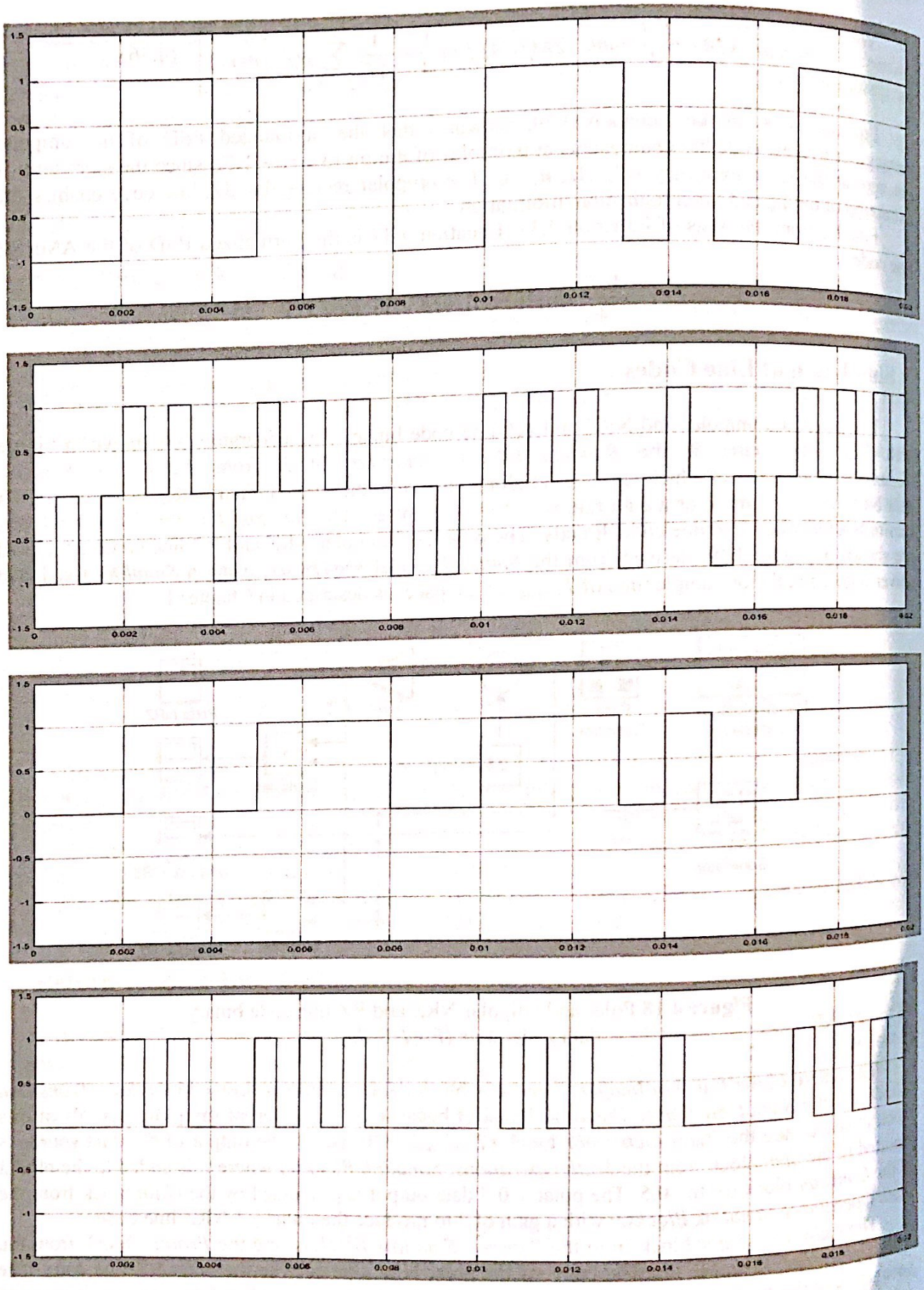

Figure 4.19 Polar NRZ (top), polar RZ, unipolar NRZ and unipolar RZ (bottom) line codes from the binary data generators (*Fig418.slx*).

The AMI NRZ and RZ and split-phase NRZ line code binary data generators are shown in Figure 4.20, derived from the *Simulink* model *Fig420.slx*, using a fixed-step solver with a simulation sample time of $T_{simulation}$ = 20 μsec, as described in Chapter 1. The unipolar NRZ generator in Figure 4.20 is included for reference. Although a *MATLAB Function Block* as described in Chapter 1 could be used to generates these lines codes, *Simulink* blocks provide adequate support.

The AMI NRZ line code requires a 1-bit memory since the pulse transmitted that represents an input binary data $b_k = 1$ alternates between $a_k = +1$ and $a_k = -1$. The *J-K Flip-Flop* block from the *Flip Flops, Simulink Extras* implements the 1-bit memory. The output of the *Random Integer Generator* block data source is processed by a *Rate Transition* block from the *Signal Attributes, Simulink Blockset* because of the different sampling periods of this block (1 msec) and the *Pulse Generator* block ($T_{simulation}$ = 20 μsec).

