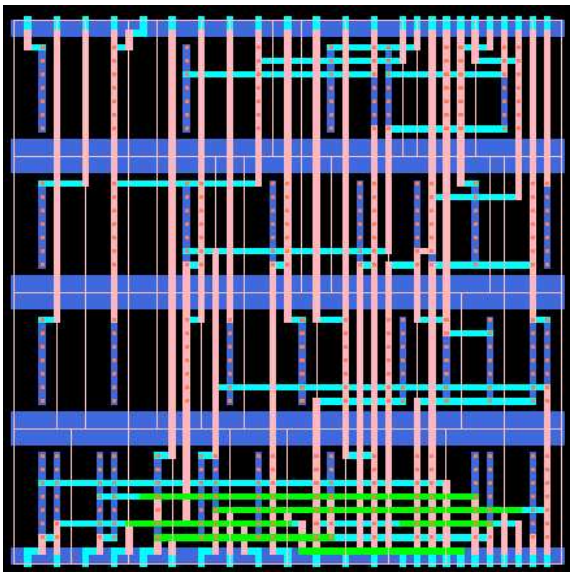
Simple FM Receiver

Arif E. Nugroho
<arif_endro@opencores.org>



VLSI Research Group ITB
LabTek VIII Institut Teknologi Bandung
Jl. Ganesha 10 Bandung 40141 West Java Indonesia

2005

Contents

List of Figures

1 Introduction

This design is a very simple implementation of FM Receiver for demodulating Frequency Modulated (FM) signal. The input signal it's self is a square wave signal that already modulated and digitized in the form 8 bit signed 2's complement.

1.1 Circuit Schematics

The works of this FM Receiver is based on PLL operation to capture FM signal. The design architecture it's self like ordinary PLL using phase detector, vco (realize using an nco), loop filter, and low pass filter.

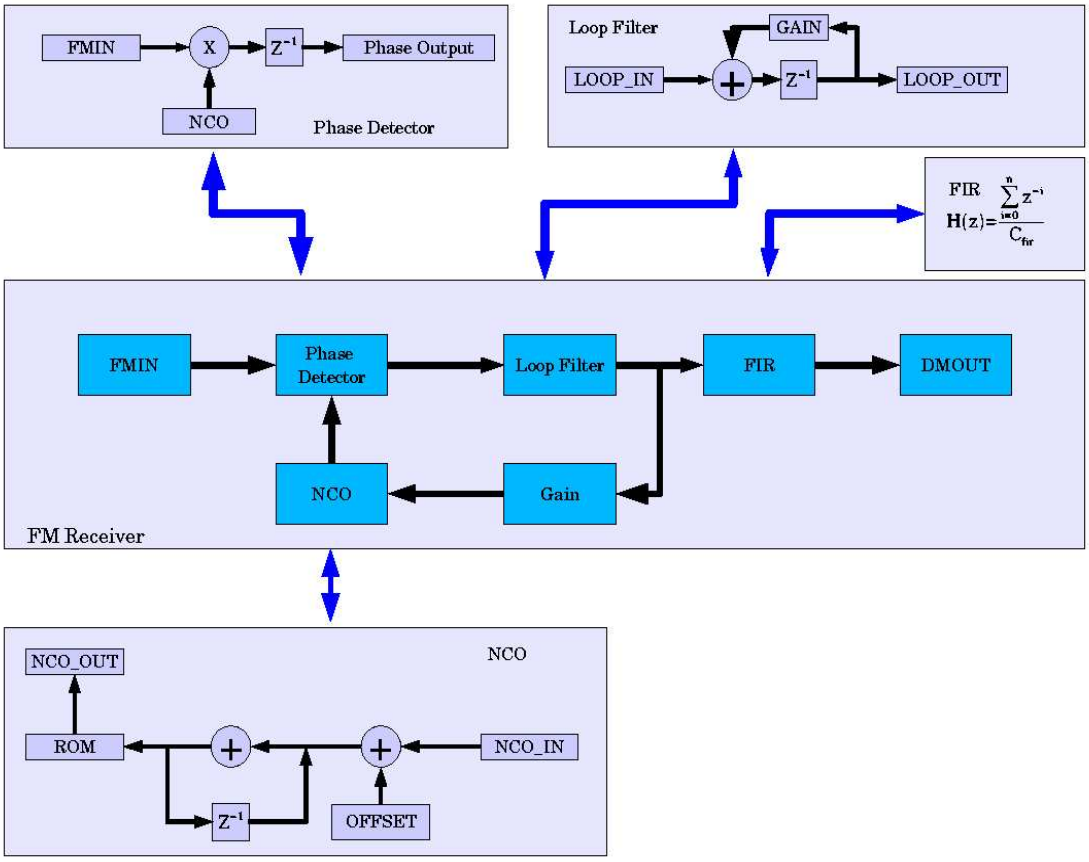


Figure 1-1: Architecture Schematics

In the PLL loop's, the FM signal it's self captured from phase difference betwee input signal and loop signal. This signal is raw signal of the original signal and the looks is not too good, After loop filtering the phase difference looks more clearly. finally, after filtering out in low pass filter the original signal can be recovered.

