Key Design Features

- Synthesizable, technology independent IP Core for FPGA, ASIC and SoC
- Supplied as human readable VHDL (or Verilog) source code
- Encodes a simple NRZ bitstream to the equivalent Manchester code and vice-versa
- Supplied as separate, independent encoder and decoder IP Cores
- DC-balanced encoding suitable for AC-coupling over large distances
- Ideal for serial comms over twisted pair or coaxial
- Encoder features an input FIFO for buffering the incoming bitstream / data packet
- Robust data and clock recovery at the decoder featuring Gaussian pulse-shaping filters for optimum SNR and noise immunity
- No complex PLL setup required. Whole design runs off a single source clock of 240 MHz
- Configurable bit rate setting of 5,10,15 or 20 Mbps on low-cost FPGAs and SoC devices[1]

Applications

- Applications requiring low-to-medium serial bandwidth
- Industrial / scientific / medical / automotive control systems
- Implementation of robust serial links on low cost devices requiring only a single bit lane
- Galvanically isolated serial links where the DC-component must be zero

Generic Parameters

Generic name	Description	Type	Valid range
depth	Input FIFO depth (Encoder)	integer	≥ 2 (must be a power of 2)
log2d	Log2 input FIFO depth (Encoder)	integer	log2 (depth)
bit_rate	Bit rate in Mbps (Encoder & Decoder)	integer	0: 5 Mbps 1: 10 Mbps 2: 15 Mbps 3: 20 Mbps

1 Higher bit-rate settings of 50 Mbps+ are available on higher-end devices. Please contact Zipcores for more information

Block Diagram

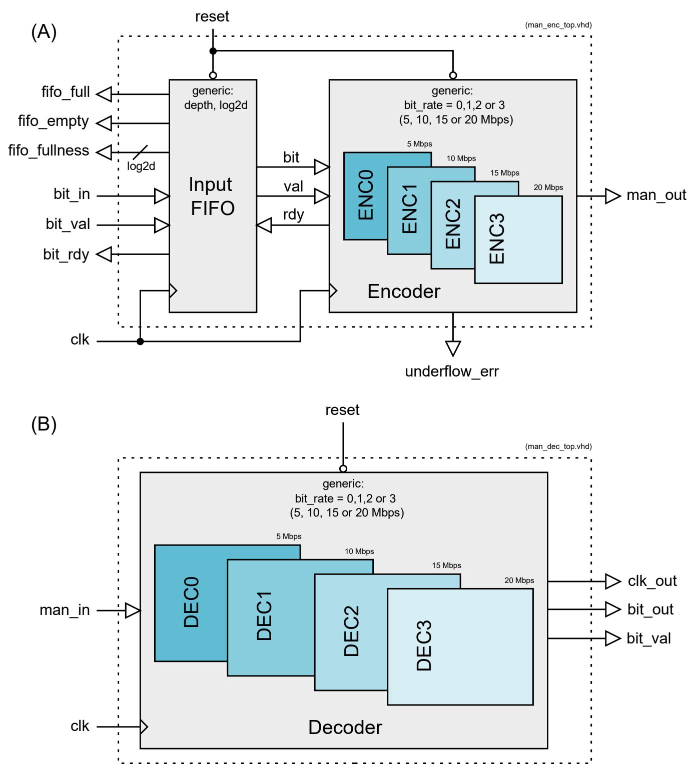


Figure 1: General architecture of the Manchester Encoder (A) and Manchester Decoder (B) IP Cores

Pin-out Description

MANCHESTER ENCODER

Pin name	I/O	Description	Active state
clk	in	Synchronous clock (240 MHz nominal)	rising edge
reset	in	Asynchronous reset	low
underflow_err	out	Indicates a starvation of bits at the encoder	high
fifo_full	out	Input FIFO full flag	high
fifo_empty	out	Input FIFO empty flag	high
fifo_fullness[log2d:0]	out	Input FIFO ‘fullness’ counter	data (unsigned number)
bit_in	in	NRZ bit data input	data
bit_val	in	Bit data input valid	high
bit_rdy	out	Bit data input ready (handshake)	data
man_out	out	Manchester encoded bit out	data

MANCHESTER DECODER

Pin name	**I/O**	**Description**	**Active state**
clk	in	Synchronous clock (240 MHz nominal)	rising edge
reset	in	Asynchronous reset	low
man_in	in	Manchester encoded bit input (asynchronous signal)	data
clk_out	out	Recovered clock signal	rising edge
bit_out	out	Recovered bit data	data
bit_val	out	Bit data output valid	high

General Description

The MAN_CODEC IP Core is a versatile encoder and decoder pair that converts a basic NRZ bitstream into a standard Manchester code and vice-versa. The encoder and decoder are provided as separate IP Cores and, as such, may be used independently or together as a combined codec unit.

