# An FPGA-Based Transceiver Module

John B. Stephensen, KD6OZH
3064 E. Brown Ave., Fresno, CA 93703
kd6ozh@arrl.net

Traditionally, hams have wired together multiple standard integrated circuits to construct radios, TNCs and other devices. However, modern devices manufactured in high volume integrate most functionality on one chip that is customized for the purpose. Feature sizes in integrated circuits have now become so small that field programmable gate arrays (FPGAs) can be used to achieve what once required a custom chip. Since the logic in the FPGA can be used in parallel rather than sequentially, it is a much more powerful tool than a microprocessor. Recent FPGAs also contain larger amounts of dedicated memory, making it possible to incorporate microprocessors within the FPGA.

My goal was to make a small high-speed digital signal processor that can be embedded in amateur radio devices. The original hardware platform was developed while I was a member of the ARRL high-speed multimedia (HSMM) working group and was described in QEX[1]. However, the combination of an FPGA and an external microcontroller was costly and inter-chip communication imposes design restrictions and a speed penalty. Therefore, I developed a soft processor that was recently described in PSR[2]. However, this CPU comprises only 8% of the logic required to implement an OFDM modem. This document describes the specialized signal processing modules that accompany the CPU. Each implements a specific algorithm, but can be customized by loading new parameters. The result is a device that looks like a custom integrated circuit that can have software downloaded to program it for specific applications.

## 1. Overview

The hardware platform is a 2.5” x 2.4” PCB with a Xilinx XC3S500E FPGA surrounded by an 80 Msps analog to digital converter (ADC), 80 Msps digital to analog converter (DAC), 100 Mbps Ethernet physical layer interface (PHY) and 4-megabits of flash memory. There is also a JTAG port, a low-speed DAC and an RS-485 port for debugging and control. The 10-pin header connects directly to 5 FPGA I/O pins that may be used for debugging, controlling analog circuitry and/or attaching an audio CODEC. The schematic diagrams are in appendix A.

The ADC analog input and DAC analog output are provided directly on 2-pin connectors and analog filtering takes place externally. This allows the use double-sided boards or even copper-clad perforated board for analog circuitry. The FPGA configuration is automatically loaded from the flash memory when power is applied. It contains three main sections – the tuner, the modem and the CPU.

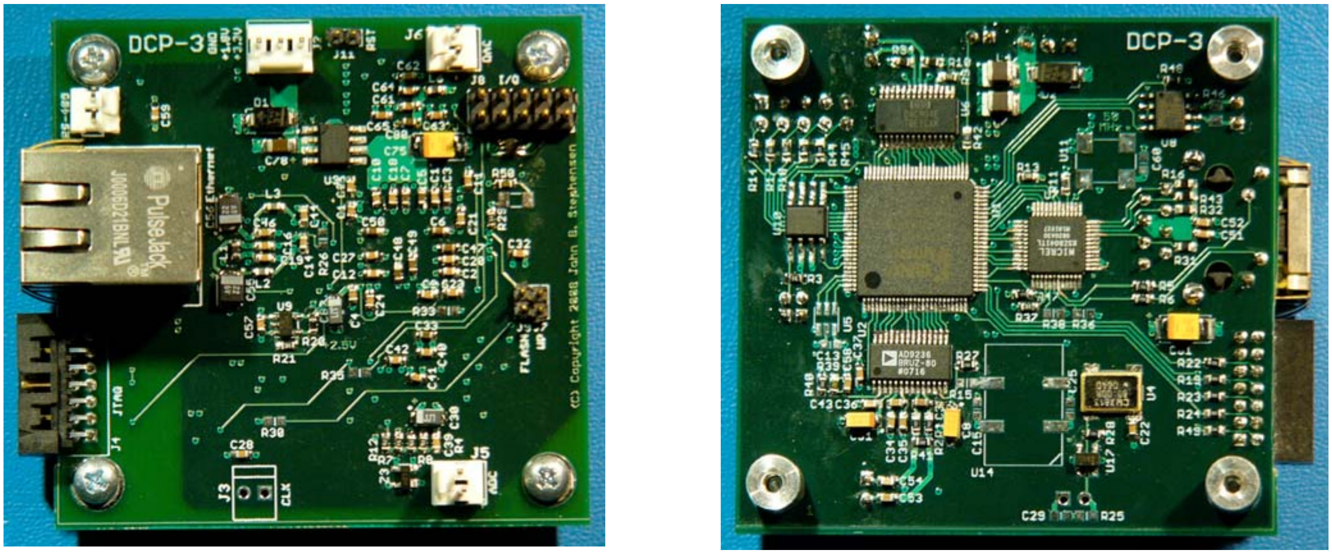


Figure 1 – DCP-3 PCB

The tuner section, shown in figure 2, interfaces to the ADC and DAC and provides digital upconversion, downconversion, filtering and signal level control. Some functions operate at twice the sampling rate in order to share resources. The single lines in the block diagram are paths where the in-phase (I) and quadrature (Q) samples are processed sequentially at 160 MHz. The double lines are paths where I and Q samples are processed in parallel at 80 MHz.

