REIHE INFORMATIK
32/95

Improved 3-Line Hardware Synchronization

M. Makhaniok, R. Männer
Universität Mannheim
Seminargebäude A5
D-68131 Mannheim

IMPROVED 3-LINE HARDWARE SYNCHRONIZATION¶

M. Makhaniok1), R. Männer2,3)

1) Institute of Engineering Cybernetics, Academy of Sciences, Surganov St. 6, 220012 Minsk, Republic Belarus
2) Lehrstuhl für Informatik V, Universität Mannheim, A5, D-68131 Mannheim, Germany
3) Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg, Heidelberg, Germany

• **Abstract.** A new procedure is proposed to synchronize processors of a distributed system, which concurrently execute a common process consisting of a sequence of operations. The procedure is an extension of that used for the 1987 IEEE Futurebus Standard. It is based on global synchronization lines and a distributed synchronizer, and requires only minor modifications of existing hardware. The procedure allows to carry out two alternative synchronization protocols. As usual, an operation may be terminated by the last processor having finished its part of the operation. Alternatively, the operation may also be terminated by the first processor being ready. Application of this second procedure, e.g., to bus arbitration, allows to reduce the arbitration time in average by a factor of 2.

1. Introduction. We consider a <u>process</u>[1] that consists of a sequence of <u>operations</u>, e.g., search, comparison or sorting, that are executed distributively by multiple processors. The processors execute their local parts of every operation. Then they are synchronized so that every operation is finished by all processors before the next one is started.

A distributed system (DS) consists of a number of processors that operate in the general case asynchronously at different speeds and with different input data. Generally a varying number of processors participate in an operation, and the duration of an operation depends on the set of participating processors and their characteristics. Assuming a high number of processors and a short duration of the operations, it is not possible to exchange these characteristics before the start of an operation. Therefore a <u>distributed hardware synchronization</u> between the processors is required so that each one can determine when the current operation will be finished; using only its own characteristics, and characteristics specified for the whole system. The proper type of synchronization is the topic of this paper.

In many cases synchronization in a DS is done using an asynchronous (hand-shake) protocol. Such a synchronization does not need a common clock and has a number of well known advantages. One of them is that the end of an operation can be determined dynamically [1] which can speed up the operation on average. This may considerably reduce unnecessary waiting time and processing overhead. The synchronization procedure introduced in this paper exploits this fact.

¶ This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) under grants Ma 1150/8-1, Ma 1150/8-2, and 436-WER-113-1-0 (438 113/117/0).

1 Here and below we underline terms that are newly defined.

2. Taub's synchronization procedure. Synchronization of processors can be done according to the following procedure that was originally designed by Taub [2,3] for the IEEE "Futurebus" backplane bus standard. In this paper we simply call it "Taub's synchronization procedure". Each processor of a DS can be considered to consist of two parts, a processing element (PE) and a synchronization unit (SU), where the lines $\mathbf{L_i}$ are used for data exchange between PEs (Fig. 1).

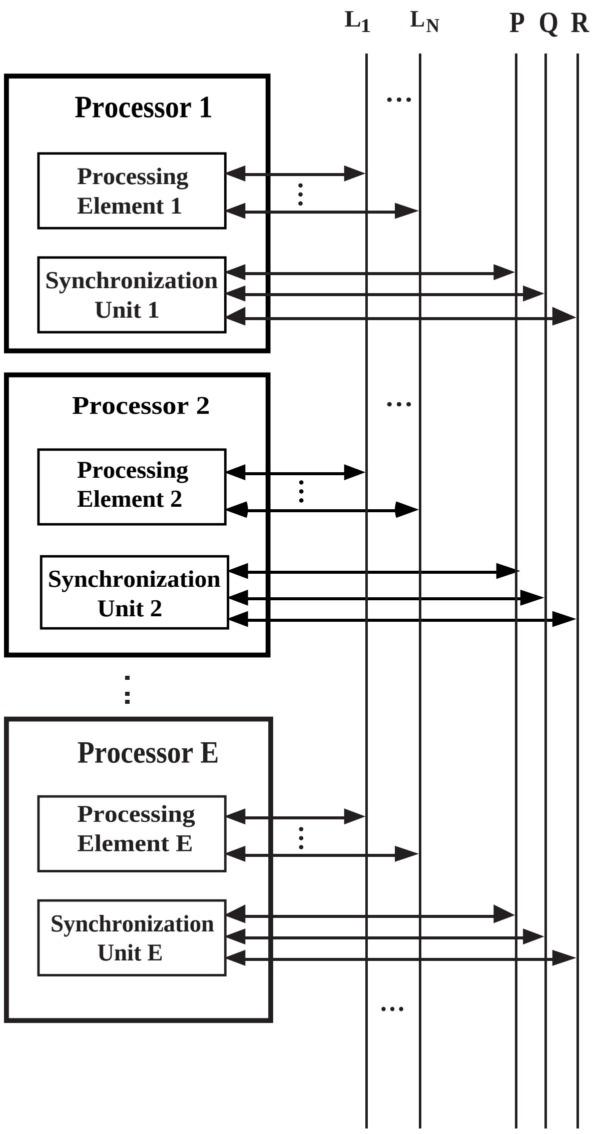


Fig. 1. A multiple processor system with distributed synchronization.

