Digital Devices and Techniques

Implementation of a Programmable Cyclic Encoder

Gavin Cameron

MSc/PGD Networks and Data Communication

June 2, 1999

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

ABSTRACT

This report contains the practical implementation of a programmable cyclic encoder.

The original design document is attached as an appendix to this document and covers the theoretical design of the encoder.

The design was captured and simulated using Mentor Graphics. Circuit diagrams and Timing waveforms are attached as appendices to this document.

INTRODUCTION

Error correcting codes are used to correct errors when messages are transmitted through a noisy communications channel.

The (n,k) codes chosen for the design are two Bose-Chaudhuri-Hocquenghem (BCH) codes for (15,7) and (15,5) codewords.

This equates to 15 bits of *codeword* containing either 7 or 5 bits of *data* and 8 or 10 bits of *parity* respectively. The two modes are selectable via a discrete signal.

DESIGN OF THE PROGRAMMABLE ENCODER

The theoretical design of the programmable cyclic encoder was completed as part of the digital devices and techniques module (see appendix 3 for the submitted report).

This report is only concerned with the implemented design and simulation of the encoder.

The two generator polynomials used were:

(15,7) code:

$$G(x) = x^8 \oplus x^7 \oplus x^6 \oplus x^4 \oplus 1$$

(15,5) code:

$$G(x) = x^{10} \oplus x^8 \oplus x^5 \oplus x^4 \oplus x^2 \oplus x \oplus 1$$

MENTOR GRAPHICS

The circuit was captured and simulated using Mentor Graphics v8 IDEA tools.

The tools used for the work were:

1. dmgr

   design manager. This tools allows control of a design, launches design tools, archives designs and takes care of configuration management of the designs.

2. da_LMS

   design architect (Library Management System). This is the schematic capture tool. It allows the user to draw symbols (of components) and connect them together with nets (wires). The LMS allows for tight regulation of the library components that users are allowed access to at various stages of design.

3. dve

   design viewpoint editor. This tool configures the design to be viewed from any *viewpoint*. This means that simulation work requires certain information and would have a viewpoint, PCB design requires different information and would have a different viewpoint. Through viewpoints, parts of the design can be masked off from the viewer to allow for block level simulation, for example. The viewpoint allows for back-annotation of any changes made to the design at, for example, PCB layout stage, without affecting the original design.

4. quicksimII

   quick simulation 2. This tool is for digital simulation. It looks at the design through the *viewpoint* created with *dve* and simulates what it sees. It can use a number of *models* to achieve this, for example, simple logic models like AND / XOR, VHDL models, or models of actual devices like an SN74F74.

SCHEMATIC CAPTURE

Refer to Appendix 1 for circuit diagrams.

It was felt that in order to simplify the layout of the design, that the design should be of a hierarchical structure. This allowed blocks of the circuit to be designed and assigned a symbol that could be used in an upper layer of the design. The symbol and the underlying circuit have exactly the same number and names of inputs and outputs.

As an example, the gen_lib (generic library) D-Type flip-flops used: the Preset input and !Q output were not required, so the preset was pulled high, and the !Q un-connected on the underlying circuit. The other connections were connected to *port_in* and *port_out* symbols which is the method for transferring signals through a hierarchical structure.

The CRC encoder part of the design is entered almost exactly as shown in figure 4 of Appendix 3, although tri-state buffers were used instead of a multiplexor for selecting which output to feedback into the circuit depending on mode of operation. This was to make the circuit easier to read, it has no effect on function.

The input polynomial (as well as going through the CRC circuit) goes through a bank of registers which are independent of the CRC circuit. This is to allow the real data from the input polynomial to delayed such that it is outputted directly before the CRC.

The selecting of output data source is via a multiplexor. The MUX is controlled by a counter with a programmable output. When the counter block is in mode (15,7), the output *DataNParity* is low for the first 15 (*n)* counts and high for an additional 8 (*n-k*). Similarly, when the counter block is in mode (15,5), *DataNParity* is low for the first 15 counts and high for an additional 10. This allows the output multiplexor to select the delayed data for the first *k* bits and the CRC for the last *n-k* bits.

In order to remove any hazards on the output caused by the count selector switching between counts, the output of the selector was registered, hence delaying the signal by 1 clock. The result of which is that the output selector was set to detect the numbers 14-22 for the (15,7) mode and 14-24 for the (15,5) mode (giving 15-23 and 15-25 respectively).

