Time, Timing and Clock in Massively Parallel Computing Systems

Victor Varshavsky,
the University of Aizu, Japan
E-mail: victor@u-aizu.ac.jp

> "Time is money."
>
> *proverb*

1 Introduction

Time is one of the most complicated and poorly understood objects of Nature, although most of us suppose we perfectly know what it is. History, psychology, art, micro- and macrophysics, cosmology and a number of other sciences deal with various aspects of Time and actually have different points of view on it. The subject of this paper is Time in artificial systems or Artificial Time. This subject is specific and obviously requires special consideration. Before I pass to the main contents of the paper, I will afford some speculations on this topic.

Starting from, most likely, Isaac Newton who clearly enough formulated this concept, Time is treated as an independent physical value (Physical Time). All the definitions of Physical Time with which I am familiar are associated in this or that way with the procedure of measuring Time[1] and the concept of simultaneity. Measurement, in its turn, is associated with its precision, so that simultaneity is an abstraction within the precision of a given measuring instrument (clock). The theory of relativity introduced the concept of event and dependence of a measured time interval on observer's position. Whatever Einstein meant, he introduced the concept of informational exchange (signal about an event). Furthermore, an event itself is discrete and a natural question arises: is physical time discrete or analogous? From the standpoint of the basic definition, it is analogous. On the other hand, it is granulated, at least, by the precision of the measuring instrument. Since any measuring process takes energy and is an intervention in the measured process, the uncertainty ratio in the form connected with time can provide the information about the minimum quantum of time and, perhaps, about the attainable accuracy of a measurement.[2] The uncertainty ratio linking energy and time is $\Delta E \Delta t \geq \hbar$. It means that the energy of interaction between a measuring instrument and the measured system can be known with an accuracy of $\hbar/\Delta t$. This is a distant limitation for today's microelectronics, but for FED[3] which are expected to be used in the second decade of the next century, the time of energetic level (state) change of ΔE has Poisson distribution with $\tau = \hbar/\Delta E$.[4]

Strange though it may seem, a definition of Time associated with a sequence of discrete events was given much earlier, as far ago as by Aristoteles. According to him, Time is a way of ordering events, reflecting cause-and-effect relationships between them. From this point of view, Time is a logical abstraction (Logical Time) and its essence is reflected in a brilliant phrase by Aristoteles: "If nothing happens, no Time."

Both physical and logical time were defined with reference to natural time associated with human-independent processes flowing in Nature. But from the very early periods of history people had to create artificial systems (devices) whose behavior and usage were associated with Time. The first example is a clock. Functioning of all known clocks is connected with processes which flow in analogous physical time. In most cases, a clock transforms analogous physical time into a sequence of discrete events. Another example is a calendar which represents systems establishing a logical order on events. However, the problem of

[1] This is natural like for any other physical variable. The measuring instrument for Time is a clock.

[2] Frankly speaking, I could not get any perspicuous explanation from physicists about the meaning of this form of uncertainty ratio. Probably, the people I asked were not experts in this field.

[3] FED — Future Electron Devices, such as Single Electron Transistor, Wave Function Rearrangement Devices, Quantum Dot Cells, etc.

[4] $\tau \cong 6 \cdot 10^{-4} p$ sec/eV. Note that for advanced microelectronic devices, characteristic energies of switching are of the order FJ (1 FJ≈600eV) and Δt has the order of fractions of a picosecond, without considering the energy of recharging the capacitors of the parasitic capacities.

Time in artificial systems faced people in all its magnitude after they began to create processes modeling tools, both analytic and technical ones.

Analytic analogous models, such as differential equations, contain analogous physical time as an independent variable without any special problems, the more so as the independent physical time definition itself was, in one way or another, associated with the development of differential calculus.

Analytic models of discrete processes, such as difference equations or finite automata, deal with discrete time, or rather with process step numbers, which at the first glance have nothing to do with natural time within this model. But this is true only for isolated systems; as soon as we try to organize or describe[5] the interaction between the system and its environment, we will have to examine the interaction between formal and natural times. This is indirectly connected with the difficulties of transition from abstract automaton model to structural one and problems with definition of an asynchronous automaton in the classical structural automata theory.[6]

As we pass on to technical systems modeling certain processes, we inevitably encounter the fact that physical devices that form a technical system are functioning in the natural physical time while the analytic processes they model are defined in the artificial formal time. Thus, the problem of introducing time into an artificial system (System Time) and problem of organizing the interaction between system time and physical time become design problems.

This problem arose for the first time in the beginning of this century, when mechanical differential analyzers were created and used in real time mode. A mechanical differential analyzer has a disc integrator as one of its basic elements (Fig.1a).[7] If the small disc of radius r moves relative to the big disc center by $R(X)$, and the rotation angle of the big disc axisis equal to X, the rotation angle of the small disc axis equals $F(X) = \int_0^X \frac{R(X)}{r} dX$. It is easy to see that the integrator acts independently of time and any variable can be used as independent[8]. If we want to deal with physical time as an independent variable, it is enough to have a permanent rotation speed ω of the big disc axis. Then $F(X) = \int_0^t \frac{\omega R(t)}{r} dt$ and the synchronous motor that provides a permanent rotation speed acts as a clock common for all the system. The situation principally changes when we pass on to electronic differential analyzers. Their basic component is an integrating amplifier (Fig.1b) whose functioning explicitly depends on physical time and all variables are functions of time. Now we need a clock again to compare the values of all the variables in the same time moment.[9] We have already some problems caused by the fact that the amplifier has its own delay and changes propagate through it with some delay. Although it is negligible for most applications, modeling very fast processes requires time scaling.[10]

The history of discrete calculating tools development began in ancient times.[11] However, the question about time in which technical facilities are functioning appeared much later. The more so as it was much later when formal models of behavior of such devices appeared. Mechanical calculating devices, such as machines by Pascal, Leibniz, Odner or Bebbige's analytic machine had a single rotation of the drive shaft[12] as a system time unit. The shaft was rotating in real physical time, but it practically was not taken into account in the process of functioning. Introducing electric drive in the beginning of the 20th century[13] did not change the situation. Due to the rigidity of an electric drive and gearing, there was no noticeable delay in information transfer. For example, not the delay but the drive shaft moment changed depending on addition carry length.[14] In calculators where addition and subtraction were automated, the sequence of calculation steps was determined by the sequence of drive shaft rotations (system clock). Curiously enough, the idea of determining system time steps by data readiness, lately called "data-flow architecture", was first used in the 18th century at Proney's calculation bureau which compiled navigation tables. A group of top-level mathematicians, including Lagrange, devel-

[5] It is probably the same within analytic models.

[6] Nothing else was invented beside *"asynchronous automaton is an automaton the transitions of which are initiated by changing the state of its input and any transition in which is completed by a stable state"*; but everybody who has lectured on structural automata theory knows how difficult, sometimes impossible, it is to answer students' questions: "What, after all, is the difference between a synchronous and asynchronous automaton?"

[7] It also performs the operation of differentiation, if the feedback circuit contains a tracing system. A delay in the feedback circuit affects only the precision of calculation.

[8] Or dependent

[9] Note again that simultaneity is attainable only within the clock accuracy.

[10] Scaled time is nothing but artificial system time.

[11] Remember notches on a tree, knots and stones used for counting and, finally, counting boards and abacus. Note that the mathematical thinking of ancient people was algorithmic; for example, in Egypt a mathematical task was considered to be solved only if an algorithm of relative calculation on abacus was found.

[12] Or a single movement of the lever in a proportional lever machine.

[13] Calculators by Reinmetall, Mercedes, Facit, Fenix, etc.

[14] This changed not the system time unit but its duration in real physical time and could be compensated by a power margin.