The general principle of Taub's synchronization procedure is that ***any** processor can start a new operation, and the operation is finished only when **all** participating processors completed their individual parts*.

Let **Z** be a set of processors in the system. It takes an individual waiting time **t(j,W)** for the **j**-th processor to complete its part of an operation **W**. Therefore the operation will be finished within the time $\mathbf{Max\{t(j,W),\ j\in Z\}}$. We call this value the global waiting time.

Taub's procedure is based on a synchronizer that uses three bus lines **P**, **Q**, **R**. Each SU is connected to these lines and can assert onto them its individual binary signals $\mathbf{p_j}$, $\mathbf{q_j}$, $\mathbf{r_j}$

through open-collector drivers. Thus every line computes the wired-OR of the signals asserted by the SUs, and all SUs read back this value. If we assume, e.g., that a line is set to a logic **0**, then it will be switched to the logic **1** state if **any** processor asserts a **1** signal. In contrast, the line will change its state from **1** to **0** only if **all** processors withdraw their **1** signals.

Suppose that the initial signals $\mathbf{p_j}$, $\mathbf{q_j}$, $\mathbf{r_j}$, $\mathbf{j \in Z}$, generated by all SUs, have the values of **0**, **0**, **1** respectively, and therefore also **P**, **Q**, **R**. Then the sequence of events in the synchronization procedure according to [2] is described by Tab. 1.

Tab. 1. Synchronization according to Taub [2,4].

#	**Protocol steps**	**Current stage of signals**				**Next stage**
		signal	lines	outputs	trigger outputs	trigger outputs
		run	P Q R	Op_1 Op_2 Op_3	s_p r_p s_q r_q s_r r_r	p q r
1	Initial condition for the 1st operation	0	0 0 1	0 0 0	0 x 0 x 1 0	0 0 1
2a	Start of the 1st operation by any PE	1	0 0 1	0 0 0	1 0 0 x 1 0	1 0 1
2b	Start of the 1st operation by other PEs	0	1 0 1	0 0 0	1 0 0 x 1 0	1 0 1
3	Execution of the 1st operation (stable state)	x	1 0 1	1 0 0	1 0 0 x 1 0	1 0 1
4	End of the 1st operation by any PE	x	1 0 1	1 0 0	1 0 0 x 1 0	1 0 0
5	End of the 1st operation by all PEs	x	1 0 0	0 0 0	1 0 1 0 0 x	1 1 0
6	Execution of the 2nd operation (stable state)	x	1 1 0	0 1 0	0 0 1 0 0 x	1 1 0
7	End of the 2nd operation by any PE	x	1 1 0	0 1 0	0 1 1 0 0 x	0 1 0
8	End of the 2nd operation by all PEs	x	0 1 0	0 0 0	0 x 1 0 1 0	0 1 1
9	Execution of the 3rd operation (stable state)	x	0 1 1	0 0 1	0 x 0 0 1 0	0 1 1
10	End of the 3rd operation by any PE	x	0 1 1	0 0 1	0 x 0 1 1 0	0 0 1
11	End of the 3rd operation by all PEs[2]	x	0 0 1	0 0 0	1 0 0 x 1 0	1 0 1

One example for the corresponding signal waveforms is shown in Fig. 2. Here three participating PEs are assumed with numbers A, B and N.

3. Alternative Synchronization Procedure. To reduce the global waiting time in the DS we introduce an alternative synchronization procedure. Unlike Taub's procedure it is based on the principle that ***<u>any</u> processor can start a new operation, and <u>any</u> processor can terminate it***. The latter is possible if a processor recognizes either that all output signals are settled or that all further changes of the signals in the DS cannot longer influence the final result of the operation[3].

[2] If **run=1** the PEs will execute the sequence of the next three operations starting from step 2a. If, however, **run=0** PEs will be waiting for initiation of the next sequence of operations that corresponds to step 1.