Also, when the *n-k* parity bits are being fed out of the CRC circuit, *DataNParity* disables the feedback path, as feedback at this stage will corrupt the remainder polynomial.

The logic required to provide a logic high for the correct sequence of counts was calculated from Karnaugh maps as follows:

Figure 1 - Karnaugh Map for (15,7) Selector

count1,count0 \ count3,count2	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	1	0	0	0
10	1	0	0	0

It should be noted that *count4* = 1 (logic one) for these numbers (16-22), but cannot be shown on a two dimensional Karnaugh map. The numbers 14 and 15 are selected by the equation:

$$Select = \overline{count4\,count3\,count2\,count1}$$

From the map, it can be seen that the equation required for the output is:

$$Select = count4\left(\overline{count3}\left(\overline{count2} + \overline{count1}\right)\right)$$

Hence, the select equation for (15,7) mode is:

$$Select = \overline{count4\,count3\,count2\,count1} + count4\left(\overline{count3}\left(\overline{count2} + \overline{count1}\right)\right)$$

Figure 2 - Karnaugh Map for (15,5) Selector

count1,count0 \ count3,count2	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	1	1	0	0
10	1	1	0	0

Again, *count4* = 1 and cannot be shown on a two dimensional Karnaugh map.

The equation for this mode is therefor:

$$Select = \overline{count4\,count3\,count2\,count1} + count4\,\overline{count3}$$

The first term of the both equations are the same, so that saves on the number of gates required as both equations can be fed from the same gate.

The output of the circuit is registered again to remove any hazards when the Data / Parity selector switches, hence delaying the output by a further clock cycle.

It was discovered in preparation for simulation, that there was a licencing problem with some of the simulation models, in particular, the model used for the 4 bit counter. The simplest way around this was to create a counter from gen_lib components. As speed is not a consideration when simulating gen_lib components, a ripple counter was constructed. This is not the most effective type of counter, but this demonstrates the flexibility of an ECAD system: when the licensing problem is fixed, the hashed together ripple counter can be replaced with a real component.

OPERATION OF THE ENCODER

The circuit must be initially reset with *ResetN* which is active low, to clear out any unwanted bits from the feedback loop that could result from a power up condition.

The signal *FiveSevenN* selects the mode and must remain static throughout the encoding sequence. A logic low selects (15,7) mode and a logic high selects (15,5).

The pre-scaled input polynomial is applied at the *Input* signal, most significant bit first.

The data is clocked through with the *Clock* signal. This is active on the rising edge, hence the input data must be present before that edge. Since gen_lib parts were used, setup and hold times are of no concern as they are all 0. However, if actual parts were used, then the setup and hold times of the registers must be adhered to or the circuit may fail.

For the first 15 (*n*) clock cycles, the polynomial is divided and also delayed by *n-k* cycles, hence the *k* data bits emerge from the output after *n-k* clock cycles (separate path from the CRC). After *n* bits, the feedback path is disabled and the output becomes the remainder polynomial *R(x)* of *P(x)* / *G(x)* for the last *n-k* bits. This requires a further *n-k* clock cycles.

The cycle can then start again for the next word.

SIMULATION OF THE ENCODER

The design was prepared for simulation using the *design viewpoint editor* to allow the simulation to see all the models associated with each symbol.

For each of the two modes, the same example input polynomials as discussed in appendix 3 were used, these were:

(15,7) mode:

$$P(x) = x^5 \oplus x^4 \oplus x \oplus 1$$

(15,5) mode:

$$P(x) = x^4 \oplus x^2 \oplus 1$$

These were pre-scaled by clocking in 0's for the last *n-k* bits of the message.

Refer to Appendix 2 for the timing waveforms of the simulation runs. As can be seen from the simulations, the output of the CRC generator combined with the data word, produces the following:

(15,7) mode:

$$T(x) = 011001111101100$$

$$= x^{13} \oplus x^{12} \oplus x^9 \oplus x^8 \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^3 \oplus x^2$$

(15,5) mode:

$$T(x) = 101011001000111$$

$$= x^{14} \oplus x^{12} \oplus x^{10} \oplus x^9 \oplus x^6 \oplus x^2 \oplus x \oplus 1$$

Which are the same as the calculated results in Appendix 3, hence proving the operation of the circuit.

CONCLUSION

The CRC generator functions as per requirements.

