**BRIEF**

<insert>

David Nakasone

davenakasone@gmail.com

This is 2 of 2 circuits.

SERIAL IN, SERIAL OUT

It is designed to input 200 encrypted 16-bit words and output 200 decrypted 16-bit words, revealing the message from the sender.

In each word, the received input represents a character.

Since the encrypted input was an unencrypted ASCII character with a key of { 2^15 + 1957 } = \*\*34725\*\* added to it:

input [ 1000 0111 1110 0110 ] is decrypted by subtracting decimal 34725 or binary 16'b 1000 0111 1010 0101

this produces the output of [ 0000 0000 0100 0001 ], revealing the character 'A'

this process continues until all the characters have been decrypted

The test bench begins by stimulating this circuit with 200 encrypted 16-bit words that would have come from the sending circuit.

As the stimuli produce output, the receiver can easily convert the binary ASCII numbers to characters and read the message.

See "20200429 encryptor #UNLV" here: https://www.edaplayground.com/x/3hWW for the circuit 1 of 2 that is able to encrypt and transmit a message to 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 decryptor (clock, abort, wordIn, charOut, readyRX); // end of port list

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

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

input [15:0] wordIn; // input as 16-bit encrypted word

output reg [15:0] charOut; // output as 16-bit ASCII character with only 7 bits used

output reg readyRX; // 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

// reg [15:0] verify [0:199]; // holds the 200 lines of 16-bit input, can use to verify

always @ (posedge clock)

begin

if (abort)

begin

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

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

readyRX <= 0; // not ready to RX

end

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

end

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

begin

case (ps)

s0: begin

charOut = 16'b0; // output is 0

readyRX = 0;

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

end

s1: begin

charOut = wordIn - 16'b1000011110100101; // decrypt current word

counter = counter + 1; // increment counter

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

else ns = s1; // keep decrypting if 200 characters have not been decrypted

readyRX = 0;

end

s2: begin

readyRX = 1'b1; // ready to receive more, last message decrypted

#100; // use counter or wait time to reflect all bits in as serial RX

// good spot for a confirmation signal

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

end

s3: begin

readyRX = 1'b1;

charOut = 16'b0; // clear receive buffer

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

ns = s0; // stand by to decrypt next RX

end

endcase

end

endmodule

</insert>

**TB**

<insert>

module TBdecrypt;

reg clock; // toggled for clock pulse

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

reg [15:0] wordIn; // 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 readyRX; // after 200 characters encrypted and placed in buffer, this starts serial TX

integer counter;

decryptor DUT (clock, abort, wordIn, charOut, readyRX);

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

initial // initial block, holds serial RX input of 16-bit encrypted words

begin

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

abort = 1'b0; // proceed to decrypt

counter = 3'd0; // just for display control

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

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

// BEGIN INPUT BUFFER (encrypted RX from sender)

#20 wordIn = 16'b1000011111101110; counter = counter + 3'd001; // encrypted input 1

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 2

#20 wordIn = 16'b1000100000001101; counter = counter + 3'd001; // encrypted input 3

#20 wordIn = 16'b1000100000000110; counter = counter + 3'd001; // encrypted input 4

#20 wordIn = 16'b1000100000011011; counter = counter + 3'd001; // encrypted input 5

#20 wordIn = 16'b1000100000001010; counter = counter + 3'd001; // encrypted input 6

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 7

#20 wordIn = 16'b1000011111011000; counter = counter + 3'd001; // encrypted input 8

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 9

#20 wordIn = 16'b1000100000011001; counter = counter + 3'd001; // encrypted input 10

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 11

#20 wordIn = 16'b1000100000010011; counter = counter + 3'd001; // encrypted input 12

#20 wordIn = 16'b1000100000011000; counter = counter + 3'd001; // encrypted input 13

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 14

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 15

#20 wordIn = 16'b1000100000001011; counter = counter + 3'd001; // encrypted input 16