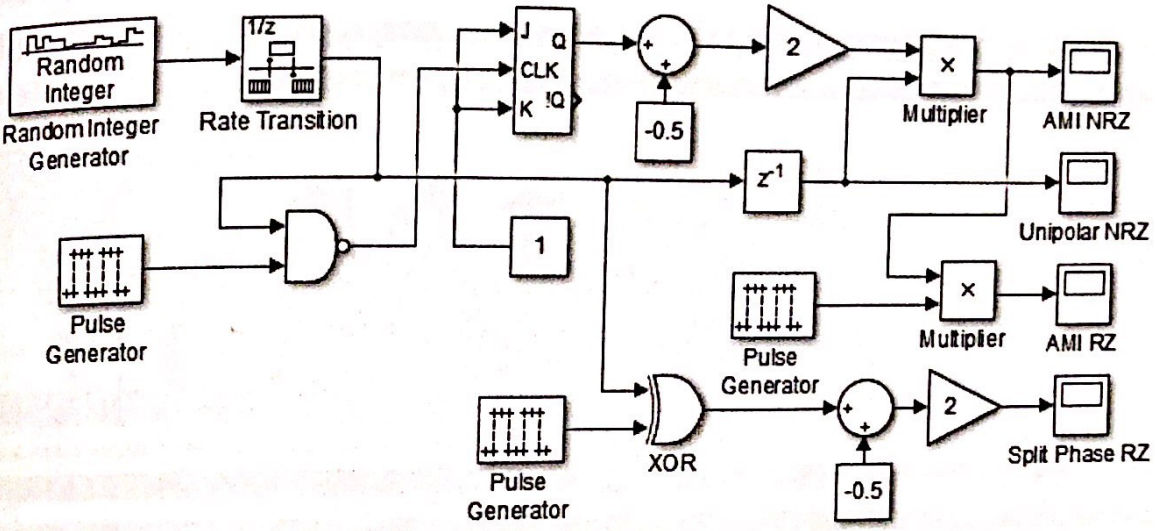

Figure 4.20 AMI NRZ and RZ and split-phase RZ line code binary data generators (*Fig420.slx*).

The output of the *Rate Transition* block as the data source is logically NANDed by the *Logical Operator* block from the *Logic and Bit Operations, Simulink Blockset* with the output of another *Pulse Generator* block to provide the *clock* input of the *J-K Flip-Flop* block. The parameters of this *Pulse Generator* block are an amplitude A = 1 V, a frequency f_o = 1 kHz, a pulse width τ = 20 μsec (the simulation sample time, $T_{simulation}$) and 0° phase offset.

The J and K inputs *J-K Flip-Flop* block are provided by a *Constant* block set to 1 block to provide a logic 1. With these control signals the *J-K Flip-Flop* block *toggles* the output state on each positive edge of the clock control signal. Since the binary data source is ANDed with a clock synchronized to the bit rate r_b = 1 kb/sec, only input binary data $b_k = 1$ toggles the *J-K Flip-Flop* block.

The output of the *J-K Flip-Flop* block is inputted to the *Sum* block from the *Math Operations, Simulink Blockset* where it is added to the output of the *Constant* block set to –0.5. The polar ± 0.5 data output is processed by the *Gain* block from the *Math Operations, Simulink Blockset* with a gain of 2 to produce the polar ± 1 AMI NRZ line code.

The *Pulse Generator* block from the *Sources, Simulink Blockset* and the *Product* block from the *Math Operations, Simulink Blockset* converts the polar AMI NRZ line code to the polar AMI RZ line code. The *Pulse Generator* block has parameters of an amplitude of 1 V, a period of 1 msec which is equal to data bit time T_b, a pulse width of 50% of the period or $T_b/2$ = 0.5 msec and a phase delay of 0°.

The split phase line NRZ line code is obtained as the logical XOR (exclusive OR) of the data source with the output of another *Pulse Generator* block with the same parameters as that for the AMI

RZ line code. The XOR is a *Logical Operator* block from the *Logic and Bit Operations*, *Simulink Blockset*.