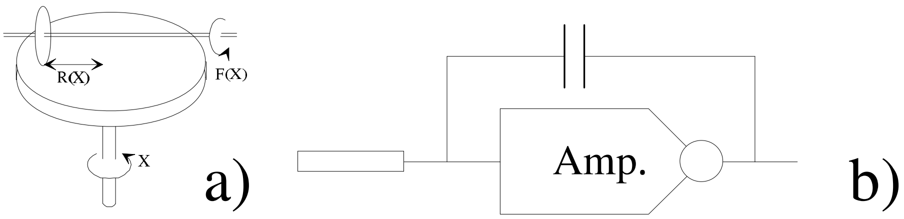

Figure 1: Integrators.

oped calculation methods (algorithm) and a system of tables. The output data of such a table was the result of a single arithmetic operation over the input data (a program). A calculation operator received his initial tables and filled out one or several output tables which were submitted to the next operator. An operator began the next step of the calculation process after he got all the initial data for this step.[15]

Obviously, the problems of interaction between physical and system (logical) times really appeared along with relay devices in industrial automatics and signalization/interlock systems on railroads. These problems were caused by the danger of races, the character of which depended on variations of times of relay contacts switching. It was the problem of races that posed the task of organizing logical behavior invariant to the behavior of components in physical time. The works by Shestakov and Gavrilov (USSR), Shannon (USA) and Nakamura (Japan) necessitated by practice led to the development of relay device theory and structural theory of finite automata (Moore, Mealy, Haffmen, etc.). The appearance of electronic lamps and transistors did not change the general situation for the time being.

The model of a finite automaton

$S(t+1) = F(S(t), X(t))$

$Y(t) = \Phi(S(t), X(t))$ (Mealy model) or

$Y(t) = \Phi(S(t))$ (Moore model),

where $S(t)$, $X(t)$ and $Y(t)$ are the state, input and output respectively, included the parameter t as the number of a process step, not associated with physical time. In fact, it already was artificial system time.[16] It was the effort to eliminate t from the automaton equations that led to the concept of asynchronous automaton for which the current state is determined by a stable solution of equation $S = F(S, X(t))$. The parameter t keeps its value of input states sequence numerator. As soon as the asynchronous automaton model was suggested and studied as applied to real design tasks, the questions immediately appeared about the linkage between the logical behavior of an automaton and physical behavior of its components. Answers to these questions were being sought at the logical level (anti-race coding, etc.).

Theory of algorithms (Markov's normal algorithms, Turing and Post machines, etc.) and abstract automata theory (Kleaney's regular events, etc.) also developed ignoring real time. These models treated system time as a step of the algorithm or a number of symbol position in the input pattern. However, some contradictions between the structural and abstract automata theories had already become evident.

Indeed, from a regular event $xxxx(xxxx)^*$, using the direct algorithm of McNotohn-Yamada, we can build a finite automaton representing this event (Fig.2a).[17] In terms of the abstract theory, all this is absolutely correct. But in terms of the structural theory, such an automaton is a generator rather than modulo-4 counter. To provide correct functioning of the automaton, we should introduce either a spacer separating two successive symbols x or a signal t of symbol change. Then, to provide the introduction of system time and correct functioning of the automaton, we should change the input alphabet and regular event, the representation of which is modulo-4 counting. Let $y = x \& t$, $w = \bar{x} + \bar{t}$ (synchronous version) or $y = x$, $w = \bar{x}$ (asynchronous version) and let the corresponding regular event be $ywywywyw(ywywywyw)^*$. A change of a regular event causes a change of the finite automaton that recognizes the event (Fig.2b).[18]

In spite of many works on asynchronous automata and obvious successes of the theory, using an external clock (synchronous implementation) appeared to be the simplest way of system time introduction[19] for many years. For the example above, so called master-

[15] Isn't it a multiprocessor data-flow computer system?

[16] Note the conceptual resemblance between the model of a finite automaton and difference equation.

[17] Modulo-4 counter.

[18] We will get the same automaton if replace y and w by yy^* and ww^* respectively; but this will be just an illusion of asynchrony within the language of regular events.

[19] Or, more precisely, transforming physical time into system time.

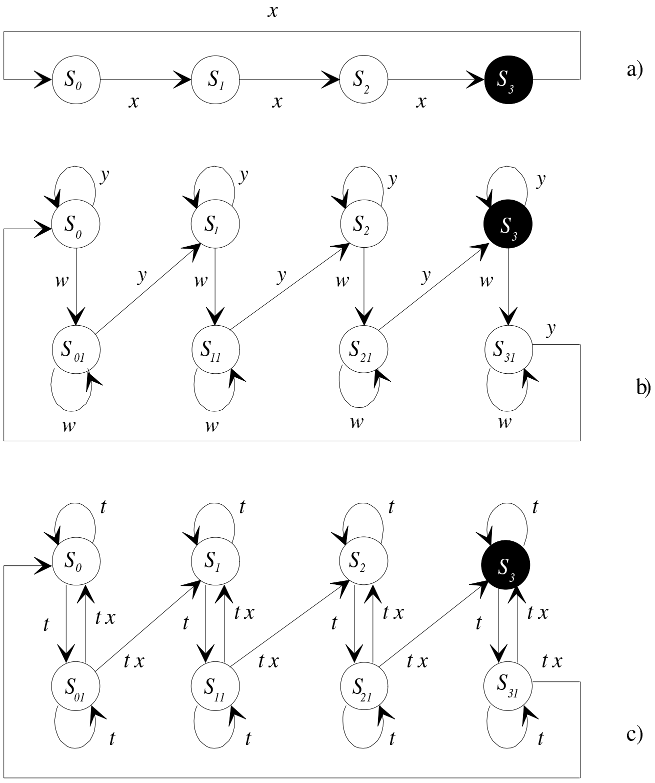

Figure 2: Inserting time into automata graphs.

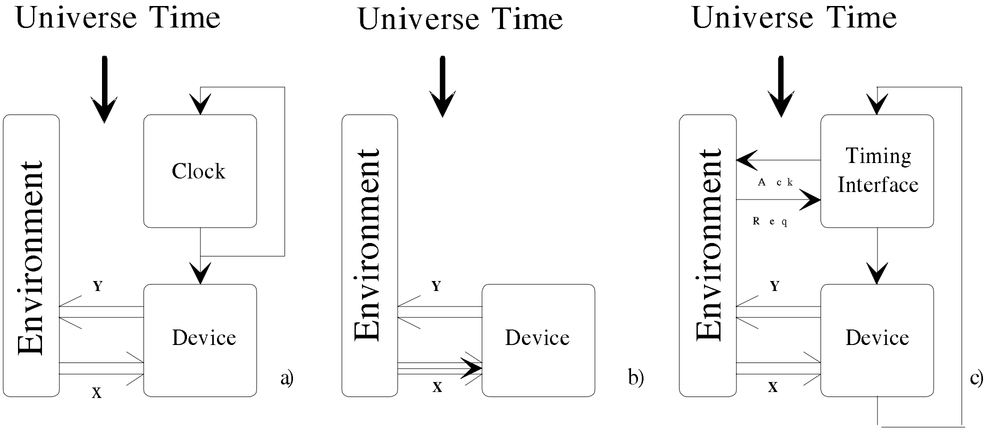

Figure 3: Time interactions with the environment.

slave implementation of the finite automaton was specified by the graph in Fig.2c.

Using an external clock or, in more exact terms, an external clock signal as an extra global variable which forms the steps of process of device discrete states change is explained by a simple structure shown in Fig.3a.

A clock is a periodical calibrating process which splits to discrete steps the processes flowing in analogous physical world time in the environment and in the device. There should be an agreement that both in the environment and in the device the duration of phase transitions from one discrete state to another does not exceed[20] the duration of a clock functioning period. Within this structure, the clock is nothing but a calibrated delay. It is important that events in the clock are caused by neither events in the environment nor those in the device. All the three objects above have, among others, only one reason of events within them, which is the flow of the world physical time. Note for the future that the absence of cause-and-effect relationships between these events or, in other words, a violation of behavior causality is one of the sources of troubles the synchronous implementation has.

In the classical asynchronous implementation (Fig.3b), the environment initiates a transient process in the device.[21] Again, the temporal behavior of the environment is largely causally independent from the device behavior. In cases when the environment is a part of the Universe, it is true and the problem of interface between system and natural times requires posing and solving special tasks.

However, if the environment or its considerable part is an artificial system, we have a right to introduce causality into the interface between them. This task is solved by so called matched implementation (Fig.3c). The essence of matched implementation suggested in the pioneer works by D.Muller is that the environment dictates the rhythm of state change to the device and vice versa. At least one output of the device changes only after the next transient process in it is completed. The development of Muller's model led to the creation of a new class of devices with causal behavior which were self-timed (speed-independent, delay-insensitive) devices.