The signal processing hardware starts with the direct digital synthesizer (DDS) whose frequency may be set with a resolution of 0.02 Hz. During reception, the input from the ADC is sequentially multiplied by the cosine and sine outputs of the DDS to generate baseband I and Q outputs. One cascaded integrator comb (CIC) filter and two finite impulse response (FIR) filters provide downsampling and establish the receiver bandwidth. A noise blanker is located between the FIR filters. The AGC levels the signals and converts from two 20-bit to two 16-bit samples. Optionally, the received signal may pass through the resampling filter to be converted to a rate that is not an integer fraction of the ADC sampling rate.

When transmitting, the resampler can convert the baseband signal to a rate compatible with the DAC. The signal may then be compressed or clipped to reduce the peak to average power ratio (PAPR) and passed on to the filter chain. The order of the filters is reversed for transmission with the two FIR filters first and the CIC filter last. Upsampled I and Q baseband signals are mixed with the cosine and sine DDS outputs and added together to produce the final output to the DAC.

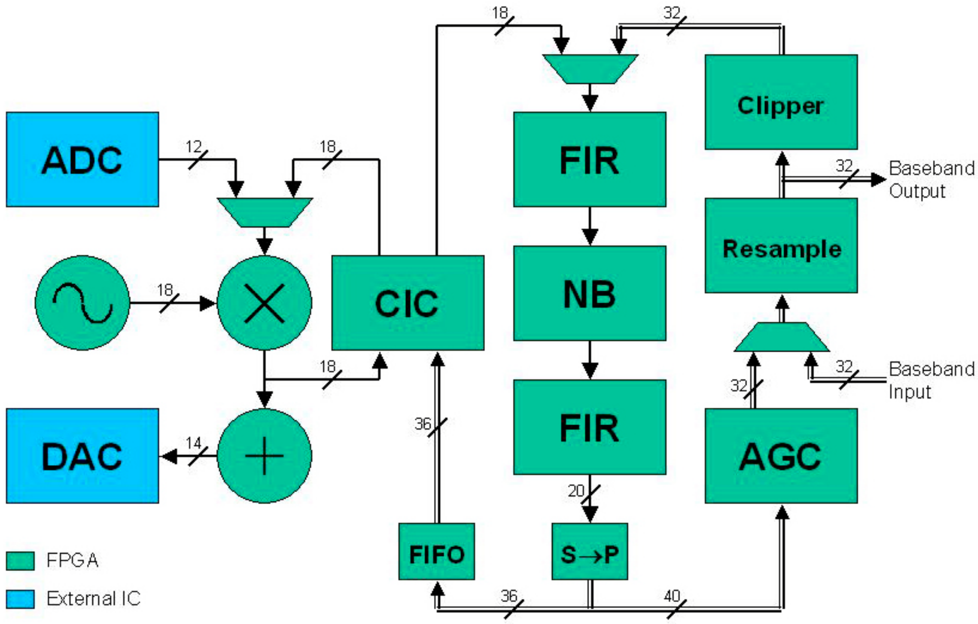


Figure 2 –Tuning, Filtering and Level Control Circuitry

The modem section, shown in figure 3, uses CORDIC (coordinate rotation digital computer) to modulate or demodulate the signal when a single carrier is used. This is an algorithm that performs trigonometric functions using only adds and shifts. For amplitude modulation (AM) and phase modulation (PM), it can rotate the input signal to alter its phase or measure the phase and magnitude of the input. For frequency modulation (FM), the input is integrated or the output is differentiated. SSB reception and transmission use a 16-bit accumulator that continuously rotates the phase. It functions as the BFO and provides 0.12 Hz resolution at 8 ksps. The CPU accesses the modem via a 15-word FIFO with two 16-bit entries containing either magnitude and phase or I and Q information.