Both the encoder and decoder work from a single synchronous clock input (*clk*) running at a fixed frequency of 240 MHz. In addition there are four different bit rates to choose from. These are: 5, 10, 15 and 20 Mbps. The bit rate setting is fixed at compile time and is controlled by the *bit_rate* parameter.

Manchester Encoder

The encoder (Figure 1-A) features an input FIFO which may be used to buffer an incoming bitstream before encoding. For instance, this is useful in order to buffer a full data packet or frame of data before transmission. The *fifo_full*, *fifo_empty* and *fifo_fullness* signals are provided to help manage the flow of data into the FIFO. The FIFO depth may be set using the *depth* and *log2d* generic parameters. If the FIFO becomes empty and the encoder is starved of data, the *underflow_err* flag is asserted high.

The input interface to the encoder features three signals with full flow-control. These are: *bit_in*, *bit_val* and *bit_rdy*. An input bit is sampled on the rising clock-edge of *clk* when *bit_val* and *bit_rdy* are both high[2]. The output of the encoder is a standard Manchester encoded bitstream on the *man_out* port.

Manchester Decoder

The decoder (Figure 1-B) accepts an asynchronous Manchester encoded bitstream (*man_in*) at the correct bit rate and generates an output bit, output valid and recovered output clock that are all synchronous with the 240 MHz system clock (*clk*).

The decoder employs a number of Gaussian pulse-shaping FIR filters in order to filter the incoming Manchester signal, recover the data clock and the original bitstream. As such, the decoder is capable of accommodating a significant amount of jitter and frequency deviation on the incoming *man_in* signal without failure.

(Note: The FIR filters have been highly optimized to have the correct characteristics and impulse response times for the chosen bitrate settings. For this reason, the user must be careful to select the correct bitrate setting for the application and also ensure that a stable 240 MHz system clock is provided – e.g. via the PLL or MMCM resources available on most modern FPGAs and SoCs).

2 Please see application note: *app_note_zc001.pdf* for more information on the valid/ready streaming protocol

Functional Timing

Figure 2 demonstrates a simple encoding example. In this case, a 4-bit nibble: “0001” is sent to the Manchester encoder for transmission. The 4 consecutive bits are qualified by the *bit_val* flag. As the input FIFO does not fill to capacity, the *fifo_full* signal remains low and the *bit_rdy* signal remains high. In this example the *bit_rate* parameter has been set to ‘3’ giving a bit rate of 20 Mbps.

