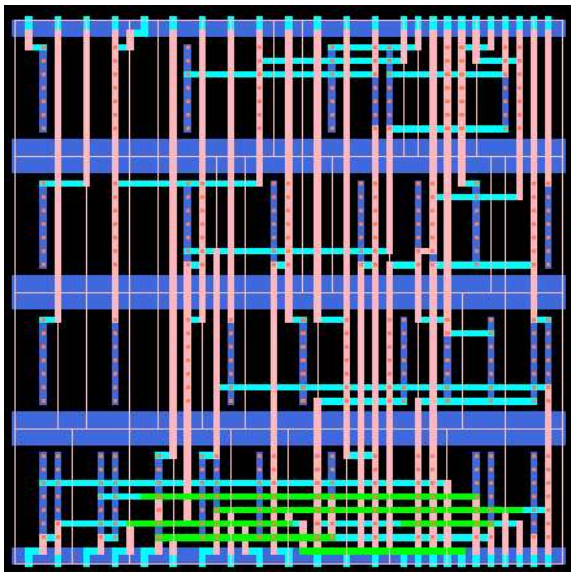
Simple FM Receiver

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 Circuit Block

Simple FM Receiver is based on PLL operation to capture FM input data. The design architecture is like ordinary PLL using basic component like phase detector, vco (realize using an nco), loop filter, and other supporting component.

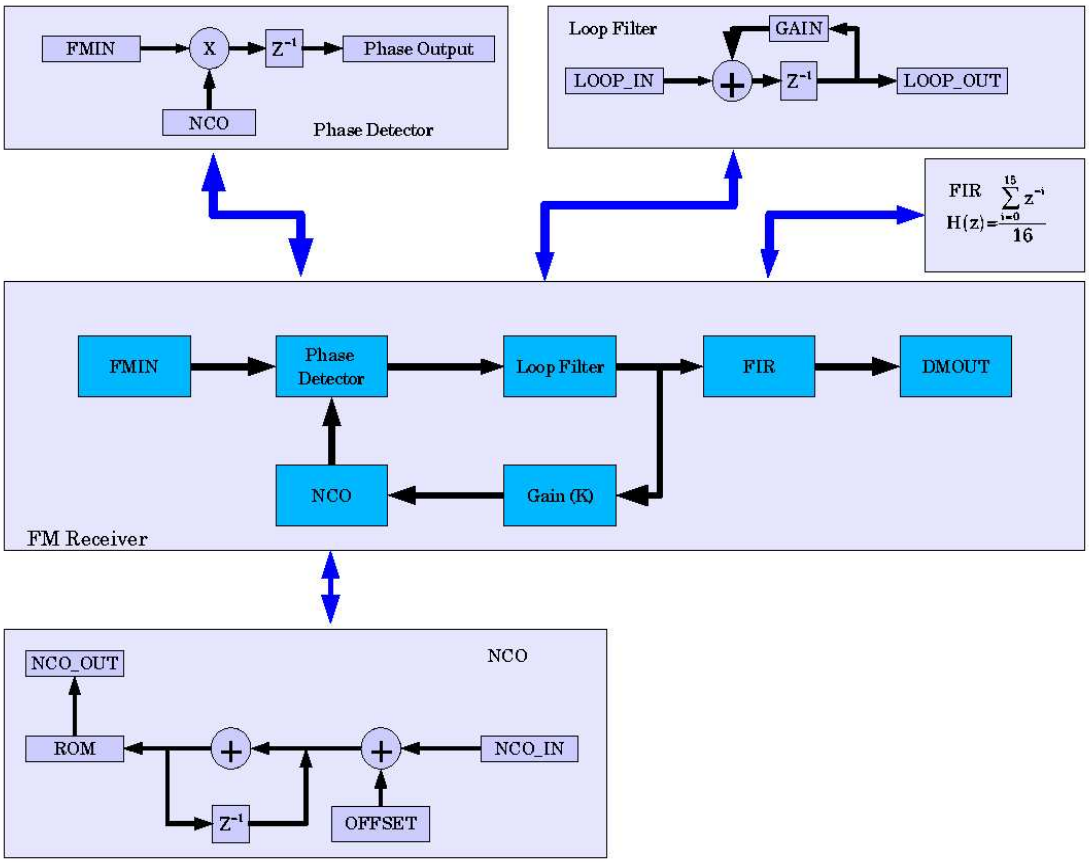


Figure 1-1: Architecture Schematics

This design is based on example circuit in **Mr. Wada** homepage with some modification.

2 Circuit Explanation

2.1 Core Component

- **fm**
  It's the main component, it's purpose is to connect many other component to form an PLL. It's function is to demodulate the FM data.

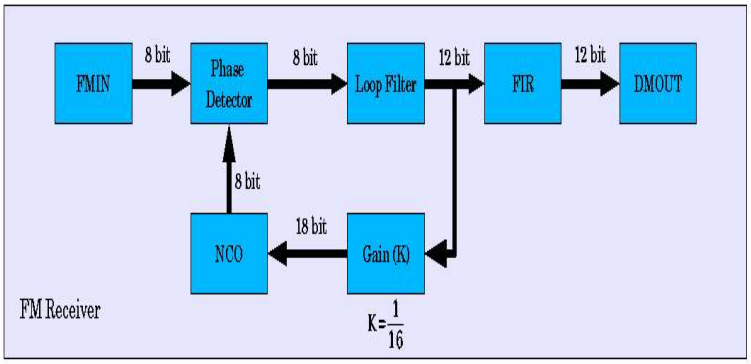


Figure 2-1: FM core component

2.2 Main Component

- **nco**
  The NCO functions it's like VCO in analog PLL. This NCO works like variable binary up-counter that controlled by input. Because its controlled by input then it's output frequency is change along with it's input value.