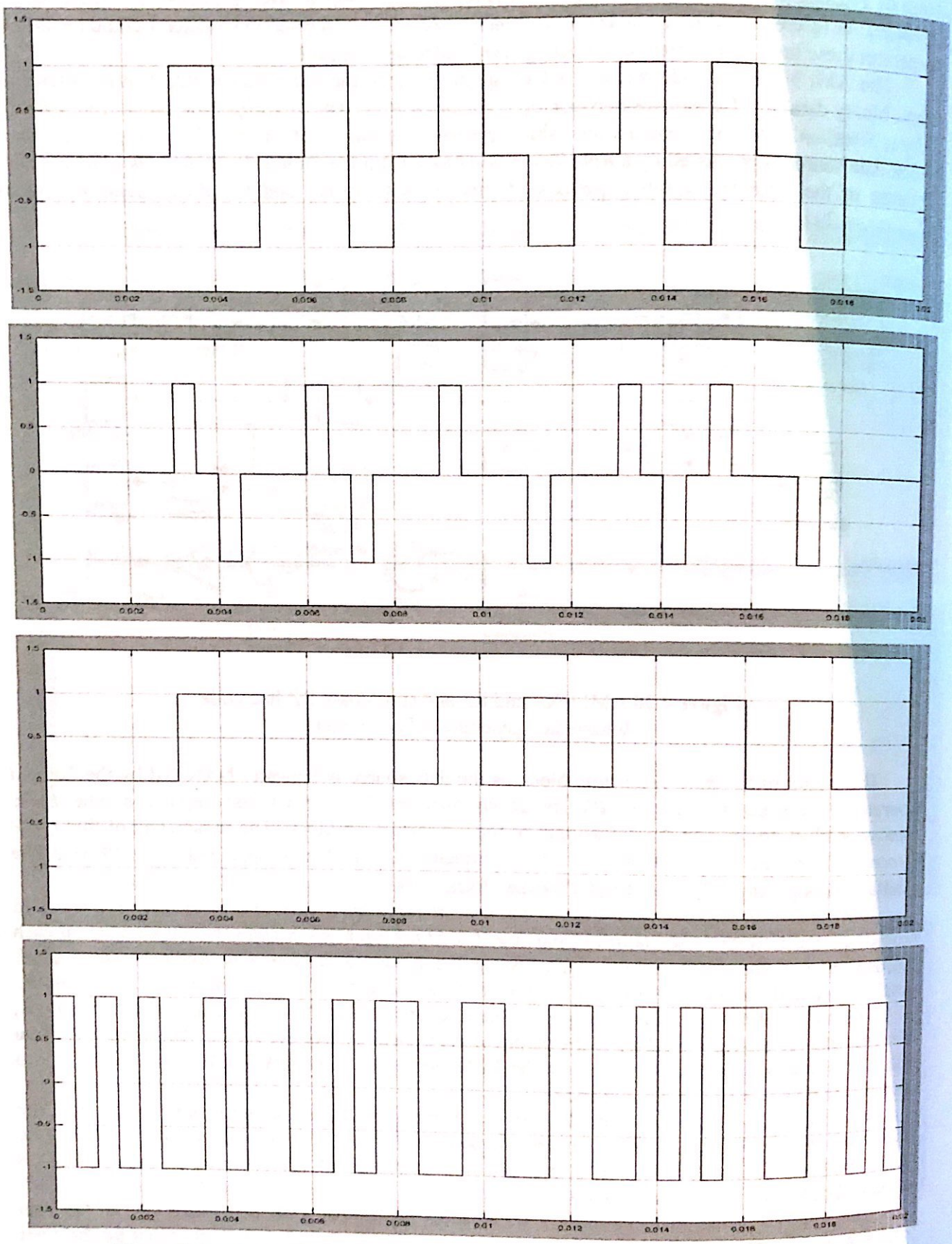

Figure 4.21 AMI NRZ (top), AMI RZ, unipolar NRZ and split-phase NRZ (bottom) line codes from the binary data generators (*Fig420.slx*).

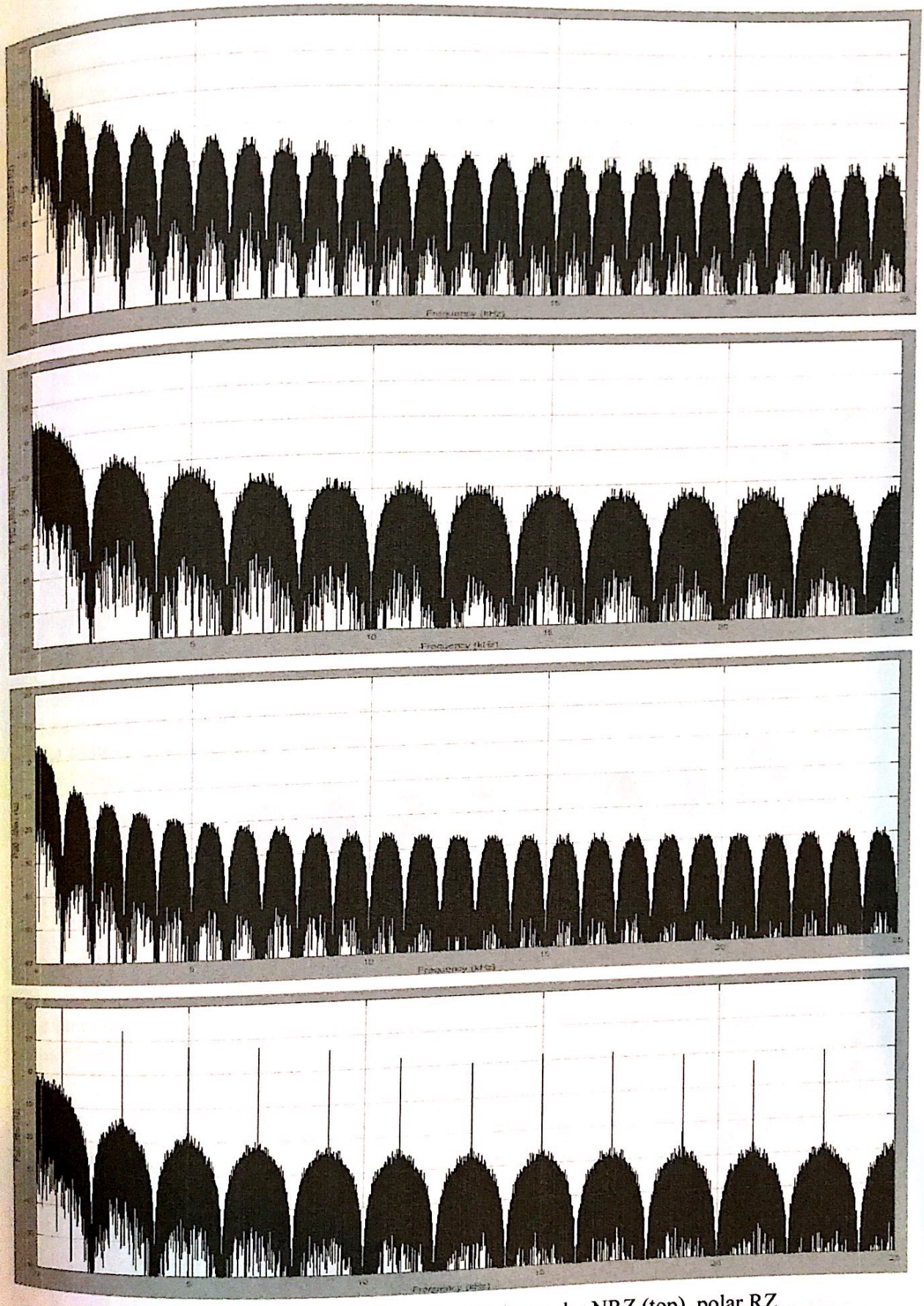

Figure 4.22 Power spectral density of a 1 kb/sec polar NRZ (top), polar RZ, unipolar NRZ and unipolar RZ (bottom) line codes (*Fig422.slx*).

