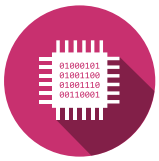


Binary representation of numbers

Labor Digital Design

Contents

1 Goal

This lab aims to better understand the binary representation of numbers. Different operators are analyzed and their behavior is tested using simulations.

The considered number representations are: Binary (unsigned and in two's complement), Hexadecimal and Decimal.

The following operations are analyzed: Inversion, Addition, Multiplication and Concatenation of numbers.

2 Circuit

The circuit **`NUM/toplevel`** (Figure 1) used in this lab contains a signal generator, which is composed of a counter generating the **`5bit`** signal **`Phase`**. This counter counts continuously and is used to generate a sine wave via a look-up table. The sine wave has an **`8bit`** resolution and is called **`Sinus`**.

This signal is then forwarded to 4 different blocks which perform the operations of inversion (**`inv[7:0]`**), addition (**`sum[7:0]`**), multiplication (**`prod[15:0]`**), and concatenation (**`concat[15:0]`**).

> Except for the sinus generator, all blocks are already implemented and are used to verify your analysis.

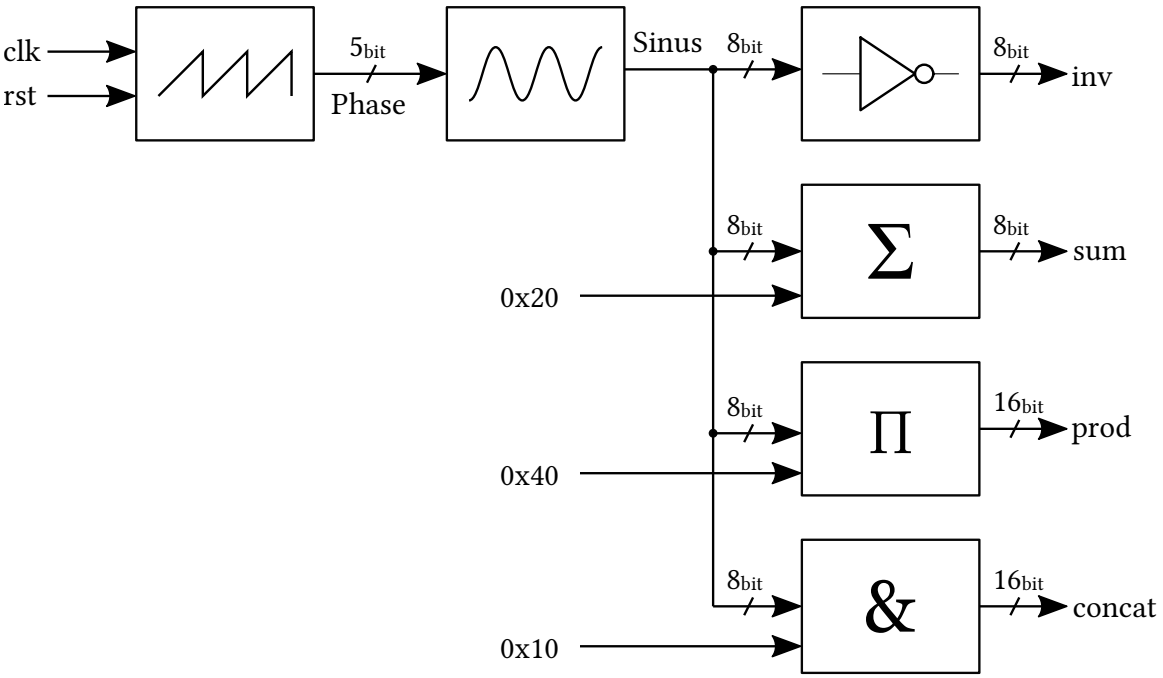


Figure 1 - Sinus generator and operators circuit

3 Sinus wave generator

To make the circuit work, a sinus wave generator must be implemented. This is achieved using an existing counter and a sine table.

