CSC8501 Coursework 1 – 2015
Convolutional Encoding
Due 28th October 2016 at 10am

Specification (what you need to do): You will build a computer program in C++ that generates a number of convolutional encoders from given design constraints and then demonstrate your computer program's behaviour.

Design constraints:

- The encoder used by your software is non-recursive, limited to three registers and two XOR gates.
- All encoder elements must be used (i.e., you can't leave out some elements).
- 2 outputs will be generated for each step of encoding and such outputs will emanate from each XOR gate (not from the registers). Therefore, XOR gates should be labelled XOR1 and XOR2: the output bit from XOR1 preceding the output bit from XOR2 in the overall output of the encoder.
- After an encoding step, and before the next encoding step proceeds, all registers shift right one place (registers are in series). The bit from the rightmost register is discarded and the input bit becomes resident in the leftmost register.
- The encoder registers always start in a zero state.

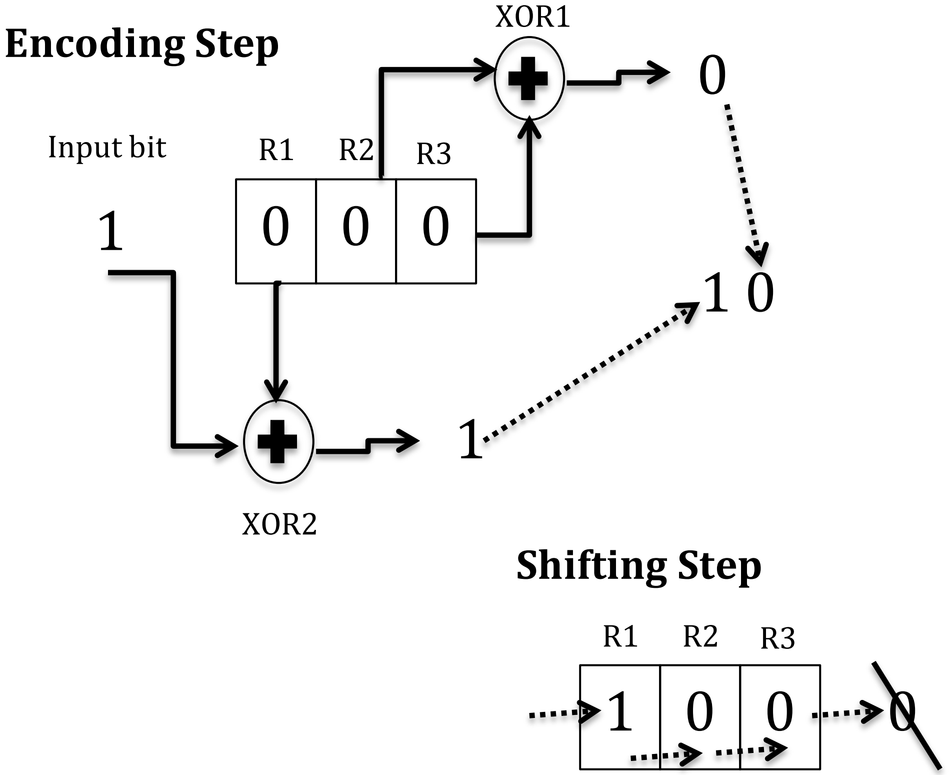

Figure 1 – Diagram showing encoding and shifting step of a valid encoder (just a sample encoder, there are many other variations).

Deliverables (what we want to see):

- C++ source code authored by the student (*submit to submission system*)
- Executable file containing solution (*submit to submission system*)
- Output files produced by encoders and their comparisons (*for demonstration only*)

Demonstration (your chance to explain and show your solution):

On Friday 28th October from 10am onwards students will demonstrate their solutions.

Learning Outcomes (what we expect you to demonstrate in a general way)

- Be capable of designing and creating programs.
- Realise inappropriate/appropriate usage of programming languages.
- Understand how to manage memory.
- To be able to create and use data structures.
- To be able to use condition statements, loops and functions.

Marking Scheme (what is worth what):

Marks are out of 25 and are awarded as follows:

- 10 Marks for achieving correct output
- 5 Marks for appropriate file input and output
- 5 Marks for user interface design
- 5 Marks for adherence to the 7 rules of programming (see lecture 1)

Additional direction (making the task clearer):

- **Task 1**: Construct a computer program that will generate a series of encoders. This set of encoders should reflect ALL permutations of encoders that are possible from the design constraints provided.
    - **If you can't create ALL permutations then create as many as you can. You need at least one to do the second coursework.**
- **Task 2**: Allow each encoder to consume all the data provided in the supplied text file and record the output of each encoder using a text file. Each encoder's output should be in a separate text file.
    - **You will be given the input data (a text file containing 0s and 1s)**.
- **Task 3**: Write a test program that reads in two encoder outputs and compares them.
- **Task 4**: Identify encoders that produce the same output.
- **Task 5**: Compare the validity of your encoder against the same encoders produced by other class members. As everyone should have the same set of encoders this should be straightforward and requires some collaboration between class members.
    - **Are results reproducible across different class members and if they are not then why not (this is part of the testing process)?**

Set by: Graham.Morgan@ncl.ac.uk