UNIVERSITÀ DI PISA

Electronics and Communications Systems

Multi-Standard Modulator

Federico Cavedoni

Academic Year
2023/2024

Index

1. Introduction 3
   Circuit Description 3
   Possible Applications 3
   Possible Architectures 4
2. Architecture description 5
3. VHDL Code 6
   Multi Standard Modulator 6
   Counter 7
   Phase Adder 8
   lut_table_65536_7bit 8
   Amplitude Multiplier 8
   QLUT Optimization 9
4. Test Phase 10
   Frequecy modulation 10
   Phase modulation 11
   Amplitude modulation 11
   Reset phase 12
   QLUT Comparison 12
5. Vivado report 13
6. Conclusion 13

1. Introduction

Circuit Description

This project presents the design and implementation of a digital **multi-standard modulator** capable of performing **Amplitude Shift Keying (ASK)**, **Frequency Shift Keying (FSK)**, and **Phase Shift Keying (PSK)** modulations.
The system employs a **Look-Up Table (LUT)** based on a **Numerically Controlled Oscillator (NCO)** that is able to generate the required waveforms for modulation.
The NCO uses a precision phase accumulator to achieve fine frequency control and efficient signal synthesis.
The modulator circuit accepts input parameters for frequency, phase, and amplitude, enabling flexible configuration for various modulation schemes. These parameters are encoded as 16-bit values for frequency and phase, and as a 4-bit value for amplitude. The output of the modulator is a 16-bit digital signal representing the modulated waveform.

Possible Applications

The designed multi-standard modulator is applicable in diverse communication systems requiring flexible digital modulation:

- **Wireless Communication**: Supports **QAM** in LTE and 5G, **PSK** in Wi-Fi, and **GFSK** in Bluetooth, enhancing data rates and spectral efficiency.
- **Data Transmission**: Enables **QAM** in DSL and cable modems, and **PAM** in optical networks, ensuring high-speed data transfer.
- **Signal Processing**: Used in Software-Defined Radios (SDRs) for dynamic modulation adaptation with **FSK**, **PSK**, and **QAM**, and in Cognitive Radios for efficient spectrum use.
- **IoT Devices**: Supports **ASK** and **PAM** for efficient communication in smart home and LPWAN devices like LoRa and NB-IoT.
- **Satellite Communications**: Facilitates **PSK** and **QAM** in DVB for broadcasting and GNSS for reliable data transmission over long distances.

Possible Architectures

The multi-standard modulator will be composed of various components that will allow you to perform the required function:

- The Numerically controlled Oscillator (NCO) which will generate a wave based on an input frequency
- The Phase Adder which will take care of adding the input phase to the generated wave
- The Look-up Table that allows you to obtain the value of the resulting wave with a simple lookup
- The Amplitude Multiplier which multiplies the signal by the input amplitude

These components allow to generate a wave and modulate it correctly based on the input frequency, phase and amplitude to obtain the modulated signal and use it for different types of communication and different protocols, as we have seen in the possible application paragraph.

2. Architecture description

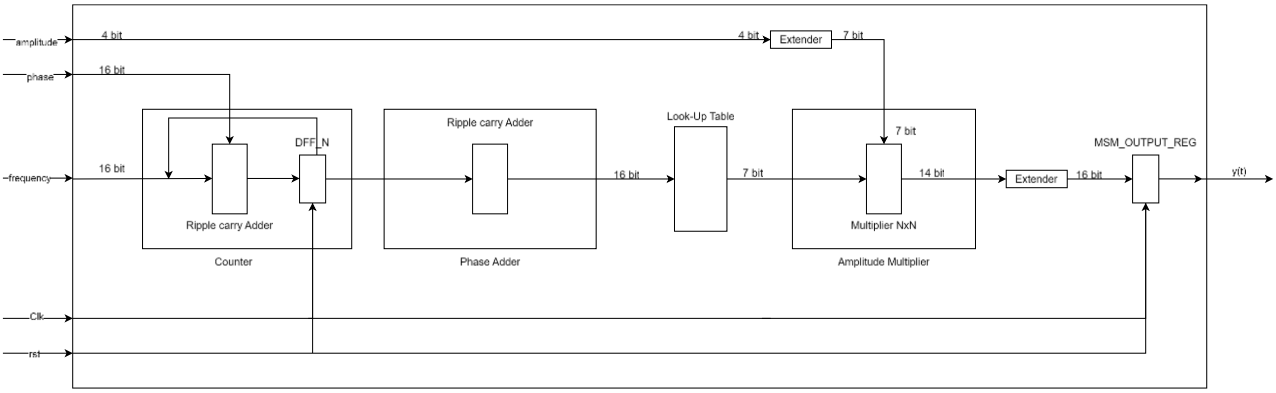

Figure 1: Architecture of the Multi-Standard Modulator

The system architecture, as introduced previously, is made up of various components that allow the correct application of the modulation functions.
As input to the system, we have the frequency word **fw** on 16 bits, the **phase** on 16 bits, the **amplitude** on 4 bits and the **clock** and **reset** signals. In output to the system, we only have the signal generated **yq** on 16 bits.