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 17

#20 wordIn = 16'b1000100000001100; counter = counter + 3'd001; // encrypted input 18

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 19

#20 wordIn = 16'b1000100000010001; counter = counter + 3'd001; // encrypted input 20

#20 wordIn = 16'b1000100000001001; counter = counter + 3'd001; // encrypted input 21

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 22

#20 wordIn = 16'b1000100000011001; counter = counter + 3'd001; // encrypted input 23

#20 wordIn = 16'b1000100000001101; counter = counter + 3'd001; // encrypted input 24

#20 wordIn = 16'b1000100000000110; counter = counter + 3'd001; // encrypted input 25

#20 wordIn = 16'b1000100000011001; counter = counter + 3'd001; // encrypted input 26

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 27

#20 wordIn = 16'b1000011111101110; counter = counter + 3'd001; // encrypted input 28

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 29

#20 wordIn = 16'b1000100000010011; counter = counter + 3'd001; // encrypted input 30

#20 wordIn = 16'b1000100000001010; counter = counter + 3'd001; // encrypted input 31

#20 wordIn = 16'b1000100000001010; counter = counter + 3'd001; // encrypted input 32

#20 wordIn = 16'b1000100000001001; counter = counter + 3'd001; // encrypted input 33

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 34

#20 wordIn = 16'b1000100000011110; counter = counter + 3'd001; // encrypted input 35

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 36

#20 wordIn = 16'b1000100000011010; counter = counter + 3'd001; // encrypted input 37

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 38

#20 wordIn = 16'b1000100000011001; counter = counter + 3'd001; // encrypted input 39

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 40

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 41

#20 wordIn = 16'b1000100000010101; counter = counter + 3'd001; // encrypted input 42

#20 wordIn = 16'b1000100000001110; counter = counter + 3'd001; // encrypted input 43

#20 wordIn = 16'b1000100000001000; counter = counter + 3'd001; // encrypted input 44

#20 wordIn = 16'b1000100000010000; counter = counter + 3'd001; // encrypted input 45

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 46

#20 wordIn = 16'b1000100000011010; counter = counter + 3'd001; // encrypted input 47

#20 wordIn = 16'b1000100000010101; counter = counter + 3'd001; // encrypted input 48

#20 wordIn = 16'b1000011111010011; counter = counter + 3'd001; // encrypted input 49

#20 wordIn = 16'b1000011110101111; counter = counter + 3'd001; // encrypted input 50

#20 wordIn = 16'b1000011111100111; counter = counter + 3'd001; // encrypted input 51

#20 wordIn = 16'b1000100000001010; counter = counter + 3'd001; // encrypted input 52

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 53

#20 wordIn = 16'b1000100000000110; counter = counter + 3'd001; // encrypted input 54

#20 wordIn = 16'b1000100000011001; counter = counter + 3'd001; // encrypted input 55

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 56

#20 wordIn = 16'b1000011111110010; counter = counter + 3'd001; // encrypted input 57

#20 wordIn = 16'b1000100000001010; counter = counter + 3'd001; // encrypted input 58

#20 wordIn = 16'b1000100000010001; counter = counter + 3'd001; // encrypted input 59

#20 wordIn = 16'b1000100000010001; counter = counter + 3'd001; // encrypted input 60

#20 wordIn = 16'b1000100000001110; counter = counter + 3'd001; // encrypted input 61

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 62

#20 wordIn = 16'b1000011111100111; counter = counter + 3'd001; // encrypted input 63

#20 wordIn = 16'b1000100000000110; counter = counter + 3'd001; // encrypted input 64

#20 wordIn = 16'b1000100000010011; counter = counter + 3'd001; // encrypted input 65

#20 wordIn = 16'b1000100000010000; counter = counter + 3'd001; // encrypted input 66

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 67

#20 wordIn = 16'b1000011111011000; counter = counter + 3'd001; // encrypted input 68

#20 wordIn = 16'b1000011111011010; counter = counter + 3'd001; // encrypted input 69

