Micro-pipeline Section For Condition-Controlled Loop

Dimitar Tyanev, Stamen Kolev, Dragomir Yanev

Abstract: This paper presents a novel methodology for design of operational computational devices, realizing a diversity of condition-controlled loop algorithmic structures. The synthesis of such a structure takes in count just the necessary for the computation signal changes, which makes the synthesis of controlling finite state machine unnecessary. Thus the resulting controlling circuit is tightly coupled with the logic one. In this sense the structure can be defined as self-controlling and can be part of micro-pipeline structure with certain delay. The methodology can be applied when designing either synchronous or asynchronous structures. The structures in either case are as fast as possible because they can be organized in pipeline fashion.

Key words: Controlling automata, computational devices, micro-pipeline.

Introduction

Nowadays the realization of the computational process goes deeper in the lower hardware levels, but there are no popular formal methodologies for the implementation of an arbitrary computational algorithm on hardware. A reason for this is the difference in hardware and software design. The authors take advantage of programmable hardware technologies like (FPGA’s, CPLD’s, etc.) and propose such a methodology.

1. Main Goal

Authors’ goal is defined in their previous papers [1, 2, 3, 4]. Basically the idea is to develop methodology for hardware realization of arbitrary computational algorithm, which is free of micro-program controlling device.

A methodology for hardware realization of one of the basic algorithmic structures, the condition-controlled loop, is presented in the present work. The block diagram of the loop is depicted below.

Pic. 1 Condition-controlled loop block diagram (CT - Condition test)

Typical for the structure is that the loop body is executed until certain condition is changed. The condition is re-evaluated and tested in each iteration of the loop. Usually the condition has the following form:

\[(\Delta \text{Var})_k \leq \varepsilon\]  \hspace{1cm} (1.1)

where \((\Delta \text{Var})_k\) is the k-th change in the control value Var, and \(\varepsilon\) is the a priori known accuracy. This is why the computation of the test condition will be part of the logic in the loop body. For simplicity we assume that sequential structure, which can be realized as pipeline or one-phase structure [1].

2. Logical structure of the condition-controlled loop

The synchronous or asynchronous hardware realization of the condition-controlled loop in principal is the same. In the variant on picture 2 the structure is synchronous and one
A phase clock generator is used. It consists of three sequentially joined elements: Multiplexer, Logic and Register. The Logic implements the computation in the loop body and calculates the new control value. The register has Edge structure and changes its output on the rising edge of the clock signal. The feed-back from the Register to the Multiplexer makes the loop computation possible. Four toggle master slave flip-flops Q1, Q2, Q3 and Q4, changing state on the falling edge of the clock signal, are used for the control of the sequential structure, forming a controlling circuit.

The loop is fully hardware implemented. The clock diagram on picture 3 shows the process of functioning of the structure and clarifies the role of the controlling circuit the synthesis of which is done according to the authors’ methodology. The one and only “Engine” of the resulting structure is the clock signal. The lack of controlling finite state machine turns it into an element, which can be very easy embedded into a micro-pipeline structure. This shows that the proposed structure can be used as micro-pipeline loop element which corresponds to authors’ basic concept in the current research.

![Diagram](image)

**Pic. 2** Condition-controlled loop hardware implementation

The structure can be in one of two states: inactive and active. In inactive state the feedback is held active by the Multiplexer MUX. In the Register R is kept the data from the last computation, which can be read. In this state the structure is clocked but no internal switches are done.

In active state the multiplexer keeps on the feedback again but this time the results from the logic are stored in the register and loop condition is re-evaluated and tested. If the condition is not satisfied the calculation continues otherwise the flip-flops Q3 and Q4 restore the inactive state of the structure.

The switch from inactive to active state is done by the input signal Ready. It sets the input T-flip-flop to \( Q1 = 1 \). Its output \( Q1 \) drives the multiplexer to set its input to Data In on the rising edge of the clock signal. Thus during the current period of the clock signal the logic will produce results which are function of the input data from Data In. At the end of the clock cycle the result is saved in the register. The result is saved on the rising edge of Wr pulse which is identical to the clock pulse in this very moment, selected by flip-flop Q2. This pulse is also used to restore the state of flip-flop Q1, which drives the multiplexer to activate once again the feedback. This happens on the falling edge of the Wr pulse, which means that the result is already saved in the register.

