Synchronous Elastic Architectures: Practical Considerations And MiniMIPS Case Study

Eliyah Kilada and Shomit Das

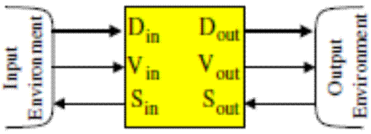


Fig. 1. Interface of an EB with one input and one output channels [2].

***Abstract*—This paper presents practical considerations that should be taken into account during the process of elasticization of any ordinary clocked system. MiniMIPS has been elasticized, synthesized and fabricated as a case study.**

***Index Terms*—Synchronous Elastic Architectures, MiniMIPS.**

I. Introduction

LATENCY Insensitive Design proposed by Carloni et. al [1] is an efficient method to realize a System On Chip (SOC) design based on re-use of Intellectual Property (IP) core modules. This is achieved by using communication channels to transfer data between modules which in turn requires a suitable protocol to be defined. The protocol in question here is the Synchronous ELastic Flow (SELF) protocol proposed by Cortadella et. al. [2]. In this paper, we start by a quick review of synchronous elastic architectures and the SELF protocol in Sections II and III, respectively. Section IV summarizes some of the common practical considerations that should be taken into account during the process of elasticization of any ordinary clocked architecture. A case study, which is MiniMIPS elasticization is then introduced in Section V. Finally, verification, fabrication and performance evaluation are described in Section VI.

II. Synchronous Elastic Architecture

An elastic system comprises elastic modules and elastic channels. Elastic modules are implemented as EBs. Fig. 1 shows the interface of an EB with single input single output channels. An EB is basically modeled as an unbounded FIFO that uses the SELF protocol for handshaking. Abstract specification is shown in Fig. 2.

Elastic channels use two control wires, 'valid' in the forward direction and 'stop' in the backward direction. These wires implement handshaking between the sender and receiver entities similar to the request/acknowledge wires in asynchronous systems. A channel can carry tokens (valid data) and bubbles (invalid data) depending on the state of the wires.

E. Kilada is with the Department of Electrical and Computer Engineering, University of Utah, Utah, USA (e-mail: Eliyah.Kilada@utah.edu).

S. Das is with the Department of Electrical and Computer Engineering, University of Utah, Utah, USA (e-mail: u0614338@utah.edu.

Manuscript received Dec 19, 2008.

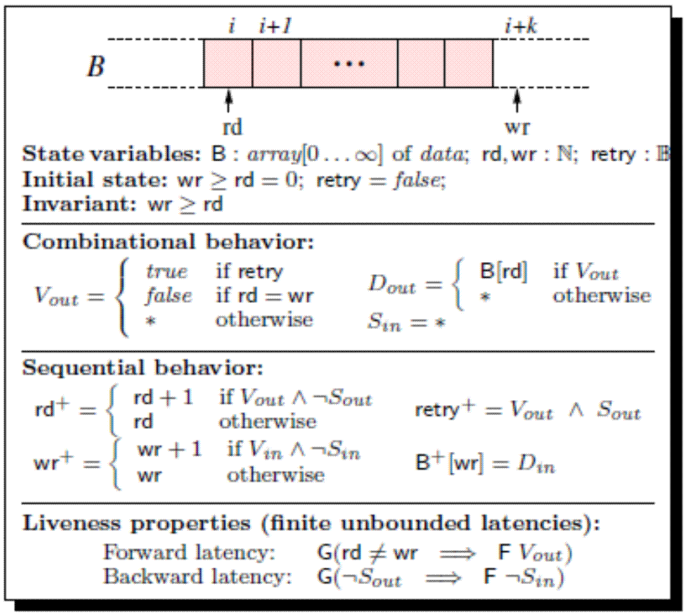


Fig. 2. EB abstract model [2].

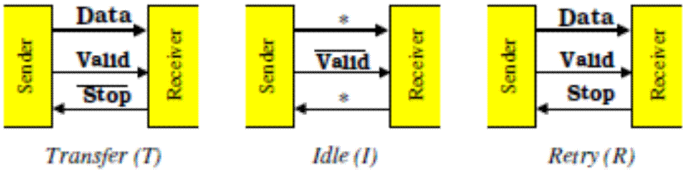


Fig. 3. The SELF protocol [2].