#20 wordIn = 16'b1000011111010011; counter = counter + 3'd001; // encrypted input 70

#20 wordIn = 16'b1000011111011011; counter = counter + 3'd001; // encrypted input 71

#20 wordIn = 16'b1000011111011101; counter = counter + 3'd001; // encrypted input 72

#20 wordIn = 16'b1000011111011011; counter = counter + 3'd001; // encrypted input 73

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 74

#20 wordIn = 16'b1000011111011010; counter = counter + 3'd001; // encrypted input 75

#20 wordIn = 16'b1000011111010110; counter = counter + 3'd001; // encrypted input 76

#20 wordIn = 16'b1000011111010011; counter = counter + 3'd001; // encrypted input 77

#20 wordIn = 16'b1000011111011000; counter = counter + 3'd001; // encrypted input 78

#20 wordIn = 16'b1000011111011110; counter = counter + 3'd001; // encrypted input 79

#20 wordIn = 16'b1000011111011000; counter = counter + 3'd001; // encrypted input 80

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 81

#20 wordIn = 16'b1000100000010011; counter = counter + 3'd001; // encrypted input 82

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 83

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 84

#20 wordIn = 16'b1000100000010001; counter = counter + 3'd001; // encrypted input 85

#20 wordIn = 16'b1000100000000110; counter = counter + 3'd001; // encrypted input 86

#20 wordIn = 16'b1000100000011001; counter = counter + 3'd001; // encrypted input 87

#20 wordIn = 16'b1000100000001010; counter = counter + 3'd001; // encrypted input 88

#20 wordIn = 16'b1000100000010111; counter = counter + 3'd001; // encrypted input 89

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 90

#20 wordIn = 16'b1000100000011001; counter = counter + 3'd001; // encrypted input 91

#20 wordIn = 16'b1000100000001101; counter = counter + 3'd001; // encrypted input 92

#20 wordIn = 16'b1000100000000110; counter = counter + 3'd001; // encrypted input 93

#20 wordIn = 16'b1000100000010011; counter = counter + 3'd001; // encrypted input 94

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 95

#20 wordIn = 16'b1000011111010111; counter = counter + 3'd001; // encrypted input 96

#20 wordIn = 16'b1000011111010101; counter = counter + 3'd001; // encrypted input 97

#20 wordIn = 16'b1000011111010111; counter = counter + 3'd001; // encrypted input 98

#20 wordIn = 16'b1000011111010101; counter = counter + 3'd001; // encrypted input 99

#20 wordIn = 16'b1000011111010101; counter = counter + 3'd001; // encrypted input 100

#20 wordIn = 16'b1000011111011010; counter = counter + 3'd001; // encrypted input 101

#20 wordIn = 16'b1000011111010111; counter = counter + 3'd001; // encrypted input 102

#20 wordIn = 16'b1000011111011100; counter = counter + 3'd001; // encrypted input 103

#20 wordIn = 16'b1000011111010110; counter = counter + 3'd001; // encrypted input 104

#20 wordIn = 16'b1000011111011000; counter = counter + 3'd001; // encrypted input 105

#20 wordIn = 16'b1000011111010101; counter = counter + 3'd001; // encrypted input 106

#20 wordIn = 16'b1000011111010101; counter = counter + 3'd001; // encrypted input 107

#20 wordIn = 16'b1000011111111111; counter = counter + 3'd001; // encrypted input 108

#20 wordIn = 16'b1000011111010011; counter = counter + 3'd001; // encrypted input 109

#20 wordIn = 16'b1000011110101111; counter = counter + 3'd001; // encrypted input 110

#20 wordIn = 16'b1000011111100111; counter = counter + 3'd001; // encrypted input 111

#20 wordIn = 16'b1000100000010111; counter = counter + 3'd001; // encrypted input 112

#20 wordIn = 16'b1000100000001110; counter = counter + 3'd001; // encrypted input 113