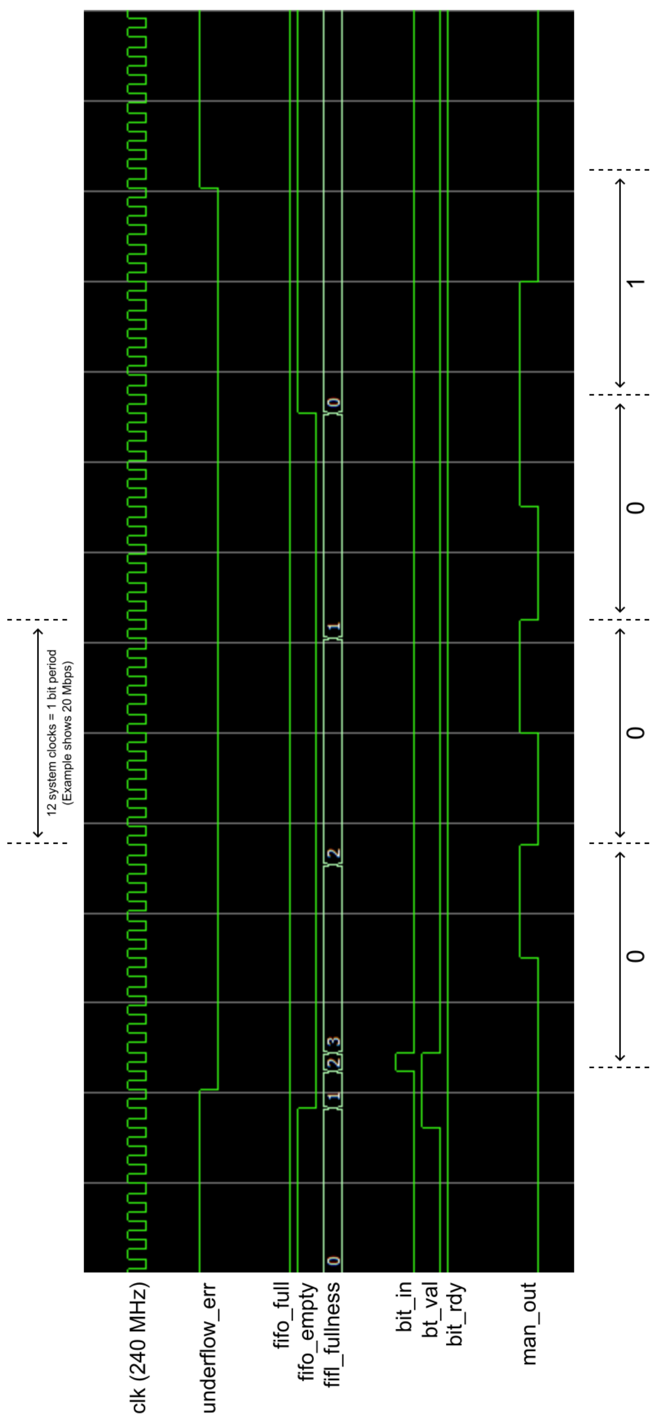


Figure 2: Manchester Encoder timing waveform (20 Mbps rate)

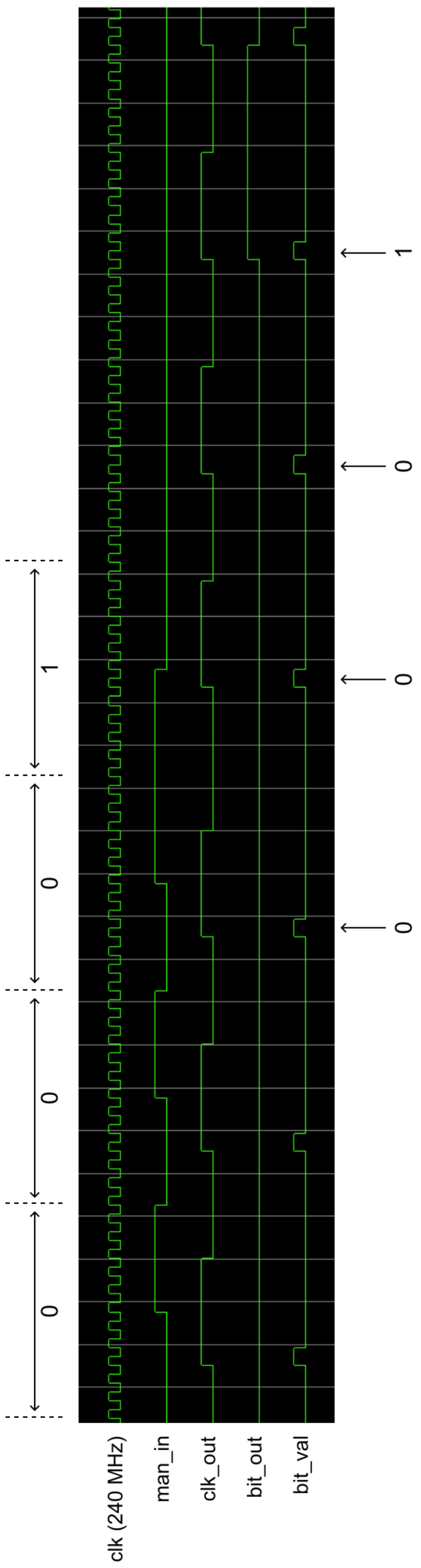


Figure 3: Manchester Decoder timing waveform (20 Mbps rate)

Following on from the encoder timing, Figure 3 shows the corresponding decoder timing waveform for the same nibble of data, "0001" and the exact same bit rate of 20 Mbps.

(Note: Due to the nature of Manchester encoding being a derivative of Binary Phase Shift Keying (BPSK), then the user must take into account the possibility of phase ambiguities at the decoder. For instance, the transmission of the sequence: 00010001 may be decoded as 11101110 and vice-versa. The ambiguity may be avoided if an extra stage of differential encoding/decoding is employed. Alternatively, a known good sync or framing sequence may be used to frame the data packets correctly and determine whether the decoded bitstream is in phase or 180° out of phase with the transmitted signal).