[3] A trivial example is the case in which a processor with the highest possible (unique) bus access priority arbitrates for bus mastership. It can stop the arbitration process immediately since it will become bus master

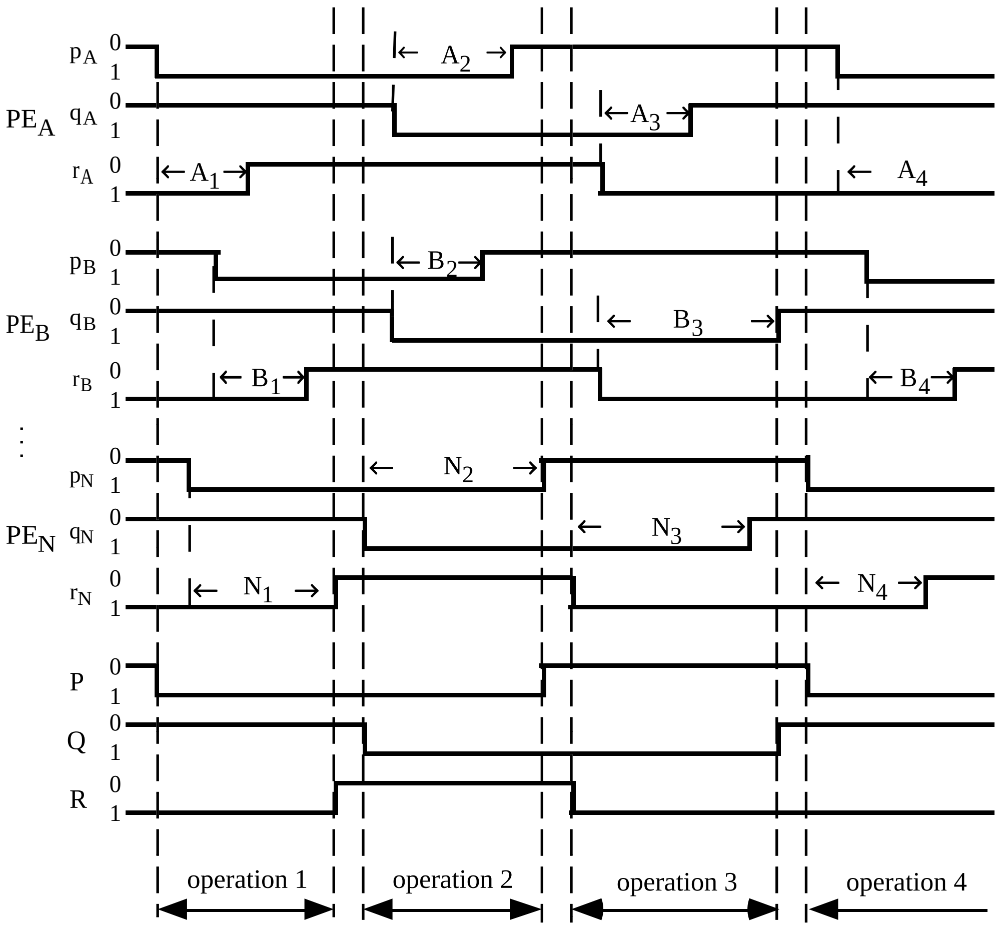
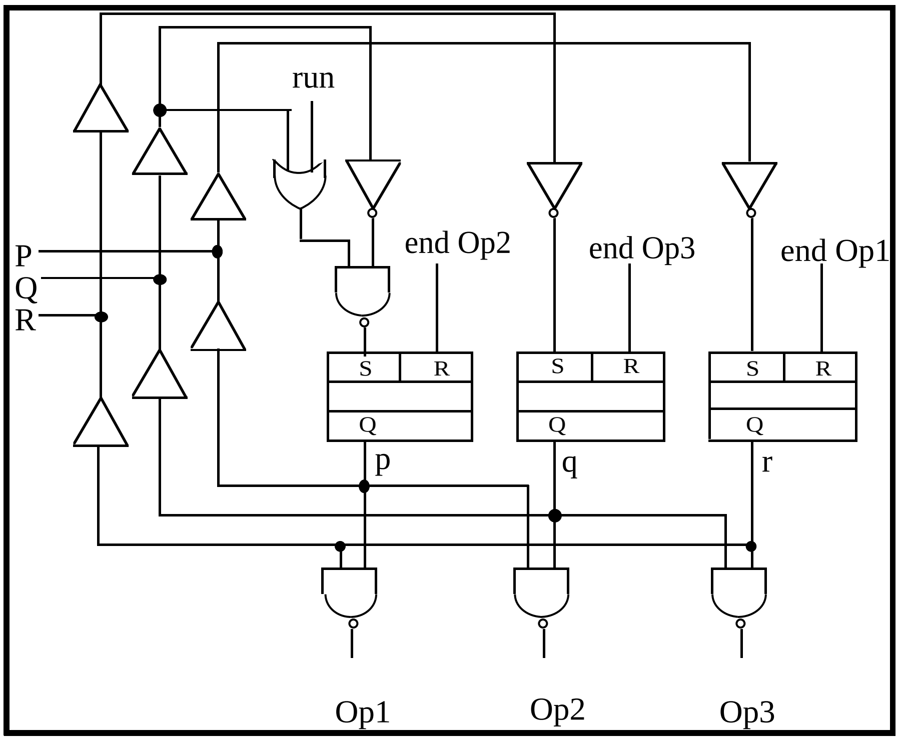


Fig. 2. Signal timing example corresponding to Taub's synchronization procedure.



definitively.

Fig. 3. Control circuit for Taub's synchronization procedure.