#20 wordIn = 16'b1000100000010011; counter = counter + 3'd001; // encrypted input 114

#20 wordIn = 16'b1000100000001100; counter = counter + 3'd001; // encrypted input 115

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 116

#20 wordIn = 16'b1000100000000110; counter = counter + 3'd001; // encrypted input 117

#20 wordIn = 16'b1000100000010011; counter = counter + 3'd001; // encrypted input 118

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 119

#20 wordIn = 16'b1000100000000110; counter = counter + 3'd001; // encrypted input 120

#20 wordIn = 16'b1000100000001110; counter = counter + 3'd001; // encrypted input 121

#20 wordIn = 16'b1000100000010111; counter = counter + 3'd001; // encrypted input 122

#20 wordIn = 16'b1000100000010101; counter = counter + 3'd001; // encrypted input 123

#20 wordIn = 16'b1000100000010001; counter = counter + 3'd001; // encrypted input 124

#20 wordIn = 16'b1000100000000110; counter = counter + 3'd001; // encrypted input 125

#20 wordIn = 16'b1000100000010011; counter = counter + 3'd001; // encrypted input 126

#20 wordIn = 16'b1000100000001010; counter = counter + 3'd001; // encrypted input 127

#20 wordIn = 16'b1000011111010001; counter = counter + 3'd001; // encrypted input 128

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 129

#20 wordIn = 16'b1000100000011000; counter = counter + 3'd001; // encrypted input 130

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 131

#20 wordIn = 16'b1000100000010010; counter = counter + 3'd001; // encrypted input 132

#20 wordIn = 16'b1000100000001010; counter = counter + 3'd001; // encrypted input 133

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 134

#20 wordIn = 16'b1000100000011001; counter = counter + 3'd001; // encrypted input 135

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 136

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 137

#20 wordIn = 16'b1000100000010001; counter = counter + 3'd001; // encrypted input 138

#20 wordIn = 16'b1000100000011000; counter = counter + 3'd001; // encrypted input 139

#20 wordIn = 16'b1000011111010001; counter = counter + 3'd001; // encrypted input 140

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 141

#20 wordIn = 16'b1000100000000110; counter = counter + 3'd001; // encrypted input 142

#20 wordIn = 16'b1000100000010011; counter = counter + 3'd001; // encrypted input 143

#20 wordIn = 16'b1000100000001001; counter = counter + 3'd001; // encrypted input 144

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 145

#20 wordIn = 16'b1000011111011010; counter = counter + 3'd001; // encrypted input 146

#20 wordIn = 16'b1000100000010001; counter = counter + 3'd001; // encrypted input 147

#20 wordIn = 16'b1000100000000111; counter = counter + 3'd001; // encrypted input 148

#20 wordIn = 16'b1000100000011000; counter = counter + 3'd001; // encrypted input 149

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 150

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 151

#20 wordIn = 16'b1000100000001011; counter = counter + 3'd001; // encrypted input 152

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 153

#20 wordIn = 16'b1000100000010101; counter = counter + 3'd001; // encrypted input 154

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 155

#20 wordIn = 16'b1000100000011001; counter = counter + 3'd001; // encrypted input 156

#20 wordIn = 16'b1000100000000110; counter = counter + 3'd001; // encrypted input 157

#20 wordIn = 16'b1000100000011001; counter = counter + 3'd001; // encrypted input 158

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 159

#20 wordIn = 16'b1000100000001010; counter = counter + 3'd001; // encrypted input 160

#20 wordIn = 16'b1000100000011000; counter = counter + 3'd001; // encrypted input 161

#20 wordIn = 16'b1000011111010011; counter = counter + 3'd001; // encrypted input 162

#20 wordIn = 16'b1000011110101111; counter = counter + 3'd001; // encrypted input 163

#20 wordIn = 16'b1000011111110101; counter = counter + 3'd001; // encrypted input 164

#20 wordIn = 16'b1000100000000110; counter = counter + 3'd001; // encrypted input 165

