VLSI LAYOUT GENERATION OF A PROGRAMMABLE CRC CHIP

Sadiq M. Sait and Mohammed Shahid K. Tanvir
Department of Computer Engineering
King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia

Abstract

VLSI layout generation of a **programmable** *CRC chip with a CRC of 16-bits is presented. The hardware of CRC generator is specified in a hardware description language (HDL). The hardware compiler and functional level simulator of HDL are used for logic synthesis. The second stage of the compilation process generates a netlist of logic gates. The netlist so produced is translated to RNL compatible netlist by a* **translator** *program. The layout subsystem of* **VPNR** *is used to generate the VLSI layout of the programmable CRC chip from RNL netlist. The design rules and technology files of MOSIS are used. The layout is viewed in* **MAGIC** *layout editor and simulated by* **irsim** *at transistor-level. The CRC chip can be used in a number of applications. These include areas such as data communications for error detection and correction, digital system testing for test pattern generation and signature analysis, and mass storage devices for parallel information transfers.*

1 Introduction

Data integrity is of prime importance in storage and transmission applications. Some of the commonly used approaches for error detection in data communications include checksums, parity checks, longitudinal redundancy code (LRC), and cyclic redundancy check (CRC) code. Out of these, CRC is very popular and is widely used because it can detect:

- all one or two bit errors
- all odd number of bit errors
- all burst errors less than or equal to the degree of the generator polynomial used
- most burst errors greater than the degree of the polynomial used

CRCs can be mathematically computed by modulo-2 division. CRC computation can be implemented both in software as well as hardware. It is possible to compute CRC code by considering a bit, a byte or even a word of the message at a time. Several algorithms based on this idea have been reported in literature. Traditionally, the CRC computations are implemented in hardware by means of a simple shift register [13], with exclusive-or (EXOR) logic gates. We have used such a shift register arrangement as shown in Figure 1. It is popularly known as linear feedback shift register (LFSR), which handles one bit of the data stream at a time.

In the following sections, we discuss the design considerations in brief, the modeling and simulation in *Universal AHPL* and the process of VLSI layout generation. The switch/transistor level simulation results are presented. The advantages of VLSI implementation and the various applications of the CRC chip are discussed. The design approach described can also be used for VLSI implementation of a parallel CRC generator. We conclude by a brief discussion on parallel CRC generation.

2 Design Considerations

Several software implementations of CRCs have been reported in literature. In [10], an emphasis has been given on parallel implementation of the code. In [11], an attempt has been made to emulate the hardware process of CRC generation in software. A byte-wise algorithm has been proposed based on look-up table techniques. The hardware implementation is preferred for several reasons namely:

(1) bit-wise software implementations are very slow, their applicability is limited only to low encoding rates and they introduce a considerable amount of delay before delivering data.
(2) byte-wise or word-wise software implementations are normally based on table look-up methods. These algorithms have a large memory and considerable CPU time requirements.
(3) hardware implementation is fast, simple and easy to realize.

It can be concluded from the results of [10], [11] and [12] that a software implementation is not feasible

Contributed Paper
Manuscript received September 8, 1993

for high speed information transfers. Thus, hardware implementation offers the necessary solution. A few hardware implementations of CRCs have also been reported in literature [1], [8]. In these implementations, an emphasis has been given on VLSI realization to exploit the high transmission bandwidth of the communication media. In the next section, we present the implementation details of a programmable CRC generator realized in VLSI.

3 Implementation

The hardware description languages (Verilog, CDL, DDL, AHPL, VHDL, ISPS etc.) have been used successfully for documentation, communication and verification [4]. They have also been used as input specification languages to design automation (DA) systems which synthesize VLSI layouts [7]. We have used UAHPL (derived from AHPL), a RTL HDL for modeling the 16-bit programmable CRC generator. UAHPL facilitates structural specification of a design. The hardware compiler and functional simulator of UAHPL are used to perform logic synthesis. A wire/gate list generated is fed to a translator program to obtain a netlist. This netlist is used by VPNR, a layout sub-system of OASIS to generate the layout. OASIS is a cell-based silicon compiler that enables the design of semi-custom testable integrated circuits [2], [5], [6]. The UAHPL model of the CRC generator, the process of translating wirelist to netlist and the layout generation are discussed in detail in the following sections.