2 Muller's model [5,6]. Causal devices.

One principal thing about Muller's approach is that he made his model non-determinate.[22] Indeed, since asynchronous behavior is generally associated with unpredictable variations in durations of transient processes from one state to another in physical time, the state of an asynchronous device in any fixed moment of physical time is not determined. In some cases, we have sufficiently full understanding about the probabilistic nature of delay variations and parameters of corresponding distributions of probabilities.[23] However, when studying logical behavior, the probabilistic approach is not of much benefit. Actually, we are interested in getting the answer to one particular question: "How and at what conditions can we get determinate behavior in logical time from non-determinate behavior in physical time?" To get the answer, the full totality of logical possibilities is analyzed in Muller's model.

A full circuit or Muller's circuit[24] is defined as a finite set S of N objects a, b, c,... , called states of the circuit. The circuit behavior is determined by the set of permissible sequences of states: $\left\{ \begin{array}{c} a(1), a(2), ... \\ b(1), b(2), ... \\ .... \end{array} \right\}$ which have the following properties:

- $a(i) \neq a(i+1) \quad \forall i$;
- If $a(1), a(2), a(3), ...$ is a permissible sequence, then $a(2), a(3), ...$ is also a permissible sequence;
- If $a(1), a(2), a(3), ...$ and $b(1), b(2), ...$ are permissible sequences and $a(2) = b(1)$, then $a(1), b(1), b(2), ...$ is a permissible sequence.[25]

A circuit can be defined by an oriented graph of possible adjacent transitions from one state to another. States $a \in S$ and $b \in S$ are in relation $a\Re b$ if there is a logical possibility of direct transition from a to b, i.e. if the graph contains an arc leading from a to b. Relation $\Re$ produces a set of neighborhoods of a $Q(a) \in S$ such that $b \in Q(a)$ if and only if $a\Re b$. The sequence $a(1), a(2), ..., a(m)$ is called $\Re$-sequence if $a(i)\Re a(i+1)$ for $1 \leq i \leq m-1$. Every trajectory on the graph has a corresponding $\Re$-sequence of states.[26]

[20] To be more specific, within the accuracy of maintaining the calibration process period, we should say "is less than."

[21] By changing the input pattern X. Note that in the synchronous model also the device actually treats the clock as a part of the environment, so that the time signal T can be included into the input pattern: $X' = \{X, T\}$.

[22] Muller's results are stated according to [7], Chapter 10. You can find there the proofs of some statements given here.

[23] For example, when a quantum state changes or when an electronic arbiter quits a metastable state.

[24] Or just "circuit."

[25] The last two properties form Markov's property which means that the current state contains all the preceding history of behavior.

[26] But not every $\Re$-sequence is permissible.

We will say that a precedes b ($a\Im b$ stands for the relation of sequence) if an $\Re$-sequence exists (a trajectory on the graph) that leads from a to b. Relation $\Im$ is reflexive (a, a is an $\Re$-sequence) and transitive (from $a\Im b$ and $b\Im c$ it follows that $a\Im c$). If $a \neq b$ and $a\Im b$, then a and b belong to a permissible sequence.

States a and b are called equivalent (aEb) if $a\Im b$ and $b\Im a$, i.e. if there is a cycle in the graph that contains a and b. The relation of equivalence splits up all the set of states S to equivalence classes A, B, C,... The relation of sequence also takes place for equivalence classes. $A\Im B$ if two states $a^* \in A$ and $b^* \in B$ exist, such that $a\Im b$. If $A\Im B$, then $\forall(a \in A)\&(b \in B) \quad a\Im b$. If $A\Im B$ and $B\Im A$, then $A = B$.

The relation $\Im$ for equivalence classes determines partial order of the classes.[27] To any permissible sequence of states corresponds only one finite sequence of equivalence classes, and for any permissible sequence of states the last equivalence class exists called finite. In a partial order on equivalence classes there is one or several maximum classes. If the finite class for a certain permissible sequence is not maximum, it is called pseudo-maximum. According to Muller's definition, a circuit is called speed-independent[28] in respect to state u if all permissible sequences starting with u have the same finite class.

Theorem (10.2.5 [ ]). *Circuit C is speed-independent in respect to u if and only if the equivalence class U ($u \in U$) precedes only one maximum class K and does not precede any pseudo-maximum class.*

Circuit C defined above is an abstract object. Structural theory provides more detailed description, in particular, representing states by signals and signals by logical elements. Structurization requires changing[29] the circuit definition.

In structural theory of automata, circuit is usually defined as a totality of n logical elements connected with each other. Every logical element has k_i inputs ($k_i \leq n$) and one output x_i. Outputs of the logical elements are called nodes of the circuit and a set of n values of variables in the nodes $X = \{x_0, x_1, ..., x_{n-1}\}$ is its state.[30] Every input of a logical element is connected[31] with only one output and there are no two outputs connected with each other. Every logical element is associated with a logical function $x'_i = F_i(X)$, where x'_i is called the next value of x_i.[32] If x'_i explicitly depends on x_i, the logical element is a memory element. Otherwise, it is a combinational element. An output of a logical element in a certain state X_j is called stable if $x_i \oplus f_i(X_j) = 0$ and excited if $x_i \oplus f_i(X_j) = 1$. In an excited state, the inputs of a logical element *have already changed*[33] while its output variable *has not changed yet.* If in a certain state X_i m variables are excited, there are 2^m logical possibilities for states change which produce a graph of full Muller's circuit. If an abstract scheme corresponds to every structurized circuit, all definitions and results obtained for an abstract model[34] can be generalized for the structurized model. However, specific features of the structurized model makes it reasonable to discuss other approaches.

According to the definition of excited and stable states of a logical element in Muller's binary model, every variable can have four values: $\{0, 1\}$ are stable values and $\{0^*, 1^*\}$ are excited values. The state diagram is called Muller's diagram. An example of Muller's diagram for the circuit:
$x_0 = x_2 + \overline{x_3}$; $x_1 = \overline{x_3}$; $x_2 = x_1\overline{x_3}$; $x_3 = x_0x_1x_2 + x_3(x_0 + x_1 + x_2)$;
in reference to the initial state 0^*0^*00 is given in Fig.4a.[35] Note that, in spite of four-valued coding, the circuit has not 4^n but 2^n possible states from which four states in our example are not attainable from the given initial state. For states a and b of Muller's diagram, the relation of immediate sequence $a\Re b$ and concept of $\Re$-sequence is naturally defined. For every $\Re$-sequence $X(1), X(2), ..., X(m)$ we can assign a sequence of cumulative states (C-states) $\alpha(1), \alpha(2), ..., \alpha(m)$. A cumulative state $\alpha(j)$ is a vector $\alpha(j) = [\alpha_0(j), \alpha_1(j), ..., \alpha_{n-1}(j)]$ of integer components $\alpha(j)$ such that $\alpha_i(1) = 0$ and $\alpha_i(j+1) = \alpha_i(j) + x_i(j) \oplus f_i(X(j))$]. In other words, a cumulative state component $\alpha(j)$ is a number of changes of variable x_i from its initial state $\alpha(1)$ up to $\alpha(j)$. The cumulative state for the example of Fig.4a is given in Fig.4b. In this diagram: $\alpha \leq \beta$ if $\alpha_i \leq \beta_i \ \forall$, $\gamma = \alpha \vee \beta$ if $\gamma_i = \max(\alpha_i, \beta_i)$ and $\gamma = \alpha \wedge \beta$ if $\gamma_i = \min(\alpha_i, \beta_i)$. When there is a partial order $\alpha \leq \beta$, the cumulative diagram is a structure. Muller proved that if this structure is semi-modular, then the behavior of the re-

[27] Note that this order is "chronological" (temporal in logical time).

[28] Independent from the durations of processes of state change in physical time.

[29] Detalization.

[30] Binary variables are usually considered, though there are no special limitations on their valuedness. For example, in [ ] multi-valued self-timed codes are discussed.

[31] Functionally linked.

[32] By the way, here we can make a preliminary remark about the difference between asynchronous and synchronous models. In a synchronous model, logical function of a logical element is defined as $x_i(t+1) = f_i(X(t))$, where t is a special variable, signal of logical time.

[33] A cause of change of the output has occurred.

[34] Full Muller's circuit.

[35] Since, as we mentioned above, simultaneity is not attainable in physical time, only adjacent transitions are shown in the diagram.

spective circuit is speed-independent. Such a circuit is called semi-modular.

Muller's diagram for semi-modular circuits has a simple local property. Let $Q(X)$ be a neighborhood of state X in a Muller's diagram such that $X \Re Y \ \ \forall Y \in Q(X)$ and let $D(X(1))$ be a set of states attainable from $X(1)$.[36] Then, for the semi-modularity of the respective circuit (cumulative diagram) it is necessary and sufficient that

if $x_i = 0^*(1^*) \quad \forall X \in D(X(1))$, then $y_i \neq 0(1) \ \forall Y \in Q(X)$.