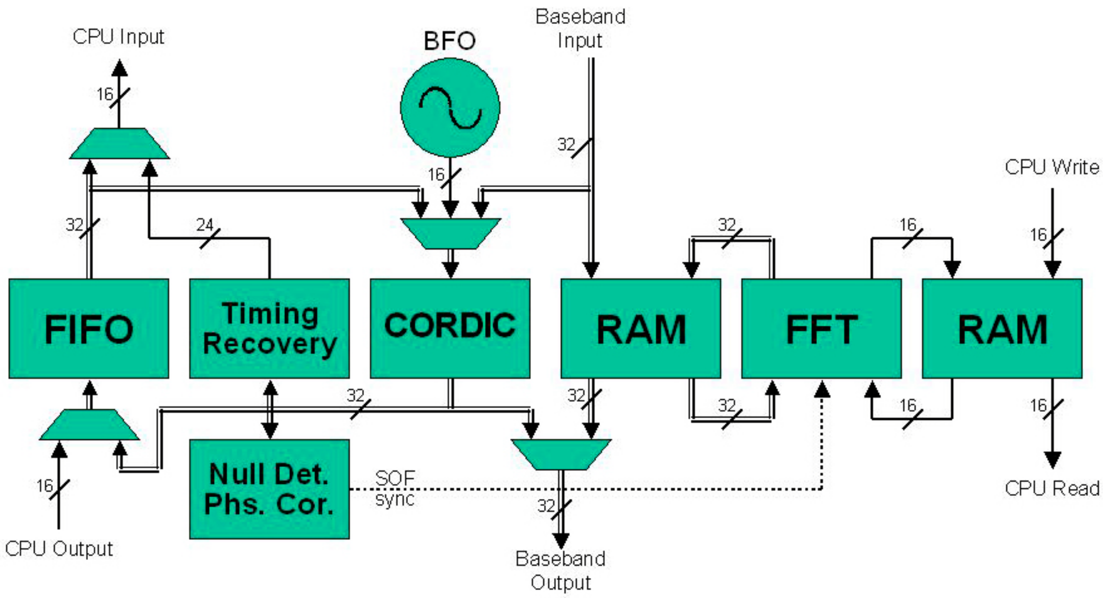


Figure 3 – Modulation and Demodulation Circuitry

The timing recovery unit generates error signals for use in optimizing the sampling of FSK and PSK waveforms. The null detector and phase correlator provide timing recovery for OFDM reception. Both are described in more detail later in this document.

A fast Fourier transform (FFT) is provided for OFDM or MFSK modems. It moves data between two RAMs while converting between time and frequency domains. The time domain buffer connects directly to the tuner and the frequency domain buffer is read or written by the CPU. When receiving, I and Q samples of the input signal accumulate in one RAM, the processing module then calculates the FFT and places the phase and magnitude for each demodulated subcarrier in the second RAM.  When transmitting, the CPU places data in the subcarrier RAM and this is converted to a series of samples in the I/Q RAM.

An 80 MHz 16-bit CPU controls the tuner and modem. It has a RISC instruction set that includes bit manipulation for protocol processing and hardware multiply, multiply-accumulate and divide for signal processing. It also allows addition, subtraction, loading or comparison of 8-bit constants in one instruction or the same operations with 16-bit constants using a prefix instruction. In addition, memory read and memory write instructions support indirect addresses with offsets while input and output instructions use direct addressing. This produces fast compact code. The complete instruction set is described in appendix B.

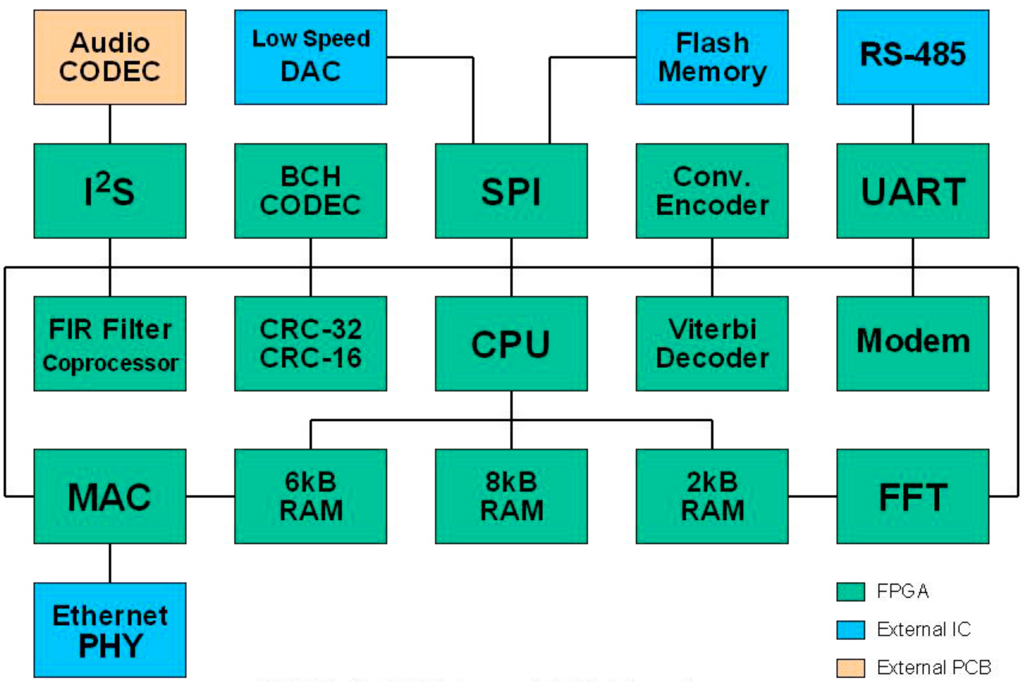


Figure 4 – CPU and Peripherals

The CPU has 8 kB of dedicated instruction and data memory that is accessed simultaneously via two ports. It is augmented by a number of modules that provide I/O and speed up common algorithms for digital communication and digital signal processing such as filtering and error detection and correction. These are accessed via I/O ports and two-port buffer memory as shown in figure 4. An advantage of this architecture is that the CPU can start multiple operations that proceed in parallel.

## 2. Tuner

The heart of the DDS is a phase accumulator that increments by the value in the center frequency register on alternate cycles of the 160 MHz clock. The upper 10 bits of the phase output address a dual-port 1024 x 18 ROM sine look up table. Adding a 90-degree offset on alternate clock cycles generates cosine and sine outputs. The ROM address inputs select adjacent entries and linear interpolation is used to smooth the output. The difference between adjacent entries is multiplied by bits 4-21 of the phase accumulator and then added to the first entry. Extensive pipelining is used with delays inserted between stages to align results. The shift registers provided in the Xilinx architecture minimize the amount of resources used as one FPGA logic cell can provide a 1 to 16 stage shift register. 8 clock cycles are used to generate each output. The algorithm was originally described in a Motorola application note[3] and spurs should be below –112 dBc.