Source File Description

All source files are provided as text files coded in VHDL. The following table gives a brief description of each file. Note that the design features four different FIR filters for the different bit rate settings. For brevity, these are all labelled filter 'X' in the table.

Source file	***Description***
fir_shapingX_pack.vhd	FIR filter package definitions file
fir_shapingX_mad.vhd	FIR filter Multiply/Add module
fir_shapingX_madl.vhd	FIR filter Multiply/Add module (odd tap)
fir_shapingX_sat.vhd	FIR filter saturation module
fir_shapingX.vhd	FIR filter main component
fir_shaping_top.vhd	FIR filter top-level module (selects one of 4 filters depending on the bit rate)
pipeline_reg.vhd	Pipeline register component
fifo_sync_bit.vhd	One-bit wide synchronous FIFO
edge_detect.vhd	Logic edge-detection circuit
pattern_gen.vhd	Programmable pattern generator module
pattern_sync.vhd	Programmable sync detect module
man_enc.vhd	Manchester encoder core component
man_enc_top.vhd	Manchester encoder top-level component
man_dec.vhd	Manchester decoder core component
man_dec_top.vhd	Manchester decoder top-level component
man_codec_bench.vhd	Top-level test bench

Functional Testing

An example VHDL test bench is provided for use in a suitable hardware simulator. The compilation order of the source code is the same as that in the source code file description (above).

The test bench instantiates four separate modules in series. These are the pattern generator (*pattern_gen.vhd*), the Manchester encoder (*man_enc_top.vhd*), the Manchester decoder (*man_dec_top.vhd*) and the pattern sync detect module (*pattern_sync.vhd*).

The pattern generator is set up to generate a data packet of 96 bits in total having the value: 0xFFFFDEADBEEFCAFEBABEFFFF

The pattern sync detect module is set up to detect the pattern:

0xDEADBEEFCAFEBABE

If the simulation runs correctly, then the output of the sync detect module (*sync_val*) should be asserted high each time the sync pattern is detected at the decoder. This should occur with each packet sent. Running the simulation for ~1 ms is sufficient to see around 100 decoded packets.

Development Board Testing

The Manchester Encoder and Decoder IP Cores were tested using a pair of low-cost Xilinx® evaluation boards separated by 20m of twisted pair cable. An Arty-A7 evaluation board was used for the encoder/transmitter and an Arty-S7 board was used as the decoder/receiver. The same test setup was implemented in hardware as the functional simulation. The setup was adjusted to send a data packet around every 5 us.

For the transmitter, a single-ended Manchester encoded signal was generated at 20 Mbps on one of the PMOD output pins. This signal was then sent via the twisted pair cable to the receiver.

The receiver also used one of PMOD headers for the Manchester input. In addition, other debug information was provided via the other PMOD headers on the S7 board. This allowed monitoring on the scope and to confirm the correct reception of data packets.

Figure 4 shows the Arty-S7 board set up as the receiver with scope probes monitoring the incoming Manchester data, packet sync flag, recovered bitstream and recovered clock.

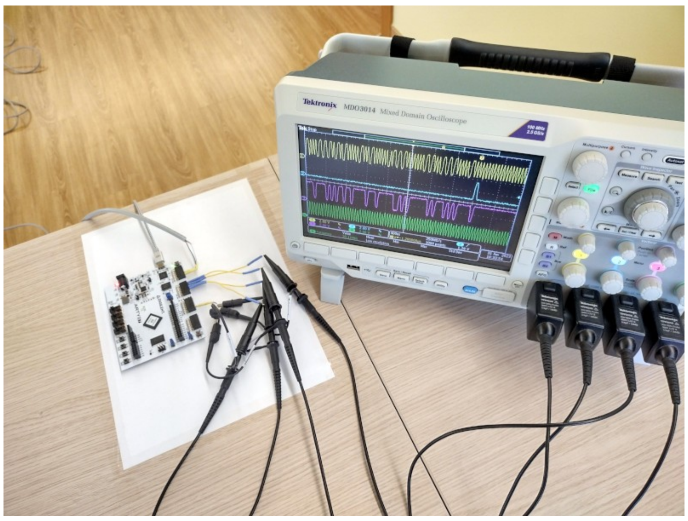
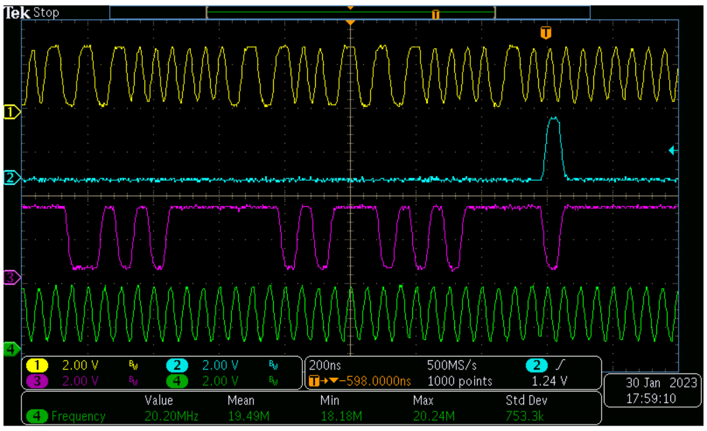