In other words, for the semi-modularity of a circuit it is necessary and sufficient that every time when the output of a logical element gets excited (a cause of change of the output occurs), the excitation is removed only by the change of the output (the cause is kept until its effect is achieved). We will call this property "causal conditionality."

Actually, a Muller's diagram contains redundant information. Instead the state change graph, we can use eventual description of the behavior, i.e. partial order on events which are changes of variables. Languages of such a description are signal transition graph (STG), change diagrams (ChD), labeled Petri nets (LPN), etc. [8,9,10]. All these languages have equal descriptive power. In Fig.4c, the STG for the example from Fig.4a is shown. The markers on its arcs indicate the current state of the circuit. For a change diagram[37], a cumulative diagram can be derived similarly to Muller's diagram and a set of algebraic properties is known which provides semi-modularity of corresponding circuits. From a ChD, one can unambiguously restore Muller's diagram. Logical functions of variables are defined by Muller's diagram in the following way: if X is a state of a working cycle in Muller's diagram[38], then if $x_i = 0(1)$, then $f_i(X) = 0(1)$ and if $x_i = 0^*(1^*)$, then $f_i(X) = 1(0)$.[39] The latter provides the task of synthesizing semi-modular circuits specified by ChD or Muller's diagrams. However, the synthesis task is not limited by that due to the following reason:

- When turning from ChD to Muller's diagrams, the latter may contain conflicting states, i.e. states in which different variables are excited though their values are the same. In such a case, extra variables are necessary to remove the conflict. Obviously, incorporating extra variables should not break semi-modularity.
- Real circuits are always synthesized in a certain functional basis. The logical functions obtained as a result of formal synthesis may not be included in the functional basis and should be represented as a superposition of basic functions. Again, the circuit should stay semi-modular in reference to the outputs of all logical elements.

It is interesting to note the connection between the algebraic properties of a cumulative Muller's diagram and logical properties of respective circuits. For example, if a given cumulative diagram is semi-modular and distributive, 2NAND-gate *or* 2NOR-gate is a functionally complete element in the class of semi-modular circuits. For non-distributive circuits[40], the functionally complete set contains elements 2NAND *and* 2NOR.

As we mentioned above, logical time is determined by a partial order on events, reflecting cause-and-effect relationships between them. For a cumulative Muller's diagram, the moment of logical time in every state is found by a simple and natural method: $T(\alpha) = \sum_{j=0}^{n-1} \alpha_j$. For a ChD and its cumulative unfolding, moments of logical time are not found so easily. Events of a ChD determine the change of states and, hence, it is natural to link logical time moments with arcs (markers). A combination of markers uniquely determines a current state and if we compare Fig.4a, b and c, we will easily see that the event $+X_0$ (excited state 0^*) is present in two moments of logical time. If the marker on arc $+X_1 \rightarrow +X_2$ has a value of logical time t, then the marker on arc $+X_0 \rightarrow +X_3$ is "spread" over the moments of time $t \rightarrow t+1$. Thus, logical time is treated in a different way for states and for events. Logical time for a state is unique for all the circuit because in every moment of physical time the circuit is in the same state.[41] Logical times for different events can be different in the same moment of physical time and the same logical time for different events can correspond to different physical time moments. Interpretation of eventual logical time depends on the agreement we will accept for this interpretation.[42] For example, if events $E_1(t_1)\&...\&E_k(t_k)$ are the cause of event $E_m(t_m)$, we can conclude that $t_m = \max(t_1, ..., t_k) + 1$.[43] In the case 3 of our example, when x_3 is taken, all the mark-

[36] Set $D(X(1))$ is sometimes called "working cycle."

[37] STG are a subset of ChD.

[38] Working cycle is a set of states attainable from the initial one.

[39] Generally, this definition is ambiguous because a function can be specified arbitrarily on states not included in the working cycle.

[40] Non-distributivity is displayed when the next excitation of a variable occurs in more than one state.

[41] This statement is not very accurate because when the state is changed, the variable passes through a continuous set of values and the border between 0 and 1 is not distinct. Analysis of the firing mode is beyond this article, but what we have said can be used with the accuracy necessary for our aims.

[42] Actually, the interpretation largely depends on the semantics of the described processes.

[43] Remember that moments of logical time mark the output arcs of the corresponding events and the violation of immediate

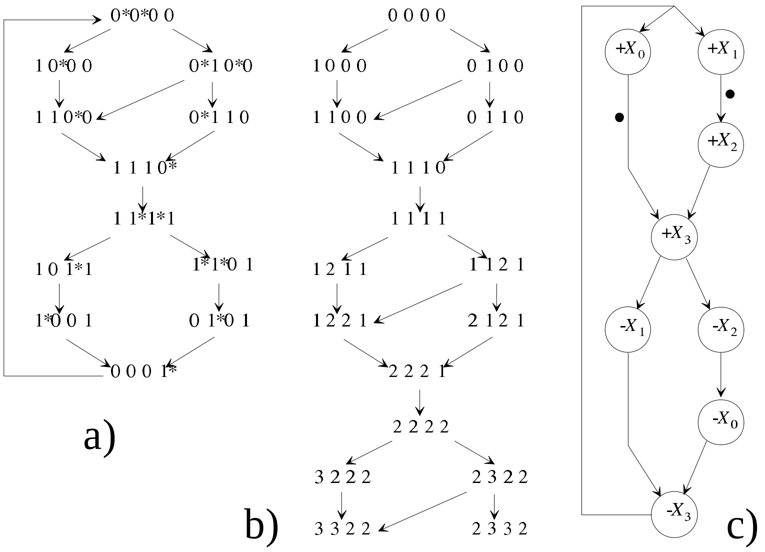

Figure 4: Muller's circuit specifications.

ers have the same value of logical time for any state of arcs markings. If x_1 or x_2 is selected as the variable forming the values of logical time, the situation becomes more complicated.

To this point we considered models of autonomous circuits and their behavior. However, in real situations any device should interact with the environment. Let us consider the simplest possibility of organizing such interaction.

Fig.5a contains the circuit specified by Muller's diagram (ChD) in Fig.4. According to the definition of Muller's circuit, variables x_i are referred to the outputs of gates and the values of each variable at the inputs of other gates are identical in any moment of physical time. This leads to Muller's assumption of delays: *all delays are referred to outputs of gates, and the dispersion of delays in wires after their branching is neglected.*[44] Speed-independence means that the circuit behavior (in terms of keeping partial order on events) will not change when an arbitrary delay is put in sequence with the output of any gate. The environment also can act like such a delay (Fig.5b). We can introduce an extra designation for the output signal of the environment e_i. If the environment is put in sequence with the output of a gate x_i, the interface between the circuit and environment is simple and evident: $e_i = x_i, \;\; x_i = \overline{e_i}$. From the environment standpoint, the circuit is modeled by an inverter with an arbitrary delay at its output. Signal e_i can be considered as the initiating (synchronizing) input of the circuit and signal x_i — as the signal informing about the completion of transient processes in the circuit. Actually, the latter is not absolutely true. If, say, x_2 is selected as the interface signal, from Fig.4c it follows that, when $x_2 = 1$, the firing of x_0 may be incomplete. The same is true for x_1 when $x_2 = 0$. The completion of these firings is checked in the next functioning cycle of the composition circuit/environment. This provides a number of new possibilities because some of the internal variables switch concurrently with the environment.[45]

Since the model of an open semi-modular circuit is an inverter with a series delay, the model of an open semi-modular circuit with a series inverter is a delay. This idea makes it possible to compose semi-modular circuits and perform their block synthesis when open semi-modular blocks with their series inverters are placed like delays in sequence with the semi-modular circuit coordinating their behavior.

A synchronous circuit with a common clock also can

sequence of time moments on the input and output arcs of a given event is not something esoteric.

[44] Muller's assumption of delays is not always true, depending on the difference in wire length after branching and on layout. The problems of circuits insensitive to wire delays are beyond the scope of this article. Muller's assumption is quite satisfactory for the further discussion.

[45] Using these possibilities is closely connected with the semantics of behavior.