- The first block we find is the **Counter** which takes the clock, the reset and the frequency word as input and is responsible for generating the wave starting from the frequency multiplied by the k value of the counter. It returns as output k*fw on 16 bits.
- After the Counter there is the **Phase Adder** which takes care of adding the input phase on 16 bits to the k*fw frequency. In output I will therefore have k*fw + phase which is the index to be sent as input to the look-up table.
- After the phase adder there is the **look-up table (LUT)** which is a pre-compiled table with fixed input-output association. The table has a 16-bit input and a 7-bit output and returns a result that depends on the input index.
- After the lookup table there is the **Amplitude Multiplier** which takes as input the result of the lookup of the look-up table on 7 bits and the amplitude on 7 bits (4 originally, extended to 7 for the operation) and returns as output the signal multiplied by the input amplitude on 14 bits (7+7 from the multiplication).
- Finally, there is the **MSM Output Register** on 16 bits (14 originally from the multiplier extended to 16) which will output the modulated wave with the parameters passed as input.

3. VHDL Code

This section analyzes the various vhdl files that make up the system

Multi Standard Modulator

The **MSM.vhd** file is the main entity of the project and contains the declarations of the various subblocks such as counter and links starting from inputs to output.

```
entity MSM is
  generic (
    N : natural := 16;
    A : natural := 4;
    P : natural := 7;
    O : natural := 16
  );
  port(
    clk   : in std_logic;  -- clock of the system
    reset : in std_logic;  -- Asynchronous reset - active high

    fw : in std_logic_vector(N-1 downto 0);  -- input frequency word
    phase: in std_logic_vector(N-1 downto 0);  -- input phase
    amplitude : in std_logic_vector(A-1 downto 0);  -- input amplitude

    yq : out std_logic_vector(O-1 downto 0)   -- output waveform
  );
end entity;
```

Figure 2: MSM.vhd module

This module also takes care of extending the signals when necessary, as in the case of the input amplitude to the multiplier and the output output to the system.

```
amp_ext <= (P-1 downto A => '0') & amplitude;
```

Figure 3: Amplitude bit extension

```
mul_ext <= (O-1 downto 2*P => multiplier_output(2*P-1)) & multiplier_output;
```

Figure 4: Multiplier output extension

Counter

The **Counter.vhd** module consists of a ripple_carry_adder and a d-flip flop. Its operation is based on the fact that at each clock the previous output of the adder is added with the new input fw and therefore in output there is k*fw with k being the number of sums made by the adder. In this way we obtain a sine wave controlled with the input fw frequency.

```vhdl
entity Counter is
  generic (
    N : natural := 8
  );
  port (
    clk       : in  std_logic;
    a_rst_h   : in  std_logic;
    increment : in  std_logic_vector(N - 1 downto 0);
    en        : in  std_logic;
    cntr_out  : out std_logic_vector(N - 1 downto 0)
  );

end entity;
```

Figure 5: Counter.vhd module

```vhdl
FULL_ADDER_N_MAP : ripple_carry_adder
  generic map (Nbit => N)
  port map (
    a    => increment,
    b    => q_h,
    cin  => '0',
    s    => fullAdder_out,
    cout => open
  );

DFF_N_MAP : DFF_N
  generic map (N => N)
  port map (
    clk     => clk,
    a_rst_h => a_rst_h,
    d       => fullAdder_out,
    en      => en,
    q       => q_h
  );

-- Connect the output
cntr_out <= q_h;
```

Figure 6: Counter module made of an adder and a dff

Phase Adder

lut_table_65536_7bit

The **lut_table_65536_7bit.vhd** module consists of a precompiled table with 16-bit input and 7-bit output. The module simply takes care of doing a lookup inside the table and returning the output associated with the input index.

```
entity lut_table_65536_7bit is
  generic (
    N : integer := 16;
    P : integer := 7
  );
  port (
    address  : in  std_logic_vector(N-1 downto 0);
    lut_out : out std_logic_vector(P-1 downto 0)
  );
end entity;
```

Figure 9: lut_table_65536_7bit module

Amplitude Multiplier

The **Amplitude_Multiplier.vhd** module takes care of multiplying the output signal from the LUT by the input amplitude in order to obtain the amplitude modulated result.

```
entity Amplitude_Multiplier is
  generic (
    N : natural := 7
  );
  port (
    a : in  std_logic_vector(N - 1 downto 0);
    b : in  std_logic_vector(N - 1 downto 0);
    mul_out  : out std_logic_vector(2*N - 1 downto 0)
  );

end entity;
```

