Peer-reviewed & Open access journal
www.academicpublishingplatforms.com
The primary version of the journal is the on-line version

ATI - Applied Technologies & Innovations
Volume 6 | Issue 1 | March 2012 | pp.1-9
http://dx.doi.org/10.15208/ati.2012.1

Superscalar micro-pipeline structure

Dimitar Tyanev
Faculty of Computing and Automation, Technical University of Varna, Bulgaria

e-mail: dstyanev@yahoo.com
postal address: 1 "Studentska" Str., Varna – 9010, Bulgaria

The problem of micro-pipeline with big latency has been analyzed. In order to obtain regular temp into micro-pipeline hardware saturation should be used. The synthesis of micro-pipeline controllers into fork dot is presented in two variants - about 2-phase and 4-phase transfer protocol.

Keywords: Micropipeline, superscalar, pipeline controllers, protocols

Introduction

The paper considers micro-pipeline with disturbed functioning due to one of the micro-pipeline stages latency. Depending on how much the latency of the "slow" stage differs than the others delay, particular pipeline sections could be in different states. For example, previous stage will remain in waiting state because its request will be unaccepted for a long time. So this stage could not return back acknowledgement signal in time. The situation will be distributed gradually to the previous stages and they also will be fixed into waiting states. If the delay is high, this condition could reach the initial stage, which will end the task load into the pipeline. Of course, it will be temporal.

Assuming that the temp is steady, after this situation into the stages there will be no request because of the big delay and lack of requests. All tasks which got over this point could finish their execution and to step down of the pipeline. As a result, released stages will be set up into initial state which could be propagated till the last pipeline stage.

A superscalar structure

If the stage executing large computations could not be decomposed and remains the same, we apply at this point a machine saturation approach (also called superscalar). By "superscalar" in this case we will understand the addition of the same micro-pipeline stage one, two or more times with parallel connections, as it is shown on Figure 1.

There is a pipeline section with three consecutive stages presented on the Figure. We accept that the stage number 2 has big latency. Four-time duplication of stage 2 is exemplary. Duplication degree will be considered later.

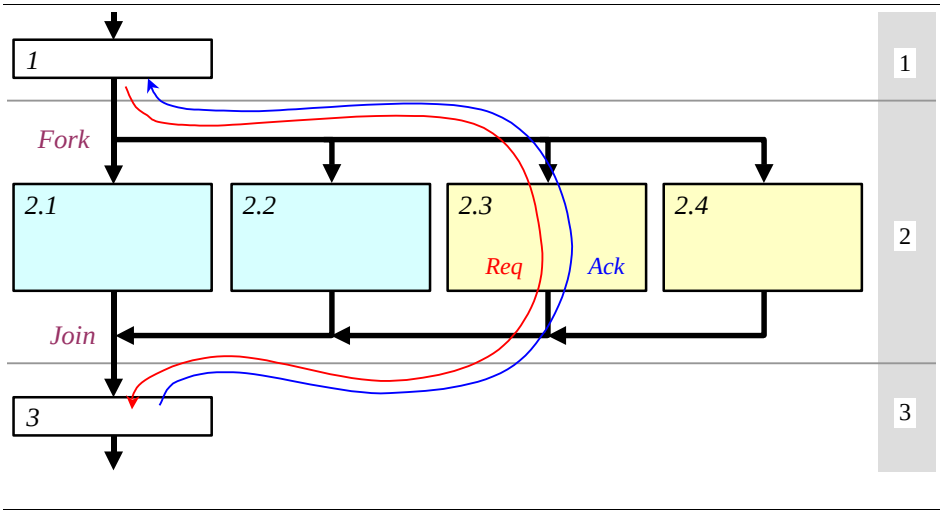
The superscalar structure has no restrictions about the type of duplicated element. This means that it could have arbitrary pipeline structure (Tyanev and Yanev, 2011).

There are two specific points set into the structure - branch dot (*Fork*) and joint dot (*Join*).