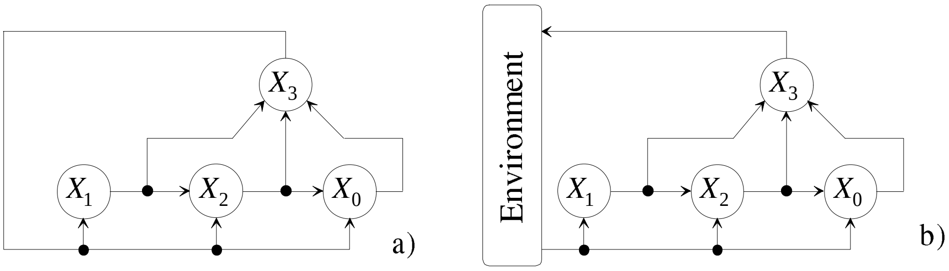

Figure 5: Muller's circuit for Fig.4 specification.

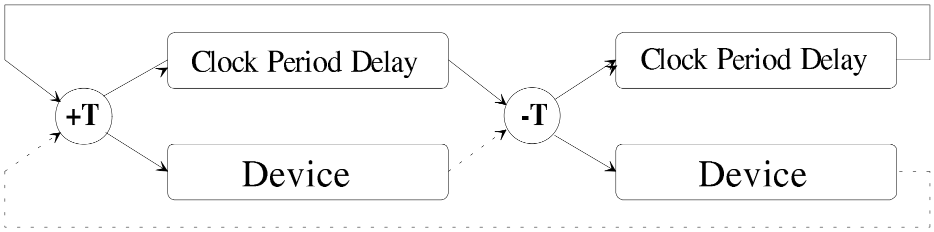

Figure 6: Signal graph for external clock timing.

be conventionally specified by a ChD (Fig.6a). Doing so, first, a calibrated delay act as a clock model and, second, if the clock period delay is a priori bigger than any possible duration of transient process in the device[46], we can neglect the links from events in the device to events $\pm T$.

3 Distributed timing strategy.

We should realize that a system of synchronization from an external clock includes not only the clock itself but also the system of delivering the clock signals to the points of their interaction with the device. It is not an exaggeration to say that as soon as we introduce not only the clock but also the clock signals delivery system[47], we pass on to the problem of space/time. For artificial systems, this problem has its specifics.[48] In VLSI and VLSI-systems, we actually deal with a wire metrics specified on the intermodular connection graph. Two modules A and B are adjacent in reference to signal x_i if a wire exists that transfers x_i from A to B.[49] Note that if signals x and y propagate from A to B via different wires, $D(AB_x \neq D(AB_y)$.

The delay of signal distribution through a wire is determined by two factors. First, by distributed RC parameters. The time constant $\tau_1 = RCl^2/2 \approx 100p\text{sec/mm}$. Second, it is determined by the limitations on the density of current in the wire. For aluminum, the density that causes migrations of atoms is equal to 10^5A/cm^2=1mA/μ^2 [50] and time of charging the wire capacitor by direct current is $\tau_2 = VCl/I \approx 100p\text{sec/mm}\cdot\text{mA}$.[51] If current density is the dominating factor, i.e. if $\frac{\tau_1}{\tau_2} = \frac{R \cdot I \cdot l}{2 \cdot V} << 1$, the wire surface can be considered to be an equipotential surface of the capacitor. In this case we have an equipotential or equichronic zone. Actually, the delays are considerably bigger since the wire is loaded on the inputs of transistors the input capacities of which per a unit of area are by 50–100 times more than that of the wire itself. The estimation of equichronic zone dimensions is complicated and depends on accepted hypotheses and criteria. But general estimation gives them the order of 1mm. From the concept of equichronic zone, regardless of its dimensions, two important conclusions follow:

[46] This is an indispensable condition of correct behavior when the device is synchronized from a common clock.

[47] And informational signals, too.

[48] In particular, for VLSI and VLSI-systems we will consider hereafter.

[49] It is by no means an esoteric way of measuring distances. For example, road maps released in New Zealand use hours and minutes of driving to specify distances between two points.

[50] This is the limitation for direct current. For alternating currents, the permissible value of density can be more by several times.

[51] When the width of the wire increases, its capacitance and permissible current increase equally.

- the totality of equichronic zones produces the totality of local times;
- the system of time signals delivery should provide mutual coordination of local times.

Using common clock to synchronize VLSI behavior imposes certain demands on the system of time signals delivery. Usually, a wire delivery system is an H-tree (Fig.7) in which the points of entering time signals into the equichronic zone are equally distant from the common source of these signals. H-tree balancing for contemporary synchronization speeds (200–400MHz) and contemporary VLSI sizes is not a simple task; every new step on this way comes harder and harder. The major problems for all H-tree modifications are the following: the dispersion of physical and technological parameters of the H-tree that leads to delays dispersion and large power necessary to recharge the H-tree wires.[52] These problems limit the number of terminal points in the H-tree and the degree of system fragmentation to modules which are treated as equichronic.

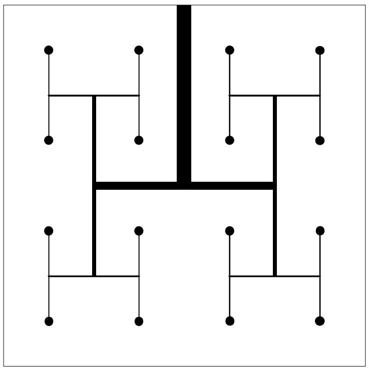

Figure 7: H-tree.

Let us go back to considering the functioning of clock as such. Any clock we know, from water and solar ones to the most precise atomic clock, are based on a certain periodic process flowing in physical time. A clock used in VLSI is a generator (clock generator) with a calibrated delay in the feedback circuit (Fig.8a).[53] What demands are imposed on the period? Only one — the period duration must be sufficient for the completion of transient processes; within this limitation the precision of period maintenance is not very important. Moreover, the period can even be elastic if we can control the delay depending on the mode of VLSI functioning (data-dependent delay) or fix the moments of transient processes completion in this or that way (Fig.8b). However, as we mentioned above, the stumbling block is the system of time signals distribution and delivery. It is evident that we should provide coordination only between local time signals for modules adjacent on the connection graph. This leads us to the idea of decentralized timing (Fig.8c). In other words, we will try to demonstrate that a distributed asynchronous clock can be created in which the passive wire system of synchrosignals delivery is replaced by an active environment. We will call such a device "Synchro-Stratum."

4 Synchro-Stratum

As a model for the next step of our discussion, we will consider a one-dimensional cellular array of N *synchronous* Moore automata A_j (modules, blocks) in which every automaton A_j has information exchange only with its two immediate neighbors (A_{j-1}, A_{j+1}). If $S_j(t)$ is a state of A_j in the moment of logical time t, then

$$S_j(t+1) = F_j[S_{j-1}(t), S_j(t), S_{j+1}(t)]. \qquad (4.1)$$

Note[54] that if the behavior of the array is specified by a system of logical equations (4.1), a proper transformation of the space of states and functions can provide arrays with equivalent behavior:

$$\begin{aligned} S_j(t+1) &= F_j[S_{j-1}(t+1), S_j(t), S_{j+1}(t)];\\ S_j(t+1) &= F_j[S_{j-1}(t), S_j(t), S_{j+1}(t+1)];\\ S_j(t+1) &= F_j[S_{j-1}(t+1), S_j(t), S_{j+1}(t+1)]. \end{aligned} \qquad (4.2)$$

In equations (4.1) and (4.2), t is an integer variable representing the global (common for all the array) logical time. Let us introduce the concept of local logical time T_j which represents the value of global logical time at the input of A_j. Doing so, the equation (4.1) looks like this:

$$\begin{aligned} S_j(T_j+1) = F_j[&S_{j-1}(T_{j-1}=T_j), S_j(T_j),\\ &S_{j+1}(T_{j+1}=T_j)]. \end{aligned} \qquad (4.3)$$

It obviously follows from (4.3) that to provide the correctness of array temporal behavior it is enough to coordinate the values of local logical times only for the immediate neighbors. In Fig.9a, the ChD for global synchronization of the array from a common synchrosignal t is given; Fig.9b contains the ChD for equivalent coordination of local times. The ChD in

[52] For example, with 32 equichronic zones and frequency 400MHz, we need about 2.5W only to recharge the capacitance of H-tree wires.

[53] For now, we will consider an isolated (autonomous) device. Interaction with the environment is a special question.

[54] This will be helpful for the further discussion.