The recovered signal it's self not perfect but it's not too far from original signal. The shape of original signal suppose to be straigh line but the recovered signal not. There is still many noise in the

recovered signal.

2 Circuit Component

2.1 Core Component

- **fm**
  The core component, the connector between many component in the PLL loops.

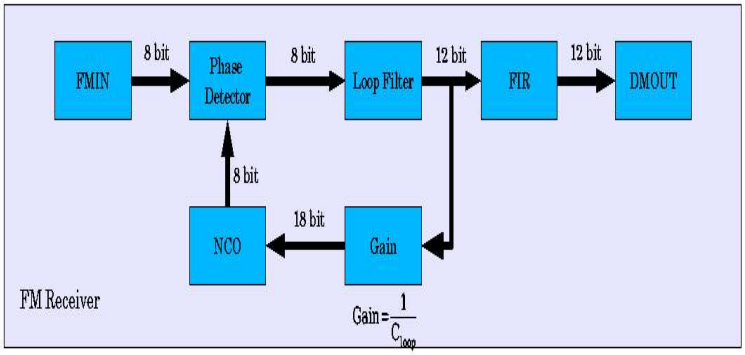


Figure 2-1: FM core component

2.2 Main Component

- **nco**
  The NCO functions it's like VCO in analog PLL.

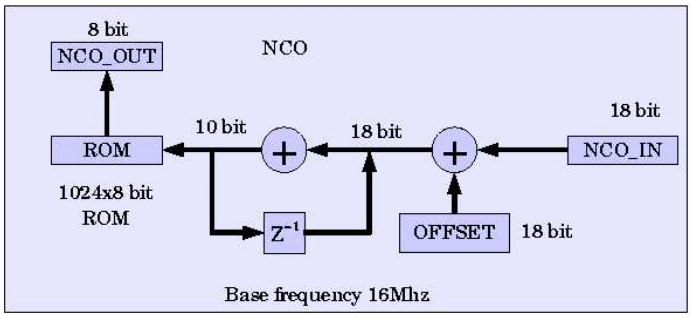


Figure 2-2: NCO block diagram

- **phase_detector**
  This component works by multiplying input signal and loop signal.

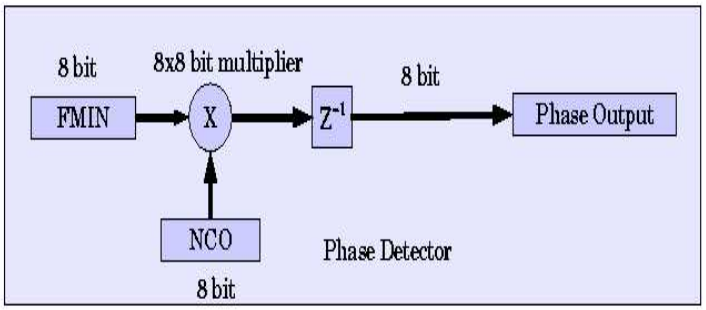


Figure 2-3: Phase detector block diagram

- **loop_filter**

  A low pass filter in the PLL loop.

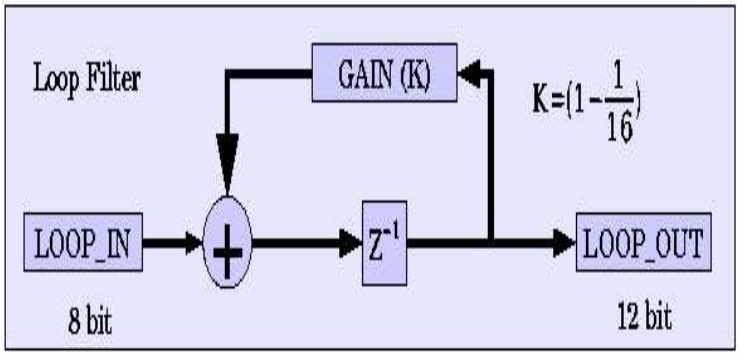


Figure 2-4: Loop filter block diagram

2.3 Arithmetic Component

- **adder**

  Adder implementation in this design is in the simplest form (e.g simple ripple carry adder), this adder implementations is the slowest compared with other type adder but this adder implementations is easy to understand. This adder is used for arithmetics operations in many component that exists in the PLL loops[1].

