), operational components (registers), and control components (automata) [1]. Control automata design has been developed in considerable detail, while formal (and thus computer-aided) register design still involves difficulties, despite its apparent simplicity. In practice, register design uses various flipflops as storage elements, and each type of flipflop can be implemented by dozens of different logic circuits with different reliability characteristics and optimization options. Moreover, writing data to a register may involve more than ten different register operations. The designer thus has to choose the hardware solution from a large number of options, allowing for the time characteristics of the input signals, as well as requirements for speed, cost-efficiency, reliability, etc.

The present article is an attempt to propose a solution of this problem through the development of a formal register design procedure originally described in [2]. The designer, using his experience and intuition, allows for a variety of circumstances and constraints, selects some register design, and describes its operating algorithm in an accessible language. The register circuit is then constructed on the basis of this algorithm. The simplicity of the procedure enables the designer to foresee the outcome of the computer-aided design procedure. By formulating the register description, the designer is thus capable of making supplementary decisions and allowing for constraints in such a way that the end result is a reliably functioning unit.

We now proceed to present the algorithmic description language and the procedure for formal register design.

REGISTER STRUCTURE

A register is a device that receives, stores, and delivers information at specified time instants. An n-digit binary number \( a_1a_2...a_n \) requires an n-bit register: each register bit contains the corresponding digit of the binary number and is implemented using a single flipflop. A simplified register structure is shown in Fig. 1. The input and output lines link the register with other network components. Let us describe in general terms each part of the register, namely the flipflops, the term former (combinational logic circuits), the input former, and the operation control former.

A flipflop is a two-state automaton with a complete system of transitions. A flipflop may have a different number of inputs and in general two outputs for signals that are one the inverted form of the other. This device can maintain one of two possible stable states, 0 or 1. The transition from state 0 to state 1 and back in a flipflop is triggered by input signals, which have the values 0 or 1. Different types of flipflops are used in design practice. Each flipflop type is characterized by its own state-change technique depending on the input signals. The different flipflop types are characterized using the standard notation for flipflop inputs, and the flipflop behavior is described by transition tables. Transition tables for common flipflops are shown in Tables 1a-h, where the letter denotes the flipflop state [3].

Table 1a shows flipflops of groups RS (RS, R, S, E), and Tables 1b-h show flipflops of types D, T, DV, TV, JK, RST, DVT, respectively. The dash in Table 1a indicates that the input combination \( R = 1, S = 1 \) is forbidden in RS flipflops; all the input patterns not shown explicitly in Tables 1g and 1h are also forbidden. The last two tables describe combined RST and DVT flipflops, which function as a T flipflop for \( T = 1 \), and as RS and DV flipflops, respectively, when 1 appears on the other inputs.

The information inputs R, S, D, T, J, K receive information that has to be stored; the control inputs V open (V = 1) or close (V = 0) the access to the flipflop through the information inputs. Clocked flipflops, in addition to information and control inputs, have special inputs that receive clock signals \( \bar{\tau} \), as well as setting inputs \( R' \) and \( S' \) that are used to reset the flipflop to state 0 and 1, respectively. Reset is performed, in particular, when the flipflop is cleared by a special signal \( \bar{N} \) before starting to operate; in the course of normal operation, the flipflop may be cleared by writing zeros through the information inputs.

A flipflop typically has two outputs. On one of the inputs (the 1-input) the signal \( a \) corresponds to the flipflop state, whereas on the other input (the 0-input) the signal \( \bar{a} \) is the inverted value of the first signal.

The term former (Fig. 1) is an input stage that receives information from other registers and devices. The incoming information is written into the flipflops by one of the write operations (many such operations may be supported in flipflops). Each operation \( i \) is executed when the register receives the signal \( U_i = 1 \), which is the condition for the execution of the corresponding operation formed in one of the control devices. When \( U_i = 0 \), the operation \( i \) is not executed.

We denote by \( f_i^j \) (resp., \( \bar{f_i^j} \)) the information written by operation \( i \) in register bit \( j \). This operation is received from 1-outputs (resp., 0-outputs) of other devices. Then the direct term \( B_i^j \) and the inverted term \( C_i^j \) of operation \( i \) are defined by the following expressions:

\[
B_i^j = U_i \& f_i^j, \tag{1}
\]

\[
C_i^j = U_i \& \bar{f_i^j}. \tag{2}
\]