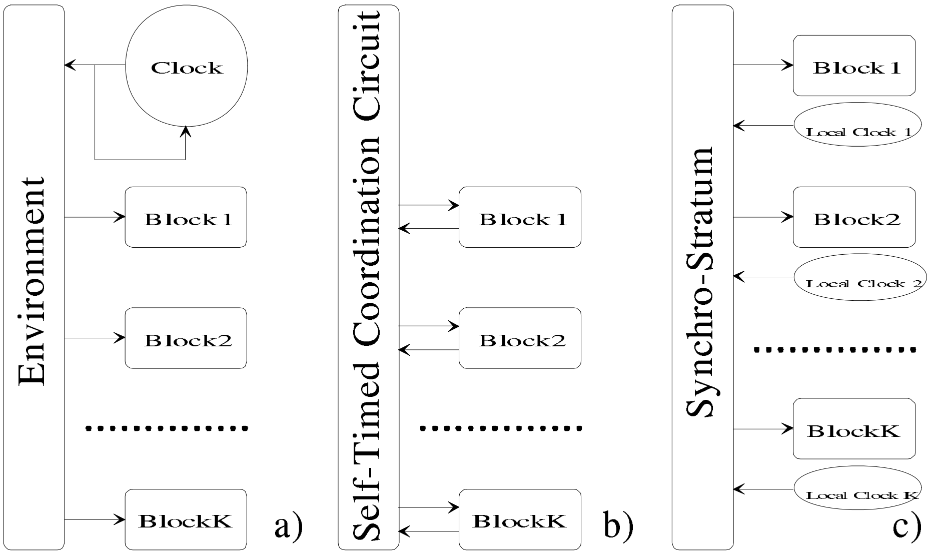

Figure 8: Types of timing organization.

Fig.9b is correct but no one circuit corresponds to it because the respective Muller's diagram contains conflicting states. To resolve the conflicts, extra variables should be introduced into the ChD, like in Fig.10. This specification produces the following circuit:

$$\begin{aligned} T_i &= \overline{c_{i-1}c_ic_{i+1} + b_i(c_{i-1} + c_i + c_{i+1})}; \\ c_i &= \overline{b_{i-1}b_ib_{i+1} + T_i(b_{i-1} + b_i + b_{i+1})}; \\ b_i &= a_{i-1}a_ib_{i+1} + c_i(a_{i-1} + a_i + a_{i+1}). \end{aligned} \tag{4.4}$$

Actually, this already proves that a Synchro-Stratum can be implemented. However, there is a number of arguments for more detailed analysis of Synchro-Stratum implementation.

- First, the circuit (4.4) is fairly complicated; for each synchronization point it contains 42 transistors in CMOS implementation[55] and, more important, it requires 6 input wires from the adjacent synchronization points and the same number of output wires to them.[56]

- In practice, various synchronization systems are used producing several synchro-signals (synchro-sequences) and every synchronous prototype leads to its own Synchro-Stratum structure.

The main idea of synchronization by signals from a clock is associated, in one way or another, with organizing master-slave behavior of circuit components. As a model for synchronization strategy consideration, we use cellular arrays in which cells are finite Moore automata built as two-register master-slave circuits (Fig.11). At one value of the clock signal T, the automaton changes its current state writing the new state from output of logic to the master register (working phase). At the other value of T, current state does not change and is written to the slave register (passive

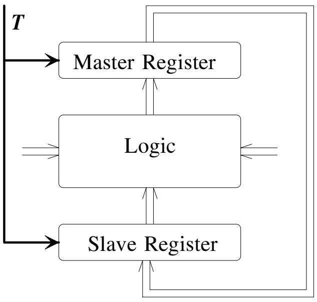

Figure 11: Master-slave automaton structure.

[55] An extra delay added by the Synchro-Stratum is equal to 6τ per a full synchronization cycle, where τ is the delay of one gate.

[56] As the number of neighbors grows, the number of wires and transistors grows proportionally.

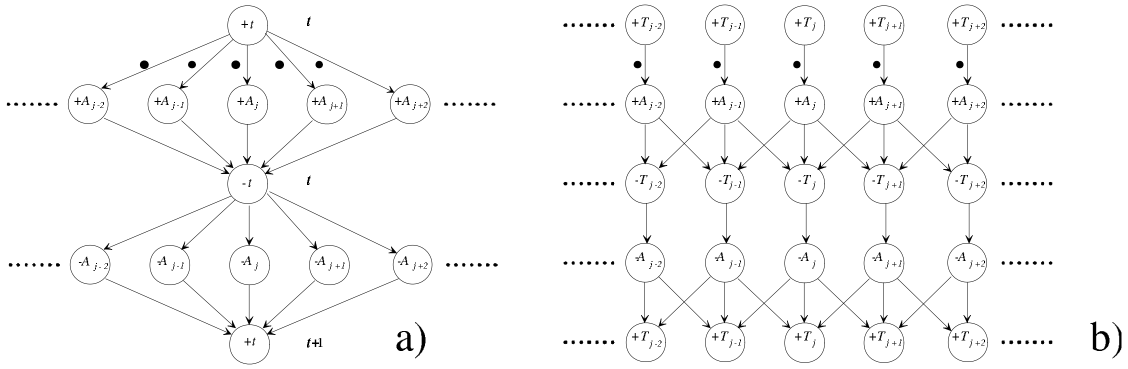

Figure 9: Change diagram for Synchro-Stratum.

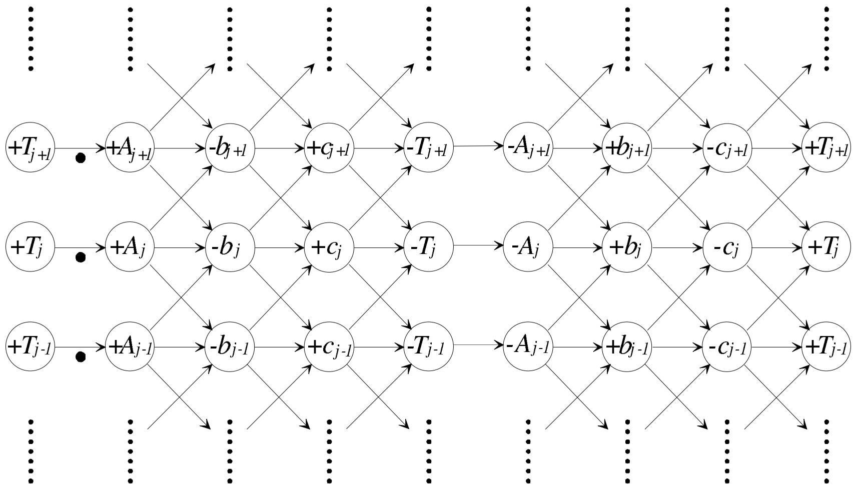

Figure 10: Change diagram for Fig.9 Synchro-Stratum implementation.

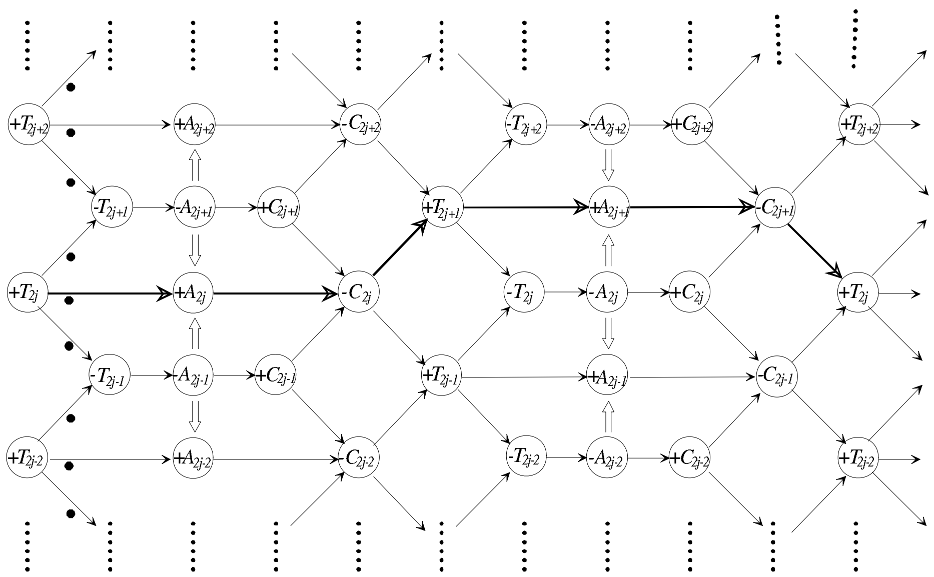

Figure 12: Change diagram for fast Synchro-Stratum.

phase). This is a two-phase functioning with different-polar control. In the working phase, the slave register state does not change keeping the preceding state of the automaton, while in the passive state the master register state does not change keeping the current state. It is essential for temporal coordination of behavior that the outputs of adjacent automata (inputs of a given automaton) do not change in the working phase. In [11–15], several implementations of Synchro-Stratum are discussed. Here, in addition to (4.4) we will discuss just one new solution. Let us consider the ChD fragment in Fig.12.

