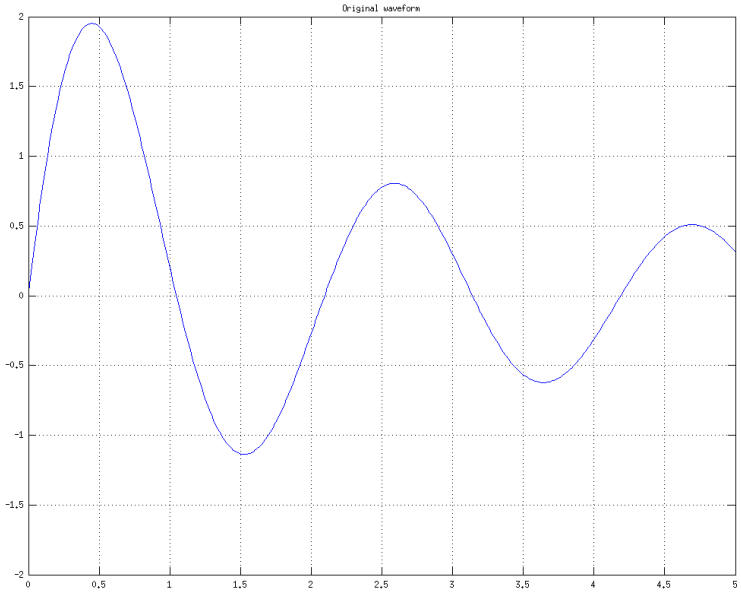
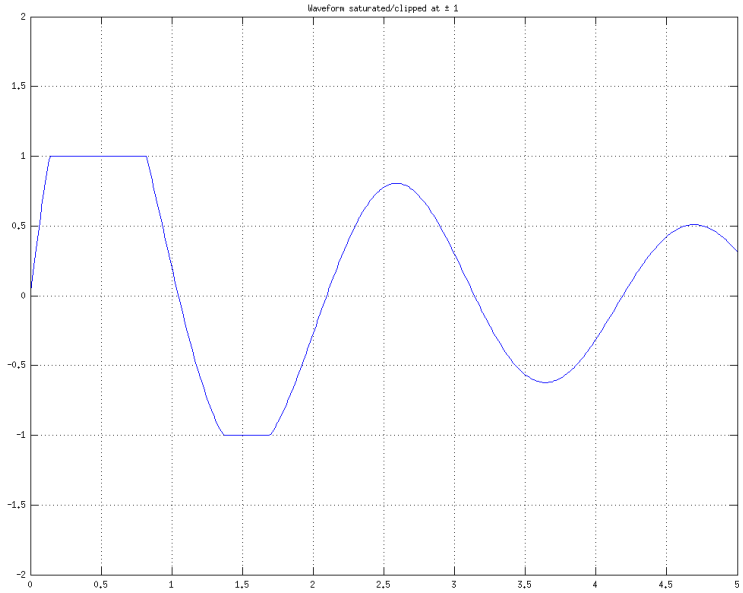
SATURATION
&
COMPRESSION

Saturation (or Clipping)

- Eliminates MSB bits
- It is common to saturate a signal after an operation which will or *may* cause the magnitude of a signal to increase (e.g., addition, subtraction, multiplication, (almost any operation), etc.)



Original waveform

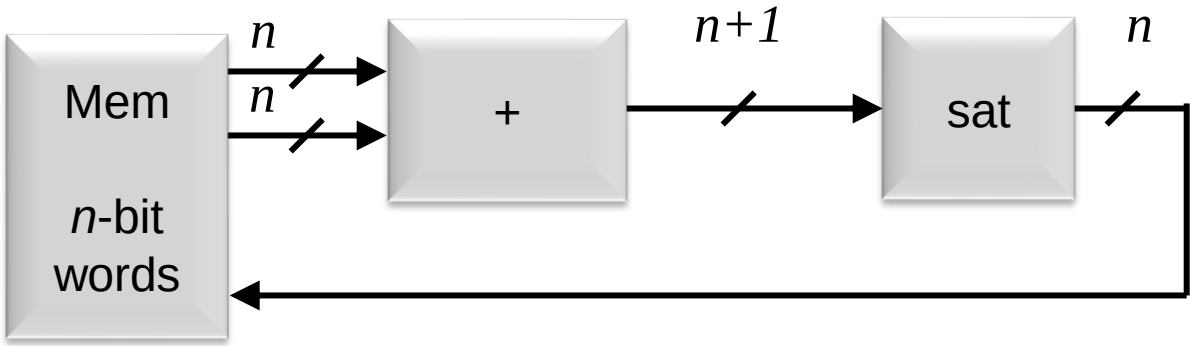


Saturated/clipped waveform

© B. Baas

Saturation (or Clipping)

