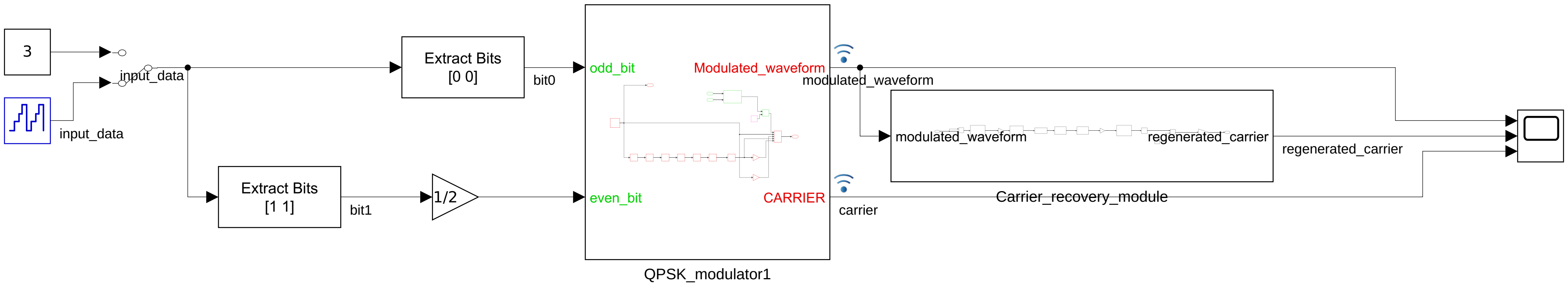
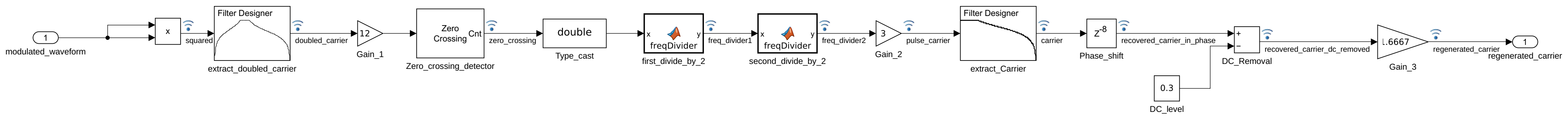
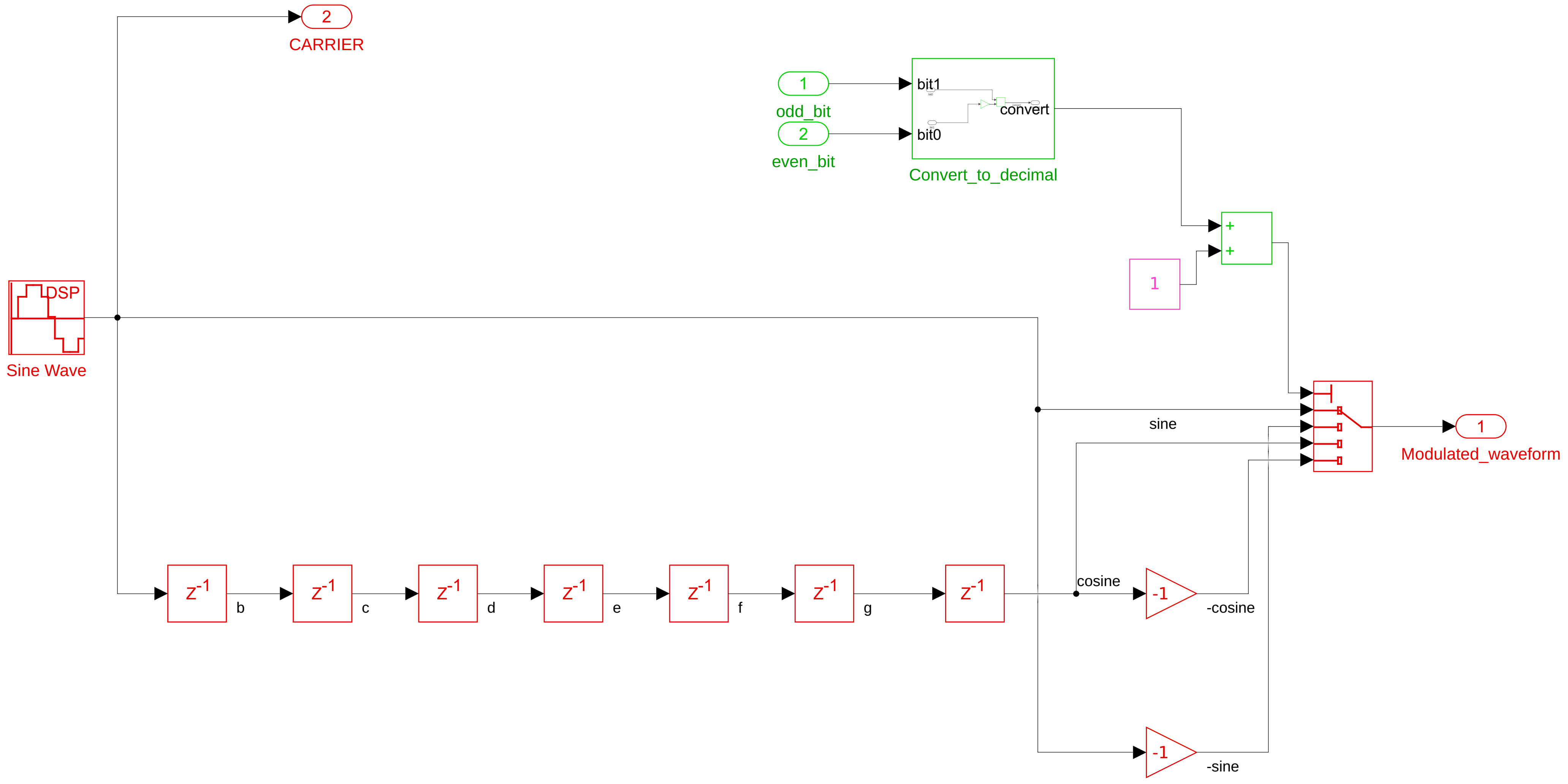
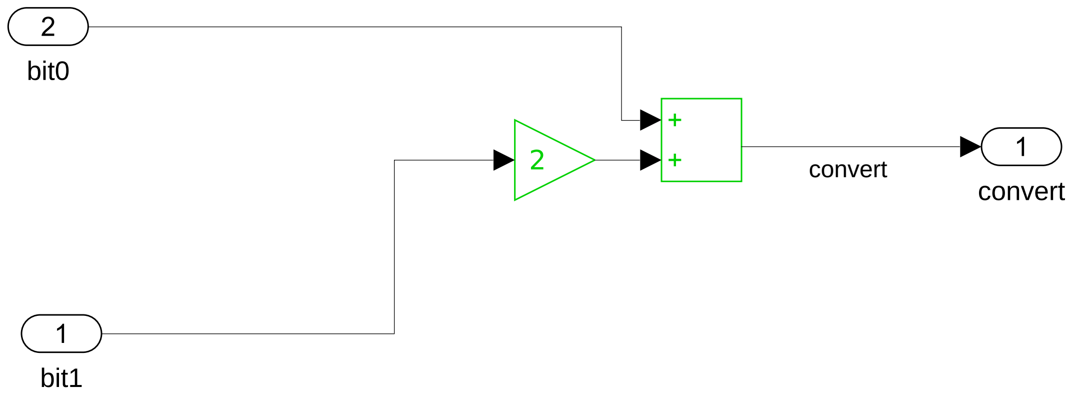