Any sequential circuit can be considered to be consisting of finite state machines (FSM). Our CRC generator is a 3-state FSM as shown in Figure 2(a). It has following characteristics:

- based on internal Exclusive-Or (IE) type LFSR
- simple to design and easy to implement
- highly programmable
- enables efficient realization in VLSI

A CRC generator is said to be *programmable* if it can generate CRCs for various generator polynomials having same or different degrees. In our implementation, the CRC generator can produce 16-bit CRC pattern for 2^{15} different generator polynomials of degree 16. The programmability is achieved by controlling the **EXOR** gates in Figure 1 by the inputs p1, p2, p3 etc. as shown in Figure 2(b). The presence or absence of an EXOR gate corresponds to the presence or absence of a term in the divisor polynomial. A vector **VEC** of 16-bits is declared as a *bus* in UAHPL model, discussed in the following section. A logic-1 at any of these lines means that the corresponding EXOR gate in Figure 1 is present and a logic-0 means that the EXOR gate is absent. This

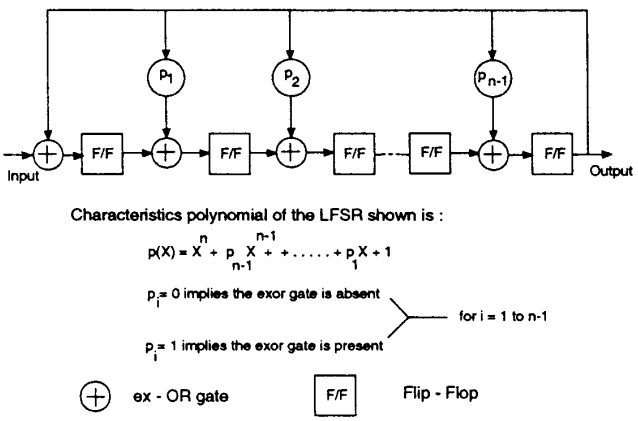

Figure 1: Shift register implementation of CRC generator

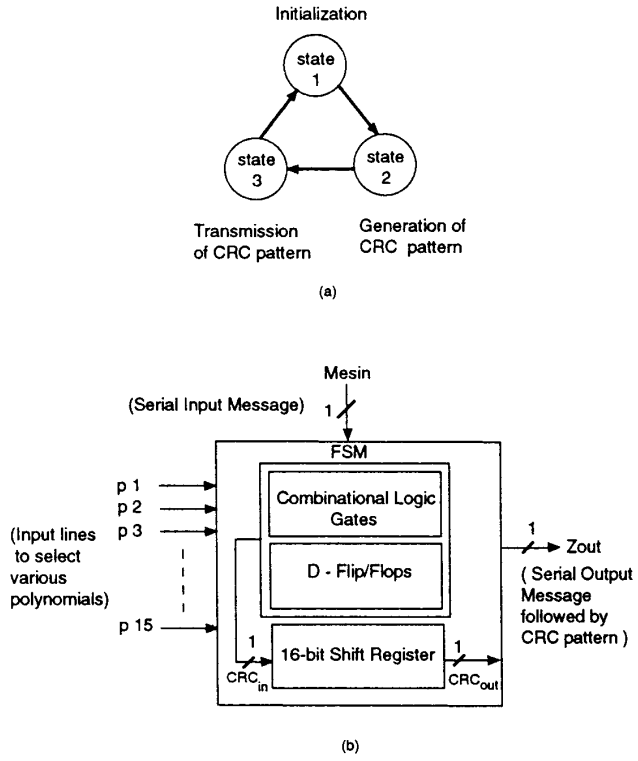

Figure 2: (a) State diagram of programmable CRC generator. (b) Structure of programmable CRC generator.

```
MODULE:CRCGENERATOR.
MEMORY:CREG{16};COUNT{6}.
BUSES: X{6};Y;ZOUT;CRCRDY.
EXBUSES: Z;COUT;VEC{16}.
EXINPUTS:CLK;RESET;START.
EXINPUTS:MESIN.
CLUNITS : INC{6}
BODY SEQUENCE: CLK.
     1 COUNT ⇐ 6$0;
     "initialize counter and CRC register"
     CREG ⇐ 16$0;
     ⇒ ↿(START)/(1)
     "goto 2 if start is high"
     2 ZOUT=MESIN;
     Y=MESIN@CREG{15};
     COUNT⇐X;
     CREG ⇐Y,CREG{0:14}@(VEC{0:14}&Y);
     "shift and Exor to get CRC"
     ⇒ ↿(& /COUNT)/(2).
     "goto 3 if all bits of count are high"
     3 COUNT ⇐X;CRCRDY= \1\;
     "CRC is ready"
     CREG ⇐ \0\,CREG{0:14};