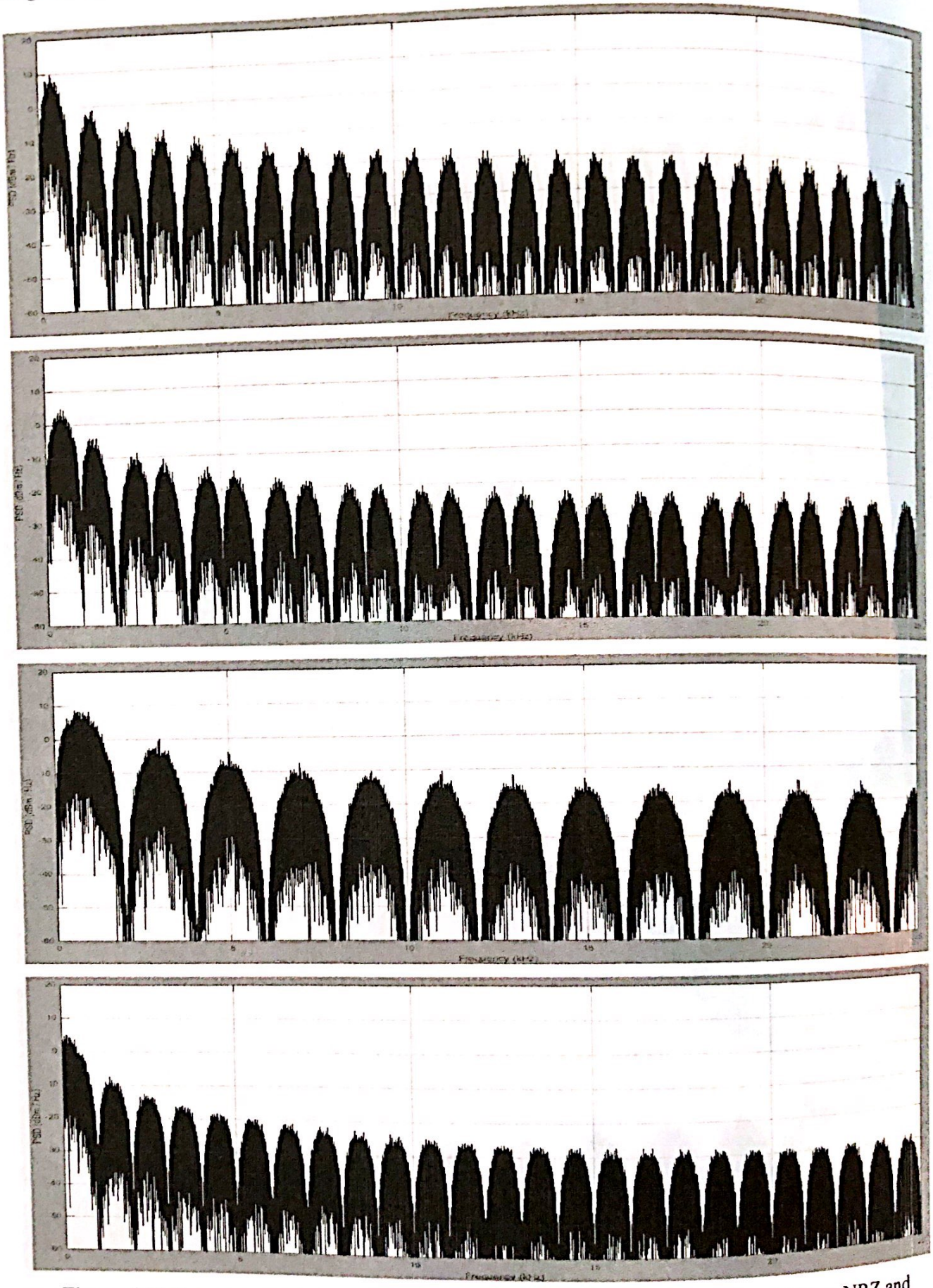

Figure 4.23 Power spectral density of a 1 kb/sec AMI NRZ (top), AMI RZ, split-phase NRZ and unipolar NRZ (for reference, bottom) line codes (*Fig423.slx*).

The AMI NRZ, the AMI RZ, the unipolar NRZ (for reference) and the split-phase NRZ line codes generated are shown in Figure 4.21, derived from the *Simulink* model *Fig421.slx*. Note that the *Pulse Generator* blocks in Figure 4.18 and Figure 4.20 are operated as sample based for the fixed-step solver in *MATLAB* and *Simulink*, as described in Chapter 1. The parameters are then derived from the nominal bit time, the pulse width and the 50 kHz simulation rate and results in a 50 sample period, a 25 or 1 sample pulse width and a delay of 0 or 1 samples.

The normalized (R_L = 1 Ω) power spectral density (PSD) of a 1 kb/sec polar NRZ, polar RZ, unipolar NRZ and unipolar RZ line code is derived from the *Simulink* model *Fig422.slx*. Setting the *Simulation Stop Time* to 5.24288 sec results in 262 144 (2^{18}) samples at an 50 kHz simulation rate and a spectral resolution of approximately 0.19 Hz and the PSD in dBm is shown in Figure 4.22.