III. SELF Protocol

The two control signals mentioned above, valid (V) and stop (S) determine three possible states in the channel, as shown in Fig. 3.

- *Transfer (V & ˜S)* : The sender provides valid data and the receiver accepts it.
- *Idle (˜V)* : The sender does not provide valid data.
- *Retry (V & S)* : The sender provides valid data, but the receiver does not accept it.

Note that in the Retry state, the sender sustains the valid data until the receiver is able to read it. Hence, the system can tolerate variable communication delays without losing data.

Also, the receiver can issue a Stop signal even if the sender does not provide Valid data.

Elastic buffers implementation we used as well as joins and forks are the same as that described in [2]. We represent a fork as a circle with a dot inside, and a join as a circle with an 'x' inside.

IV. Practical Considerations

In this section we introduce some common practical considerations that should be taken into account during the process of elasticization of any ordinary clocked architecture.

A. Reconvergent Fanouts Between Registers

Consider the reconvergent fanout between two registers A and B in Fig. 4a. Datapath channels are drawn in solid lines while control channel pairs (i.e., stop and valid signals combined) are drawn in dotted lines in the direction of the valid signals. One could be easily tempted to have only one control channel pair (i.e., stop and valid) to represent the three reconvergent paths (i.e., 1, 2 and 3) as shown in Fig. 4a. This would, indeed, work fine, except when a bubble is inserted in one path. The same circuit with a bubble inserted in path 2 is shown in Fig. 4b. Apparently, the control plane doesn't have sufficient information to control the flow of data in paths 1,2 and 3. Hence, this circuit will malfunction. Hence, one control channel pair can not represent more than one datapath channel in presence of bubbles. The safest way to elasticize reconvergent fanouts is to have a separate control channel pair corresponding to each datapath channel. This way, any number of bubbles could be inserted any where. Yet, a more efficient workaround - for the problem of Fig. 4 - is to have only two separate control channel pairs to represent the three datapaths; the first to represent the datapath channel with a bubble (i.e., datapath 2) and the other to represent both datapaths 1 and 3. This is illustrated in Fig. 4c. In general, there is a need for one control channel pair for every datapath where a bubble would be inserted and only one control channel pair to represent all the other datapaths that don't have bubbles.

B. Control Channels Optimizations

Fig. 5a shows a typical example from our case study, MiniMIPS. MiniMIPS has four 8-bit instruction registers, namely, I1, I2, I3 and I4. These four registers are fed one at a time based on the enable signal IRWrite. IRWrite is generated by the MiniMIPS controller. Fig. 5b shows a good elasticization scheme of that circuit. Again, one could be easily tempted to optimize the control channel pairs in Fig. 5b to the scheme shown in Fig. 5c. In Fig. 5c the four fork branches of ExMEM (ExMEM is the control channel pair that corresponds to MemData) as well as the four 2-input joins at the input of the four instruction registers have been replaced by one 2-input join between ExMEM and C (C is the control channel pair that corresponds to the MIPS controller output IRWrite). Fig. 5c will, indeed, work fine but it suffers from a serious problem if bubbles are to be inserted. What if a bubble is required to be inserted just before the D input of I3 (i.e., at wire x shown in Fig. 5a For the circuit in Fig. 5b, the bubble will be inserted at wire x1. Yet, for the circuit in Fig. 5c, there is no place in the control plane that corresponds to x. One may say that x2 would be the right place to insert the bubble, no? That's wrong! A bubble at x2 means a bubble in the datapath of both ExMEM and C to I3, while a bubble at x, means a bubble only in the datapath of ExMEM to I3, while C-I3 datapath is not affected. Again, more caution is required while trying to optimize control channels especially when bubbles are expected to be inserted.

C. Memory And Register File Elasticization

From the control point of view, both memory and register file in a microprocessor could be considered as combinational units [2]. Yet, one problem could arise in the following situation. Suppose that the write enable port is active but invalid (i.e., the valid control signal of that port is inactive) , and at the same time, there is invalid data available at the write data port. There is no way for the ordinary memory or register file to prevent this invalid data from being stored. Hence, it is a must that the write enable port of these storage elements be ANDed with the valid signal of the relevant write ports. This way, we can prevent invalid data storage.