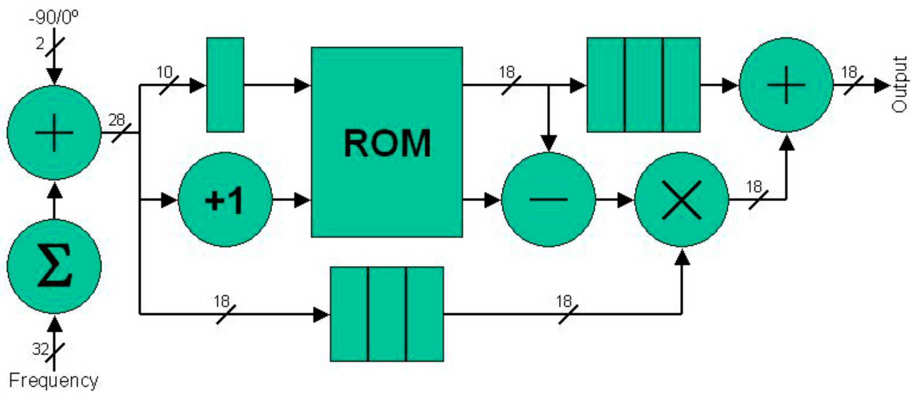


Figure 5 – DDS VFO

The CIC filter, shown in figure 6, can downsample or upsample by an integer value between 10 and 640, converting between 80 Msps and 8000-125 ksps. A CIC filter is used as it can provide large decimation and interpolation ratios while using only addition and subtraction. It is essentially a moving average filter that has been optimized to use less hardware. Instead of summing a fixed number of samples in each pass, the process has been simplified to summing all samples in an accumulator (integration) and then calculating the difference between accumulator states at two different times – the first and last taps of the moving average. The frequency response of the differentiator is a comb shape so that portion is called a comb filter. The accumulator can overflow as long as there are enough bits to cover the time span over which the filter averages. Cascading multiple filters improves the frequency response and the integrators and differentiators can be grouped together. This simplifies downsampling as intervening samples can be ignored. The differentiators need only have one delay register as long as they are enabled only for each output sample. The accumulators must process all input samples. When upsampling the differentiators precede the integrators.

The CIC circuitry contains four 56-bit integrators and four 28-bit differentiators for each channel. The integrators are split in two and the carry between the two 28-bit halves is buffered to minimize propagation delays. The CIC filter has an inherent gain that varies with the interpolation or decimation factor so this is offset by setting a gain factor that controls a shifter and multiplier. The CIC gain is the third power of the interpolation factor or the fourth power of the decimation factor.

When receiving, the minimum CIC gain is $10^4$ and the maximum gain is about $1.68 \times 10^{11}$. The received data input (RDI) is multiplied by 0-1024 and shifted by 0-15 bits by two 4-input multiplexers before being applied to the first integrator. The shifter provides a 43-bit output so it is expanded to 56 bits by adding 13 sign bits. This gives a gain range of $2^{25}$ or about $3.3 \times 10^7$. The top 28 bits of the last integrator are then fed to the first differentiator and the received data output (RDO) is obtained from the top 18 bits of the last differentiator. Thus, the input signal is expected to grow by 38 bits ($2.75 \times 10^{11}$) by the time that it reaches the output. When the minimum decimation factor of 10 is used, the gain compensation can be set to $2.75 \times 10^7$. When the maximum decimation factor of 640 is used, the gain compensation can be set to $1.63 \times 10^0$.