The normalized (R_L = 1 Ω) PSD of the AMI NRZ and RZ, split-phase NRZ and unipolar NRZ (for reference) line codes, derived from the *Simulink* model *Fig423.slx*, are shown in Figure 4.23 in dBm. The normalized PSD of these line codes are obtained by the *Spectrum Scope* block from the *Signal Processing Sinks*, *Signal Processing Blockset*.

From Equation 4.25, Equation 4.29, and Equation 4.33, the PSD of the polar NRZ and unipolar NRZ line codes in Figure 4.22 and the AMI NRZ line code in Figure 4.23 show approximate spectral nulls at multiples of the data rate $r_b = 1/T_b = 1$ kHz. The PSD of the unipolar NRZ line code shows the spectral impulse at $f = 0$ Hz (DC), while the PSD of the AMI NRZ line code has a magnitude of 0 at $f =$ 0 Hz.

The PSD of the unipolar RZ line code in Figure 4.22 shows spectral impulses not only at $f = 0$ Hz (DC), but at odd multiples ($(2n + 1)$ kHz, $n = 1, 2, \ldots$) of the data rate $r_b = 1/T_b = 1$ kHz. The spectral impulses at the even multiples ($2n$ kHz, $n = 1, 2, \ldots$) of the data rate are suppressed because of the spectral nulls at twice the data rate because of the sinc^2 term in Equation 4.36. These spectral impulses facilitate the synchronization of the received bit rate r_b in a receiver.

From Equation 4.35, Equation 4.36, and Equation 4.37, the PSD of the polar RZ and unipolar RZ line codes in Figure 4.22 and the AMI RZ line code in Figure 4.23 show approximate spectral nulls at multiples of twice the data rate $2r_b = 2/T_b = 2$ kHz. However, the PSD of the AMI RZ line code also shows additional spectral nulls at multiples of the data rate $r_b = 1/T_b = 1$ kHz because of the $\sin^2$ term in Equation 4.37. From Equation 4.34, the split-phase NRZ PSD also shows approximate spectral nulls at multiples of twice the data rate $2r_b = 2/T_b = 2$ kHz but has a magnitude of 0 at $f = 0$ Hz (DC).

Pulse Code Modulation

Pulse code modulation (PCM) represents a sampled and quantized analog baseband signal as a sequence of encoded baseband pulses using line codes [Carlson08]. A PCM baseband system is shown in Figure 4.24, derived from the *Simulink* model *Fig424.slx*, using a fixed-step solver with a simulation sample time of $T_{simulation}$ = 1.5625 μsec, as described in Chapter 1. The simulation rate then is 1/ $T_{simulation}$ = 640 kHz, which is an integer multiple of the PCM bit rate r_b = 64 kb/sec here.

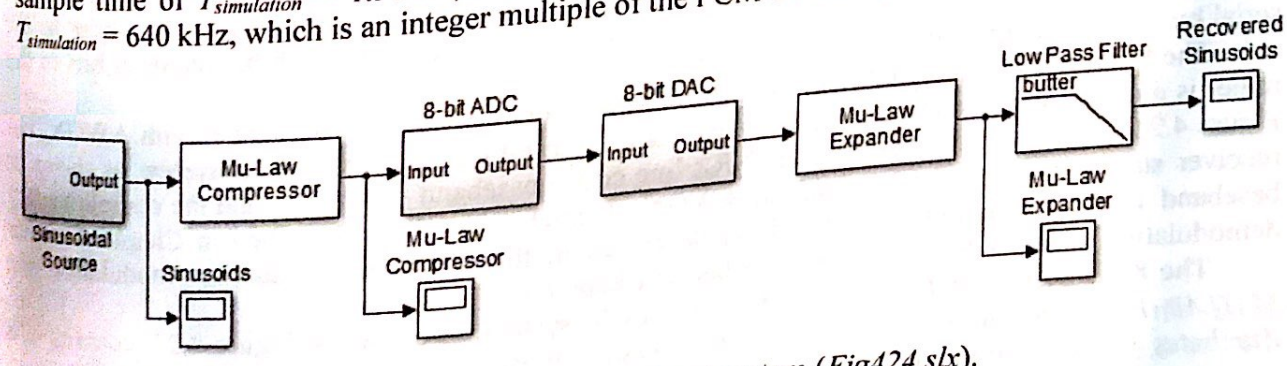

Figure 4.24 8-bit PCM baseband system (*Fig424.slx*).

The PCM baseband digital communication system in Figure 4.24 uses an NRZ line code with a μ-Law compressor, 8-bit analog-to-digital converter (ADC) with serial data output, 8-bit digital-to-analog converter (DAC) with serial data input, μ-Law expander and a lowpass filter (LPF). The digital communication channel is a direct connection between the baseband transmitter and receiver without AWGN.

The simulated sinusoidal source *Simulink Subsystem* is the sum of three sinuoids with parameters of an amplitude and frequency of ± 1 V at 500 Hz, ± 0.5 V at 1.5 kHz and ± 0.2 V at 2.5 kHz and a peak amplitude of approximately 1.15 V, as shown in Figure 4.2. The *μ-Law Compressor* block and *μ-Law Expander* block, both from the *Source Coding, Communication Blockset*, have parameters of $\mu = 255$ and a peak signal magnitude $V_{max} = 1.28$ V, as given by Equation 4.10.