We assume that the pipeline structure's current state is as follows: into stage 2.1 there is a task (k-1) and into stage 2.2 - task *k*. Task (k+1) is into stage 1. Stages 2.1

and 2.2 are busy and are executing large computations. We consider as well that the other parallel stages 2.3 and 2.4 have finished their computations and have given the results through stage 3 down to the pipeline. So these two stages are in initial state and could accept new tasks to execute.

FIGURE 1. 4-TIMES SUPERSCALAR



Each of the parallel stages, after their computations finish, can give request to the stage 3's pipeline controller into joint dot. This stage has to operate with a lot of the previous stages, which means that its pipeline controller is non-linear. This is formulation of the first particular problem - to synthesize pipeline controller into superscalar structure joint dot.

On other hand, each parallel stage should start new task regardless of the other stages. So every parallel stage has to be controlled by its own pipeline controller. If the controller of free parallel stage receive request, it will write the initial data and after the end of computations will give request to the next stage. In this meaning, the functionality of such controllers is not fully used and they are simple linear controllers.

Stage 1 indicates the obtained result by sending request. This request has to be attended into one of the parallel stages. So into branch dot for this request there is a unique situation created. It is expressed into specific control organization. Because every parallel stage has to be served independently into this point, the request from stage 1 has to be given to only one stage chosen through the free stages which emitted acknowledgement signal.

So there is the second problem's formulation - to synthesize pipeline controller which can solve the selection problem about the acknowledgement signals coming from the parallel stages and to start the chosen stage by writing into its initial register the data from stage 1.

Pipeline controller into join dot

At the stage 3 input there are the data buses from all branches combined. A lot of requests are given to the pipeline controller at this stage - in the example from Figure1 they are 4. It is impossible to stipulate the order and the moment of these

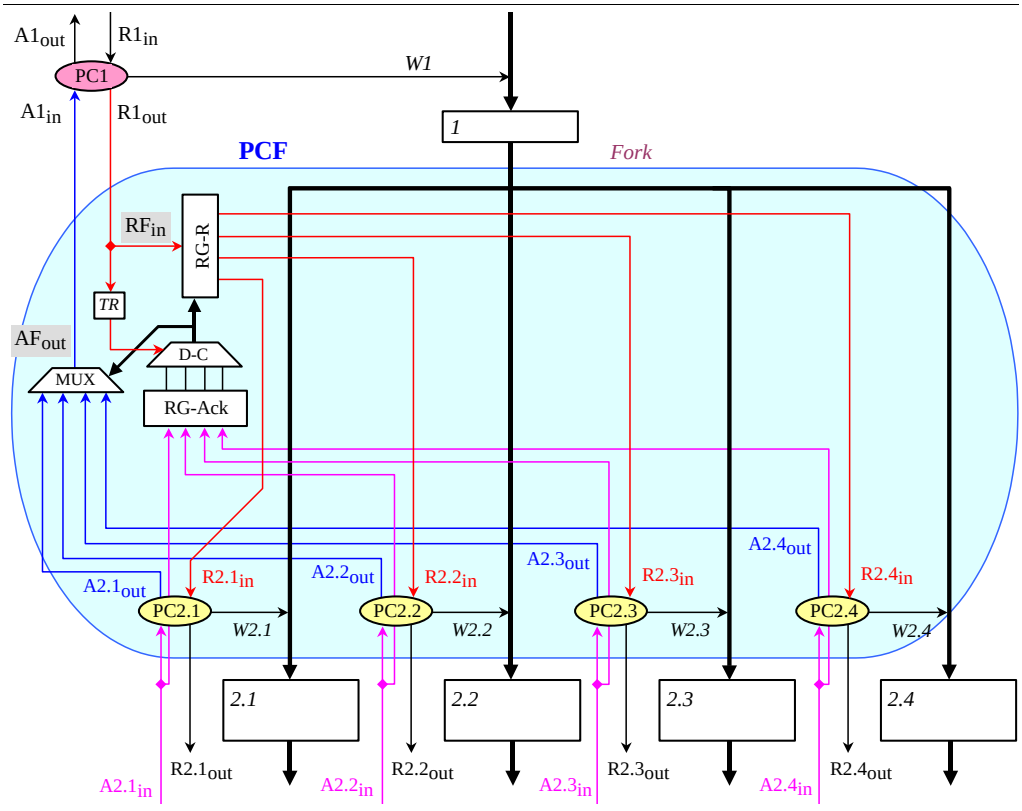
requests' arrival. It depends on the order the parallel stages has started, on the data each stage is working with and finally on the computational delay. The following conclusions were made based on these considerations:

1. Working conditions of the pipeline controller at this point are analogous to the conditions at joint dot into any other non-linear structure. Then the problem for synthesis of pipeline controller to control the joint dot in superscalar structure, formulated above, has analogous decision (Tyanev and Popova, 2011);
2. The order in which the tasks are coming to the common stage 3 probably will not correspond to the order the respective tasks were started. This imposes the result identification and recovery of their correct order. The problem for results order recovery is formulated and solved in (Tyanev and Bozhikova, 2011).

Pipeline controller into fork dot

Figure 2 shows the logical structure of the synthesized pipeline controller at the branch dot.

FIGURE 2. PIPELINE CONTROLLER AT FORK DOT STRUCTURE



We already showed that parallel micro-pipeline stages are controlled by their own pipeline controllers. When one stage finishes its computations, it gives request to the joint dot controller. These requests are noted into the structure as *R2.x* (*x* stands for arbitrary number between 1 and 4, in this case). When given request *x* is accepted by joint dot controller, the results from corresponding stage 2.x are written to the input fixing register of the joint dot stage (stage 3). The controller *PC2.x* receives in response acknowledgement signal with the same number *A2.x*. Thus the data transfer of 2.x stage finishes and it is ready for the next start. The event has to be remembered because there are lots of stages. This is why each returned acknowledge *A2.x* is fixed into register RG-Ack, which has number of flip-flops equal to the superscalar degree. Each bit in this register, set in 1, means that the corresponding stage into the parallel structure is in waiting state for request from stage 1. Every bit in this register set to 0 means that the corresponding stage is either still computing or has given request to the next stage, but this request is still unaccepted, i.e. the corresponding stage is not ready for next start.

So, when stage 1 finishes its computations it generates the request $R1_{out} \equiv RF_{in}$. This request has to receive one of the free micro-pipeline stages in the parallel structure. It doesn't matter which stage exactly. Hence, here we have the problem for the selection of one stage among all free. The selection problem has to be solved with the appearance of request RF_{in}, which has two active edges in the case of 2-phase protocol (falling and rising). The new request appearance is fixed by edge-detector into special flip-flop, noted as TR, which is set into single state. Single signal from the normal value output of this flip-flop is led to the scheme *D-C* (*Daisy-Chain*) as a selection signal (*Choice*) (Procopov, S.P., Tyanev, D.S. 2009). This logical scheme implements selection among all possible candidates based on the principle "first detected". The selection is kept on the *D-C* output until the fall of *Choice* signal active 1. The logical combination from the output of this scheme is used as a code controlling MUX switching.

Request RF_{in} has to be allocated and directed to the selected stage. Despite of this, its value have to be kept till the next time because at this moment there will be other requests RF_{in}, which have to be allocated to another stages. Then the allocation of request RF_{in} cannot be done only by de-multiplexer. The allocated request has to be fixed into a flip-flop. Register RG-R into the structure is for this purpose. Signal *Choice* is created by TR flip-flop.

Selected request RF_{in} is transformed into request $R2.x_{in}$ and reaches the pipeline controller *PC2.x* of the selected free stage, which switches and emits the signal *W2.x* to write of the data coming from stage 1 data bus. Switched controller PC2.x returns through multiplexer MUX acknowledgement signal AF_{out} intended to PC1 controller. This ends the connection session and the TR flip-flop has to be reset. The corresponding flip-flop into RG-Ack has to be reset as well, which erases the stage that has started this procedure from the list of free stages. The same stage could be registered into RG-Ack as free when transfers the result through the joint dot, i.e. when receives again the acknowledgement *A2.x*.

