**BRIEF**

<insert>

David Nakasone

davenakasone@gmail.com

This is 1 of 2 circuits.

SERIAL IN, SERIAL OUT

It is designed to input 200 unencrypted 16-bit words and output 200 16-bit words that encrypt the original input.

In each word, the input represents a 7-bit ASCII character that the sender wants to transmit, starting with the LSB of the 16-bit word. The remaining bits are 0.

The 16-bit word is encrypted on each rising clock edge.

For encryption, the 16-bit word just adds { 2^15 + 1957 } = \*\*34725\*\* to the input, producing an output. In binary, the key is 16'b 1000 0111 1010 0101

for example, 'A' input = ASCII 65 (decimal), so input is [ 0000 0000 0100 0001 ]

and the input is encrypted to (decimal) 34725 + 65 = 34790, so the output is [ 1000 0111 1110 0110 ]

the process continues until all 200 words have been encrypted

if the message is less than 200 words, the NULL character will be encrypted

The test bench stimulates this circuit by acting as the buffer for an arbitrary plain text input stream.

It is what the sender used to input his message.

Using these 200 unencrypted 16-bit words as stimuli, the input is changed between clock edges.

As the stimuli provide input to the DUT and output is recorded by the test bench, this represents the point at which the receiver gets the encrypted message.

Since the transmission was encrypted, the received data will not be coherent unless the receiver has the proper KEY.

See "20200429 decryptor #UNLV" here: https://www.edaplayground.com/x/2j7c for the circuit 2 of 2 that is able receive and decrypt a message from this circuit.

\* The sender and receiver (both parties) should each have an encryptor and decryptor circuit so they can both transmit and receive

\* The encryption method is customizable to virtually anything, but both parties must use the same encrypt and decrypt method

\* This code is synthesizable at the RTL level and can be targeted to ASICs or FPGAs

\* Icarus Verilog 0.9.6, YOSIS ABC produces map and netlist/cell library

\* Logic synthesis on front-end and place/route on back-end were successful

</insert>

**DESIGN**

<insert>

module encryptor (clock, abort, charIn, charOut, readyTX); // end of port list

input clock; // gets the 16-bit character every positive edge

input abort; // cease and clear when = 1, synchronus

input [15:0] charIn; // input as 16-bit ASCII character with only 7 bits used

output reg [15:0] charOut; // output as 16-bit encrypted word

output reg readyTX; // when circuit is ready to transmit, "readyT" = 1

parameter s0 = 0, s1 = 1, s2 = 2, s3 = 3; // states

reg [1:0] ps, ns; // present state, next state registers

initial ps = 2'b00; // start in state 0

reg [7:0] counter = 8'b0; // count to 199, reflects when all 200 lines encrypted, state control

//reg [15:0] carray = 16'b0; // takes input value to a register...for verification, but going to need more states

always @ (posedge clock)

begin

if (abort)

begin

ps <= s0; // go to state 0 if abort = 1

charOut <= 16'b0; // output is set to 0

readyTX <= 0; // not ready to TX

end

else ps <= ns; // otherwise go to next state

end

always @ (charIn, ps) // when present state or input changes

begin

case (ps)

s0: begin

charOut = 16'b0; // output is 0

readyTX = 0;

ns = abort==0 ? s1 : s0; // if about is 0, encryption can begin

end

s1: begin

charOut = charIn + 16'b1000011110100101; // encrypt current character

counter = counter + 1; // increment counter

if (counter == 8'b11001000) ns = s2; // once 200 characters encrypted, proceed

else ns = s1; // keep encrypting if 200 characters have not been encrypted

readyTX = 0;

end

s2: begin

readyTX = 1'b1; // ready to transmit encrypted message

#100; // use counter or wait time to reflect all bits leaving as serial TX

// good spot for a confirmation signal

ns = s3; // go to the "done state"

end

s3: begin

readyTX = 1'b1;