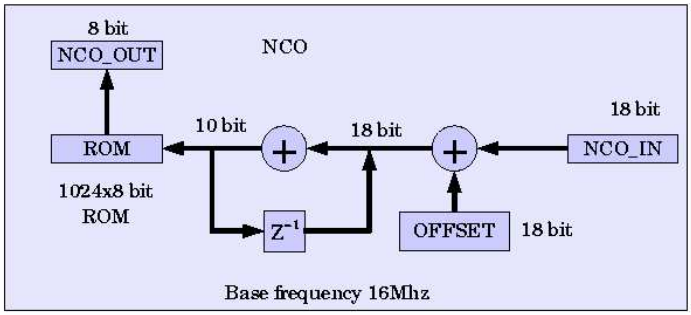


Figure 2-2: NCO block diagram

- **phase_detector**
  It's functions is to detect the phase different between input signal and signal from nco. This component is operate by multiplying input signal and output NCO.

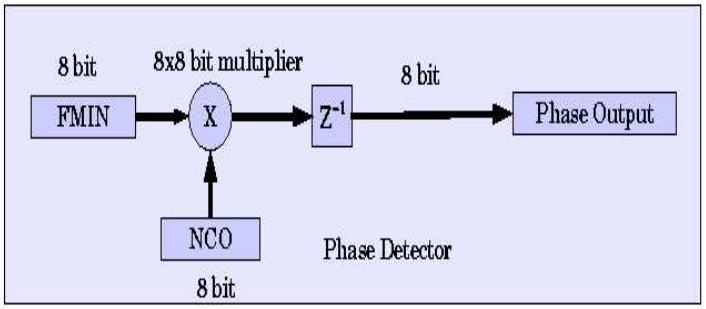


Figure 2-3: Phase detector block diagram

- **loop_filter**
  The filter that exists in the PLL loop. It's mathematical functions look like this.

$$Y(z) = X(z)\frac{z^{-1}}{1-(1-\frac{1}{16})z^{-1}}$$

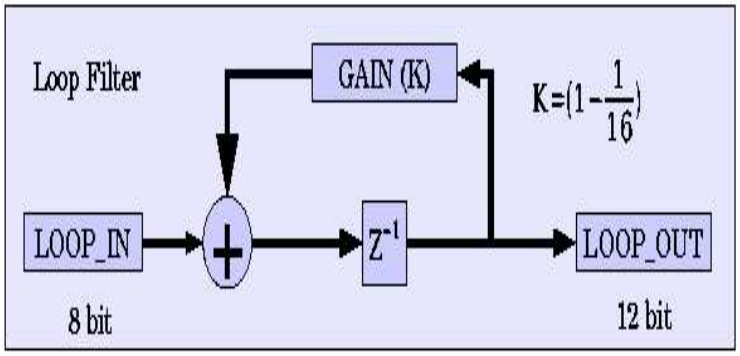


Figure 2-4: Loop filter block diagram

- **fir**
  This is the Low Pass filter type FIR. Realization of filter using direct FIR Transform 16 tap.

$$Y(z) = X(z)\frac{1}{16}\sum_{i=0}^{15} z^{-i}$$

2.3 Basic Component

- **adder**
  Many customized adder have been used on this design. This adder is used for many aritmetic op-

eration, special purpose adder were used for aritmetics operation in 2's complement or unsigned number.

- **subtractor**
  The subtractor that used on the loop filter, we realize the loop filter constant multiplier e.g. (15/16) by (1 - 1/16) so we need a substractor to do this task. Actually we implement the substractor by using ordinary adder, but the augend is 2's complement of subtractor.

```
          - X = 2's (X)       (negatif value is equal to 2's of it's value)
        X - Y = X + 2's (Y)
```

- **multiplier**
  Multiplier is used on the phase detector to multiply input signal and signal nco to get the phase different. This multiplier is implemented using simple addition of two operand, this multiplier need 8 stage of addition to perform operation on 8 bit input operand. This multiplier is the slowest component in this design, because this operations takes 8 stages of additions to complete single multiplications.

```
operand0          = XXXX_XXXX
operand1          = XXXX_XXXX
_____________________________
...
...       -> 8 stage addition
...
_____________________________
result = XXXX_XXXX__XXXX_XXXX
         ^^^^^^^^^-> 8 bit output
```

- **full adder**
  The very basic component that build all module. This component is implemented using this relations.

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

3 Information

3.1 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.

3.2 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***
- **ModelSim 6.0** - *The Simulator*
- **ISE Xilinx 6.3i** - *The Synthesizer*
- **FPGA Xilinx XC2V2000-6-FF896** - *The Implementor*
- **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: 1.0 **24th January 2005**