The corresponding alternative synchronization procedure is also based on a synchronizer that uses three open-collector bus lines **P**, **Q**, **R**. Suppose as above that the initial signals $\mathbf{p_j}$, $\mathbf{q_j}$, $\mathbf{r_j}$, generated by all PEs, have the values of **1**, **0**, **0** respectively, and therefore also **P**, **Q**, **R**. Then the sequence of events of the synchronization procedure is given by Tab. 2.

Tab. 2. Synchronization protocol for the alternative procedure.

#	**Protocol steps**	**Current stage of signals**							**Next stage**
		signals from PE			lines	signals to PE		trigger outputs	trigger outputs
		run	trm	ack	P Q R	Op_1 Op_2	stop	s_p r_p s_q r_q s_r r_r	p q r
1	Initialization of the system	0	x	x	1 0 0	0 0	0	1 0 0 x 0 0	1 0 0
2	System is initialized (stable state)	0	x	x	1 0 0	0 0	0	0 0 0 x 0 0	1 0 0
3a	Start of the 1st operation by any PE	1 x	x x	x x	1 0 0 1 0 1	0 0 0 0	0 0	0 0 0 x 1 0 0 1 0 x x 0	1 0 1 0 0 1
3b	Start of the 1st operation by other PEs	0	x	x	1 0 1	0 0	0	0 1 0 x x 0	0 0 1
4	Execution of the 1st operation	x	x	x	0 0 1	1 0	0	0 x 0 0 x 0	0 0 1
5a	The 1st operation is finished by any PE	x	1	0	0 0 1	1 0	0	0 x 1 0 x 0	0 1 1
5b	The first operation is finished by other PEs	x	x	0	0 1 1	0 0	1	0 x 1 0 0 0	0 1 1
6	Confirmation of the end of the 1st operation	x	0	1	0 1 1	0 0	1	0 x x 0 0 1	0 1 0
7	Execution of the 2nd operation (stable state)	x	x	x	0 1 0	0 1	0	0 0 x 0 0 x	0 1 0
8a	The 2nd operation is finished by any PE	x	1	0	0 1 0	0 1	0	1 0 x 0 0 x	1 1 0
8b	The first operation is finished by other PEs	x	x	0	1 1 0	0 0	1	1 0 0 0 0 x	1 1 0
9	Confirmation of the end of the 1st operation	x	0	1	1 1 0	0 0	1	x 0 0 1 0 x	1 0 0
10	End of a pair of operations	x	x	x	1 0 0	0 0	0	0 0 0 x 0 0	1 0 0

A control circuit of a SU for the alternative synchronization is presented in Fig. 4. The example of signal timing (Fig. 5) clarifies the interaction between the processors participating in the described synchronization procedure.

It is possible to combine Taub's synchronization procedure with the one suggested in this paper. This is important for applications where both synchronization principles must be used to execute one process, i.e., where some operations are finished when individual processors terminate, and others only when all participating processors are ready with the current operation. In this case a control circuit with four common lines can be used (Fig. 6).

As mentioned above, it is possible sometimes to take a decision about the result before the end of an operation, i.e., before the resulting signals appear on the output lines $\mathbf{L_i}$ of the DS.

This can happen, e.g., in an arbitration operation as discussed in the following section.