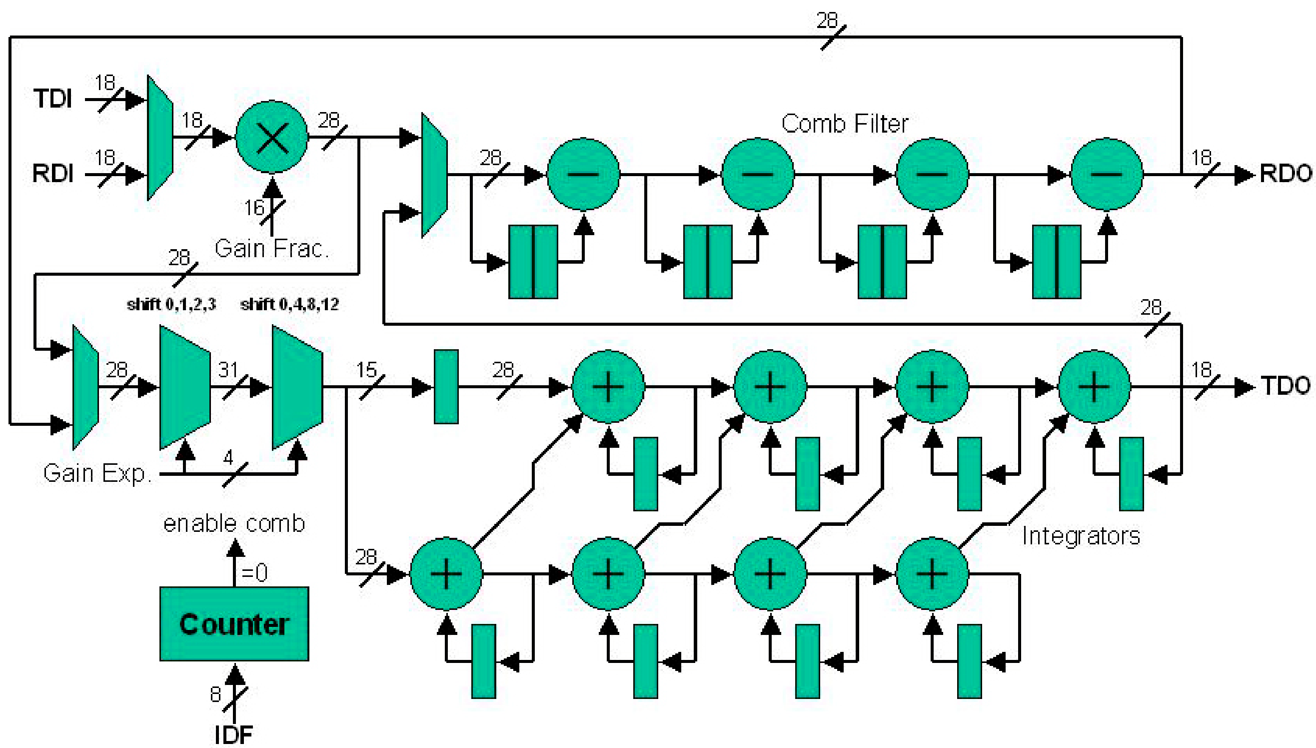


Figure 6 – CIC Filter

The transmit data input (TDI) is multiplied by 0-8 before being applied to the bottom 21 bits of first differentiator. The upper 7 bits are copies of the sign bits. This is necessary as the required resolution grows by at least 1 bit per state. The full 28-bit differentiator output is then shifted by 0-15 bits and applied to the first integrator. This gives a gain adjustment range of $2^{18}$ or about $2.6 \times 10^5$. The minimum CIC gain is $10^3$ and the maximum gain is about $2.6 \times 10^8$. The total gain can be kept at about $2.6 \times 10^8$, so the 18-bit input grows to 46 bits. Consequently the transmit data output (TDO) is tapped down on the last integrator and the top 10 bits are ignored.

Two FIR filters follow the CIC filter. The first is used for downsampling by a factor of 2-50 and the second is used to establish the shape of the final passband. It may also downsample or upsample by a factor of up to 20 depending on the steepness of the filter skirts. The first FIR filter outputs 18-bit results and the second FIR filter outputs 20 bit results. A CIC filter is essentially a FIR filter with all coefficients equal to one and has a fixed frequency response of which only a fraction of the center portion is flat. FIR filters operate by multiplying a set of signal samples (data) by a set of positive and negative coefficients and summing the products. They consume more resources than the CIC but this allows tailoring the frequency response to specific requirements. Both FIR filters use 24-bit coefficients to reduce the level of spurious responses. Spurs are down 4-5 dB per coefficient bit depending on the filter shape factor and the amount of downsampling or upsampling.

