SATURATION
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](image1)

![Saturated/clipped waveform](image2)
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
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
Saturation (or Clipping)

- Matlab code that produced previous example waveforms
- Copy, paste, and try it!
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
Saturation (Clipping)

• A saturator checks for 3 possibilities:
  – $in > SAT_{HI}$ or $in \geq SAT_{HI}$
  – $in < SAT_{LO}$ or $in \leq SAT_{LO}$
  – else pass through

• Think of a saturator as a three-input mux
Saturation (Clipping)

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:

\[
\begin{align*}
\text{if} \ (\text{in}[\text{MSB}:\text{MSB}-1] == 2\text{'}b01) \\
&\quad \text{out} = \text{SAT_HI}; \\
\text{else if} \ (\text{in}[\text{MSB}:\text{MSB}-1] == 2\text{'}b10) \\
&\quad \text{out} = \text{SAT_LO}; \\
\text{else} \\
&\quad \text{out} = \text{in}[\text{MSB}-1:0];
\end{align*}
\]

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