Figure 4: Test bench setup showing the Arty-S7 board used as a receiver for the Manchester encoded signal at 20 Mbps

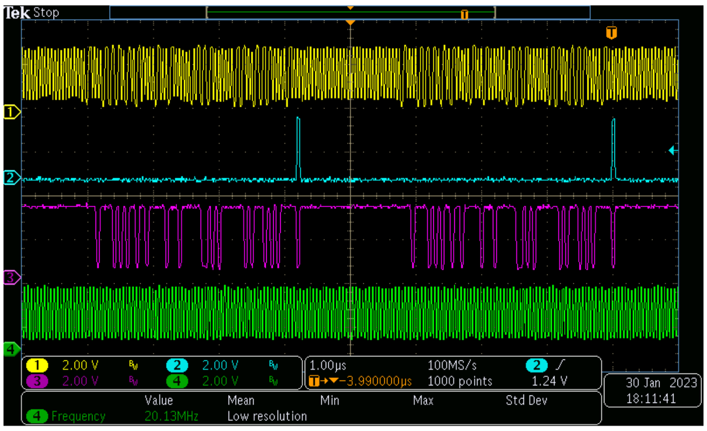
Figure 5 shows a screenshot of the scope traces for the received packet.



1: Input, 2: Sync, 3: Data, 4: Clock

Figure 5: Close-up of scope traces at the receiver

Figure 6 shows two consecutive packets sent in a row. The bit rate was fixed at 20 Mbps for all tests. Other bit rates were also tested to verify correct operation.



Two consecutive decoded packets

Figure 6: Scope traces for 2 packets in a row

Synthesis and Implementation

The files required for synthesis and the design hierarchy is shown below:

Manchester Encoder top-level component:

- man_enc_top.vhd
  - fifo_sync_bit.vhd
    - pipeline_reg.vhd
  - man_enc.vhd

Manchester Decoder top-level component:

- man_dec_top.vhd
  - man_dec.vhd
    - fir_shaping_top.vhd
      - fir_shapingX.vhd
        - fir_shapingX_mad.vhd
        - fir_shapingX_madl.vhd
        - fir_shapingX_sat.vhd
    - fir_edge_detect.vhd

The Manchester Encoder and Decoder IP Cores are designed to be technology independent. As a benchmark, synthesis results have been provided for the AMD / Xilinx® 7-series FPGAs and SoCs. Synthesis results for other FPGAs and technologies can be provided on request.

Trial synthesis results are shown for a combined encoder/decoder solution with the generic parameters set to: *depth* = 512, log2d = 9 and *bit_rate* = 3 (20 Mbps).

For this particular IP Core, then the system clock frequency is fixed at 240 MHz. Generally, higher-end devices are capable of much higher clock frequencies. Please contact Zipcores if you require a solution that supports bit rates in excess of 20 Mbps. In addition, we can also provide custom solutions adapted to different system clock frequencies other than 240 MHz.

Resource usage is specified after place and route.

XILINX® 7-SERIES FPGAS

Resource type	***A-7***	***K-7***	***V-7***	***V-US+***
Slice Register	626	626	625	626
Slice LUTs	844	844	844	838
Block RAM	0	0	0	0
DSP48	12	12	12	12
Occupied Slices	552	576	573	254 (CLB)
Clock freq (fixed)	240 MHz	240 MHz	240 MHz	240 MHz

Revision History

Revision	***Change description***	***Date***
1.0	Initial revision.	20/08/2021
1.1	Minor code changes and source clean-up.	14/03/2022
1.2	Major release. Added programmable bitrate settings of 5,10,15 and 20 Mbps. Added input FIFO to the encoder	23/02/2023