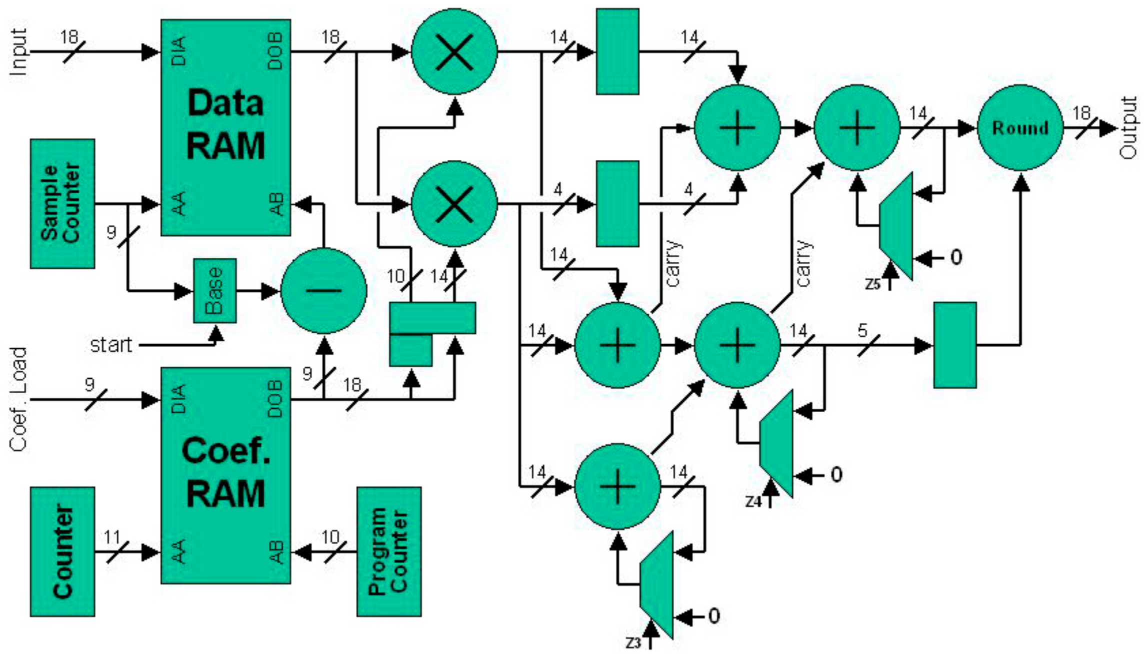


Figure 7 – FIR Filter

Figure 7 shows the common architecture of the two FIR filters. Each uses two dual-port 18k block RAMs. One port of the data RAM is used to store samples as they arrive. The sample counter determines the address used and it always incremented after each write. When the filter starts the contents of the sample counter are saved and used as the base address for the other port which is used to retrieve samples for processing by the filter. I and Q samples are stored serially in the RAM and are accessed on opposite phases of the 80 MHz master clock. The filter runs on a double-rate clock at 160 MHz.

The second block RAM stores instructions consisting of a 9-bit index, a 24-bit coefficient, a write enable bit and an end of filter bit. The CPU may load instructions via a 9-bit wide port. Instructions addressed by the program counter are retrieved over two clock cycles and are used to process I samples on one clock cycle and Q samples on the next clock cycle. Data is read from the data RAM at the address obtained by subtracting the index from the base address register. It is then multiplied by the coefficient using two dedicated 18x18 multipliers and two adders to sum the partial products. This creates a 42-bit product that is summed alternately in two 42-bit accumulators. The accumulators are split into three 14-bit sections to reduce carry propagation delays and addition occurs over 3 clock cycles. Each section contains an adder and a multiplexer. This provides a 2-clock delay so two channels are supported. The multiplexers have one port tied to zero so that they can be used to switch between loading and accumulating by the signals Z3, Z4 and Z5. The final sum of products is then rounded to either 18 or 20 bits.

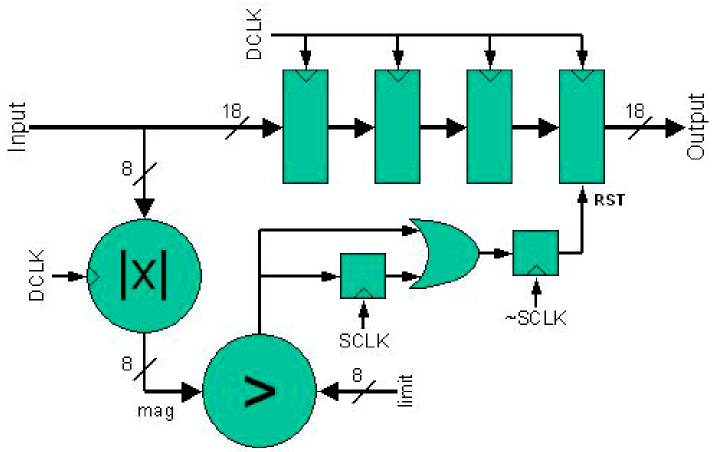


Figure 8 – Noise Blanker

The noise blanker is shown in figure 8. It generates the absolute value of I samples and Q samples by complementing negative values. These are compared to a limit and if it is exceeded in either axis the output is set to zero. Two registers clocked on opposite edges of the single-rate clock (sclk) align the comparitor output with the two related samples. The signal is delayed in three registers and the fourth register is reset as the offending samples pass by. This logic is placed between the FIR filters so that blanking occurs prior to the steep-skirted final filter. This minimizes stretching of noise pulses.

**Tuner Command**

<table>
<tr><th></th><th>Register</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>58</td><td rowspan="2">Center Frequency</td><td colspan="16">LSW</td></tr>
<tr><td>59</td><td colspan="16">MSW (update)</td></tr>
<tr><td>5A</td><td>CIC Gain</td><td colspan="10">Integer</td><td colspan="6">Fraction</td></tr>
<tr><td>5B</td><td>CIC Int./Dec.</td><td colspan="4">Exponent</td><td></td><td></td><td colspan="10">Interpolation/Decimation (10-640) - 1</td></tr>
<tr><td>5C</td><td>Noise Blanker</td><td></td><td colspan="7">Threshold (default = 127)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>5D</td><td>FIR Decimation</td><td></td><td></td><td colspan="6">FIR 2 Decimation - 1</td><td></td><td></td><td colspan="6">FIR 1 Decimation - 1</td></tr>
<tr><td>5E</td><td>FIR Coef. Load</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="9">¼ Instruction</td></tr>
<tr><td>5F</td><td>FIR Select</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>RST</td><td>FIR</td></tr>
</table>

CIC Gain = integer.fraction x $2^{exponent}$, max. receive gain = $2^{38}/d^4$, max. transmit gain = $2^{28}/i^3$
RST: reset FIR while loading coefficients
FIR instructions written 9-bits at a time: 8-0, 17-9, 26-18, 35-27

**Tuner Status**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 58 | Overflow Flags | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ADC | MIX | DAC | FL1 | FL2 |

**FIR Instruction Format**

<table>
<tr><td>35</td><td>34</td><td>33</td><td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td>E</td><td>W</td><td colspan="9">Data Index</td><td colspan="24">Coefficient</td></tr>
</table>

W = write data to output
E = end of filter (place 1 instruction before last coefficient)

Figure 9 – Tuner Configuration Registers

The tuner is configured via the 8 registers shown in figure 9. The center frequency is a 32-bit word, loaded LSW first in two 16-bit segments. The FIR instructions are loaded into the filter selected by the FIR bit in 9-bit segments, LSW first. The RST bit resets the FIR address counter and inhibits filter operation during loading. The two FIR decimation values initialize counters that delay the start of the FIR filters until a specified number of new samples have been accumulated in data memory. Interpolation is handled by putting multiple write instructions in the filtering programs.