D. Synthesis Of Elastic Buffers

It has been observed that the synthesis of elastic buffers by design compiler results in a serious issue. For both of the low and high latches of the synthesized EB, the G input is always delayed with respect to the D input. Therefore, before G turns off, D changes its value. Hence, latches latch the wrong values. This caused the synthesized EB's to fail when sdf back annotation is taken into account.

To work around, the elastic buffer controllers have been protected from being optimized during synthesis. This is done by annotating them with set_size_only attribute.

V. Case Study: MiniMIPS

MIPS (Microprocessor without Interlocked Pipeline Stages) is a 32-bit architecture with 32 registers, first designed by Hennessy [3]. The MiniMIPS is an 8-bit subset of MIPS. It is fully described in [4]. MiniMIPS uses an 8-bit datapath. Only 8 registers are implemented and the program counter (PC) is also 8-bit long. Being a RISC architecture, the MIPS Instruction Set is less taxing to implement.

In this Section, we use the MiniMIPS as a case study of elasticization. Each functional module in the architecture is treated as an IP core and elastic buffers (EBs) are used as handshaking interfaces. This allows for tolerance in variability in the computation and communication delays between modules. Fig. 6 shows the ordinary clocked version of the MiniMIPS. The control plane of the elastic clocked version of MiniMIPS is shown in Fig. 7.

From the elastic control point of view, even the MiniMIPS controller signals (e.g., RegWrite, IRWrite, ..etc) are considered part of the data plane and they need their own control channels. Mapping between datapath signals in the ordinary

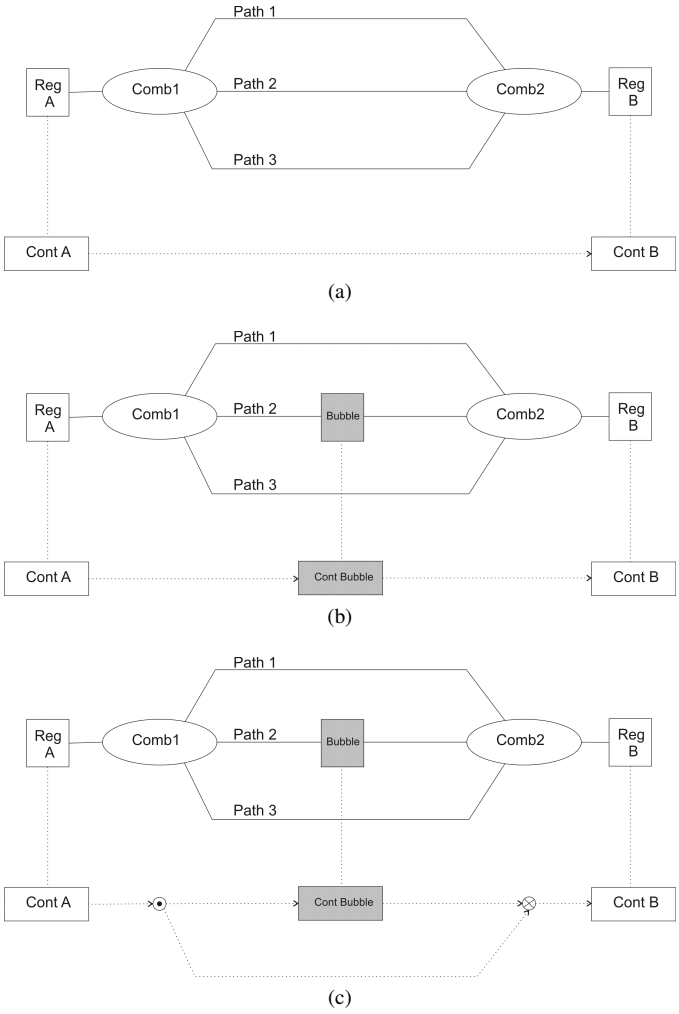


Fig. 4. Elasticization of reconvergent fanouts.