The output of the *μ-Law Compressor* block is inputted to the 8-bit ADC *Simulink Subsystem* provides quantized analog data at a sampling rate $f_s = 8$ kHz as binary NRZ serial data output with $r_b = 8$ bits × 8 kHz = 64 kb/sec, as shown in Figure 4.25. The 8-bit ADC consists of a *Sample and Hold* block from the *Signal Operations, Signal Processing Blockset*, the 8-bit *Uniform Encoder* block from the *Quantizers, Signal Processing Blockset*, an *Integer to Bit Converter* from the *Utility Blocks, Communication Blockset* and a *MATLAB Function Block.*

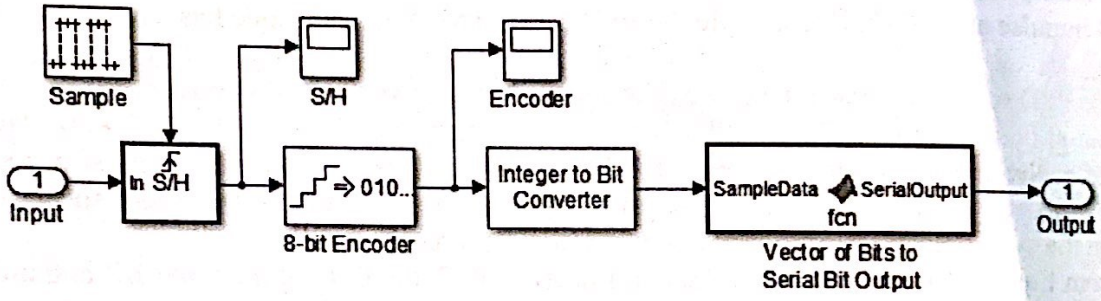

Figure 4.25 8-bit ADC *Simulink Subsystem* of the 8-bit PCM system in Figure 4.24.

The 8-bit ADC *Simulink Subsystem* sample clock is provided by the *Pulse Generator* block to the *Sample and Hold* block with parameters of an amplitude of 1 V, a period of 125 μsec ($f_s = 8$ kHz), a nominal pulse width of 10% of the period and 0 sec phase delay. The 8-bit *Uniform Encoder* block converts the simulated analog baseband signal to a binary number with parameters of 8 bits of resolution, unsigned integer output, a maximum positive input voltage $V_{maxp} = 1.27$ V and a maximum negative input voltage $V_{maxn} = -1.28$ V.

The output of the 8-bit *Uniform Encoder* block is a *MATLAB* and *Simulink* vector integer data type in the range 0 to 255 which is converted to a vector of 8 binary bits by the *Integer to Bit Converter* block from the *Utility Blocks, Communication Blockset*. The output of the *Integer to Bit Converter* block is inputted to a *MATLAB Function Block* which converts the 8-bit vector of bits to the serial binary NRZ output, as shown in Figure 4.26.

The baseband PCM digital communication system does not feature a channel with AWGN, but rather is directly connected to the serial binary NRZ input 8-bit DAC *Simulink Subsystem*, as shown in Figure 4.24. The direct link between the NRZ line coded baseband transmitter and the complimentary receiver structure can be broken and an AWGN channel inserted, as described in Chapter 2, other baseband line codes could be utilized, as described in this Chapter, or bandpass modulation and demodulation methods employed, as described in Chapter 3.

The 8-bit DAC *Simulink Subsystem* of the PCM system, as shown in Figure 4.27, consists of a *MATLAB Function Block*, a *Data Type Conversion* block from the *Signal Management, Signal Attributes* blockset from the *DSP System Toolbox* and an 8-bit *Uniform Decoder* block from the *Quantizers, Signal Processing Blockset*. The *MATLAB Function Block* converts the serial binary NRZ input to an 8-bit unsigned integer output, as shown in Figure 4.28.

The *Data Type Conversion* block then provides the requisite data type manipulation and outputs an unsigned 8-bit integer. The *Uniform Decoder* block converts the input integer to a number with parameters of 8 bits of resolution, a maximum positive output voltage $V_{maxp} = 1.27$ V and a maximum negative output voltage $V_{maxn} = -1.28$ V, which is inputted to the *μ-Law Expander* block as shown in Figure 4.24.

The output of the *μ-Law Expander* block is inputted to an *Analog Filter Design* block from the *Filter Designs, Signal Processing Blockset* which provides the lowpass filter (LPF) to smooth and recover the analog signal, as shown in figure 4.24. The parameters of the LPF are selected to be a 9-pole Butterworth filter with a cutoff frequency f_{cutoff} of 3.75 kHz. The maximum frequency in the simulated analog signal input is 2.5 kHz and the Nyquist frequency is $f_s/2 = 4$ kHz here. The 8-bit baseband PCM system thus produces an analog signal output in the final range of –1.28 V to 1.27 V for the recovered sinusoids.

8-bit ADC/Vector of Bits toSerial Bit Output

```
1     function SerialOutput = fcn(SampleData)
2     %#codegen
3 -   persistent SmpNum
4
5 -   if isempty(SmpNum)
6 -       SmpNum = 1;
7     end
8
9 -   SerialOutput = SampleData(SmpNum);
10 -  SmpNum = SmpNum + 1;
11 -  if (SmpNum == 9)
12 -      SmpNum = 1;
13    end
14
15    end
```

Figure 4.26 8-bit vector of bits input to NRZ serial bit output for the *MATLAB Function Block* in Figure 4.25.