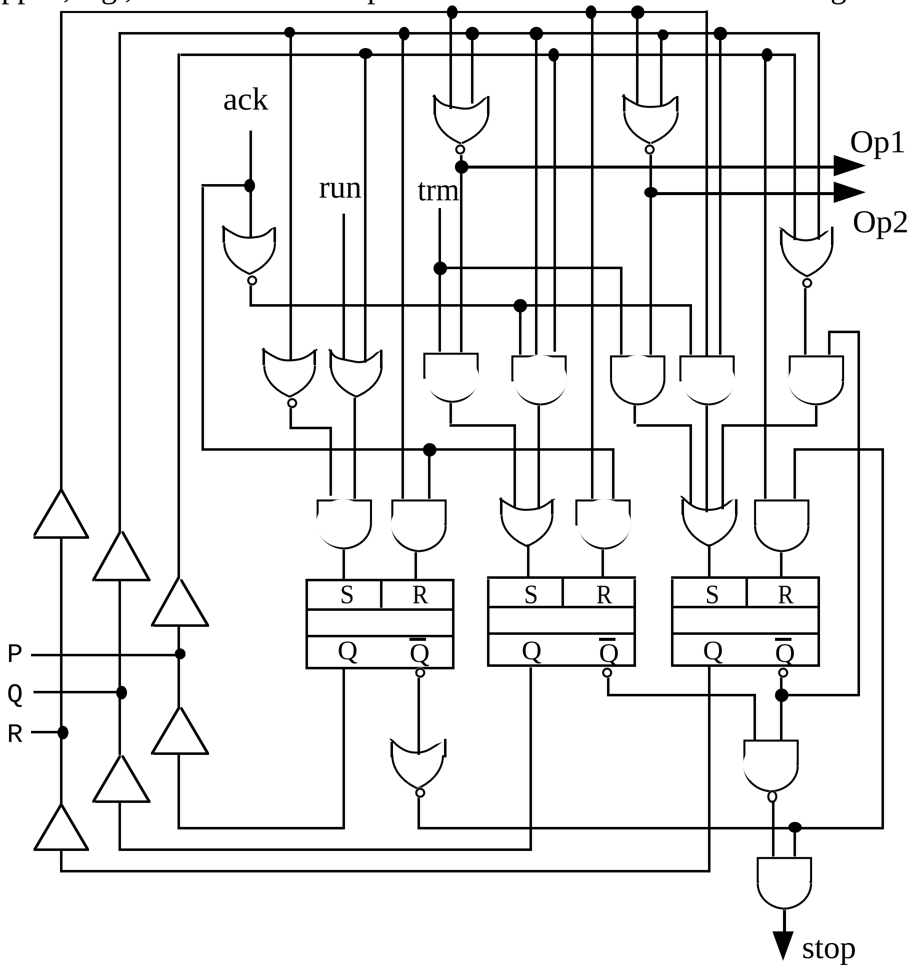


Fig. 4. Control circuit of a synchronization unit for the alternative synchronization procedure.

4. Synchronization of the arbitration operation. Taub's synchronization procedure was especially designed for the arbitration operation [4,5]. The objective of arbitration is to find a processor having the highest arbitration priority, i.e., the highest arbitration word, among all processors participating in the operation. The arbitration operation is executed by a distributed arbitration circuit that consists of PEs connected by open-collector arbitration lines $\mathbf{L_1,\ldots,L_N}$ (Fig. 1). After begin of the operation all PEs assert the bits of their unique arbitration words $\mathbf{A_j} = <\mathbf{a_{j,1},\ldots,a_{j,N}}>$ onto the corresponding arbitration lines. Until the end of the operation the PEs simultaneously compare their own bit values $\mathbf{a_{j,i}}$ with the current signals on the arbitration lines. If a line $\mathbf{L_i}$, $\mathbf{i \in Z}$, is set to a logic **1** and the corresponding bit $\mathbf{a_{j,i}}$ of the word is **0** then the PE **j** withdraws all less significant bits $\mathbf{a_{j,i+1},\ldots,a_{j,N}}$ from the lines $\mathbf{L_{i+1},\ldots,L_N}$. If this condition is not longer true the PE asserts the withdrawn bits again.

Every PE consists of combinatorial logic which compares all bits of its arbitration word with the signals on the corresponding arbitration lines. Let $\mathbf{T(j)}$ be the gate delay time to process one bit for $\mathrm{PE}_\mathbf{j}$. The computation of the wired-ORs on the bus lines requires an additional bus-propagation delay $\boldsymbol{\tau}$. This delay is identical for all arbitration lines [3]. Obviously, as the arbitration scheme considered is a combinatorial circuit, each PE starts to compare every bit automatically when the signals on the lines belonging to more significant bits are changed [5-7]. Thus the operation can be considered as a sequential procedure.