From this ChD we have the following circuit:

$$C_i = \overline{C_{i-1}A_iC_{i+1}};$$
$$T_i = \overline{T_{i-1}T_{i+1}(C_{i-1} + C_{i+1})}. \qquad (4.5)$$

It follows from (4.5) that the complexity of this Synchro-Stratum is 14 transistors per synchronization point in CMOS-implementation, 4 input and 4 output wires for the interface between the cells of the Synchro-Stratum. By this parameters, it is inferior to the best one we know [15, Fig.5][57] However, the circuit (4.5) has a remarkable property which is the extreme performance. Indeed, if we consider master-slave structure of automaton, we will notice that the durations of transient processes are different in different phases. In the working phase, the transient process duration consists of the logical circuit delay and delay of writing into the register. In the passive phase, the transient process contains only writing into the register. Let us presume that in the ChD (Fig.12) the working cycle in the automata is accomplished when $T_j = 1$[58] and that the delay in the logical circuit is $\tau_{\log ic} \geq 2\tau_{gate}$. Then the full cycle of synchronization is determined by the way shown in Fig.12 by bold arrows. It is easy to see that the extra delay added by the Synchro-Stratum is equal to $4\tau_{gate}$ per a full cycle of T_j change.[59]

We will consider the full cycle of local time signal change $\rightarrow -T_j \rightarrow +T_j$ as one step of the local logical time. Let us introduce designations 1(k) and 0(k) for values $T_j(k) = (1,0)$ and let $T_j(1) = 1 \ \forall j$ in the initial moment of time. Then the cumulative unfolding of the ChD in Fig.12 has a projection on T_j as shown in Fig.13a. The equations are different for even and odd automata:

$$S_{2j}(t+1) = F_{2j}[S_{2j-1}(t), S_{2j}(t), S_{2j+1}(t)];$$
$$S_{2j+1}(t+1) = F_{2j+1}[S_{2j}(t+1), S_{2j+1}(t), S_{2j+2}(t)] \qquad (4.6)$$

Analyzing the diagram in Fig.13a allows us to get interesting conclusions about temporal behavior of the system. If we fix A_1 in the state 1(1), then the

[57] 10 transistors, one input and one output wire for Inter-Stratum interface.

[58] The signs ⇑ and ⇓ in the ChD indicate time and direction of information transfer from the master registers of A_{j-1} and A_{j+1} to the logical circuit of A_j.

[59] Note that for a minimum circuit [15,Fig.5], the extra delay added by the Synchro-Stratum is equal to $10\tau_{gate}$ and delay of the circuit (4.5) is obviously minimum.

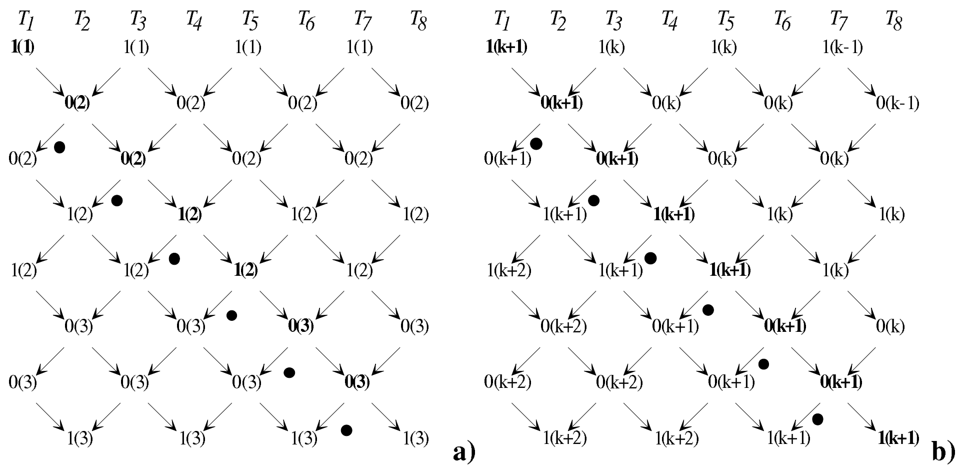

Figure 13: Local logical time distribution.

system[60] will come to a stable state with logical times distributed as 1,2,2,2,2,3,3,3,3,4... It follows from this that both in the stable state and during functioning[61] different logical times can exist in different points of the Synchro-Stratum. Furthermore, if we can control the work rhythm of some automaton or the respective cell of the Synchro-Stratum, this point of the array will be a rhythm driver for all the array and "time waves" will propagate from it over the array. We can try to organize the behavior so that the value of logical time is the same for all the automata in the stable state and on the edge of a "time wave." Such a possibility is provided by the diagram in Fig.13b. In this diagram, a step of logical time is formed for odd automata by cycle $\rightarrow +T_j \rightarrow -T_j$ and for odd automata by cycle $\rightarrow -T_j \rightarrow +T_j$. As one can see from Fig.13b, the automata equations have period 4:

$$\begin{aligned}
S_{4j}(t+1) &= F_{4j}[S_{4j-1}(t), S_{4j}(t), S_{4j+1}(t)]; \qquad (4.7)\\
S_{4j+1}(t+1) &= F_{4j+1}[S_{4j}(t), S_{4j+1}(t), S_{4j+2}(t)];\\
S_{4j+2}(t+1) &= F_{4j+2}[S_{4j+1}(t), S_{4j+2}(t), S_{4j+3}(t)];\\
S_{4j+3}(t+1) &= F_{4j+3}[S_{4j+2}(t), S_{4j+3}(t), S_{4j+4}(t)].
\end{aligned}$$

Thus, changing the form of the automata equations, we can pass from parallel synchronization to wave synchronization.

The transition from one-dimensional array to two-dimensional array implies only that there will be more neighbors and the equations will look like this:

$$\begin{aligned}
C_{i,j} &= \overline{C_{i-1,j}C_{i,j-1}A_{i,j}C_{i+1,j}C_{i,j+1}};\\
T_{i,j} &= \overline{T_{i-1,j}T_{i,j-1}T_{i+1,j}T_{i,j+1}\times}\\
&\quad \overline{(C_{i-1,j}+C_{i,j-1}+C_{i+1,j}+C_{i,j+1})}. \qquad (4.8)
\end{aligned}$$

A Synchro-Stratum can also be synthesized for an arbitrary connection graph. Logical functions for Synchro-Stratum elements are naturally generalized for the case of an arbitrary graph. Let V_j be the set of A_j neighborhood indices on the connection graph, i.e. $V_j = \{i\}$ $\forall i$ such that A_i is informationally connected with A_j. Then the functions of Synchro-Stratum elements are:

$$C_j = \overline{A_j \prod_{i\in V_j} C_i}; \quad T_j = \overline{\prod_{i\in V_j} T_i \bigcup_{i\in V_j} C_j} \qquad (4.9)$$

To organize interaction like that in Fig.12 (4.9), the necessary and sufficient condition of correct behavior of the Synchro-Stratum is that the connection graph is bichromatic [12]. However, for a Synchro-Stratum like that in Fig.10 (4.4), there is no such a limitation and the functions of its elements are[62]:

$$T_j = \overline{\prod_{i\in V_j, i=j} c_i + b_j \bigcup_{i\in V_j, i=j} c_j};$$

$$c_j = \overline{\prod_{i\in V_j, i=j} b_i + T_j \bigcup_{i\in V_j, i=j} c_j};$$

$$b_j = \overline{\prod_{i\in V_j, i=j} a_i + c_j \bigcup_{i\in V_j, i=j} a_j}. \qquad (4.10)$$

[60] As you can see from the markers distribution in the figure.

[61] When different automata have different durations of transient processes.