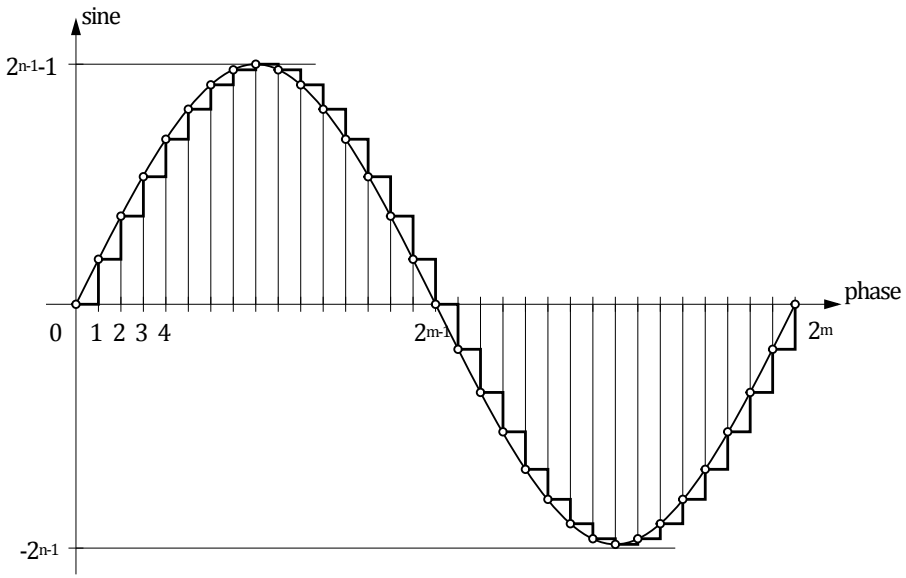


Figure 2 - Correlation between phase and sinus

3.1 Sine table

The sine table is a look-up table that provides the corresponding sine value (**`sine[7:0]`**) for each value of the phase (**`phase[4:0]`**) (Figure 2). The phase of the counter is considered as an unsigned 5-bit number and varies between 0 and 31. The sine value is a signed 8-bit number and varies between −127 and +127, and is coded in 2's complement.

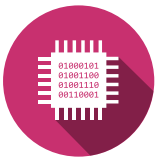
In the block **`NUM/sinewaveGenerator`**, a Very Highspeed Integrated Circuit Hardware Description Language (VHDL) code is provided. For each value of the phase, a sine value must be added.

```
1  ARCHITECTURE studentVersion OF sinewaveGenerator IS
2  BEGIN
3
4    table : process(phase)
5    begin
6      case to_integer(phase) is
7        when  0 => sine <= to_signed(16#00#,8);
8        when  1 => sine <= to_signed(16#00#,8);
9        ...
```

Listing 1 - Extract from the VHDL code of the sinus generator

The code line 7 can be understood as follows:

If **`phase`** has the value 0 (**`when 0 =>`**), the sine value (**`sine <=`**) is set to an 8-bit 2's complement value (**`to_signed(...,8)`**), the value is represented in hexadecimal (**`16#00#`**). This means that in the case of line 7, the sine value for the phase **`0`** is set to **`0`**.



Create an Excel table and calculate the sine value for each value of the phase (`0..=31`).

Complete the VHDL code of the table that creates the sine from the counter's phase.

3.2 Simulation

Perform a simulation to verify the correct behavior of the function generator. Your test bench is called **`sinewave_tb`**. The simulation should output the correct sine value for each phase value.

Simulate the Testbench **`NUM_test/sinewave_tb`** with the simulation file **`$SIMULATION_DIR/NUM1.do`**.

4 Operations

In this section, the different operations, inversion, addition, multiplication, and concatenation, are analyzed (Figure 1).

In the following sections, you must sketch the output curve of the different operations. The figures below can be used to sketch your curves (Figure 3, Figure 4, Figure 5 and Figure 7). The curves must contain the characteristic points on which the operations are applied to the signal.

4.1 Inverter

The inverter inverts the signal **`sinus[7:0]`** into **`inv[7:0]`**. The inversion is performed by the bitwise negation of the signal.

Sketch in the following Figure 3 the temporal behavior of the output signal (**`inv[7:0]`**) of the block.