As a conclusion of the above considerations, we can define the pipeline controller into branch dot in superscalar structure as controller with distributed logic. Its structure contains common and distributed section.

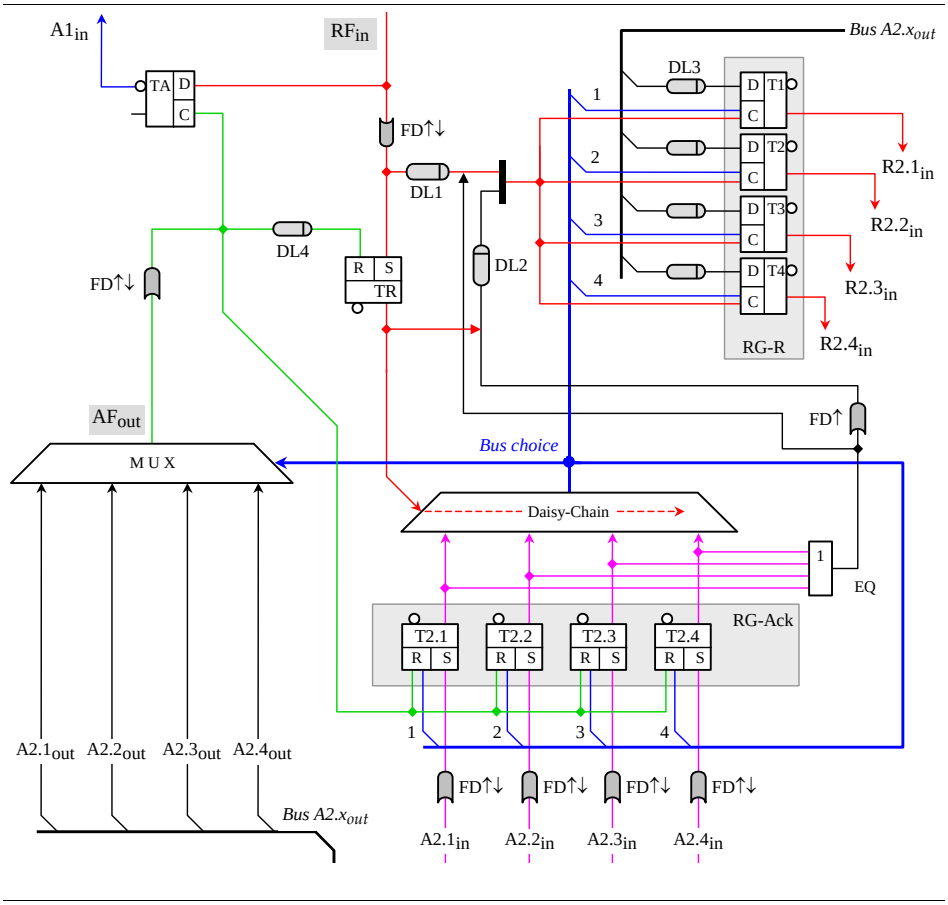
Circuit of 2-phase pipeline controller

It is known (Tyanev and Yanev, 2011), that branching out of the stage request previous to the branch dot in 2-phase protocol conditions is embarrassed by lack of correspondence between logical values of the dialog signals from source and destination. In this case, the

reason is the unknown switching history of the destination pipeline controller, which is not unique into the parallel structure. Then it is necessary to convert the logic value of the current request.

Figure 3 shows the common section of the pipeline controller logic scheme to control superscalar structure. The parallel stages controllers are not shown although they are part of this.