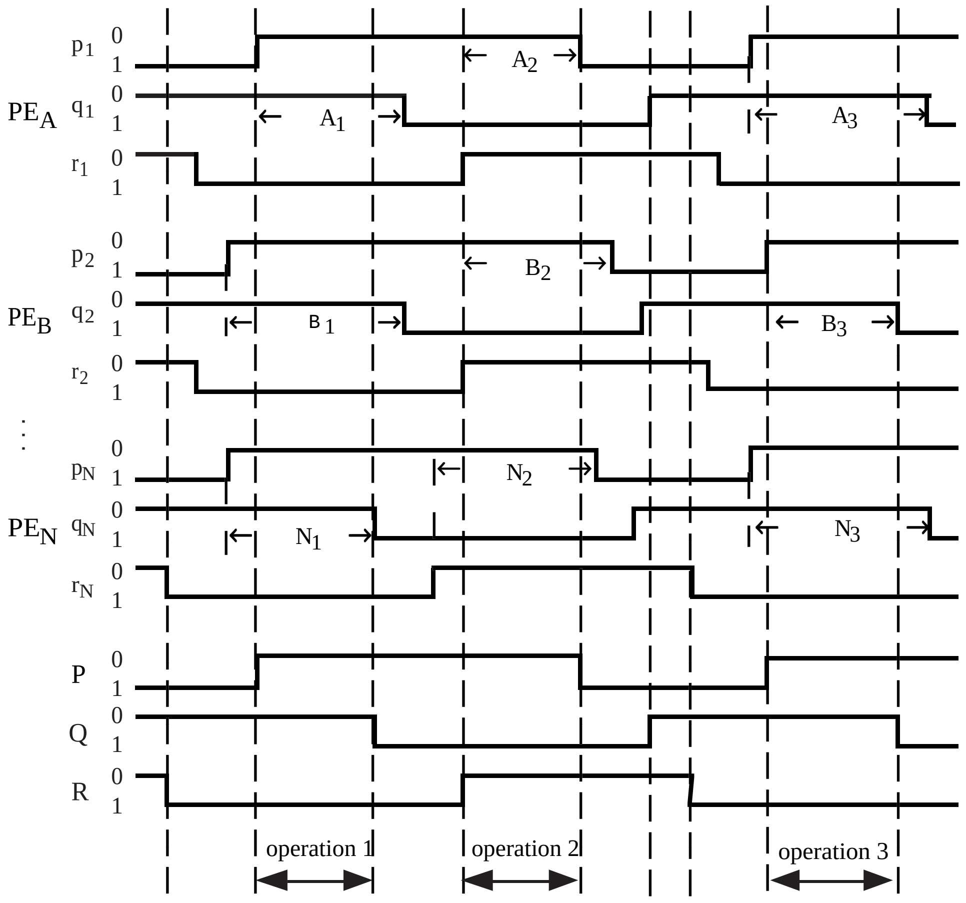


Fig. 5. Signal timing example for the alternative synchronization procedure.

The duration of the operation is $\mathbf{t(Z,W)} = \sum_{\mathbf{j\in Z}} \mathbf{C_j(Z,W)T(j)} + \sum_{\mathbf{i=1,\dots,N}} \boldsymbol{\tau}_{\mathbf{i}}$,

where

$\mathbf{C_j(Z,W)} = \sum_{\mathbf{i=1,\dots,N}} \mathbf{d_{i,j}}$, is the number of the individual propagation delays $\mathbf{T(j)}$ introduced in the operation $\mathbf{W}$ by the $\mathbf{j}$-th PE,

$\mathbf{T_L \leq T(j) \leq T_H}$, where $\mathbf{T_L}$ and $\mathbf{T_H}$ are upper and lower bounds identical for the whole DS,

$\mathbf{d_{i,j} = 1}$ if the $\mathbf{j}$-th PE takes longest to complete the $\mathbf{i}$-th distributed part of the operation $\mathbf{W}$,

$\mathbf{d_{i,j} = 0}$ else, so that $\sum_{\mathbf{j\in Z}} \mathbf{d_{i,j}=1}$.

On one hand it is very difficult to find exactly the minimal upper bound for the duration $\mathbf{t(Z,W)}$. The reason is the lack of information on the current set $\mathbf{Z}$ of PEs participating in the operation, and on their individual speed characteristics, i.e., $\mathbf{T(j)}$. On the other hand the PEs can find the minimal upper bound of the duration of the operation if they cooperate according to the following algorithm.

After the PEs have recognized the begin of the operation, they assert their $\mathbf{1}$ signals $\mathbf{p_j}$ onto the wired-OR line $\mathbf{P}$. During execution each PE $\mathbf{j}$ waits for a specified individual period of time $\mathbf{t(j,W)}$ before it switches $\mathbf{p_j}$ from $\mathbf{1}$ to $\mathbf{0}$. After the slowest PE has switched its $\mathbf{p_j}$ to $\mathbf{0}$, line $\mathbf{P}$ changes to the $\mathbf{0}$ state and all PEs recognize the end of the operation. In modular systems where different PEs can be combined arbitrarily, **t(j,W)** must be independent of **Z** and of the technical characteristics of other PEs. Therefore – according to Taub's synchronization procedure – has the global waiting time for the end of the operation to be chosen as **Max{t(j,W), j∈ Z}**.