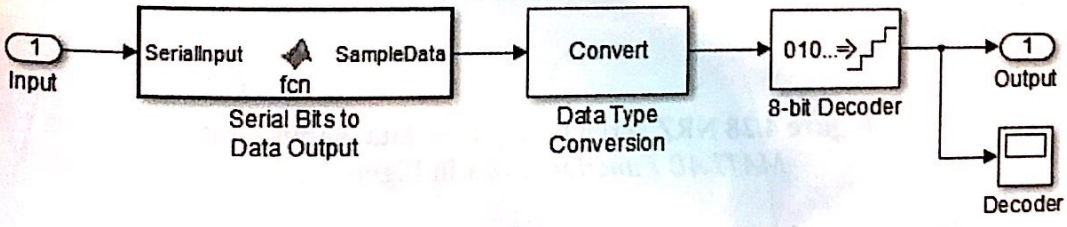

Figure 4.27 8-bit DAC *Simulink Subsystem* of the 8-bit PCM system in Figure 4.24.

The simulated analog signal input is shown in Figure 4.2 and the output of the *μ-Law Compressor* block is shown in Figure 4.17 (top). The output of the *Encoder* block of the 8-bit ADC *Simulink Subsystem* in the range from 0 to 255 and the output of the *Decoder* block of the 8-bit DAC *Simulink Subsystem* in the range from –1.28 V to 1.27 V is shown in Figure 4.29.

The 8-bit DAC *Simulink Subsystem* output reproduces the quantized data that is processed by the *μ-Law Compressor* block and the 8-bit ADC in Figure 4.24. The similarity of these signals demonstrates that the 8-bit parallel data at 8 ksamples/sec is transmitted correctly as a serial 64 kb/sec NRZ signal from the ADC to the DAC in Figure 4.24.

Note that most of the quantization levels in the 8-bit DAC output data are at amplitudes approaching V_{max} = 1.28 V here and are due to the effect of the *μ-Law Compressor* block, as given by Equation 4.10 and shown in Figure 4.15. The output of the *μ-Law Expander* block then restores the essential detail of the analog signal input at these amplitudes, as shown in Figure 4.29.

8-bit DAC/Serial Bits toData Output

```
1    function SampleData = fcn(SerialInput)
2    %#codegen
3
4 -  persistent SmpDat SmpNum SmpDatOld
5
6 -  if isempty(SmpNum)
7 -      SmpNum = 0;
8 -      SmpDat = 0;
9 -      SmpDatOld = 0;
10 -     SampleData = 0; %#ok<NASGU>
11   end
12
13 -  SmpNum = SmpNum + 1;
14 -  if (SmpNum == 9)
15 -      SmpNum = 1;
16 -      SampleData = SmpDat;
17 -      SmpDatOld = SmpDat;
18 -      SmpDat = SerialInput;
19   else
20 -      SampleData = SmpDatOld;
21 -      SmpDat = SmpDat + SerialInput*2^(SmpNum-1);
22   end
23
24   end
```

Figure 4.28 NRZ serial bit input to data output for the *MATLAB Function Block* in Figure 4.27

The 9-pole Butterworth LPF removes the quantized steps in the output of the *μ-law Expander* block, as shown in Figure 4.29 (bottom). The restored simulated analog signal displays a phase delay of approximately 0.4 msec. The PCM data transmission delay in sampling and acquisition is $2/f_s$ = 0.25 msec. The LPF is responsible for the remainder of the delay of approximately 0.15 msec.

After an initial *overshoot*, there is also a slight degradation in the peak amplitude of the restored analog signal due to the 8-bit resolution, companding and LPF of the PCM digital communication system. A comparison of the original analog signal, as shown in Figure 4.2, and the recovered analog signal, as shown in Figure 4.29 (bottom), shows a slight distortion in the waveform.

