Implementation of Fast CRC Calculation

Tomas Henriksson and Dake Liu

Dept. of Electrical Engineering, Linköpings universitet
SE-581 83 Linköping, Sweden
Tel: +46-13-28{8956, 1256}, Fax: +46-13-139282, e-mail: {tomhe, dake}@isy.liu.se

Abstract - CRC is important for error detection in communication systems. With transmission speeds of several Gb/s the high-speed implementation is a bottleneck. A circuit with two parallel calculation units has been implemented in a 0.35 micron process. They use 32 bits and 64 bits parallel input respectively. Chip measurements prove throughput higher than 5.76 Gb/s, which indicates that 10 Gb/s throughput is possible in more modern processes.

I. Introduction

Cyclic redundancy check (CRC) is widely used for error detection in communication protocols, such as Ethernet and ATM. With the increase in network speed to several Gb/s the CRC computation becomes a bottleneck in the system implementation.

The basic CRC algorithm is specified in a bit-serial way, but it has long been known that parallel implementations offer better performance [1]. This comes to the cost of larger silicon area.

CRC is a general method for detecting errors and the algorithm can be instantiated with different length and characteristic polynomials. For Ethernet and ATM the same 32 bit long polynomial is used. That is

$$g(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+ \\ +x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x^{1}+1$$

Therefore it is most interesting to make high speed implementations for that specific instance of the CRC algorithm. The standards specify that the implementation must start by resetting the state register and then perform a polynomial division. That requires complex combinational logic in the critical loop and in [2] it was shown that the implementation can be made simpler and thus faster by setting the state register to a modified start value and then use simpler combinational logic in the critical loop. This way also requires that 32 zeros are fed to the calculator after the actual data in order to get the correct value.

In this paper an implementation of that algorithm using 32 bit and 64 bit parallel input is described. Chips have been manufactured and measured results are presented. Section II describes the implementation and section III stresses the special features of the design. The results are presented and discussed in section IV and finally conclusions are drawn in section V.

II. Implementation

The circuit was implemented by using RTL descriptions in VHDL and synthesis by Cadence Ambit Buildgates. Place and Route was done in Cadence Silicon Ensemble and physical verification was performed by the Cadence DIVA tool. The manufacturing process was AMS 0.35 micron process. The standard cell library was supplied by Europractice.

The VHDL code consists of 5 parts, as shown in Fig. 1. The input registers capture the 36 input signals every positive clock edge. There are 4 lanes each consisting of one control bit and 8 data bits, so the interface is similar to the XGMII of the 10 Gigabit Ethernet. The implementation can handle any number of bytes in the packet.

The input8_2 unit splits the data to 64 data bits in parallel and generates an enable signal for the crc8 unit. The crc4 unit calculates the CRC by adding 4 new bytes every clock cycle and the crc8 unit calculates the same CRC by adding 8 new bytes every second clock cycle. The purpose of making two implementations on the same chip is just to be able to compare them.

The output multiplexer selects the output from crc4 or crc8, so only one of them can be observed at a time.

III. Design Features

Both calculation units can handle a last word of data that contains any number of bytes. This is managed by actually having several computation units in parallel and selecting the correct value based on the input control signals and by having a finite state machine, that controls the computations.

There is another possible solution to this problem, which is to use several 8 bit input computation units that are connected in series. Both alternatives were considered and synthesis has been made for both of them, but only the prior one was selected for chip fabrication. The alternatives for crc4 are explained in Fig. 2 and Fig. 3. For crc8 the same principle is used. The second alternative (Fig. 3) has longer critical path than the prior one and therefore cannot handle as high throughput, on the other hand it consumes smaller silicon area. For this chip throughput was more important than silicon area and therefore the architecture in Fig. 2 was selected.

The equations for the CRC calculation with different input widths were generated by a C-program, which takes only a

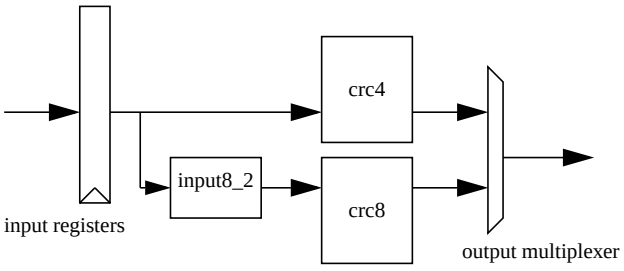

Fig. 1. Block diagram that shows the structure of the implementation.