There were two errors from the initial design.

1. The feedback path originally was enabled permanently, causing corruption of the remainder polynomial *R(x)* and hence the CRC.
2. The direct data had to be delayed by *n-k* cycles (instead of just 1 as initially thought) to allow it to emerge from the output just before the CRC.

It was discovered after completion of the circuit that the extra *n-k* clock cycles could have been eliminated by feeding the input into the back end of the linear feedback shift register instead of the front end. This has the bonus of providing the pre-scaling of the input polynomial at the same time as dividing it by the generator equation *G(x)* so that the circuit would only require *n* clock cycles instead of *n+(n-k)* as present.

REFERENCES

1. Theory and Practice of Error Control Codes, Richard E. Blahut, Addison Wesley Publishing Company, 1984.

2. A First Course in Coding Theory, Raymond Hill, Clarendon Press, Oxford, 1988.

3. A Commonsense Approach to the Theory of Error Correcting Codes, Benjamin Arazi, MIT Press, Cambridge, 1987.

APPENDIX 1 - Circuit Diagrams

Figure 3 - CRC encoder top level circuit diagram

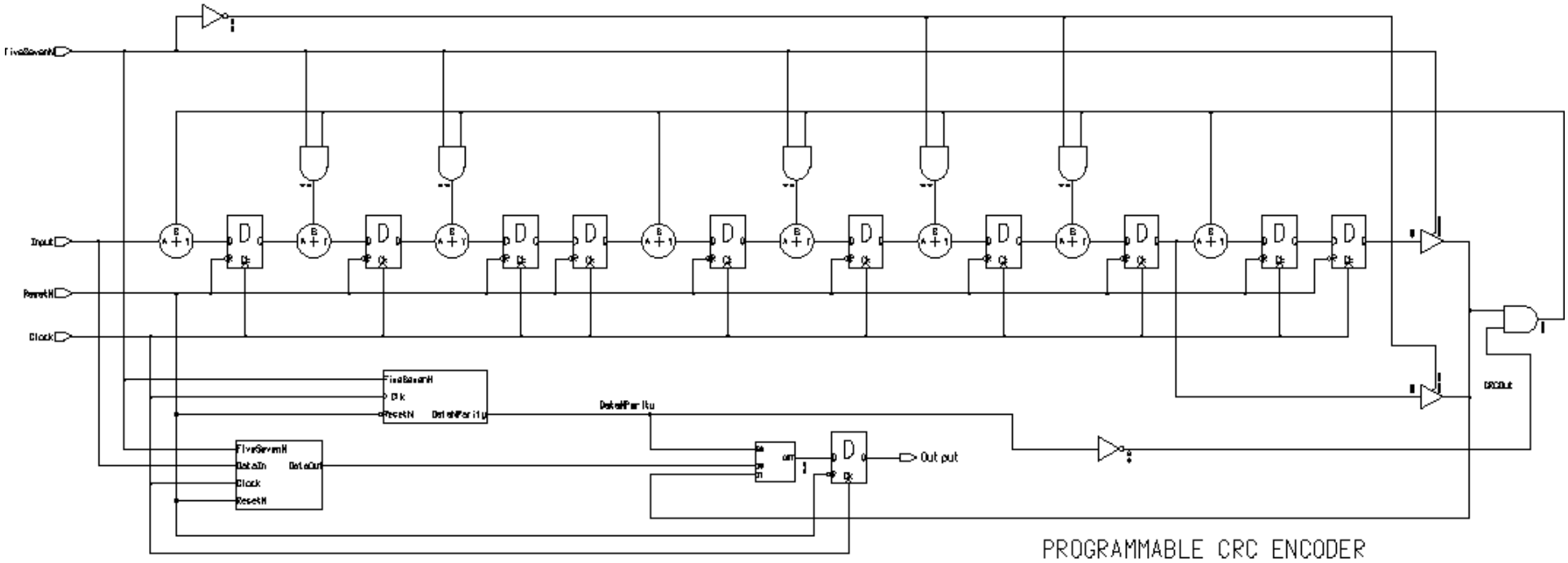


Figure 4 - DataNParity Selector

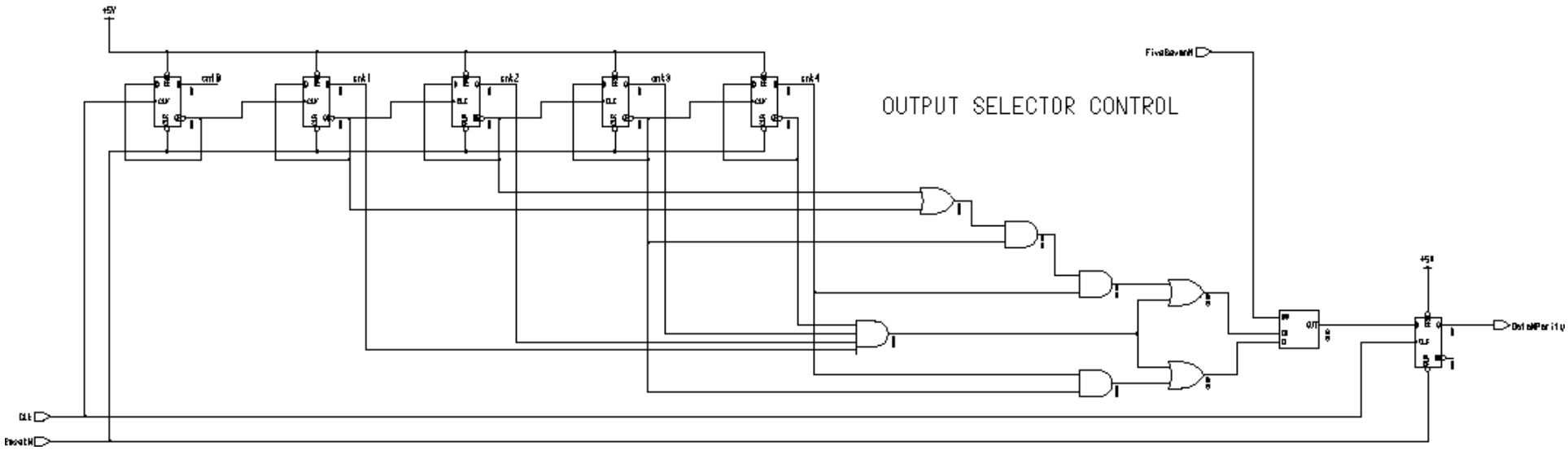
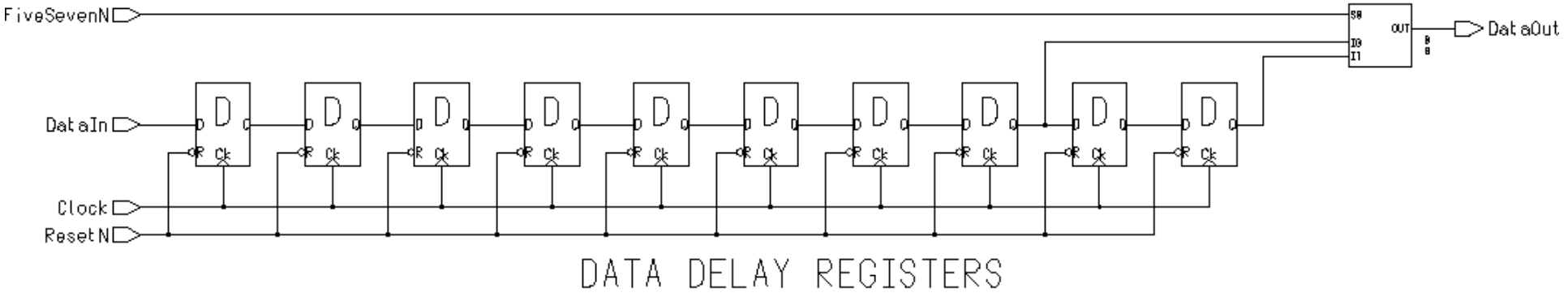