clocked MiniMIPS and the control channel pairs in the elastic clocked MiniMIPS should be self explanatory for most signals. It must be noted that some optimization have been made in the control channels and, hence, bubbles are not allowed to be inserted in some places. For example, IorD, MemRead and MemWrite have one common control channel pair. Therefore, bubbles can't be inserted at any of these signals - as explained in Section IV-A. RFWrite, in Fig. 7, is the register-file-write control channel pair. RFWrite_valid must be active if data are going to be written in the register file as explained in Section IV-C. Therefore, RFWrite_valid has been ANDed with RegWrite inside the register file. To demonstrate the correctness of the elastic architecture, several bubbles have been inserted in the design (they are not shown in Fig. 7). Sample positions are:

- 3 Bubbles in the memory (i.e., at ExMEM).
- 2 Bubbles just after the AluOut (i.e., L) Register.
- 3 Bubbles just before the A register.

VI. Verification, Fabrication And Performance Evaluation

Both the ordinary clocked and elastic (without bubbles) MiniMIPS have been synthesized, placed and routed and fabricated. The UofU standard cell library for the 0.5 μm process has been used.

Images of the fabricated chips are shown in Fig. 8. The functionality has been verified using the testbench specified in [4] on LV514 tester. Both ordinary clocked and elastic MiniMIPS chips had the following characteristics, at 5V supply:

- Minimum clock period is less than or equal to 20 ns. In fact, this is the minimum permissible clock period on LV514 tester.
- The *adr* output port delay is 10.5 ns.

Schmoo plots for the clock cycle and *adr* output port delay are shown in Fig. 9. The schmoo plots are the same for both chips (i.e., ordinary clocked and elastic MiniMIPS).

References

[1] L. Carloni, K. Mcmillan, and A. L. Sangiovanni-VincentelliR, "Theory of latency insensitive design," in *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 20, no. 9, Sep 2001, pp. 1059–1076.
[2] J. Cortadella, M. Kishinevsky, and B. Grundmann, "Synthesis of synchronous elastic architectures," in *ACM/IEEE Design Automation Conference*, July 2006, pp. 657–662.
[3] J. H. et al., "The MIPS Machine," in *COMPCON*, 1982, pp. 2–7.
[4] N. Weste and D. Harris, *CMOS VLSI design: a circuit and systems perspective*, 2004.