FIGURE 3. CIRCUIT OF 2-PHASE PIPELINE CONTROLLER AT FORK DOT

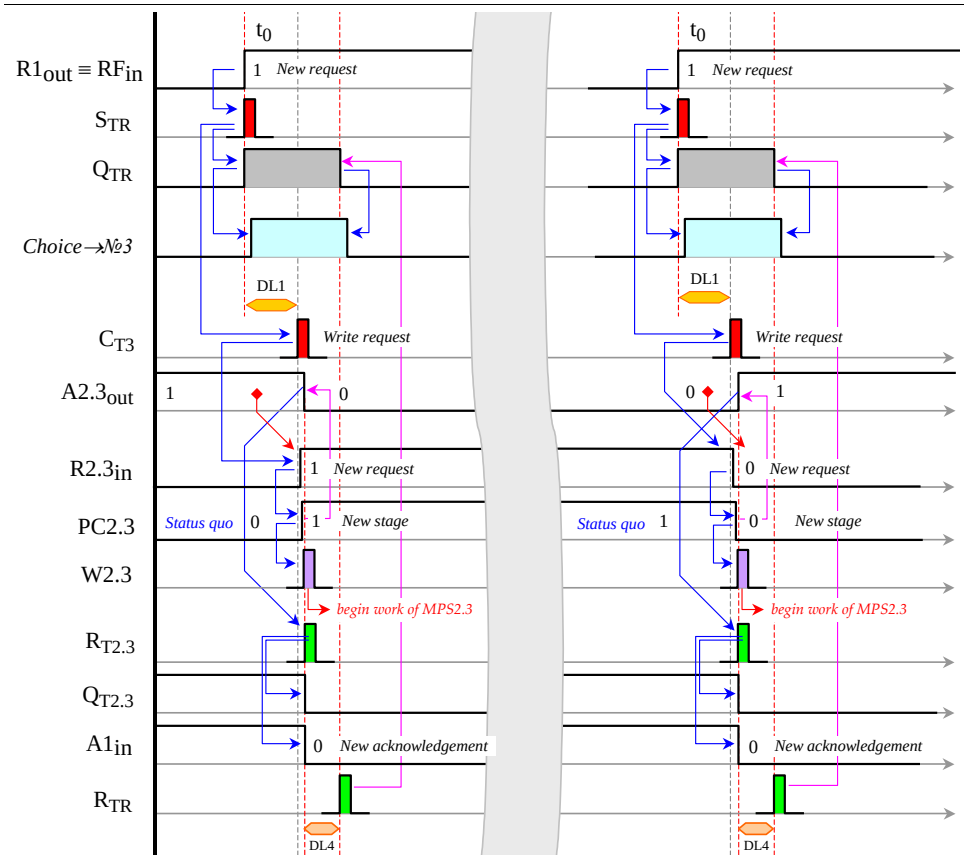


All flip-flops in RG-Ack are asynchronous RS-*Latch* flip-flops which are set to initial state by *Reset* signal not shown on the Figure 3. Impulses out of edge-detectors FD↑↓ on their S-input are used to fix the appearance of acknowledgement A2.x from stage controller at joint dot. Each flip-flop from this register is set to zero individually depending on selection code delivered by *Bus choice*. The R-input is conjunction of all signals passed to it.

Request register RG-R is built by synchronous D-*Latch* flip-flops. Selection code from *Bus Choice* defines in which flip-flop will be written the new request. Writing impulse on C-input is conjunction of all signals passed to it. The value to be written is about the acknowledgement signal supported by the selected controller in moment $A2.x_{out}$. The request value this controller has to receive is the same, i.e. $R2.x_{in}=A2.x_{out}$ and is delivered by *Bus A2.x*.

The AF_{out} switching is caught by edge-detecting schemes (both falling and rising) FD↑↓. The short impulse from the edge-detector is used to set to zero the TR flip-flop and the corresponding T2.x flip-flop into RG-Ack. Let the stage 2.3 to be selected, as it is shown on Figure 3. Figure 4 presents the time-diagram of its starting.

FIGURE 4. START OF MICRO-PIPELINE STAGE 2.3



The process is shown from the time t_0, when a micro-pipeline stage completes its calculations. Depicted are two variants – when the status quo of the controller PC2.3 is 0 and when is 1. After the new request arrival ($RF_{in}=1$) there is an strobe impulse arise on S-input of the TR flip-flop. This flip-flop is switched to 1 in order to implement a free stage selection.