The separation of the moments in time of data saving and multiplexer switching makes sure that there will be no race conditions in the feedback in this initial moment.

At the end of the second clock cycle, after starting the computational process, the flip-flop Q2 restores its state to \( Q2 = 0 \). Flip-flop Q2 is also used to set flip-flop Q3 in state 1.
Flip-flop Q3 is used as mask over the inverted clock signal. The selected pulse sequence is used for saving the intermediate results obtained from the logic circuit in the loop body.

The indication of the end of the computational process and the setting of the structure in inactive state is task of flip-flop Q4. It is set in state one synchronously with the falling edge of the clock signal together with the presence of the valid end of loop condition (CT=1). As a result the flip-flops Q3 and Q4 are set in their initial states. This blocks the clock signal passed to the register and the structure appears to be in inactive state having the result in the register ready to be read.

The concrete logic functions for the control appear to be as follows:

- Data saving in the register:
  \[ \text{DataIn} = (\text{CLK} \cap \text{Q2}) \cup (\overline{\text{CLK}} \cap \text{Q3}) \]. (2.1)

- Flip-flop Q1 sets in state 1 on:
  \[ \text{S1} = \text{Ready} \cap \text{CLK} \]. (2.2)

As we can see on figure 2, the signal \text{Ready} is temporary internally blocked disjunction not(Q2\cup Q3), which is reason for the appearance of second starting pulse. This flip-flop is set to its initial zero state by the pulse:

\[ T1 = Q2 \cap \overline{\text{CLK}} \]. (2.3)

- Flip-flop Q2 is controlled by the pulses:
  \[ T2 = (Q1 \cap \overline{\text{CLK}}) \cup (Q2 \cap \overline{\text{CLK}}) \]. (2.4)

- Flip-flop Q3 is controlled by the pulses:
  \[ T3 = Q2 \cap \overline{\text{CLK}} \]. (2.5)

And is set to initial state either by de signal \text{Reset}, or by the pulse (Q4\cap\text{CLK}), according the logic function:

\[ R3 = \overline{\text{Reset}} \cup (Q4 \cap \text{CLK}) \]. (2.6)

- Flip-flop Q4 is controlled by the pulses:
  \[ T4 = (CT \cap \text{CLK}) \cup (Q4 \cap \text{CLK}) \]. (2.7)

This shortly described process is depicted on the time diagram on picture 3.
3. Experiment

The possibility for practical realization of the proposed, in this work, thesis was experimented using programmable logic. It was used Xilinx ISE 10.1 development environment. The structure was coded in VHDL and Spartan 3 device was used. For the simulation ModelSimXE III 6.2g is used. Because of the specifics of the used device a specific approach was applied, while creating the experimental project [13]. In the end the desired structure was implemented and the algorithm described on picture 2 and picture 3 behaved as expected. The corresponding model behavior is shown on picture 4. The identical behavior of the proposed structure and the model proves the stated thesis of the research.

![Pic. 4 Experiment model time diagram](image)

Conclusion

As with the structures, which were object of the previous authors’ research works, the main complication appeared to be the use of one phase clock generator. The authors believe that the use of multi phase generators will simplify the design of the structure and especially the controlling circuit.

The presented in this paper research complies with the results published by the authors earlier. These results prove the possibility for hardware realization of self-controlled computational structures, free of controlling finite state machine, which makes the authors more confident in their belief that the hardware realization of any arbitrary algorithm as self-controlling is possible. Yet there are algorithmic structures, which are not investigated in the sense of the main authors’ idea.

References

   [http://bis-21pp.acad.bg/events/events.htm](http://bis-21pp.acad.bg/events/events.htm);

[2]. Тянеев Д. С., Колев С. И., Йосифов В., Метод за реализация на апаратни самоуправляващи се циклически структури, Годишник на ТУ-Варна, Юбилеен сборник “45 години ТУ-Варна”, 2007, ISSN 1311-896X, стр. 130-135;


[7]. Sutherland I. E., Micropipelines,