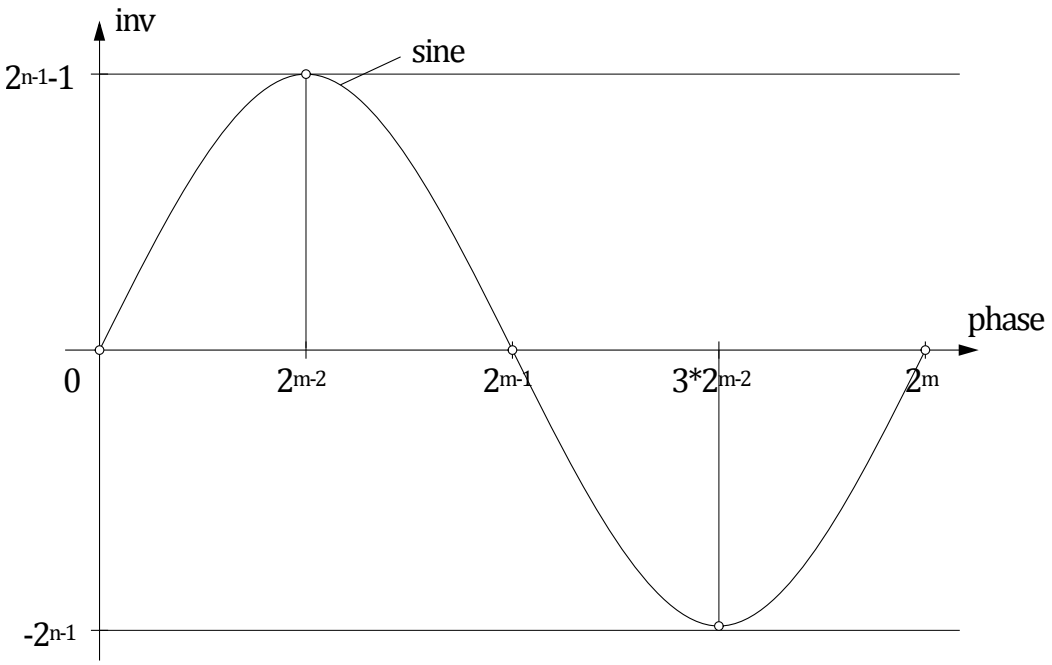
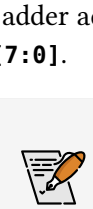


Figure 3 - Inverter

4.2 Adder

The adder adds the constant $20_h = 32_{10}$ to the signal **`sinus[7:0]`** and outputs the result in signal **`sum[7:0]`**.



Sketch in the following Figure 4 the temporal behavior of the output signal (**`sum[7:0]`**) of the block.