charOut = 16'b0; // clear transmit buffer

//charIn = 16'b0; // clear input buffer

ns = s0; // stand by for next

end

endcase

end

endmodule

</insert>

**TB**

<insert>

//`timescale 1ns / 100ps // unit in ns....#1 is a 1ns delay 10% error not synthesizable

module TBencrypt;

reg clock; // toggled for clock pulse

reg abort; // starts in 1, 0 to encrpyt, made = 1 to clear and abort

reg [15:0] charIn; // the 16-bit representation of the input character, see input buffer

wire [15:0] charOut; // the 16-bit representation of the encrypted input character, output

wire readyTX; // after 200 characters encrypted and placed in buffer, this starts serial TX

integer counter;

encryptor DUT (clock, abort, charIn, charOut, readyTX);

always #10 clock = ~clock; // set clock

initial // initial block, holds serial input of characters

begin

clock = 1'b0; // start it at 0

abort = 1'b0; // proceed to encrypt

counter = 3'd0; // just for display

$display (" TIME(ns) index value(encrypted)");

#5 charIn = 16'b0000000000000000; // stops Mealy lag...starts the offset, but THROW AWAY, not part of TX

#20 charIn = 16'b0000000001001001; counter = counter + 3'd001; // input 1

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 2

#20 charIn = 16'b0000000001101000; counter = counter + 3'd001; // input 3

#20 charIn = 16'b0000000001100001; counter = counter + 3'd001; // input 4

#20 charIn = 16'b0000000001110110; counter = counter + 3'd001; // input 5

#20 charIn = 16'b0000000001100101; counter = counter + 3'd001; // input 6

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 7

#20 charIn = 16'b0000000000110011; counter = counter + 3'd001; // input 8

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 9

#20 charIn = 16'b0000000001110100; counter = counter + 3'd001; // input 10

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 11

#20 charIn = 16'b0000000001101110; counter = counter + 3'd001; // input 12

#20 charIn = 16'b0000000001110011; counter = counter + 3'd001; // input 13

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 14

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 15

#20 charIn = 16'b0000000001100110; counter = counter + 3'd001; // input 16

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 17

#20 charIn = 16'b0000000001100111; counter = counter + 3'd001; // input 18

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 19

#20 charIn = 16'b0000000001101100; counter = counter + 3'd001; // input 20

#20 charIn = 16'b0000000001100100; counter = counter + 3'd001; // input 21

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 22

#20 charIn = 16'b0000000001110100; counter = counter + 3'd001; // input 23

#20 charIn = 16'b0000000001101000; counter = counter + 3'd001; // input 24

#20 charIn = 16'b0000000001100001; counter = counter + 3'd001; // input 25

#20 charIn = 16'b0000000001110100; counter = counter + 3'd001; // input 26

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 27

#20 charIn = 16'b0000000001001001; counter = counter + 3'd001; // input 28

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 29

#20 charIn = 16'b0000000001101110; counter = counter + 3'd001; // input 30

#20 charIn = 16'b0000000001100101; counter = counter + 3'd001; // input 31

#20 charIn = 16'b0000000001100101; counter = counter + 3'd001; // input 32

#20 charIn = 16'b0000000001100100; counter = counter + 3'd001; // input 33

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 34

#20 charIn = 16'b0000000001111001; counter = counter + 3'd001; // input 35

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 36

#20 charIn = 16'b0000000001110101; counter = counter + 3'd001; // input 37

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 38

#20 charIn = 16'b0000000001110100; counter = counter + 3'd001; // input 39

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 40

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 41

#20 charIn = 16'b0000000001110000; counter = counter + 3'd001; // input 42