Figure 5 - Data Delay Registers



APPENDIX 2 - Timing Diagrams

Figure 6 - Simulation run of (15,7) code

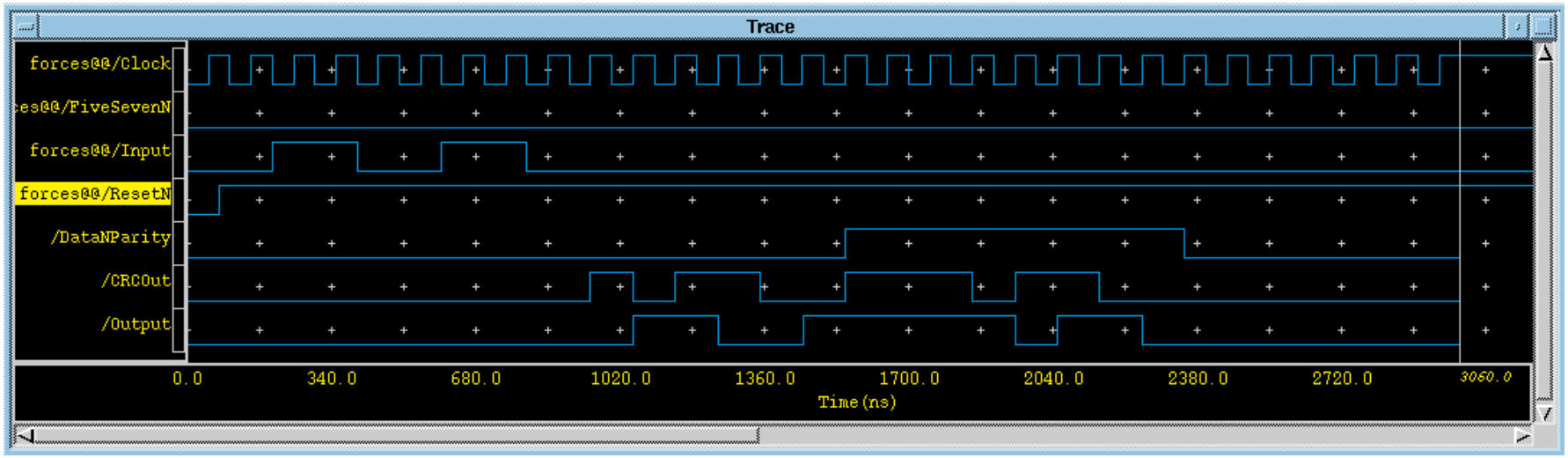
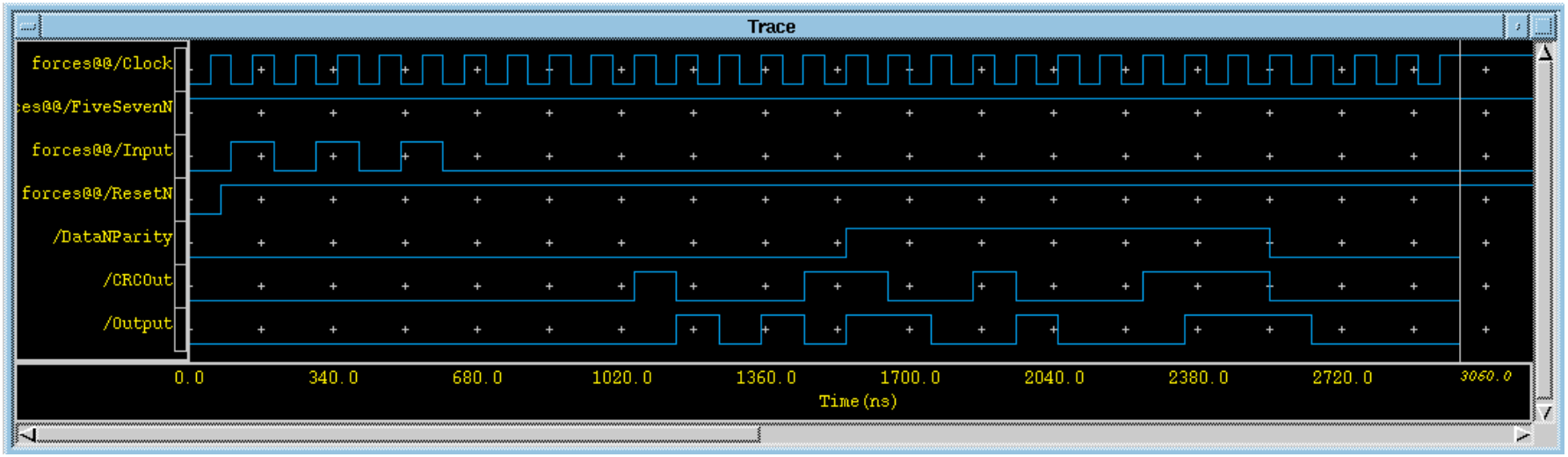


Figure 7 - Simulation run of (15,5) code



APPENDIX 3 - Design of Programmable Cyclic Encoder