[62] One more problem here, as well as in the case (4.9), is mapping, i.e. representing a function as a superposition of the basic elements. This is beyond the scope of this article. We will just note that for (4.9) this task is principally simpler.

5 Conclusion

Now we can pose some questions. The first one is: "What kind of advantages are provided by Synchro-Stratum?" We know that a decomposition as such cannot provide any functional possibilities. Indeed, we never can treat the composition of an automaton A_j with a set of internal states $\{S_{ij}\}$ and a Synchro-Stratum cell T_j with a set of internal states $\{X_{ij}\}$ as one automaton W_j with the set of internal states $\{Y_{ij}\} = \{S_{ij}\} \times \{X_{ij}\}$ and the respective function of state transitions. Moreover, using sophisticated procedures of state assignment and automata minimization we can disguise the Synchro-Stratum in the initial representation. However, we will have to solve this problem from the start for every new cellular array. The sense of decomposing a system to Automata Stratum and Synchro-Stratum is separating the tasks of organizing functional and temporal behavior.[63] It is this separation that in fact allowed us to prove[64] that for any synchronous prototype of a cellular array we can build its asynchronous version in a uniform way. This is the engineering aspect of the problem. Conceptually, Synchro-Stratum is the carrier of logical time, creating the field of local logical times in the system. The concept of Synchro-Stratum offers strong possibilities for reasoning and speculations on the topic of artificial system time, which are not completely used yet.

The second question is no less important: "Is the presence of a synchronous prototype and Synchro-Stratum sufficient for building an asynchronous version?" The answer is a very definite no. Indeed, asynchronous behavior must be causally conditioned. It means that synchronized blocks (automata) must have an interface with the Synchro-Stratum with direct or mediated (indirect) handshake (see Fig.8). In this handshake, a signal of local logical time can and should be considered as a request signal. In response to it, the block (automaton) should produce the acknowledgment signal allowing the Synchro-Stratum to form the next step of local time. Additions and corrections that should be made to provide using the synchronous prototype block in interaction with the Synchro-Stratum are the matter of a particular design task. The designer can use a wide range of possible solutions, from self-timing to parallel incorporated delay. The important thing is that the task can be solved in a decentralized way, using independent and, possibly, different methods for different blocks. Such a freedom in organizing local temporal interface supplements the set of the known synchronization systems:

- **FS** — fully synchronous (common clock)
- **FA** — fully asynchronous (self-timing, etc.)
- **LAGS** — local asynchronous / global synchronous
- **GALS** — global asynchronous / local synchronous

by one more, which is **GALA** — global asynchronous / local arbitrary (system with Synchro-Stratum).

And, finally, the last question: "Can any task which is solved by a synchronous cellular array, be solved by a respective array with Synchro-Stratum?"

Answering to this question is associated with treating and understanding the concept of logical time. One of the difficult problems for asynchronous cellular arrays is the task of interaction between the edges of waves propagating in opposite directions when implementing wave propagation algorithms. Though this task is solvable[65], trying to solve it in the structure of asynchronous arrays[66] often leads to conflict[67] situations. Introducing Synchro-Stratum removes the problem of asynchronous design. If the interaction algorithm can be implemented in the synchronous prototype, it automatically can be implemented in its asynchronous version. In the synchronous version however, simultaneity in physical time is treated as simultaneity in logical time[68], and therefore the tasks associated in this way or another with synchronization in physical time cannot be transferred from synchronous arrays to asynchronous arrays in principal. A typical example is Mychill's fired squad synchronization problem [16,17]. For a synchronous cellular automata array, this task is formulated as follows:

> We have a one-dimensional Moore automata array with bi-directional interaction. Each of N automata has n internal states (n does not depend on N). In the initial moment of time $T = 0$, all the automata are in the passive state S_0. In the moment $T = 1$, the external initializing signal arrives at the extreme automaton of the array. In the moment $T = 2N$, all the automata must simultaneously turn to the final state S_f.[69]

[63] This separation is one of the attractive features of the synchronous model.

[64] Though the article does not contain this statement explicitly.

[65] This follows from all the contents of this article. See also [15].

[66] For example, in the case of counter-flow architecture [18].

[67] Arbitration.

[68] With an precision of one synchro-cycle duration.

[69] All the squad must simultaneously shoot.

All the known solutions of this tasks are based on consecutive dividing array cuts in two by comparing the speeds of signals propagation. Since these speeds (delays of the signals in the automata) are measured in logical time units, going from the synchronous prototype to asynchronous array is simple and evident. However, the moment of synchronization (when every automaton turns to the final state) occurs when its local time is $2N$. As we mentioned above, in an asynchronous array the same values of local logical time can exist in different moments of physical time. So, the task loses its original sense.

The given task loses its original sense for an external observer which is a part of the external physical world with the unified physical time. But if the external observer is an artificial system with its logical organization, one can find sensible interpretations of asynchronous (in physical time) solution of this task. In any case, firing squad problem demonstrates the difficulties that can occur when organizing real-time interface between an artificial system and the external physical world. But this is a subject for another article.

Literature

1. Reichenbach H., *The Direction of Time*, Ed. M. Reichenbach, University of California Press, Berkeley and Los Angelos, 1958.

2. Reichenbach H., *The philosophy of Space & Time*, New York, 1958.

3. von Wrigt G.H., *Explanation and Understanding*, London, 1971.

4. Yang Z., Marsland T., *Global States and Time in Distributed Systems*, IEEE Computer Societe Press, Los Alamos, Ca., 1994.

5. Muller D.E., *A theory of asynchronous circuits*, Report of University of Illinois, No66, 1955.

6. Muller D.E. and Bartky W.C., *A theory of asynchronous circuits, I, II.*, Report of University of Illinois, No75, 1956, No78, 1957.

7. Miller R.E., *Switching Theory, vol. II, Sequential circuits and machines.*, Jonh Wiley & Sons Inc., 1966.

8. Varshavsky V., *Hardware Support of Parallel Asynchronous Processes*, Helsinki University of Technology, Digital Systems Laboratory, Series A: Research Reports, No2, Sept. 1987.

9. *Self-Timed Control of Concurrent Processes*, Ed. By V.Varshavsky, Kluwer Academic Publishers, 1990.

10. Kishinevsky M., Kondratyev A., Taubin A., Varshavsky V., *Concurrent Hardware*, John Wiley & Sons, 1994.

11. Varshavsky V., Chu T.-A., *Self-Timing — Tools for Hardware Support of Parallel, Concurrent and Event-Driven Process Control*, Proceedings of the Conference on Massively Parallel Computing Systems (MPCS'94), May 1994, pp. 510–515.

12. Varshavsky V., Marakhovsky V., Chu T.-A., *Logical Timing (Global Synchronization of Asynchronous Arrays)*, Parallel Algorithm/Architecture Synthesis International Symposium, Aizu-Wakamatsu, Japan, IEEE CS Press, March 1995, pp. 130–138.

13. Varshavsky V., *Asynchronous Interaction in Massively Parallel Computing Systems*, IEEE First ICA3PP, Proceedings of IEEE First International Conference on Algorithms and Architectures for Parallel Processing, vol.2, Brisbane, Australia, pp. 951–953.

14. Varshavsky V., Marakhovsky V., Chu T.-A., *Asynchronous Timing of Arrays with Synchronous Prototype*, Proceedings of the Second International Conference on Massively Parallel Computing Systems (MPCS'96), May 1996, pp. 47–54.

15. Varshavsky V., Marakhovsky V., *Global Synchronization of Asynchronous Arrays in Logical Time*, Proceedings of the Second Aizu International Symposium on Parallel Algorithm/Architecture Synthesis, Aizu-Wakamatsu, Japan, IEEE CS Press, March 1997, pp. 207–215.

16. Goto E., *A minimum time solution of the firing squad problem, Dittoed course notes for applied mathematics 298*, Harvard University, May 1962.

17. Varshavsky V., Marakhovsky V., Peschansky V., *Synchronization of Interacting Automata*, Math. System Theory, vol.4, No3, 1970.

18. Sproull R.F., Sutherland I.E., Molnar C.E., *Counterflow Pipeline Processor Architecture*, Technical Report SMLI TR-94-25, SUN Microsystems Laboratories Inc. CA 94043, April 1994.

19. Varshavsky V., *Logic Design and Quantum Challenge*, Proceedings of International Workshop on Physics and Computer Modeling of Devices Based on Low-Dimensional Structures. Aizu-Wakamatsu, Japan, IEEE CS Press, November 1995.