#20 charIn = 16'b0000000001101001; counter = counter + 3'd001; // input 43

#20 charIn = 16'b0000000001100011; counter = counter + 3'd001; // input 44

#20 charIn = 16'b0000000001101011; counter = counter + 3'd001; // input 45

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 46

#20 charIn = 16'b0000000001110101; counter = counter + 3'd001; // input 47

#20 charIn = 16'b0000000001110000; counter = counter + 3'd001; // input 48

#20 charIn = 16'b0000000000101110; counter = counter + 3'd001; // input 49

#20 charIn = 16'b0000000000001010; counter = counter + 3'd001; // input 50

#20 charIn = 16'b0000000001000010; counter = counter + 3'd001; // input 51

#20 charIn = 16'b0000000001100101; counter = counter + 3'd001; // input 52

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 53

#20 charIn = 16'b0000000001100001; counter = counter + 3'd001; // input 54

#20 charIn = 16'b0000000001110100; counter = counter + 3'd001; // input 55

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 56

#20 charIn = 16'b0000000001001101; counter = counter + 3'd001; // input 57

#20 charIn = 16'b0000000001100101; counter = counter + 3'd001; // input 58

#20 charIn = 16'b0000000001101100; counter = counter + 3'd001; // input 59

#20 charIn = 16'b0000000001101100; counter = counter + 3'd001; // input 60

#20 charIn = 16'b0000000001101001; counter = counter + 3'd001; // input 61

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 62

#20 charIn = 16'b0000000001000010; counter = counter + 3'd001; // input 63

#20 charIn = 16'b0000000001100001; counter = counter + 3'd001; // input 64

#20 charIn = 16'b0000000001101110; counter = counter + 3'd001; // input 65

#20 charIn = 16'b0000000001101011; counter = counter + 3'd001; // input 66

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 67

#20 charIn = 16'b0000000000110011; counter = counter + 3'd001; // input 68

#20 charIn = 16'b0000000000110101; counter = counter + 3'd001; // input 69

#20 charIn = 16'b0000000000101110; counter = counter + 3'd001; // input 70

#20 charIn = 16'b0000000000110110; counter = counter + 3'd001; // input 71

#20 charIn = 16'b0000000000111000; counter = counter + 3'd001; // input 72

#20 charIn = 16'b0000000000110110; counter = counter + 3'd001; // input 73

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 74

#20 charIn = 16'b0000000000110101; counter = counter + 3'd001; // input 75

#20 charIn = 16'b0000000000110001; counter = counter + 3'd001; // input 76

#20 charIn = 16'b0000000000101110; counter = counter + 3'd001; // input 77

#20 charIn = 16'b0000000000110011; counter = counter + 3'd001; // input 78

#20 charIn = 16'b0000000000111001; counter = counter + 3'd001; // input 79

#20 charIn = 16'b0000000000110011; counter = counter + 3'd001; // input 80

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 81

#20 charIn = 16'b0000000001101110; counter = counter + 3'd001; // input 82

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 83

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 84

#20 charIn = 16'b0000000001101100; counter = counter + 3'd001; // input 85

#20 charIn = 16'b0000000001100001; counter = counter + 3'd001; // input 86

#20 charIn = 16'b0000000001110100; counter = counter + 3'd001; // input 87

#20 charIn = 16'b0000000001100101; counter = counter + 3'd001; // input 88

#20 charIn = 16'b0000000001110010; counter = counter + 3'd001; // input 89

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 90

#20 charIn = 16'b0000000001110100; counter = counter + 3'd001; // input 91

#20 charIn = 16'b0000000001101000; counter = counter + 3'd001; // input 92

#20 charIn = 16'b0000000001100001; counter = counter + 3'd001; // input 93

#20 charIn = 16'b0000000001101110; counter = counter + 3'd001; // input 94

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 95

#20 charIn = 16'b0000000000110010; counter = counter + 3'd001; // input 96

#20 charIn = 16'b0000000000110000; counter = counter + 3'd001; // input 97