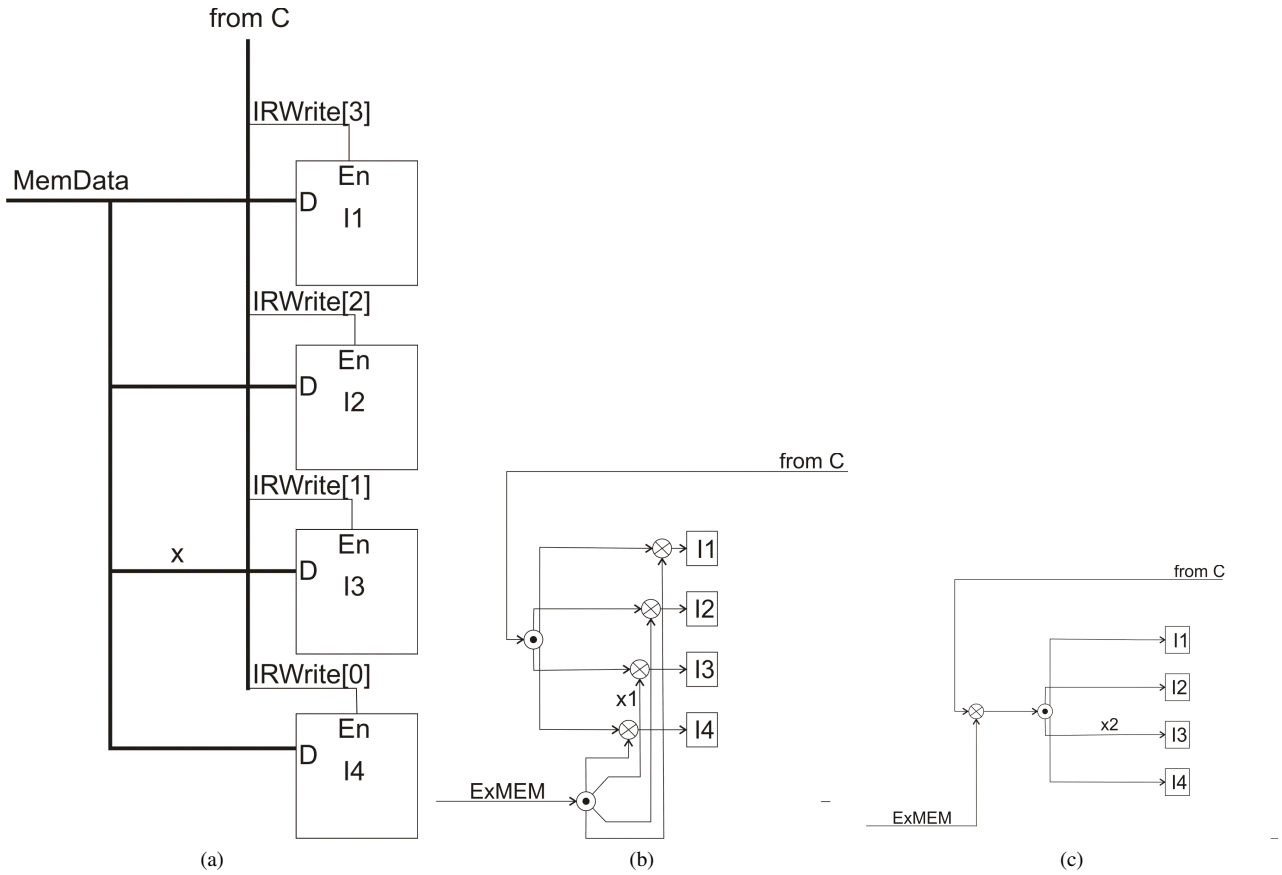


Fig. 5. Control channels bad-optimization example.