- **subtractor**

  Subtractor implementation used in this design is a modified version of adder. This is because the subtractor it's self a manipulations of adder[2].

[1] I will use a different type adder for next time, if you have any suggestion email me :)

[2] in 2's complement implementation negatif value of a number is equal to it's 2' complement. so subtraction is equal to addition to it's 2's complement

- **multiplier**

  Multiplier is used on the phase detector to multiply input signal and signal nco to get the phase different.

- **full adder**

  Full Adder, everyone already know about it's. In this design the full adder implementation is implemented using following relations[3]:

```
sum          <=  ((addend xor augend) xor carry_in);
carry        <=  ((addend xor augend) or  (carry_in and (addend or augend)));
```

3 Waveform

Here is the output waveform of the design.

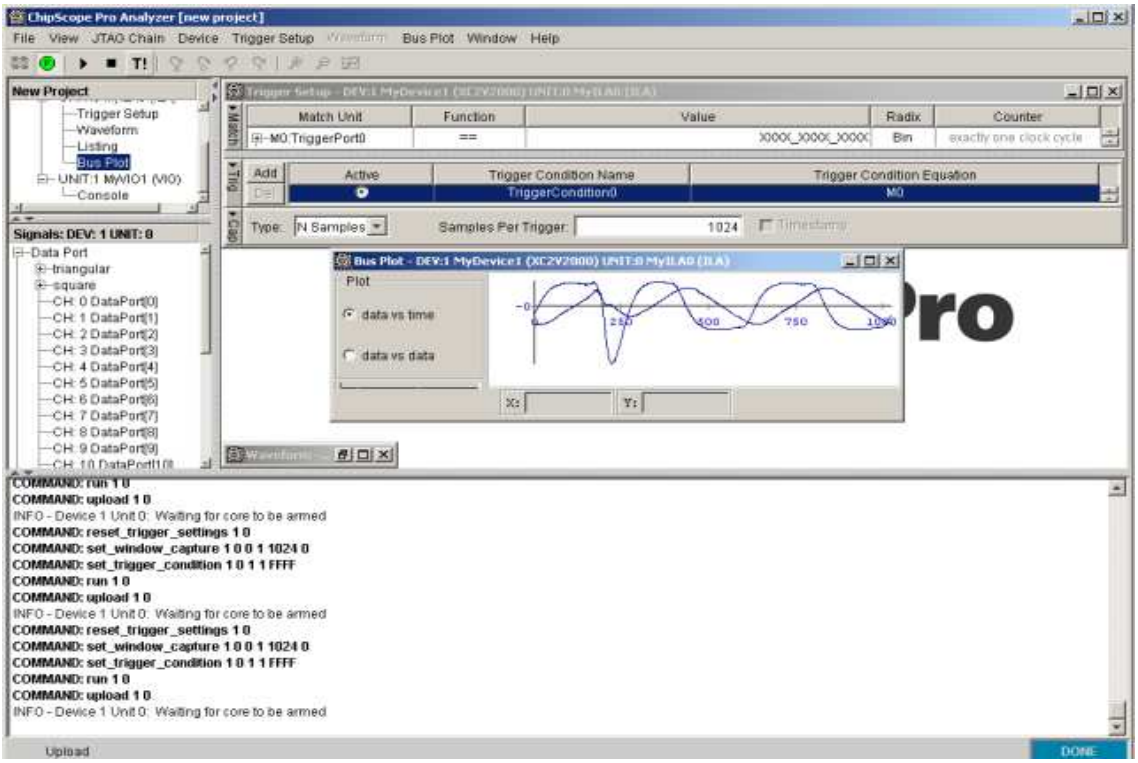


Figure 3-1: Waveform preview

[3]because full adder input/output relations can have many form, if you have better implementation just let me know :)

4 Information

4.1 Warranty

NO WARRANTY

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.2 TOOLS

- **ALLIANCE CAD SYSTEM** developed by **ASIM** team at ©**LIP6**/Université Pierre et Marie Curie, `http://asim.lip6.fr/recherche/alliance` - ***The primary VHDL Analyser for Synthesize***
- **ISE Xilinx 6.3i** - ***The Synthesizer***
- **VIM - Vi IMproved** - ***The Editor***
- LaTeX- ***The Typesetter***

References

[1] Tom Wada, ***All Digital FM Receiver***, at http://www.ie.u-ryukyu.ac.jp/~ wada/design05/

Version: 1.2 **29th March 2005**