The strobe impulse C_{T3} appears on C-input with delay DL1 and writes in T3 flip-flop the value of new request $R2.3_{in}$=1. A pipeline controller PC2.3 switching to 1 follows because on its input there are two ones ($A2.3_{in}$=1, $R2.3_{in}$=1). The controller switching generates writing data impulse W2.3 and the edge of returned acknowledgement generates with a delay DL4 the resetting impulse R_{TR} ($A2.3_{out}\rightarrow AF_{out}\rightarrow$FD↑↓) (look Figure 2). With data write starts the parallel stage computations.

Specific situation at fork dot

If there are always free stages into the parallel structure when request from stage 1 appears, the process will be as it is shown on figure 4. But the situation when the request from stage 1 cannot make a choice is possible as well. This could occur when there are no free stages in the parallel structure, i.e. all the computations are still carried. And yet the request RF_{in} has appear and it will try to write into RG-R register. Because there are no free stages, the write in this case has not to be done. It is forbidden by the *EQ* signal, which appears always as an active zero and in the time when there are no free stages registered into RG-Ack.

In these conditions, the appearance of stage 1 request causes its fixing into the TR flip-flop which sets 1 on its output. This one as a selection signal *Choice* stays set, but there is no real selection because RG-Ack has zero content. So, there is request by stage 1 but the attempt to be allocated and led into the superscalar structure fails. The event is abortive and cannot be repeated. Remind that in this condition the *EQ* signal has blocked the possibility to write into RG-R.

Way out of this situation is possible only when any of the 2.x parallel stages are released. Detection of this event into RG-Ack leads to switching of *EQ* signal to one. Except this, the selection scheme (which is still supported active by TR flip-flop) creates the selection code and this code writes the value of *R2.x* request from stage 1 into RG-R with delay DL3. In this way, newly released parallel stage will be started again. After the acceptance of stage 1 data, the acknowledgement $A2.x_{out}=AF_{out}$ is sent back. Thus, the stage 1 receives long awaited acknowledgement and the scheme in the common section is restored - TR and T2.x flip-flops are reset.

The reset TR flip-flop forbids writing into RG-R always when there is still old value of RF_{in}. Such writing attempt could be done by *EQ* signal when despite of the lack of free stages some are registered as free, but in this moment stage 1 is still busy and didn't produce new request. The lack of requests is indicated by the TR flip-flop zero content.

Circuit of 4-phase pipeline controller

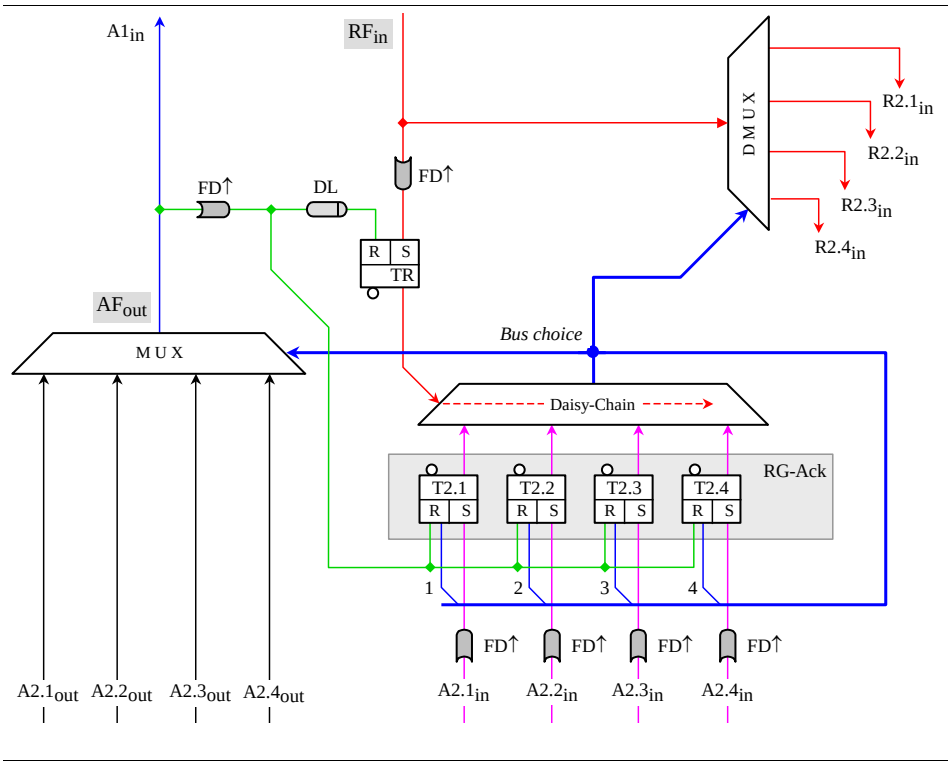
The variants of pipeline controller at joint and branch dotes with 4-phase transfer protocol are synthesized based on the same formulation, described in previous sections. In this case, the active level of *Req* and *Ack* signals is only one and this is 1. This means that the necessary edge-detecting schemes to recognize events will work only on rising edge.