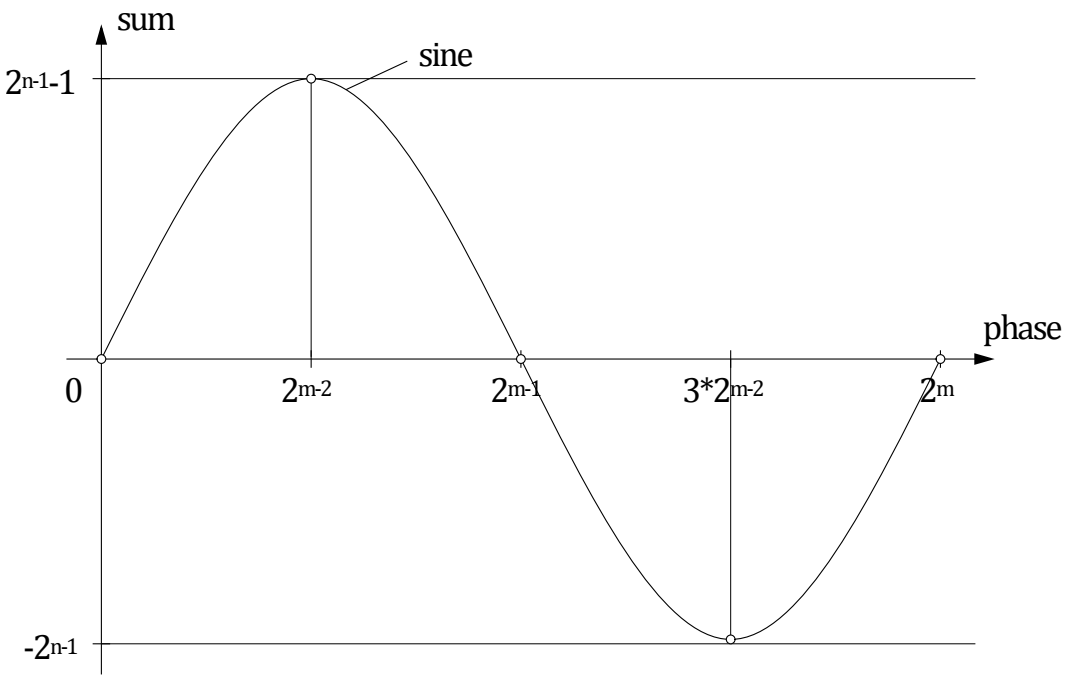


Figure 4 - Adder

4.3 Multiplier

The multiplier multiplies the signal **`sinus[7:0]`** by the constant $40_h = 64_{10}$ and outputs the result in signal **`prod[15:0]`**. Note that the result is a 16-bit signal.

Sketch in the following Figure 5 the temporal behavior of the output signal (**`prod[15:0]`**) of the block.

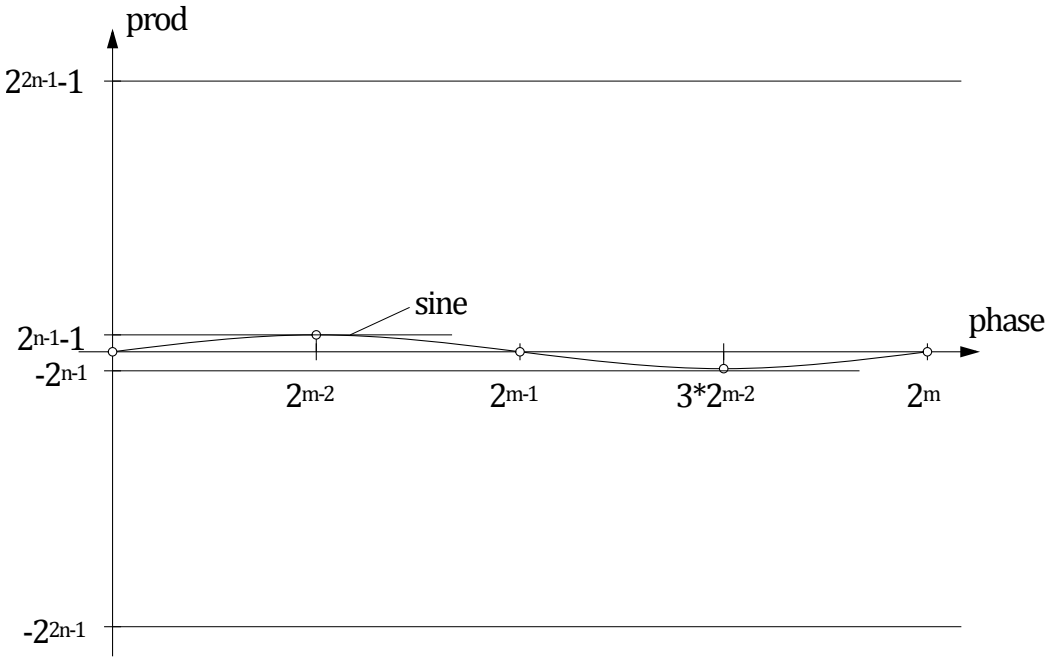


Figure 5 - Multiplier

4.4 Concatenation

The concatenator adds the concatenation constant (10_h) at the end of the signal. The result is output in signal **`concat[15:0]`**. Note that the result is a 16-bit signal.

A concatenation is an operation that connects two signals together. In the following case, 2 signals of 8-bit are combined into a 16-bit signal (Figure 6).