     "append the CRC pattern to message"
     ZOUT=CREG{15};
     ⇒ ↾(& /COUNT {2:5 }),COUNT{1}})/(3,1).
ENDSEQUENCE
     CONTROLRESET(RESET)/(1);
     X=INC(COUNT);
     Z=ZOUT;
     COUT=CRCRDY.
END.
```

Figure 3: UAHPL description of programmable CRC generator.

way, any desired generator polynomial of degree-16 can be specified. Thus, besides modulo-2 division, CRCs for many popular generator polynomials of degree 16 such as CRC-16, CRC-SDLC (IBM,CCITT), CRC-16 reverse and CRC-SDLC-16 reverse can be generated.

4 UAHPL model of CRC generator

The UAHPL description of CRC generator is shown in Figure 3. The declaration of inputs, outputs, internal and external busses, memory elements/registers, control logic unit (CLU) and the clock precedes the actual body of the CRC generator. The code corresponding to the three states of the FSM is enclosed within **BODY SEQUENCE** and **END** keywords.

The initialization is done in the 1^{st} STATE. In the 2^{nd} STATE, a 64-bit message is supplied sequentially on line ZOUT and a 16-bit CRC pattern is simultaneously generated and stored in a register, CREG. When all the message bits have been processed, the CRC pattern is ready by that time and it can be serially appended to the message data stream for transmission. The CRC pattern is serially appended to the message in 3^{rd} STATE. The generator then goes to the initial state to perform the same task described for another message, if any.

The 16-bit CRC pattern generated is available in 66^{th} clock pulse. The transmission of 16^{th} bit takes place in 81^{st} clock pulse. Thus, the number of clock cycles required to generate and transmit any CRC in our implementation, is the sum of the size of the message and the degree of the generator polynomial used.

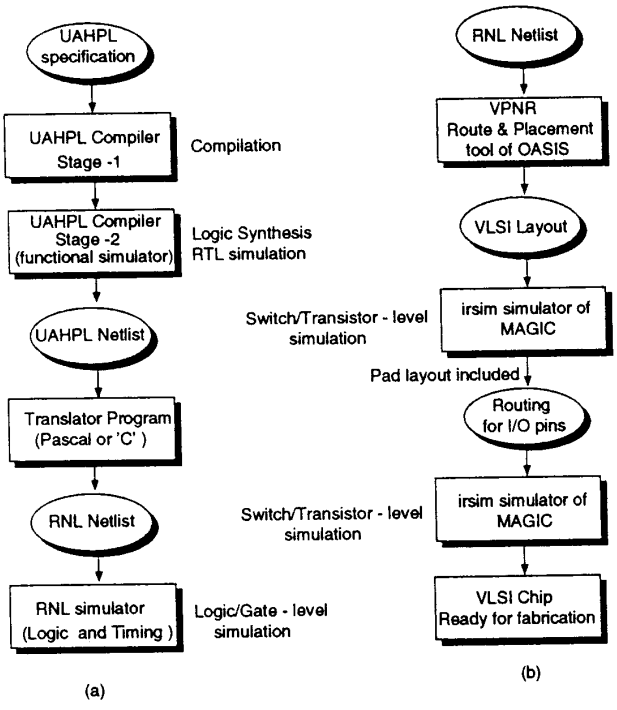

Figure 4: (a) Translation of UAHPL model to RNL netlist. (b) Translation of RNL netlist to Layout.