The status register contains overflow flags that are set when an overflow occurs and reset when read. The ADC overflow comes from the overflow output on the ADC. The MIX and DAC flags signal overflow in the quadrature mixer and adder, respectively. FL1 and FL2 are overflow flags from the two FIR filters.

**3. AGC, Resampler and Clipper/Compressor**

The AGC module levels the output and reduces the word size to 16 bits. It implements a complete hang AGC system in hardware. The desired AGC output level, attack time, release time, hang threshold and hang time are all programmable by the CPU. The AGC gain may vary between 0 and 96 dB and the CPU may read the current value.

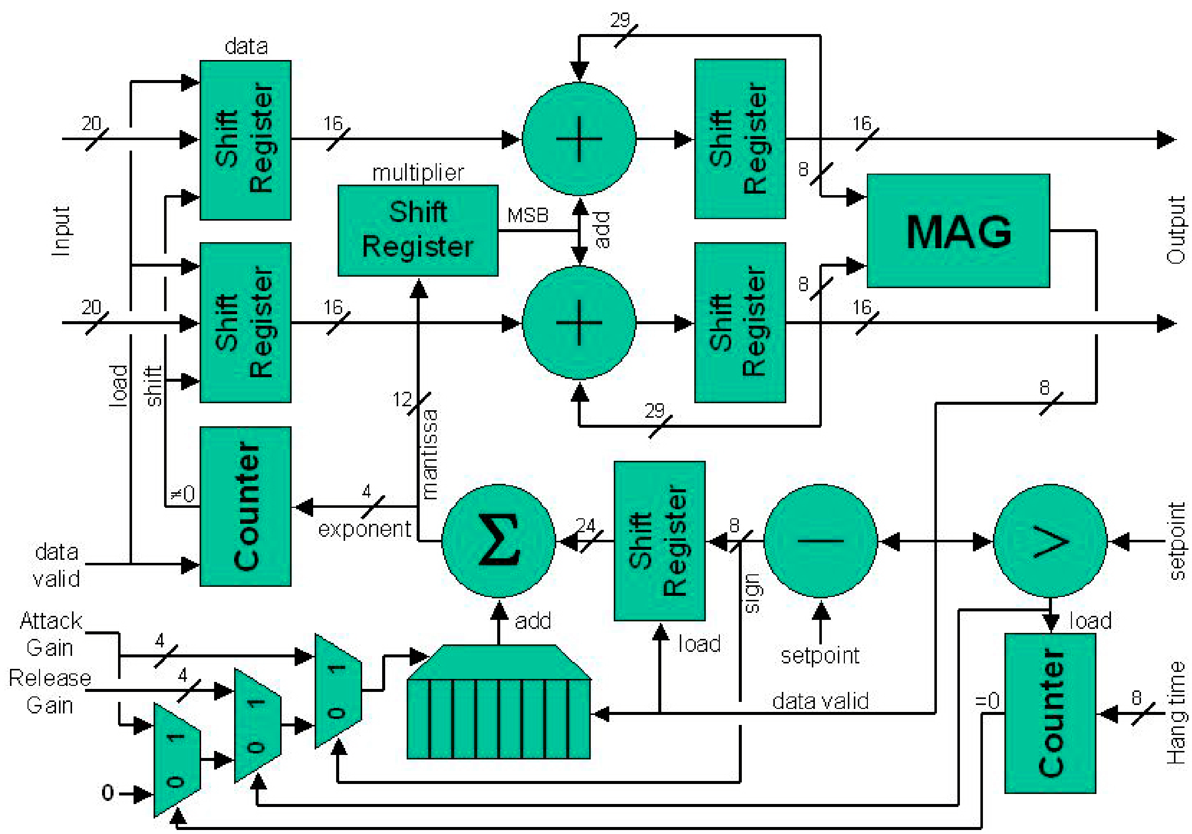


Figure 10 - Automatic Gain Control

The AGC logic is shown in figure 10 and makes extensive use of serial arithmetic to reduce resources and power requirements. This is possible because the signal has been fully filtered and down-sampled to 2.5 Msps or less. The circuit operates at 80 MHz so 32 clock cycles are available for each sample. I and Q samples are loaded into two shift registers and then shifted left for a number of clock cycles determined by the counter. The counter is loaded with the gain exponent at the same time that data is loaded. It then counts down to zero and stops the shifting, multiplying the data by the binary exponent. Not shown is logic that detects and flags overflows. The gain mantissa is used in two serial multipliers to multiply by a factor of 1 to 2. It is loaded into another shift register and the bits are shifted out MSB first. If a bit is one the shifted signal samples are added to the shifted contents of another register. The accumulated sum shifts left on each clock cycle. After 13 cycles, the results are available at the output.