- Saturation is a fundamental method to reduce the size of a word, such as after arithmetic operations
  - For example to maintain the word width for memory storage



- When *saturated*, bits are removed from the *MSB end* of the input word

```
XXXXXXXX
  YYYYY
```

© B. Baas

Saturation (or Clipping)

- Saturation is actually a 2-step process:
  1. Saturate the input to a maximum SAT_HI value and to a minimum SAT_LO value
  2. While it makes a lot of sense to choose SAT_HI and SAT_LO such that there are redundant MSB bits that can be dropped (shorten the word), this need not always be the case. When the saturation operation creates redundant and therefore unnecessary MSB bits, they should be dropped.
- It is often efficient to perform both steps simultaneously
- Ex: Output is saturated to a reduced-word size
  - input 4-bit 2's complement Range is [–8, +7]
  - output 3-bit 2's complement Range is [–4, +3]
- Ex: Output word size is not reducible
  - input 4-bit 2's complement Range is [–8, +7]
  - output saturated to [–5, +5] Requires 4 bits in the output word, so no word width reduction is possible

© B. Baas

Saturation (or Clipping)

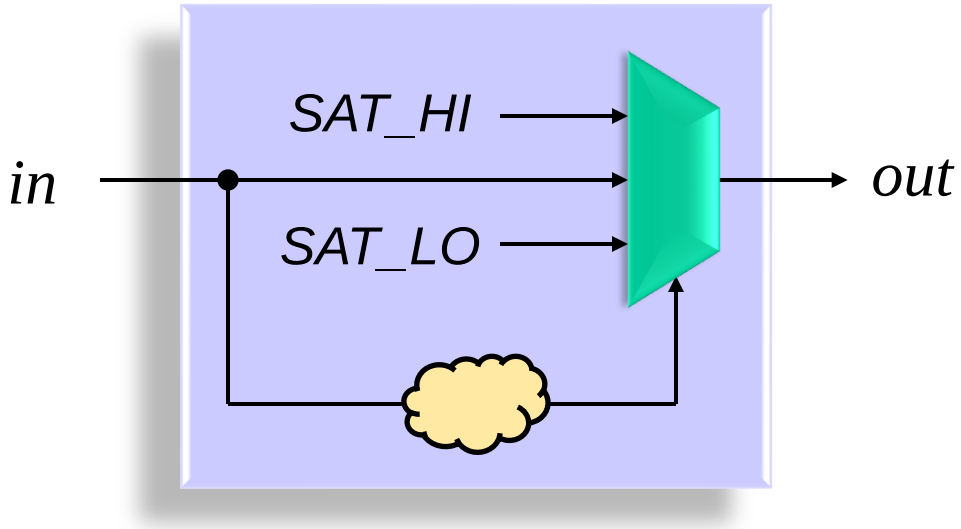
- Matlab code that produced previous example waveforms
- Copy, paste, and try it!

```
% Example saturated/clipped waveform
% 2009/02/03  Written

SatHi    =  1.0;
SatLo    = -1.0;
stepsize = 0.01;
a        = 0 : stepsize : 5;
index    = (a + stepsize) * (1/stepsize);  % matlab indexes start at 1 :-(
index    = round(index);                   % clear out VERY small offsets

b = 2.9 * sin(a*3) ./ (a+1);               % constants chosen to look nice

% plot original waveform
figure(1);clf;
plot(a,b);
grid on;
axis([0 5 -2 +2]);
title('Original waveform');

%print -dtiff 1.tiff     % 75 KB
print -dpng  1.png       % 16 KB, both look equally sharp

% saturate/clip original waveform
for l = index,
  if     b(l) > SatHi
    c(l) = SatHi;
  elseif b(l) < SatLo
    c(l) = SatLo;
  else
    c(l) = b(l);
  end
end

% plot saturated waveform
figure(2);clf;
plot(a,c);
grid on;
axis([0 5 -2 +2]);
title('Waveform saturated/clipped at \pm 1');

%print -dtiff 2.tiff     % 75 KB
print -dpng  2.png       % 16 KB, both look equally sharp
```

© B. Baas

Saturation (Clipping)

- A saturator checks for 3 possibilities:
  - $in > \text{SAT\_HI}$ or $in \geq \text{SAT\_HI}$
  - $in < \text{SAT\_LO}$ or $in \leq \text{SAT\_LO}$
  - else pass through
- Think of a saturator as a three-input mux



© B. Baas

Saturation (Clipping)

- Example:
  - 4-bit input
  - 3-bit output after saturation
- Both 0011 and 1100 could be either the "saturation" selection or the "pass through" selection and the output will be the same (correct).