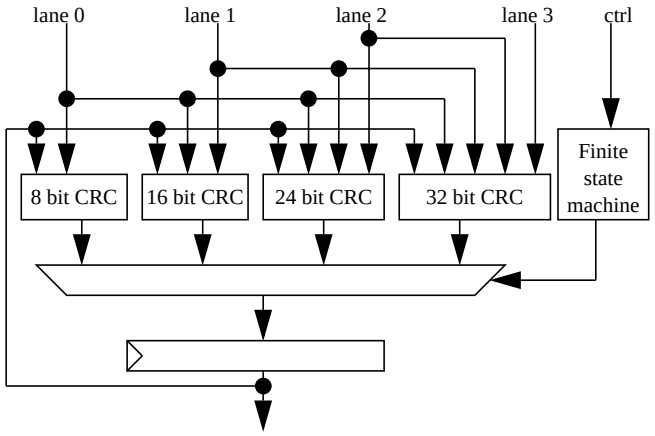

Fig. 2. The architecture selected for implementation

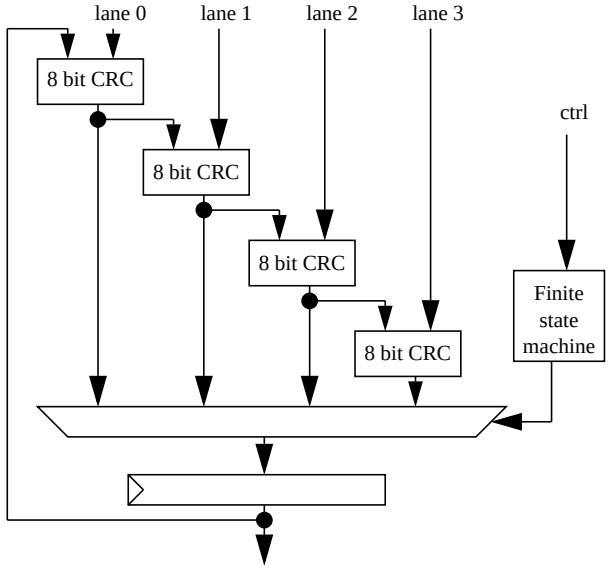

Fig. 3. The alternative architecture, not selected for implementation

polynomial as the input. This was necessary since there are 128 equations required for the crc8 unit. Actually units with up to 20 bytes input in parallel were also generated, but these were too complex to be implemented efficiently.

In order to improve the throughput of the circuit, some techniques were used. First, flip-flops in the state register that have high fanout were cloned. Second, the most critical paths were identified and logic was pushed over the register boundary to relax the critical paths. This, of course, had to be compensated for at the output.

The observability of the circuit is further improved by having the possibility to select internal states with the output multiplexer. This is not shown in Fig. 1, but the output multiplexer has actually 4 inputs of 32 bits each, 2 of those are the CRC values from crc4 and crc8 respectively. The other 2 are built up by internal states which can be used for testing purposes.

IV. Results and Discussion

The total chip implemented in 0.35 micron technology uses 7.73 mm^2 silicon area, see Fig. 4. It has 84 pads and is pad limited. The area of the core is 2.87 mm^2.

3 chips have been mounted on specially designed test boards and measurements were conducted with Agilent 16700 logic analysis system. The limitation in the testing was the testing equipment, which can only supply 180 MVectors/s. The results at supply voltage of 2.5 V are shown in table I. Because 32 bits are processed every clock cycle a throughput of 5.76 Gb/s is managed. All three chips operated at 180 MHz for supply voltage from 2.2 V to 2.7 V. The crc8 circuit on chip #1 did, however, not work correctly.

Why the chips did not work as fast at supply voltages above 2.7 V is hard to explain. The crc4 on chip #1 operated correctly at 130 MVectors/s at 3.3 V supply voltage, which is the nominal supply voltage for the process.

Overall the manufactured chips worked better than expected. Static timing analysis estimated a maximum throughput of 5.5 Gb/s for crc4 and 5.2 Gb/s for crc8, using typical process parameters in typical operating conditions.

The reason why crc8 is estimated to support less throughput than crc4 is that it uses an enable signal to control the operation and therefore has to use more complex flip-flops.

V. Conclusions

A CRC chip has been designed and manufactured. More than 5.76 Gb/s throughput can be achieved in a 0.35 micron process. The results thereby indicate that 10 Gb/s throughput will be possible in more modern processes.

References

[1] T.-B. Pei and C. Zukowski, "High-speed parallel CRC circuits in VLSI", *IEEE Transactions on Communications*, Vol. 40, pp. 653-657, April 1992.

[2] R. J. Glaise and X. Jacquart, "Fast CRC Calculation", *Proceedings of IEEE International Conference on Computer Design: VLSI in Computers*, pp. 602-605, 1993

TABLE I Measurement results

Chip #	crc4 throughput	crc8 throughput
1	5.76 Gb/s	Failure
2	5.76 Gb/s	5.76 Gb/s
3	5.76 Gb/s	5.76 Gb/s

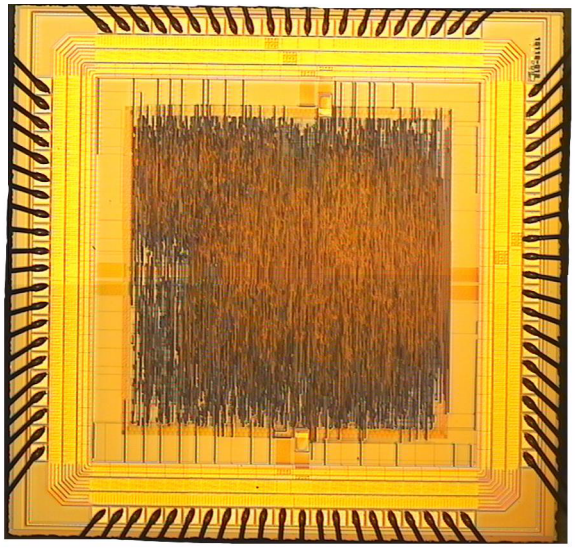

Fig. 4. Chip photo. The chip area is 7.73 mm^2.