5 Layout Generation and Verification

5.1 Logic Synthesis

A hardware compiler and a functional-level simulator were used to compile and simulate the UAHPL model of CRC generator respectively. After the UAHPL code is compiled by Stage-1 compiler, it is simulated at the RTL level. The Stage-2 compiler is used for

logic synthesis. It generates a logic netlist of the hardware circuit. This netlist consists of combinational logic gates and three types of D flip-flops (set, reset and control flip-flops) with enable and asynchronous set and reset inputs. A translator program written in 'C', translates the UAHPL netlist to RNL compatible netlist form which can be simulated by **RNL**, a timing and logic-level simulator [14]. This is illustrated in Figure 4(a).

5.2 Logic/Gate-level Simulation

The **RNL** netlist was simulated at the logic/gate-level by performing **RNL** simulation and the results were observed on **SigView**, a simulation (signal) viewing program.

5.3 Translation of netlist to layout

The standard cell library of OASIS consists of scalable **CMOS** cells compatible with 2μ SCMOS technology of MOSIS [9]. The layout sub-system of **OASIS** called **V**anilla **P**lace a**N**d **R**oute (VPNR) generates the standard cell layout from the RNL netlist. VPNR uses a library of pre-designed standard cells to make the layout. It also incorporates testability by including scan path based testing circuitry in the layout. It also performs consistency checking of the routing and placement phases of the design. The process of translation of netlist to layout is illustrated in Figure 4(b). The layout of the CRC chip is shown in Figure 6. The area of the chip is $2252 \times 2222\ \mu m^2$. It has over 3,000 transistors.

5.4 Transistor-level Simulation

A simulation was performed to verify that the layout works. The circuit was extracted from the layout using **MAGIC**'s hierarchical circuit extractor. We simulated the layout at the transistor (switch) level using *irsim*, a switch-level simulator of **MAGIC** layout editor for CMOS circuits [3]. The simulation results were observed on another tool called **ana**, a timing analyzer. The transistor-level simulation results were compared with the RTL/gate-level simulation results and verified. The **MOSIS** pad frame was included in the layout. The design rules and technology files of MOSIS have been used. The interactive routing for the I/O pins of chip was done with MAGIC. The chip was simulated again with *irsim*. The timing diagram results were compared again to RTL/gate-level and the transistor-level simulation results performed earlier to make sure that the chip will function properly when fabricated. It is observed that the simulation results are in conformity with the functional level simulation. This verifies the design methodology and greatly enhances the chances that the chip will work after fabrication. The transistor-level simulation results of the layout obtained using *irsim* are shown in Figure 5. The 64-bits message consisting of all 1's is given serially on pin Mesin. The generator polynomial selected, generates CRC-CCITT of the given message. The CRC pattern generated is 'A6E1' (in HEX) and is appended serially after the data message as shown in Figure 5. The CRC chip has been simulated at a speed of about 600 MHz.

6 Applications

The proposed programmable CRC chip can be used in various applications. These include

- error detection and correction in data communications
- pseudo-random test pattern generation
- signature analysis
- adaptable to support multiple input shift register (MISR) implementation
- modifiable to generate parallel CRC

The VLSI implementation of CRC chip has many advantages. It has a very small area and high speed coupled with flexibility. It can be easily integrated into existing or future communication systems. Such a highly programmable chip can be placed for data compression in disk controllers, I/O channels and protocol processors. The placement of VLSI chips in protocol processors will greatly reduce the data transmission cost. Likewise, the VLSI chip will improve the access and retrieval times of disk controllers. The single chip CRC is also very attractive from the stand-point of built in self test (BIST). In a conventional method of testing, large storage is needed to store many test responses. In BIST, an approach called compact testing based on CRC codes, is used to avoid this high storage requirement by storing the compressed responses. The length of the response data is equal to the degree of the generator polynomial.