0111 SAT_HI = 011
0110 SAT_HI = 011
0101 SAT_HI = 011
0100 SAT_HI = 011
0**011** *either sat or pass*
0010 in
0001 in
0000 in
1111 in
1110 in
1101 in
1**100** *either sat or pass*
1011 SAT_LO = 100
1010 SAT_LO = 100
1001 SAT_LO = 100
1000 SAT_LO = 100

© B. Baas

Saturation (Clipping)

Example: 4-bit input, ready for truncation to a 3-bit output after saturation

- If 0011 is pass through and 1100 is pass through, then the hardware can just look for when the MSB and MSB–1 bits are different.
- When the two bits are different, the MSB can not be simply dropped—the output must be saturated.
- Example verilog to saturate 1 bit:
  - ```
    if (in[MSB:MSB-1] == 2'b01)
      out = SAT_HI;
    else if (in[MSB:MSB-1] == 2'b10)
      out = SAT_LO;
    else
      out = in[MSB-1:0];
    ```

0111 SAT_HI = 011
0110 SAT_HI = 011
0101 SAT_HI = 011
0100 SAT_HI = 011
0011 *either sat or pass*
0010 in
0001 in
0000 in
1111 in
1110 in
1101 in
1100 *either sat or pass*
1011 SAT_LO = 100
1010 SAT_LO = 100
1001 SAT_LO = 100
1000 SAT_LO = 100

© B. Baas
    ```

Multi-Bit Saturation (Clipping)

- The method to saturate more than one bit is similar
- To saturate *S* bits, look for when the *S*+1 MSB bits are not all the same value
- This make intuitive sense—*S* bits can not be removed unless the *S*+1 MSB bits are all identical
- Example verilog to saturate 2 MSB bits:
  - ```
    if (in[MSB:MSB-2] == 3'b000 || in[MSB:MSB-2] == 3'b111)
      out = in[MSB-2:0];          // pass through w/o 2 MSB bits
    else if (in[MSB] == 1'b0)     // positive
      out = SAT_HI;
    else                          // negative
      out = SAT_LO;
    ```

© B. Baas
    ```

Saturation Bias Effects

- Saturation with simple hardware will usually clip to:
  - (+) 01111…111
  - (-) 10000…000
- But this treats positive saturated samples differently than negative saturated samples
- In effect, this creates a bias in the saturated output samples
- This may cause problems
  - Circuits sensitive to a DC bias; e.g., a signal path containing an accumulator or some RF circuits
  - Effect is worse for signals that saturate frequently
  - Effect is worse for outputs with narrow word widths

Examples	
11-bit	3-bit
+1023	+3
-1024	-4

© B. Baas

Saturation Bias Effects

- In cases when the non-symmetric rounding is not acceptable, clipping must be done in a symmetric manner
- That is, **`SAT_LO = -SAT_HI`**

	Examples 11-bit	3-bit
(+) 01111…111	+1023	+3
(-) 10000…001	-1023	-3

- The SAT_LO comparison is now more complex: the saturation detection circuit in the critical path must now look at all bits in the input word
- The case of **`in == (SAT_LO - 1)`** must be detected
- Example: 6-bit input saturated to a 5-bit output with values ±15
  - Requires special detection of **`in == -16`** (in which case out = –15)
  - Can be detected as the special saturation case when **`in == 110000`** (–16 alone) or **`in == 11000x`** (–16 and –15)