3
input_data
input_data
Extract Bits
[0 0]
bit0
Extract Bits
[1 1]
bit1
1/2
odd_bit
even_bit
Modulated_waveform
CARRIER
QPSK_modulator1
modulated_waveform
carrier
modulated_waveform
regenerated_carrier
Carrier_recovery_module
regenerated_carrier


1
modulated_waveform
x
squared
Filter Designer
extract_doubled_carrier
doubled_carrier
12
Gain_1
Zero Crossing
Cnt
Zero_crossing_detector
zero_crossing
double
Type_cast
x
freqDivider
y
first_divide_by_2
freq_divider1
x
freqDivider
y
second_divide_by_2
freq_divider2
3
Gain_2
pulse_carrier
Filter Designer
extract_Carrier
carrier
z^-8
Phase_shift
recovered_carrier_in_phase
+
−
DC_Removal
0.3
DC_level
recovered_carrier_dc_removed
.6667
Gain_3
regenerated_carrier
1
regenerated_carrier

```
function y = freqDivider(x)
% freqDivider  Frequency divider using persistent 2-sample input buffer
%
%   y = freqDivider(x) processes a single input sample x (scalar), and
%   returns a scalar output y, which goes high on every second rising edge.
%
%   This function is stateful and maintains internal buffer and edge counter
%   across calls using persistent variables.

    % Declare persistent variables
    persistent buf riseCount

    % Initialize persistent variables on first call
    if isempty(buf)
        buf = [0, 0];      % input buffer: [x(i-2), x(i-1)]
        riseCount = 0;     % rising edge counter
    end

    % Shift input buffer
    buf = [buf(2), x];

    % Default output
    y = 0;

    % Detect rising edge: x(i-1) <= 0 and x(i) > 0
    if buf(1) <= 0 && buf(2) > 0
        riseCount = riseCount + 1;

        if mod(riseCount, 2) == 0
            y = 1;  % Output high on every 2nd rising edge
        end
    end
end
```

```
function y = freqDivider(x)
% freqDivider  Frequency divider using persistent 2-sample input buffer
%
%   y = freqDivider(x) processes a single input sample x (scalar), and
%   returns a scalar output y, which goes high on every second rising edge.
%
%   This function is stateful and maintains internal buffer and edge counter
%   across calls using persistent variables.

    % Declare persistent variables
    persistent buf riseCount

    % Initialize persistent variables on first call
    if isempty(buf)
        buf = [0, 0];      % input buffer: [x(i-2), x(i-1)]
        riseCount = 0;     % rising edge counter
    end

    % Shift input buffer
    buf = [buf(2), x];

    % Default output
    y = 0;

    % Detect rising edge: x(i-1) <= 0 and x(i) > 0
    if buf(1) <= 0 && buf(2) > 0
        riseCount = riseCount + 1;

        if mod(riseCount, 2) == 0
            y = 1;  % Output high on every 2nd rising edge
        end
    end
end
```


2
CARRIER
1
odd_bit
2
even_bit
bit1
bit0
convert
Convert_to_decimal
+
+
1
DSP
Sine Wave
sine
1
Modulated_waveform
z-1
b
c
d
e
f
g
cosine
-1
-cosine
-sine


2
bit0
1
bit1
2
+
+
convert
1
convert