We discuss the design of parallel CRC generator in the next section.

7 Design of Parallel CRC Generator

The LFSR based CRC generator discussed has certain drawbacks. First of all, the size of the message handled and the CRC pattern generated is fixed. Secondly, it can not fulfill the requirements of very high speed (HS) data transfers. Finally, it needs a parallel to serial converter. With the advancement of the transmission technology, the bandwidth of the

transmission media has increased by several orders of magnitude. In view of the increasing speed of transmission links, it is very important to detect single and burst errors of the transmitted message by sophisticated error-detection/correction techniques. This is true specially for HS fiber optic links and digital systems having parallel information transfers such as microprocessor bus, computer network interfaces and storage devices. To achieve error correction/detection of HS systems, we need to implement some parallel algorithm for CRC generation in hardware.

A parallel CRC generator works as follows. It takes the first N-bits of the input message and generates an intermediate CRC corresponding to the N-bits processed. After this, the next N-bits of the message alongwith the intermediate CRC pattern previously generated will be processed. This process will continue until all the data bits of the message have been processed. This method does not take into account the size of the message. Infact, the message size need not be known in advance. The correct CRC code corresponding to the message transmitted upto any moment of time, is always available. The VLSI realization of a parallel CRC generator can work at a very high speed and it can be used to generate CRC for any generator polynomial and any data message. It does not need a parallel to serial converter. The VLSI implementation is highly attractive for HS fiber optic links and parallel information transfers. It can be shared by several transmission lines. Considering simplicity and reduced design time of the approach presented, we plan to implement the parallel CRC in VLSI.

8 Conclusion

In this paper, we presented VLSI layout generation of a programmable CRC generator from its UAHPL description. The design and implementation details were discussed. Besides a small area and achieving a high speed, the CRC chip is highly programmable. It can be used in various data communication and data compression applications. Simulation has been carried out at various levels to verify the design process and ensure that the chip will work after fabrication. The chip consists of 27 I/O pins and is fully testable. The **CIF** file obtained from the layout is ready to be sent for fabrication. The approach discussed can be used to translate any UAHPL model to a VLSI layout corresponding to its hardware description. It reduces the design time and enables VLSI realization of semi-custom integrated circuits in a very simple, efficient and straightforward manner. The chip is being fabricated at **ORBIT Semiconductor Inc.**, USA.

References

[1] Guido Albertengo and Riccardo Sisto. Parallel CRC Generation. *IEEE Micro*, 10(5):63–71, October 1990.

[2] F. Brglez, D. Bryan, J. Calhoun, and R. Lisanke. Automated Synthesis for Testability. *IEEE Transaction on Industrial Electronics*, 36(2):263–277, May 1989.

[3] Robert N. Mayo et.al. DECWRL/Livermore Magic Release, Digital Western Research Laboratory, September 1990.

[4] Fredrick J. Hill. AHPL: Then and Now. *IEEE Design and Test of Computers*, pages 73–75, June 1992.

[5] Gershon Kedem, Franc Brglez, and Krzysztof Kozminski. ASIC Design with OASIS. *Proceedings IEEE/ISCAS*, 4(4):2580–2583, 1990.

[6] Gershon Kedem and Krzysztof Kozminski. A Standard Cell Based Silicon Compiler. *IEEE 1986 Custom Integrated Circuit Conference*, pages 120–124, May 1986.

[7] M. Masud and Sadiq M. Sait. Universal AHPL- a language for VLSI Design Automation. *IEEE Circuits and Devices Magazine*, September 1986.

[8] Amar Mukherjee, M. A. Bassiouni, and N. Ranganathan. Improving Bandwidth of Communication Controllers. *IEEE International Conference on Communications*, 3(3):1390–94, 1988.

[9] Open Architecture Silicon Implementation Software, MCNC.

[10] A. K. Pandeya and T. J. Cassa. Parallel CRC lets many lines use one circuit. *Computer Design*, 14(9):87–91, September 1975.