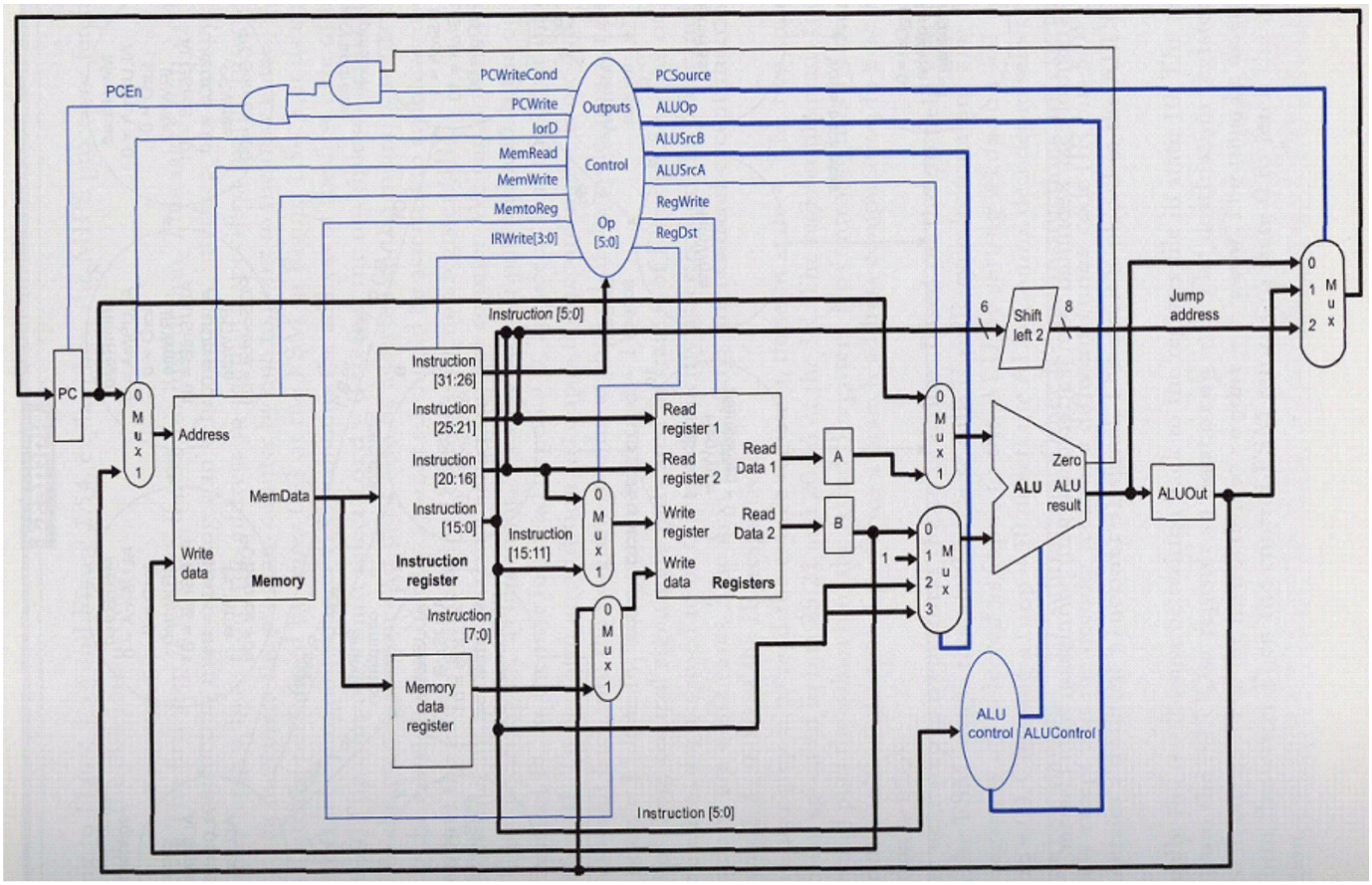


Fig. 6. Block diagram view of the ordinary clocked MiniMIPS [4].