Figure 10: Amplitude multiplier module

The module is composed of a multiplier that multiplies the signal and the amplitude, both on 7 bits, obtaining the result on 14 bits. The width is first extended from the original 4 bits to 7 bits.

```
MULTIPLIER_N : multiplier_NxN
  generic map (N=> N)
  port map (
      A => a,
      B => b,
      P => mul_nxn_out
  );

-- Connect the output
mul_out <= mul_nxn_out;
```

Figure 11: Amplitude multiplier composed of a multiplier_NxN

QLUT Optimization

To perform an optimization regarding the space occupied by the LUT, which with 16 input bits begins to have considerable dimensions, the QLUT Optimization technique was adopted which consists in using only the first quadrant of the lookup table, thus saving a large amount of memory, denying the input and/or output of the LUT depending on the value of the last two bits of the input signal.

```
lut_address <= signal_out(N-3 downto 0) when (signal_out(N-2) = '0') else not(signal_out(N-3 downto 0));

LUT_16384 : qlut_table_16384_7bit
  generic map (N => N-2, P => P)
  port map(
    address  => lut_address,
    lut_out => lut_output
  );

  lut_output_mux <= lut_output when (signal_out(N-1) = '0') else not(lut_output);
```

Figure 12: QLUT Optimization

This technique allowed, with the same results, to have a considerable saving on occupied memory and lookup time.

4. Test Phase

To check the correct functioning of the multi-standard modulator, a testbench was developed to display the signal generated as the frequency, amplitude and phase vary and to check the correct functioning of the reset signal.

The testing phase was divided into various phases, each of which varied an input to then study the wave as it varied and ensure that the modulation occurred correctly.

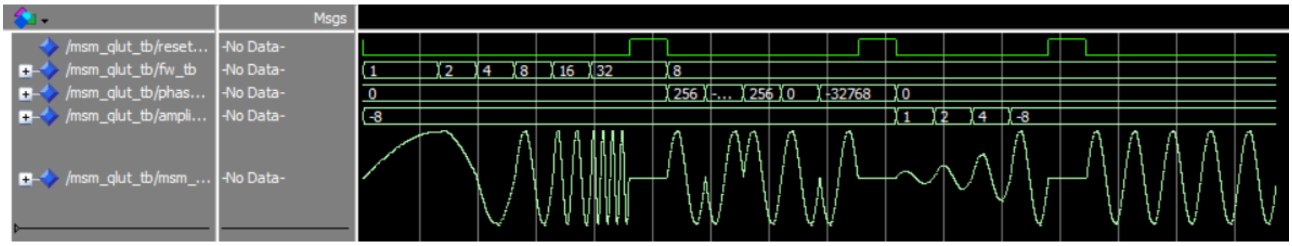

Figure 13: Test phase

Frequecy modulation

In this first phase the frequency is varied. It is clearly seen that as the input frequency varies, the output wave also varies and is therefore modulated in frequency.

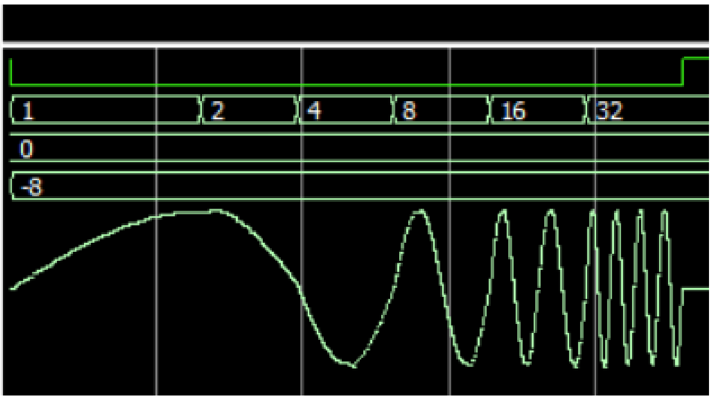

Figure 14: Frequency varying

Phase modulation

In this phase the phase of the wave is varied based on the input phase. It can be clearly seen that as the input varies, the phase of the outgoing wave also varies, which is therefore modulated in phase.