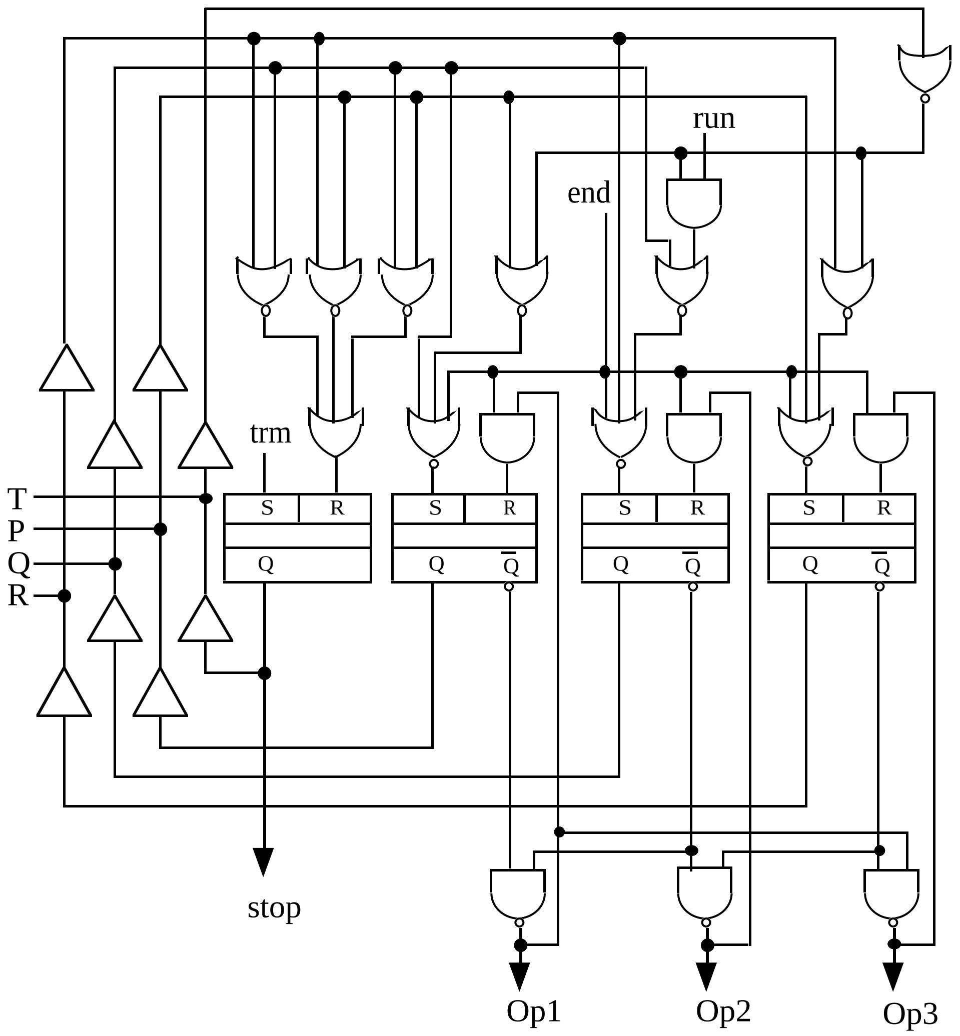


Fig. 6. Control circuit of a SU for combined synchronization.

After this time all output signals of the participating PEs are settled. Obviously the operation is executed correctly if the global waiting time **Max{t(j,W), j∈ Z}** is not shorter than the duration of the operation. This means that the condition

$$\mathbf{Max}\ \{t(j,W),\ j\in Z\} \geq \sum_{j\in Z} C_j(Z,W)T(j) + \sum_{i=1,\dots,N} \tau_i, \qquad (1)$$

must be true for all possible **T(j)**, $T_L \leq T(j) \leq T_H$. The less this value the higher is the performance of the DS. The alternative synchronization procedure allows to reduce this global waiting time.

Relation (1) guarantees the correct result of the operation. However the PEs generally know neither which PEs participate in the current operation nor their the individual propagation delays. This means that the values **t(j,W)** cannot be found from (1) directly. Therefore –

according to [7,8] – every PE **j** has to take into account relations (1) for every possible set of participants and to assume the worst case, i.e., the maximal individual waiting time

$$\mathbf{t(j,W) = NT(j) + \left(\left\lfloor \frac{N}{2} \right\rfloor + 2\right)\tau}. \tag{2}$$

In this case the global waiting time $\mathbf{Max\{t(j,W),\ j\in Z\}}$ is usually much higher than the actual duration of the operation. This can degrade the speed of the system considerably. The probability for this to happen is the higher the more PEs participate in the operation.

Let us consider for example an arbitration operation with $\mathbf{N=4}$. If we use Taub's synchronization procedure then follows from (2) that the global waiting time $\mathbf{t(j,W)}$ is equal to $\mathbf{Max\{4T(j)+4\tau,\ j\in Z\}}$[4]. However the PE having the maximal arbitration word **<1111>** could correctly decide to get next bus mastership immediately after begin of the operation. Also, the PE having the arbitration word **<0111>** must wait only the time $\mathbf{T_H+\tau}$ and can then terminate the operation. For such cases the alternative synchronization procedure has considerable speed advantages.

If a PE recognizes at the time $\mathbf{t_a(j,W)}$ that every bit $\mathbf{a_{j,i}}$ of its arbitration word $\mathbf{A_j}$ is greater or equal than the value of the signal on the corresponding bus line $\mathbf{L_i}$, it concludes that it has the highest priority among the arbitrating PEs. This happens usually before the arbitration lines are settled. The minimal waiting time is obtained if the PE acts as the next bus master from this time on. The second column of Tab. 5 gives for every PE the time $\mathbf{t_a(j,W)}$ after which it knows the arbitration result independently of the set **Z** of participating processors and their speed.

However the other PEs still have to wait for the result of the arbitration operation, i.e., they have to wait until the arbitration lines have settled. For that, the new bus master can terminate the operation using the alternative synchronization procedure. For this case the third column of Tab. 5 gives the individual waiting time $\mathbf{t_b(j,W)}$ for all PEs which, again, is independent of the set **Z** of participating processors and their speed.

If the alternative synchronization procedure is used, i.e., if every processor can terminate the operation, the average global waiting can be reduced by 50% as shown in Tab. 5. This is the case either if all processors are of comparable speed, i.e., all **T(j)** have similar values, or if the number of participating processors is high.