[11] Aram Perez, Wismer, and Becker. Byte-wise CRC Calcutations. *IEEE Micro*, 3(3):40–50, June 1983.

[12] T. V. Ramabadran and S. S. Gaitonde. A tutorial on CRC Computations. *IEEE Micro*, 8(4):62–75, August 1988.

[13] William Stallings. *Digital Data Communication Techniques*. Macmillan Publishing Co., 1987.

[14] VLSI Design Tools Reference Manual Release 3.1, NW Laboratory for Integrated Systems FR-35 , University of Washington, February 1987.

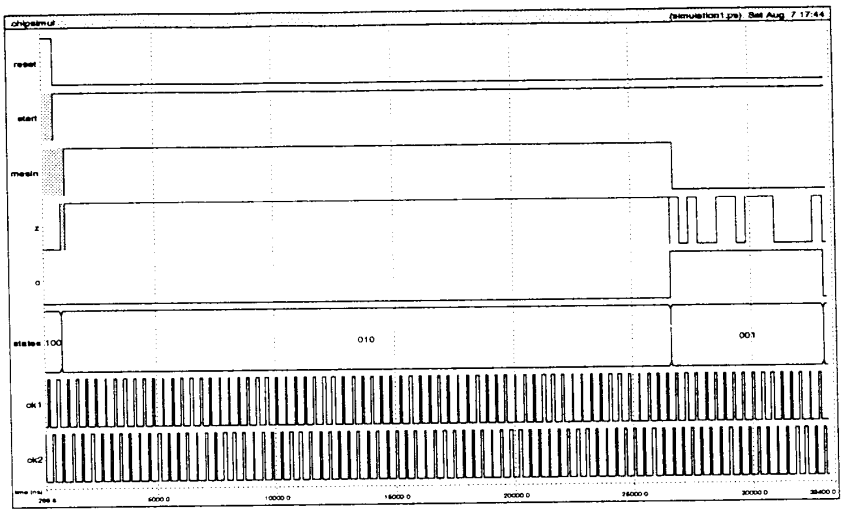

Figure 5: Transistor-level Simulation Results of CRC-CCITT

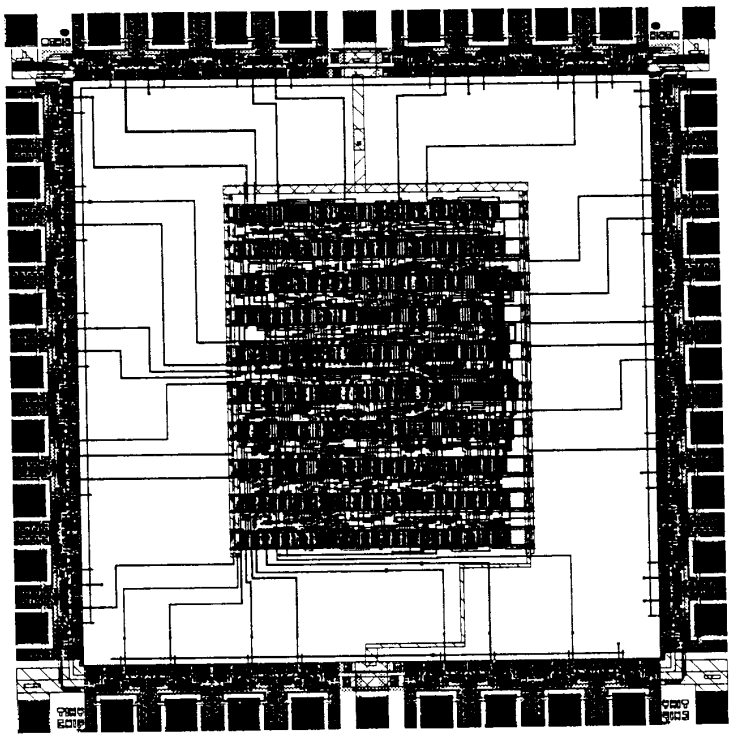

Figure 6: Layout of 16-bit programmable CRC Generator Chip