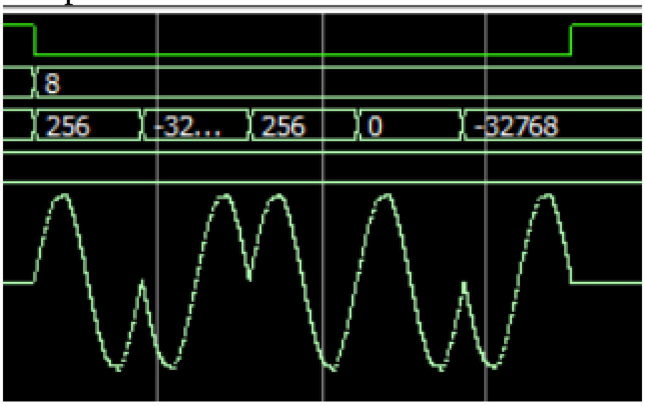

Figure 15: Phase varying

Amplitude modulation

In this phase the amplitude input to the system is varied. It is clearly seen that as the input amplitude varies, the output wave also varies, which is therefore modulated in amplitude.

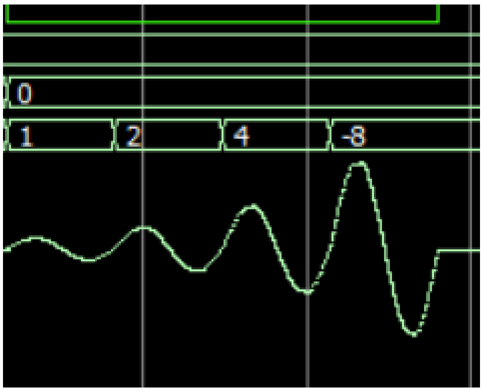

Figure 16: Amplitude varying

Reset phase

Tests were also done for the reset signal. As you can see, when the reset signal is active the system output is zero, therefore it can be stated that the reset is working correctly.

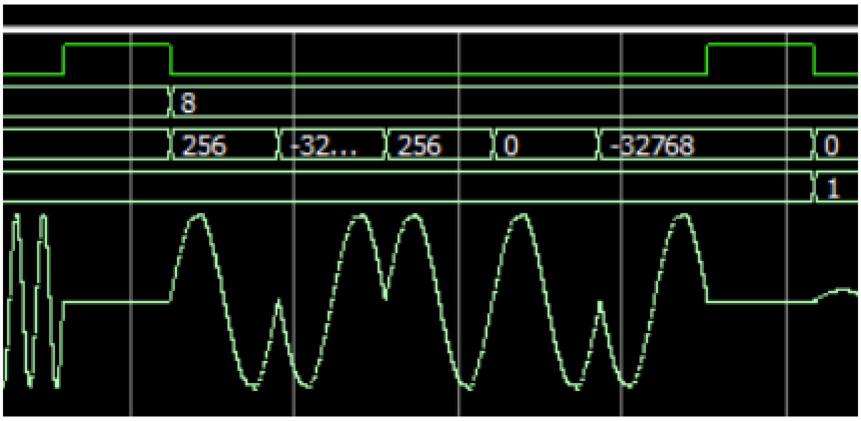

Figure 17: Reset working

QLUT Comparison

To prove that the results obtained with the LUT and the QLUT are identical, both waves were printed simultaneously. We can clearly see that the wave resulting from both lookup tables is identical and therefore we managed to obtain the same result with a lookup table that is a quarter of the original one, thus saving memory and obtaining much lower lookup times.

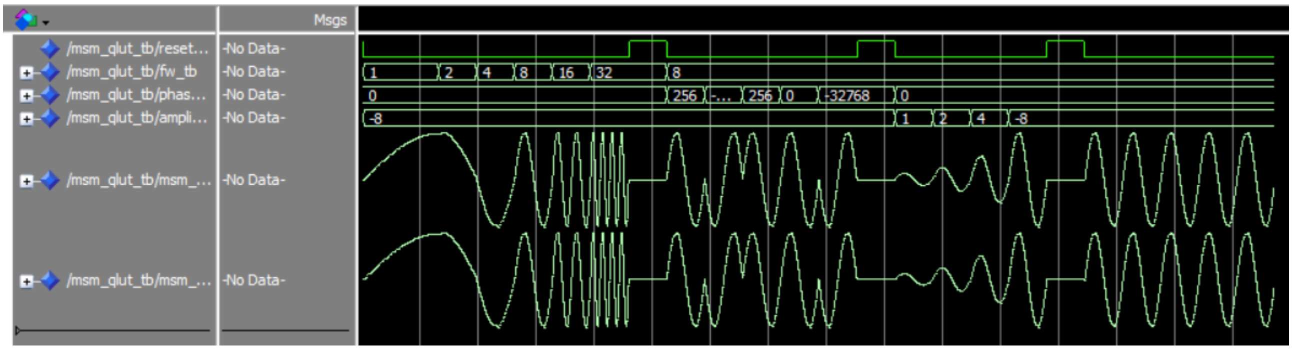

Figure 18: QLUT Comparison

5. Vivado report

6. Conclusion