© B. Baas

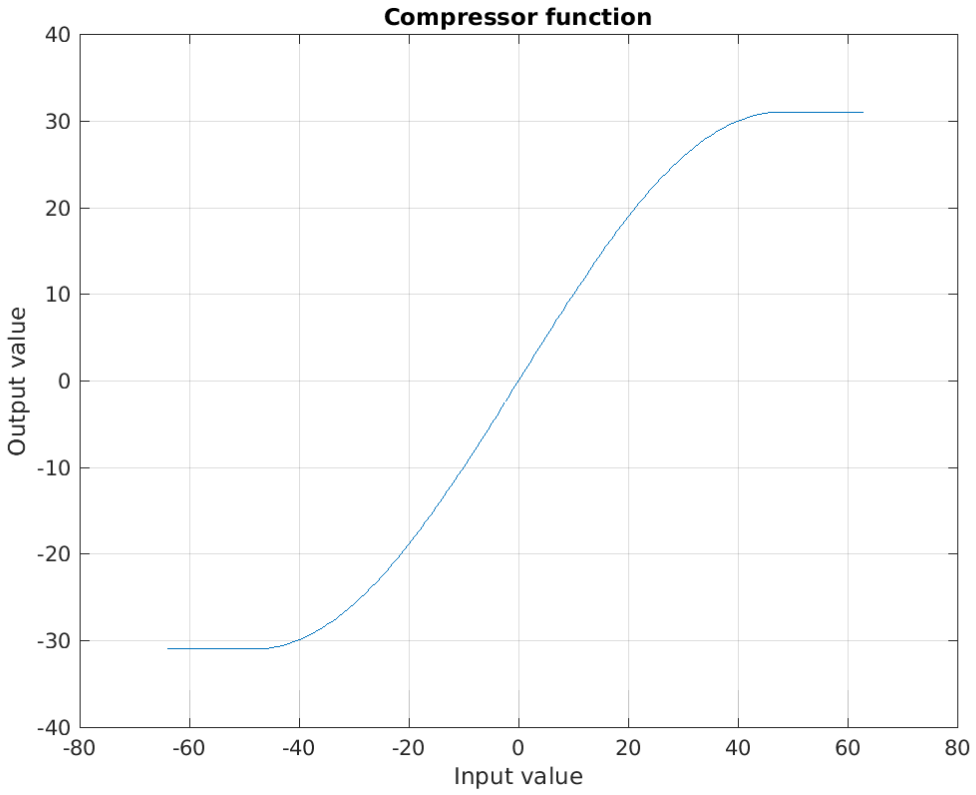
Compression: Gentle Saturation

- In some cases, the harsh corners of a saturated waveform produce undesirable characteristics
- A solution is to more gently saturate using a transfer function with a rounded shape that allows the signal to gently enter the saturation region
- Commonly called “Compression”
- Negative aspects:
  - The signal is distorted at a smaller magnitude
  - The hardware is much more complex
  - The smoothing function likely requires a lookup table

© B. Baas

Compression

- This is the transfer function of a compression function using **`sin()`** for the rounded sections and constant saturated values for large negative and large positive input values
- See the next slide for the matlab



© B. Baas

Compression

- This is the matlab code that generates the plots on the previous slide
- Copy, paste, and try it yourself!

```
% compressor.m
%
% This module compresses the values [-64:63] (which are the same values
% representable by a 7-bit 2's complement number) into an output range
% of [-31:+31] which almost fills the range of a 6-bit 2's complement
% number.
%
% Outputs are compressed according to a piecewise function comprised of
% partial sin waveforms and saturated regions.  They are not quantized or
% rounded in any other way.
%
% 2015/03/05  Cleaned up and documented better


PrintOn = 1;               % set to 1 to print tiff and png of figure(1)
XVals   = [-64:63];        % range of a 7-bit 2's complement number
XOffset = (1 - XVals(1)); % offset needed for array index to begin at 1
Xsat    = 48;              % +/- x value where full saturation begins
SatVal  = 31;              % both neg and pos. max +/- range of 6-bit 2's compl

for l = XVals,
  if     l < -Xsat
    b(l+XOffset) = -SatVal;
  elseif l < 0
    b(l+XOffset) = SatVal * sin((l/Xsat)*(pi/2));
  elseif l < Xsat
    b(l+XOffset) = SatVal * sin((l/Xsat)*(pi/2));
  else
    b(l+XOffset) = +SatVal;
  end
end

figure(1);clf;
plot(XVals, b);
hold on;
grid on;
%plot([-24:24], [-24:24]*(pi/2), 'r');   % not slope=+1 line
xlabel('Input value');
ylabel('Output value');
title('Compressor function');
%if (PrintOn)  print -dtiff compressor.tiff;  end
if (PrintOn)  print -dpng compressor.png;    end
```

© B. Baas