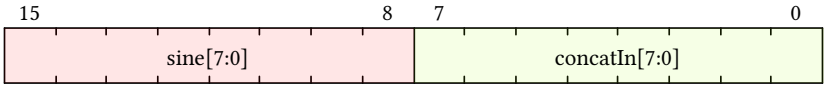


Figure 6 - Example of a concatenation

Sketch in the following Figure 7 the temporal behavior of the output signal (**`concat[15:0]`**) of the block.

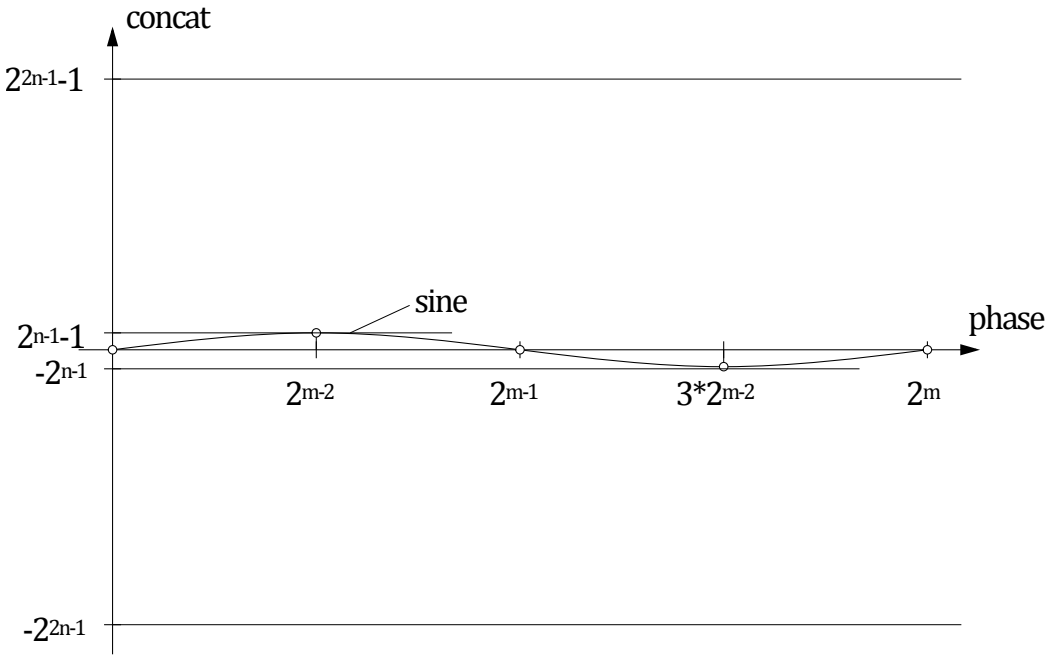


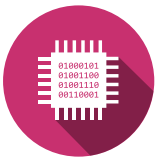
Figure 7 - Concatenation

4.5 Verification

Perform a simulation to verify the correct operation of the analyzed operators.

For each operator, find the corresponding mathematical equation. Use this information to validate the correct operation of the analyzed operators.

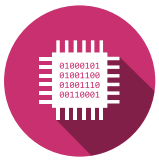
Simulate the Testbench **`NUM_test/sinewave_tb`** with the simulation file **`$SIMULATION_DIR/NUM2.do`**.



5 Checkout

Before leaving the laboratory, ensure you have completed the following tasks:

- [ ] Circuit Design
  - [ ] Verify that your Sinus generator **`/NUM/sinewaveGenerator`** generates a valid Sinuswave
- [ ] Operations
  - [ ] For each Operation have a figure drawn containing the most important points of the curve
  - [ ] Represent each operation with a mathematical formula
  - [ ] Verify your drawings against the simulation results
- [ ] Documentation and Projectfiles
  - [ ] Ensure all steps (design, convertions, simulations) are well-documented in your lab report.
  - [ ] Save the project to a USB stick or the shared network drive (**`\\filer01.hevs.ch`**).
  - [ ] Share files with your lab partner to ensure work continuity.



Glossary

VHDL **– Very Highspeed Integrated Circuit Hardware Description Language** 3, 3, 4