

```

// *****
// **
// ** ALAW2LIN.v - A-LAW TO LINEAR 2'S COMPLEMENT CODE TRANSLATOR
// **
// *****
// **
// **          COPYRIGHT (c) 2001 YOUNG ENGINEERING
// **          ALL RIGHTS RESERVED
// **
// ** THIS PROGRAM IS CONFIDENTIAL AND A TRADE SECRET OF YOUNG ENGINEERING. THE RECEIPT OR
// ** POSSESSION OF THIS PROGRAM DOES NOT CONVEY ANY RIGHTS TO REPRODUCE OR DISCLOSE ITS
// ** CONTENTS, OR TO MANUFACTURE, USE, OR SELL ANYTHING THAT IT MAY DESCRIBE, IN WHOLE OR IN
// ** PART, WITHOUT THE SPECIFIC WRITTEN CONSENT OF YOUNG ENGINEERING.
// **
// *****
// ** Revision      : 1.0
// ** Modified Date : 11/01/2001
// ** Revision History:
// **
// ** 11/01/2001: Initial design
// **
// *****
// **          TABLE OF CONTENTS
// **
// **-----
// **  DECLARATIONS
// **-----
// **  FORMAT CONVERSION
// **-----
// **  1.01: A-Law Input Inversion
// **  1.02: A-Law to Linear Table
// **  1.03: Linear 2's Complement
// **
// *****

```

```
`timescale 1ns/10ps
```

```
module ALAW2LIN (DataI, DataO);
```

```

input  [07:00] DataI; // data input - A-law
output [12:00] DataO; // data output - linear 2's complement

```

```

// *****
// **  DECLARATIONS
// **-----

```

```

wire [07:00] DataI_N; // alternating bit-wise inverted input
wire [12:00] DataO; // linear data output - 2's complement

reg [12:00] LinearData; // linear data - unsigned

```

```

// *****
// **  FORMAT CONVERSION
// **-----

```

```
// 1.01: A-Law Input Inversion
```

```
assign DataI_N = DataI ^ 8'h55;
```

```
// 1.02: A-Law to Linear Table
```

```

always @(DataI_N) begin
  casex(DataI_N)
    8'b1111xxxx : LinearData = {2'b11,DataI_N[03:00],7'b1000000}; // + full scale
    8'b1110xxxx : LinearData = {3'b101,DataI_N[03:00],6'b100000};
    8'b1101xxxx : LinearData = {4'b1001,DataI_N[03:00],5'b10000};
    8'b1100xxxx : LinearData = {5'b10001,DataI_N[03:00],4'b1000};
    8'b1011xxxx : LinearData = {6'b100001,DataI_N[03:00],3'b100};
    8'b1010xxxx : LinearData = {7'b1000001,DataI_N[03:00],2'b10};
    8'b1001xxxx : LinearData = {8'b10000001,DataI_N[03:00],1'b1};
    8'b1000xxxx : LinearData = {8'b10000000,DataI_N[03:00],1'b1}; // + zero
  end

```

```

8'b0000xxxx : LinearData = {8'b00000000,DataI_N[03:00],1'b1}; // - zero
8'b0001xxxx : LinearData = {8'b00000001,DataI_N[03:00],1'b1};
8'b0010xxxx : LinearData = {7'b0000001,DataI_N[03:00],2'b10};
8'b0011xxxx : LinearData = {6'b000001,DataI_N[03:00],3'b100};
8'b0100xxxx : LinearData = {5'b00001,DataI_N[03:00],4'b1000};
8'b0101xxxx : LinearData = {4'b0001,DataI_N[03:00],5'b10000};
8'b0110xxxx : LinearData = {3'b001,DataI_N[03:00],6'b100000};
8'b0111xxxx : LinearData = {2'b01,DataI_N[03:00],7'b1000000}; // - full scale
endcase
end

// -----
//      1.03:  Linear 2's Complement
// -----

assign Data0 = LinearData[12] ? {1'b0,LinearData[11:00]} : {1'b1,(~LinearData[11:00] + 1)};
endmodule

```