#20 wordIn = 16'b1000100000011110; counter = counter + 3'd001; // encrypted input 166

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 167

#20 wordIn = 16'b1000011111101001; counter = counter + 3'd001; // encrypted input 168

#20 wordIn = 16'b1000100000010111; counter = counter + 3'd001; // encrypted input 169

#20 wordIn = 16'b1000011111010011; counter = counter + 3'd001; // encrypted input 170

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 171

#20 wordIn = 16'b1000011111101100; counter = counter + 3'd001; // encrypted input 172

#20 wordIn = 16'b1000100000010111; counter = counter + 3'd001; // encrypted input 173

#20 wordIn = 16'b1000100000001010; counter = counter + 3'd001; // encrypted input 174

#20 wordIn = 16'b1000100000001100; counter = counter + 3'd001; // encrypted input 175

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 176

#20 wordIn = 16'b1000100000011100; counter = counter + 3'd001; // encrypted input 177

#20 wordIn = 16'b1000100000001101; counter = counter + 3'd001; // encrypted input 178

#20 wordIn = 16'b1000100000001010; counter = counter + 3'd001; // encrypted input 179

#20 wordIn = 16'b1000100000010011; counter = counter + 3'd001; // encrypted input 180

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 181

#20 wordIn = 16'b1000100000001001; counter = counter + 3'd001; // encrypted input 182

#20 wordIn = 16'b1000100000010100; counter = counter + 3'd001; // encrypted input 183

#20 wordIn = 16'b1000100000010011; counter = counter + 3'd001; // encrypted input 184

#20 wordIn = 16'b1000100000001010; counter = counter + 3'd001; // encrypted input 185

#20 wordIn = 16'b1000011111010011; counter = counter + 3'd001; // encrypted input 186

#20 wordIn = 16'b1000011110101111; counter = counter + 3'd001; // encrypted input 187

#20 wordIn = 16'b1000011111101100; counter = counter + 3'd001; // encrypted input 188

#20 wordIn = 16'b1000011111110100; counter = counter + 3'd001; // encrypted input 189

#20 wordIn = 16'b1000011111110100; counter = counter + 3'd001; // encrypted input 190

#20 wordIn = 16'b1000011111101001; counter = counter + 3'd001; // encrypted input 191

#20 wordIn = 16'b1000011111000101; counter = counter + 3'd001; // encrypted input 192

#20 wordIn = 16'b1000011111110001; counter = counter + 3'd001; // encrypted input 193

#20 wordIn = 16'b1000011111111010; counter = counter + 3'd001; // encrypted input 194

#20 wordIn = 16'b1000011111101000; counter = counter + 3'd001; // encrypted input 195

#20 wordIn = 16'b1000011111110000; counter = counter + 3'd001; // encrypted input 196

#20 wordIn = 16'b1000011110100101; counter = counter + 3'd001; // encrypted input 197

#20 wordIn = 16'b1000011110100101; counter = counter + 3'd001; // encrypted input 198

#20 wordIn = 16'b1000011110100101; counter = counter + 3'd001; // encrypted input 199

#20 wordIn = 16'b1000011110100101; counter = counter + 3'd001; // encrypted input 200

// END INPUT BUFFER (encrypted RX from sender)

#20 counter = counter + 3'd001;

#20 counter = counter + 3'd001; abort = 1; // disables circuit, ready to get another transmission

#100 $finish;

end

// BEGIN OUTPUT BUFFER (decrpyted RX displayed as ASCII)

always @ (posedge clock) // always block reflects serial decrypted RX of encrypted inputs provided

begin

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

begin

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

//$write ("%d", counter);

//$display (" %b", wordOut);

$write ("%s",charOut);

end

else if (counter == 200) $display("decryption complete");

else if (counter == 201)

begin

$write ("\n");

$display (" { reset, ready to decrypt next RX }");

end

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

end

// END OUTPUT BUFFER (decrpyted RX displayed as ASCII)

endmodule

</insert>