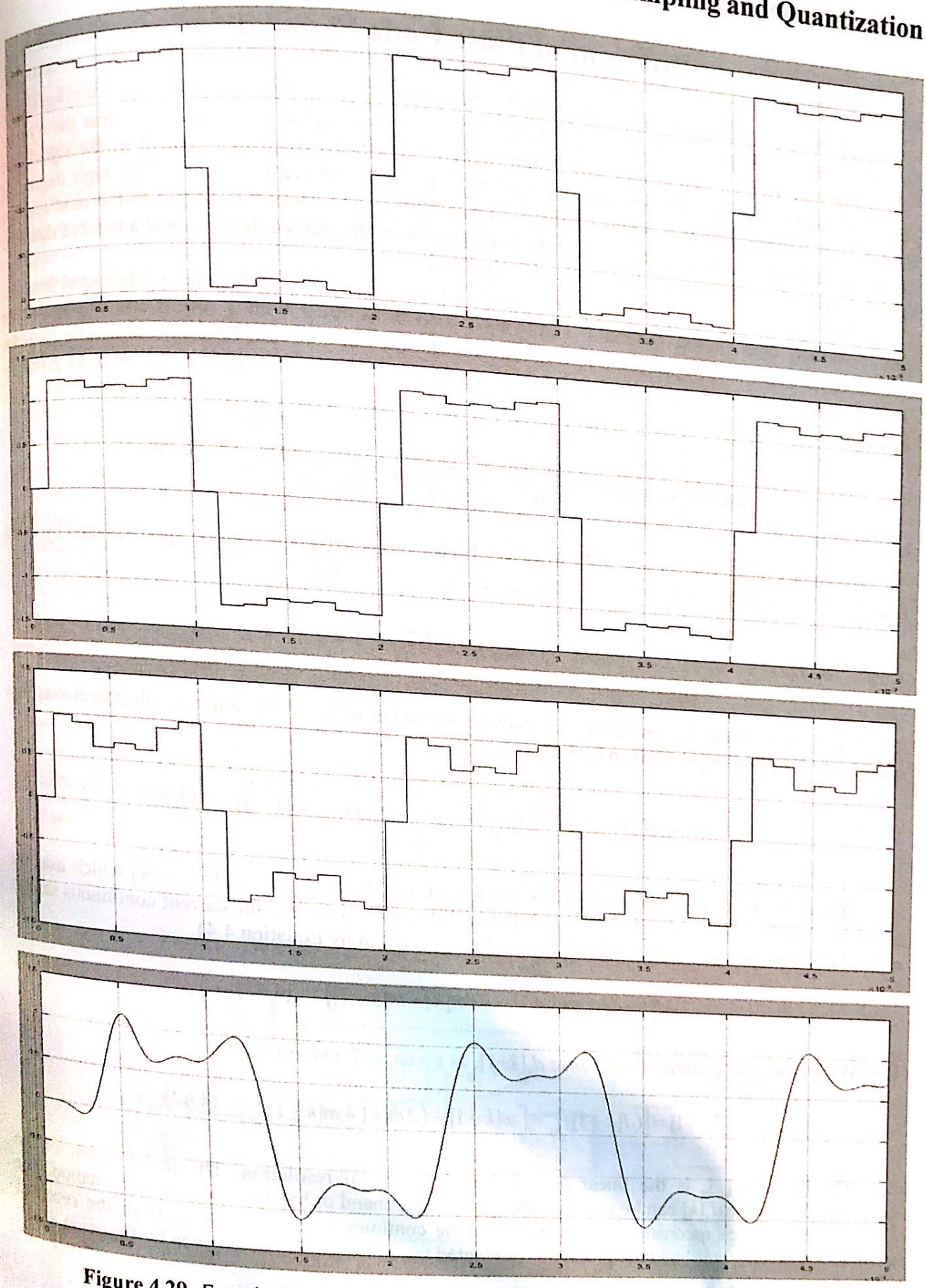

Figure 4.29 *Encoder* block output (top), *Decoder* block output, *μ-Law Expander* block and LPF output (bottom) of the 8-bit PCM system (*Fig424.slx*).

Differential Pulse Code Modulation

Differential pulse code modulation (DPCM) is an extension of baseband *delta modulation* (DM), as described in Chapter 2. If an analog baseband signal is sampled at a rate higher than the *Nyquist sampling rate* ($f_s > 2\ f_{max}$ where f_{max} is the highest significant frequency content in the signal) the sampled signal has a high degree of *correlation* between adjacent samples. This high degree of correlation implies that the signal does not substantially vary from one sample to the next and the *difference* between adjacent samples can be used to encode the transmitted signal at a reduced data rate which lowers the transmission bandwidth [Carlson08].

The correlation of the sampled signal allows the *prediction* of future values of the signal from the past behavior. The analog baseband signal $m(t)$ then is sampled with a continuous amplitude (not quantized) at the rate $f_s = f_{DPCM} = 1/T_s$ Hz. The next sample $m(t + T_s)$ can be predicted exactly from the Taylor series expansion for this signal (assuming that $m(t)$ has continuous derivatives), as given by Equation 4.38.

$$m(t+T_s) = m(t) + T_s \frac{\mathrm{d}\, m(t)}{\mathrm{d}t} + \frac{T_s^2}{2!}\frac{\mathrm{d}^2 m(t)}{\mathrm{d}t^2} + \frac{T_s^3}{3!}\frac{\mathrm{d}^3 m(t)}{\mathrm{d}t^3} + \ldots \qquad (4.38)$$

The exact prediction of $m(t + T_s)$ from the Taylor series expansion can be approximated by using only the first two terms of Equation 4.38, as given by Equation 4.39.

$$m(t+T_s) \approx m(t) + T_s \frac{\mathrm{d}\, m(t)}{\mathrm{d}t} \qquad (4.39)$$

Equation 4.39 can be rendered as a discrete sequence with $t = kT_s$ and the kth continuous (not quantized) sample as $m[k]$ is given by Equation 4.40.

$$m[k+1] \approx m[k] + T_s \left[\frac{m[k] - m[k-1]}{T_s}\right] = 2\,m[k] - m[k-1] \qquad (4.40)$$

This is the *first order linear predictor* for the current continuous sample $m[k+1]$ which uses the two prior samples $m[k]$ and $m[k-1]$ [Lathi09]. The difference d between the current continuous sample $m[k+1]$ and the continuous first order linear prediction is given by Equation 4.41.

$$d[k+1] = m[k+1] - \left(2m[k] - m[k-1]\right) \qquad (4.41)$$

DPCM transmits the *quantized difference* $d_q[k+1]$, as given by Equation 4.42.

$$d_q[k+1] = \left[(d[k+1])\right]_q = \left[m[k+1] - \left(2m[k] - m[k-1]\right)\right]_q \qquad (4.42)$$

The function $[\ ]_q$ is the linear quantizer with n bits of resolution. The transmission of the quantized difference $d_q[k]$ can be accomplished by any baseband or bandpass modulation method. The DPCM receiver then reconstructs an *estimate* of the continuous sample $m_e[k+1]$ from the received quantized difference $r_q[k]$ and the prediction generated from past estimates, as given by Equation 4.43.

$$m_e[k+1] \approx r_q[k+1] + 2\,m_e[k] - m_e[k-1] \qquad (4.43)$$