The common section of controller logic scheme at branch dot corresponds to the presented on Figure 2. The changes are due to the constant 1-value of actual signals. For example there is no need to convert the actual request value in the 4-phase case. And because the request RF_{in} is set to zero by receiving the acknowledgement AF_{out}, its storage becomes unduly. Synthesized logic scheme of the pipeline controller common section is presented on Figure 5.

TR flip-flop fixes the request RF_{in} arrived from stage 1. After the selection code switches de-multiplexer DMUX, the request is given as R2.x signal and reaches PC2.x controller. This controller is switched and returns acknowledgement $A2.x_{out}$ which go out of multiplexer MUX as signal AF_{out}. Scheme restoration follows with delay DL.

If there is no selected stage after RF_{in} registration (selection code is zero), the scheme remains in this state until there is no free stage fixed. With the free stage appearance the request (kept by controller PC1 into 1) goes through de-multiplexer and then the process follows the described procedure.

FIGURE 5. CIRCUIT OF 4-PHASE PIPELINE CONTROLLER AT FORK DOT



The scheme of the pipeline controller to control joint dot is similar to the synthesized for 4-phase protocol in (Tyanev and Popova, 2011).

Conclusion

The machine saturation (superscalar structure) is hard to achieve as it was shown. It is expensive in terms of machine costs.

Let to clarify how to define the duplication degree for the pipeline element into the parallel structure. This estimation is directly related to the performance of target structure. Assuming that the steady pipeline temp is evaluated with typical delay T_{DL} and the delay of considered pipeline element is S_{DL}, the duplication degree will be defined as

$$N = \left\lfloor \frac{S_{DL}}{T_{DL}} \right\rfloor .$$

With this N-times duplication we can be sure that for each task finishing its computations into stage 1 (according exemplary structure of figure 1) there will be at least one free and waiting 2.x stage. The parallel structure eliminates the pipeline's bottle-neck.

Presented in this paper logical schemes and all related descriptions we accept as a method to synthesize superscalar micro-pipeline structures.

References

Procopov, S., Tyanev, D., 2009. "Hardware implementation of strategies for servicing queues," Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech'09), Ruse, Bulgaria, ISSN: 1313-8936, pp.I.3-1(8)

Tyanev, D., Yanev, D., 2011. "Non-linear asynchronous micro-pipelines," International Conference on Computer Systems and Technologies, (CompSysTech'11), Vienna, Austria, ACM ISBN: 978-1-4503-0917-2, ACM Press, pp.38-44

Tyanev, D., Popova, S., 2011. "Branch management into micro-pipeline joint dot", Applied Technologies and Innovations, Vol.5, pp.11-26

Tyanev, D., Bozhikova, V., 2011. "Algorithm for micropipeline buffer control," Applied Technologies and Innovations, Vol.4, pp. 12-21