#20 charIn = 16'b0000000000110010; counter = counter + 3'd001; // input 98

#20 charIn = 16'b0000000000110000; counter = counter + 3'd001; // input 99

#20 charIn = 16'b0000000000110000; counter = counter + 3'd001; // input 100

#20 charIn = 16'b0000000000110101; counter = counter + 3'd001; // input 101

#20 charIn = 16'b0000000000110010; counter = counter + 3'd001; // input 102

#20 charIn = 16'b0000000000110111; counter = counter + 3'd001; // input 103

#20 charIn = 16'b0000000000110001; counter = counter + 3'd001; // input 104

#20 charIn = 16'b0000000000110011; counter = counter + 3'd001; // input 105

#20 charIn = 16'b0000000000110000; counter = counter + 3'd001; // input 106

#20 charIn = 16'b0000000000110000; counter = counter + 3'd001; // input 107

#20 charIn = 16'b0000000001011010; counter = counter + 3'd001; // input 108

#20 charIn = 16'b0000000000101110; counter = counter + 3'd001; // input 109

#20 charIn = 16'b0000000000001010; counter = counter + 3'd001; // input 110

#20 charIn = 16'b0000000001000010; counter = counter + 3'd001; // input 111

#20 charIn = 16'b0000000001110010; counter = counter + 3'd001; // input 112

#20 charIn = 16'b0000000001101001; counter = counter + 3'd001; // input 113

#20 charIn = 16'b0000000001101110; counter = counter + 3'd001; // input 114

#20 charIn = 16'b0000000001100111; counter = counter + 3'd001; // input 115

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 116

#20 charIn = 16'b0000000001100001; counter = counter + 3'd001; // input 117

#20 charIn = 16'b0000000001101110; counter = counter + 3'd001; // input 118

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 119

#20 charIn = 16'b0000000001100001; counter = counter + 3'd001; // input 120

#20 charIn = 16'b0000000001101001; counter = counter + 3'd001; // input 121

#20 charIn = 16'b0000000001110010; counter = counter + 3'd001; // input 122

#20 charIn = 16'b0000000001110000; counter = counter + 3'd001; // input 123

#20 charIn = 16'b0000000001101100; counter = counter + 3'd001; // input 124

#20 charIn = 16'b0000000001100001; counter = counter + 3'd001; // input 125

#20 charIn = 16'b0000000001101110; counter = counter + 3'd001; // input 126

#20 charIn = 16'b0000000001100101; counter = counter + 3'd001; // input 127

#20 charIn = 16'b0000000000101100; counter = counter + 3'd001; // input 128

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 129

#20 charIn = 16'b0000000001110011; counter = counter + 3'd001; // input 130

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 131

#20 charIn = 16'b0000000001101101; counter = counter + 3'd001; // input 132

#20 charIn = 16'b0000000001100101; counter = counter + 3'd001; // input 133

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 134

#20 charIn = 16'b0000000001110100; counter = counter + 3'd001; // input 135

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 136

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 137

#20 charIn = 16'b0000000001101100; counter = counter + 3'd001; // input 138

#20 charIn = 16'b0000000001110011; counter = counter + 3'd001; // input 139

#20 charIn = 16'b0000000000101100; counter = counter + 3'd001; // input 140

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 141

#20 charIn = 16'b0000000001100001; counter = counter + 3'd001; // input 142

#20 charIn = 16'b0000000001101110; counter = counter + 3'd001; // input 143

#20 charIn = 16'b0000000001100100; counter = counter + 3'd001; // input 144

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 145

#20 charIn = 16'b0000000000110101; counter = counter + 3'd001; // input 146

#20 charIn = 16'b0000000001101100; counter = counter + 3'd001; // input 147

#20 charIn = 16'b0000000001100010; counter = counter + 3'd001; // input 148

#20 charIn = 16'b0000000001110011; counter = counter + 3'd001; // input 149

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 150

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 151

#20 charIn = 16'b0000000001100110; counter = counter + 3'd001; // input 152