5. Conclusions. The suggested synchronization procedure allows to use different hardware synchronization procedures. Using this procedure, it is also possible to identify the end of the operation dynamically and to reach a minimal upper bound for the duration of asynchronous operations. This can considerably reduce the processing time in distributed processing systems. One advantage of the suggested synchronization is that it does not require modifications of standard hardware, i.e., backplanes, connectors, etc. The only requirement is to

[4] Using a modification of Taub's synchronization procedure [7], the global waiting time can be reduced to $\mathbf{T_H+\tau}$ if only one processor participates in the arbitration.

extend slightly the logic of the distributed synchronization circuits.

Tab. 5. Duration of the arbitration operation with 4-bit arbitration words.

Arbitration number	Individual waiting time for alternative synchronization	Termination time for alternative synchronization	Individual waiting time for Taub's synchronization
	$t_a(j,W)$	$t_b(j,W)$	$t(j,W)$
A_{15} = 1 1 1 1	0	$T(15)+\tau$	$4T(j)+4\tau$
A_{14} = 1 1 1 0	$T(14)+T_H+2\tau$	$T(14)+T_H+2\tau$	$4T(j)+4\tau$
A_{13} = 1 1 0 1	$T(13)+T_H+2\tau$	$2T(13)+T_H+3\tau$	$4T(j)+4\tau$
A_{12} = 1 1 0 0	$T(12)+2T_H+2\tau$	$T(12)+2T_H+3\tau$	$4T(j)+4\tau$
A_{11} = 1 0 1 1	$T(11)+T_H+2\tau$	$2T(11)+T_H+3\tau$	$4T(j)+4\tau$
A_{10} = 1 0 1 0	$T(10)+2T_H+3\tau$	$2T(10)+2T_H+4\tau$	$4T(j)+4\tau$
A_9 = 1 0 0 1	$T(9)+T_H+2\tau$	$2T(9)+T_H+3\tau$	$4T(j)+4\tau$
A_8 = 1 0 0 0	$T(8)+2T_H+3\tau$	$T(8)+2T_H$	$4T(j)+4\tau$
A_7 = 0 1 1 1	$T_H+\tau$	$T_H+\tau$	$4T(j)+4\tau$
A_6 = 0 1 1 0	$T(6)+T_H+2\tau$	$T(6)+T_H+2\tau$	$4T(j)+4\tau$
A_5 = 0 1 0 1	$T(5)+T_H+2\tau$	$2T(5)+T_H+3\tau$	$4T(j)+4\tau$
A_4 = 0 1 0 0	$T(4)+2T_H+3\tau$	$T(4)+2T_H+3\tau$	$4T(j)+4\tau$
A_3 = 0 0 1 1	$T_H+\tau$	$T_H+\tau$	$4T(j)+4\tau$
A_2 = 0 0 1 0	$T(2)+T_H+2\tau$	$T(2)+T_H+2\tau$	$4T(j)+4\tau$
A_1 = 0 0 0 1	$T_H+\tau$	$T_H+\tau$	$4T(j)+4\tau$
A_0 = 0 0 0 0	$T_H+\tau$	$T_H+\tau$	$4T(j)+4\tau$

References

1. M. Makhaniok, V. Cherniavsky, R. Männer, O. Stucky, Massively Parallel Realization of Logical Operations in Distributed Parallel Systems; in: H. Burkhart (Ed.): Proc. CONPAR ´90 - VAPP IV, Lect. Notes in Comp. Sci. **457**, Springer, Heidelberg (1990), 796-805.
2. D.M. Taub, Hardware method of synchronizing processes without using a clock, Electron. Lett., No. 19, Vol. **19**, 1983, 772-773.
3. D.M. Taub, Improved Control Acquisition Time for the IEEE 896 Futurebus, IEEE Micro, **7** (3), 1987, 52-62.
4. D.M. Taub, Clockless synchronization of distributed concurrent processes, IEE Proc. E, Vol. **139**, No. 1, 1992, 88-92.
5. ANSI/IEEE Std. 896.1-1987, IEEE Standard Backplane Bus Specification for Multiprocessing Architectures: Futurebus.
6. M. Makhaniok, R. Männer, Synchronization of Parallel Processes in Distributed Systems;

in: L. Bougé, M. Cosnard, Y. Robert, D. Trystram (Eds.): Proc. CONPAR ´92 - VAPP V, Lyons, France; Lect. Notes in Comp. Sci. **634**, Springer, Berlin (1992) 157-162

7. Taub M.D., Corrected Settling Time of the Distributed Parallel Arbiter, IEE Proc. E, Vol. **139**, No. 4, 1992, 348-354.
8. Kipnis S., Analysis of Asynchronous Binary Arbitration on Digital Transmission-Line Busses, IEEE Trans. Comput., Vol. **43**, No. 4, 1994, 484-489.