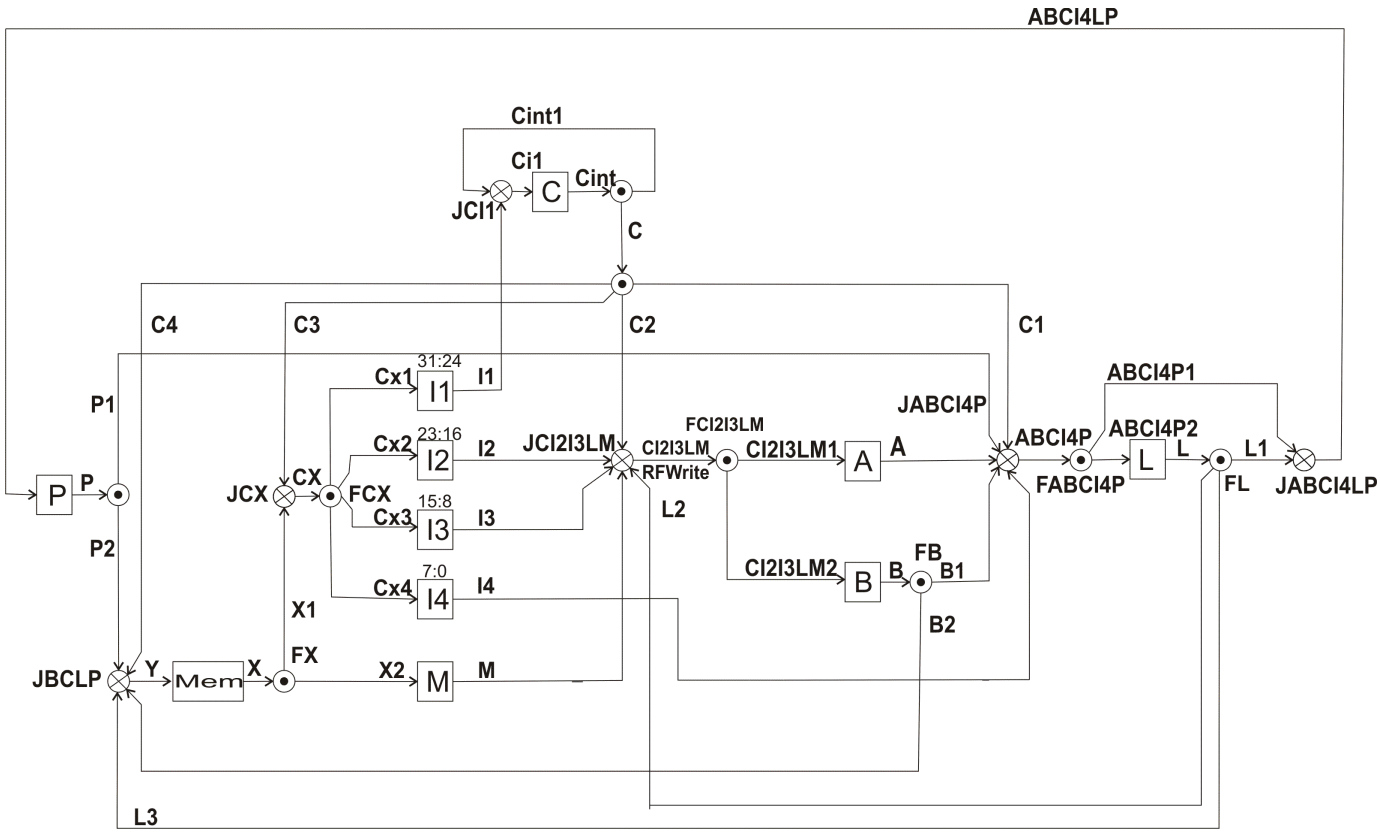
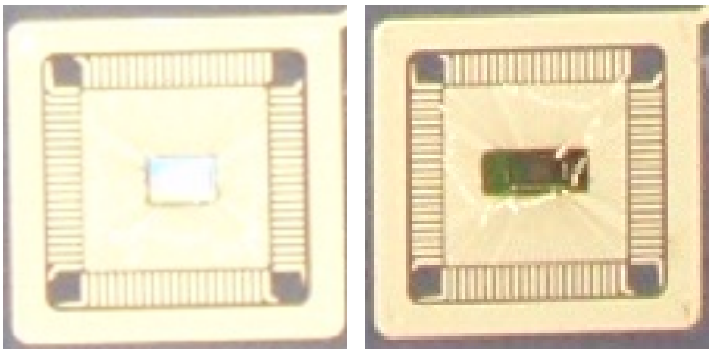
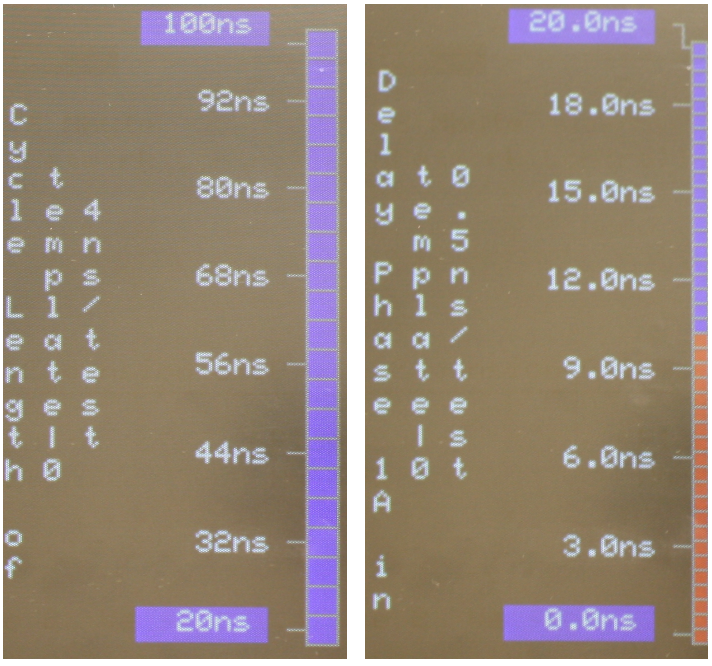


Fig. 7. Control network of the elastic clocked MiniMIPS.



(a) Ordinary clocked MiniMIPS chip. (b) Elastic clocked MiniMIPS chip.

Fig. 8. Fabricated chips images.



(a) Schmoo plot for the clock cycle. (b) Schmoo plot for the *adr* output port delay.

Fig. 9. Fabricated chips schmoo plots. Schmoo plots are the same for both ordinary clocked and elastic MiniMIPS.Red boxes are for failed tests, while blue are for passed ones.