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 153

#20 charIn = 16'b0000000001110000; counter = counter + 3'd001; // input 154

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 155

#20 charIn = 16'b0000000001110100; counter = counter + 3'd001; // input 156

#20 charIn = 16'b0000000001100001; counter = counter + 3'd001; // input 157

#20 charIn = 16'b0000000001110100; counter = counter + 3'd001; // input 158

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 159

#20 charIn = 16'b0000000001100101; counter = counter + 3'd001; // input 160

#20 charIn = 16'b0000000001110011; counter = counter + 3'd001; // input 161

#20 charIn = 16'b0000000000101110; counter = counter + 3'd001; // input 162

#20 charIn = 16'b0000000000001010; counter = counter + 3'd001; // input 163

#20 charIn = 16'b0000000001010000; counter = counter + 3'd001; // input 164

#20 charIn = 16'b0000000001100001; counter = counter + 3'd001; // input 165

#20 charIn = 16'b0000000001111001; counter = counter + 3'd001; // input 166

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 167

#20 charIn = 16'b0000000001000100; counter = counter + 3'd001; // input 168

#20 charIn = 16'b0000000001110010; counter = counter + 3'd001; // input 169

#20 charIn = 16'b0000000000101110; counter = counter + 3'd001; // input 170

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 171

#20 charIn = 16'b0000000001000111; counter = counter + 3'd001; // input 172

#20 charIn = 16'b0000000001110010; counter = counter + 3'd001; // input 173

#20 charIn = 16'b0000000001100101; counter = counter + 3'd001; // input 174

#20 charIn = 16'b0000000001100111; counter = counter + 3'd001; // input 175

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 176

#20 charIn = 16'b0000000001110111; counter = counter + 3'd001; // input 177

#20 charIn = 16'b0000000001101000; counter = counter + 3'd001; // input 178

#20 charIn = 16'b0000000001100101; counter = counter + 3'd001; // input 179

#20 charIn = 16'b0000000001101110; counter = counter + 3'd001; // input 180

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 181

#20 charIn = 16'b0000000001100100; counter = counter + 3'd001; // input 182

#20 charIn = 16'b0000000001101111; counter = counter + 3'd001; // input 183

#20 charIn = 16'b0000000001101110; counter = counter + 3'd001; // input 184

#20 charIn = 16'b0000000001100101; counter = counter + 3'd001; // input 185

#20 charIn = 16'b0000000000101110; counter = counter + 3'd001; // input 186

#20 charIn = 16'b0000000000001010; counter = counter + 3'd001; // input 187

#20 charIn = 16'b0000000001000111; counter = counter + 3'd001; // input 188

#20 charIn = 16'b0000000001001111; counter = counter + 3'd001; // input 189

#20 charIn = 16'b0000000001001111; counter = counter + 3'd001; // input 190

#20 charIn = 16'b0000000001000100; counter = counter + 3'd001; // input 191

#20 charIn = 16'b0000000000100000; counter = counter + 3'd001; // input 192

#20 charIn = 16'b0000000001001100; counter = counter + 3'd001; // input 193

#20 charIn = 16'b0000000001010101; counter = counter + 3'd001; // input 194

#20 charIn = 16'b0000000001000011; counter = counter + 3'd001; // input 195

#20 charIn = 16'b0000000001001011; counter = counter + 3'd001; // input 196

#20 charIn = 16'b0000000000000000; counter = counter + 3'd001; // input 197

#20 charIn = 16'b0000000000000000; counter = counter + 3'd001; // input 198

#20 charIn = 16'b0000000000000000; counter = counter + 3'd001; // input 199

#20 charIn = 16'b0000000000000000; counter = counter + 3'd001; // input 200

#20 counter = counter + 3'd001;

#100 $finish;

end

always @ (posedge clock)

begin

//$write ("%s" , charOut); // was used to test, use if needed

if( counter > 0 && counter <= 200)

begin

$write ("%d ", $time);

$write ("%d", counter);

$display (" %b", charOut);

end

else if (counter > 200) $display("transmission complete");

else $display ("transmission to follow:");

end

endmodule

</insert>