The AGC gain is set by the top 16 bits of a 24-bit accumulator that integrates the difference between the magnitude of the output signal and the AGC setpoint. The magnitude determination logic is described further in the compressor/clipper section of this document. Loop gain is controlled by a serial multiplier created by shifting the output of the setpoint subtracter. It multiplies by a power of two that is determined by a time delay in a shift register. Xilinx logic elements may be configured as a 16-bit shift register with an output multiplexer. That multiplexer determines the number of stages used, the time delay and the gain. There are three possibilities for gain. If the output signal is higher than the setpoint, the attack gain is selected. If the signal is below the setpoint but above the hang threshold, the release gain is used. If the signal is below the hang threshold and the hang timer has not expired, the loop gain is set to the minimum. This allows the AGC gain to be held constant for a period after the signal disappears and then be rapidly reduced after that period. This mechanism reduces AGC pumping on CW and SSB signals[4]. It is also useful for OFDM signals that use null symbols as delimiters.

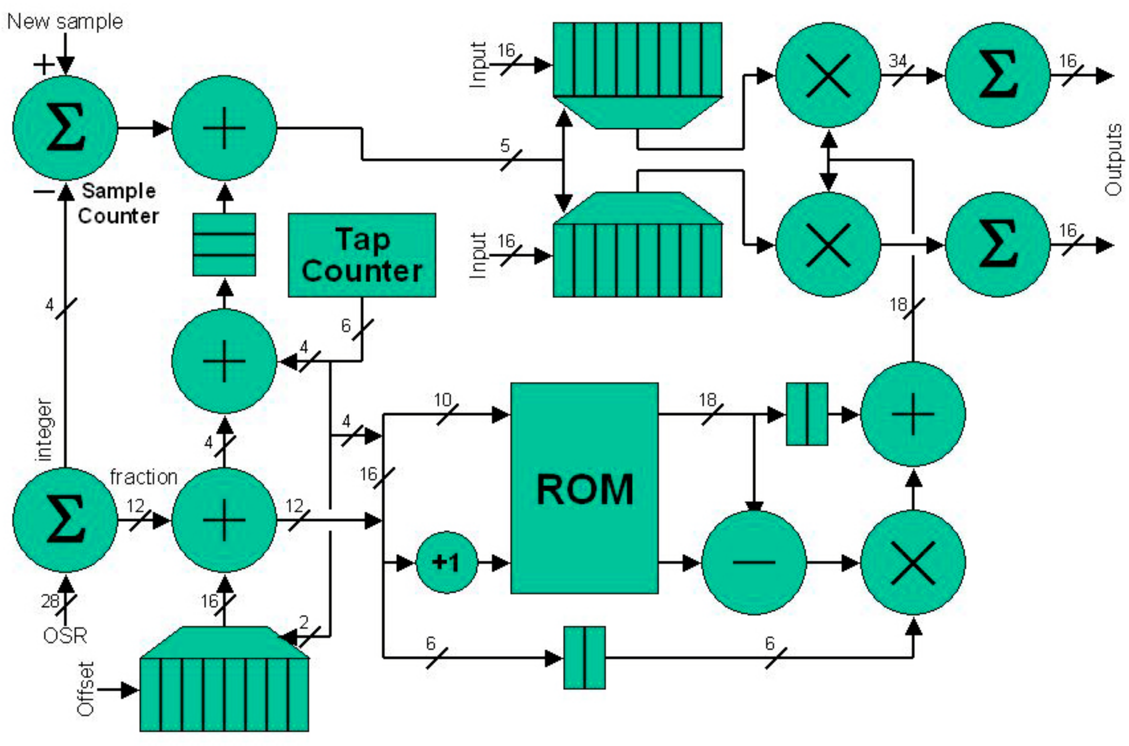


Figure 11 - Resampling Filter

A resampling filter may be switched in for applications using baud rates that are not an integer fraction of the ADC and DAC sampling rates. The output sample rate (OSR) may be increased or decreased as specified by a 4-bit integer and a 24-bit fraction that determine the time between output samples. The output sample rate is the input sample rate divided by this 28-bit number and may be set to an accuracy of 0.06 PPM.

The phase accumulator controls the resampling filter. It is incremented by the OSR before the start of each filtering cycle. If a new input sample is needed the integer portion of the output will be non-zero. New samples are requested by decrementing the sample counter by that value. The sample counter will go negative if the required samples are not present and the processing pipeline will stall. It will then continue when new samples arrive and the sample counter output goes positive. The integer portion of the phase accumulator output is treated as overflow and does not accumulate between cycles.

The upper 12 bits of the fractional portion of the phase accumulator are added to the phase offset from the offset shift register. The lower 12 bits of the result are used to select the coefficients to be used in the filtering process. The upper 4 bits are added to the data index from the tap counter so the sequence of data samples to be processed to be shifted backwards in time. The resampling filter implements a 16-tap filter with 64 possible coefficients for each tap. The tap counter increments from 0 to 15, 31, 47 or 63 depending on whether there are 1, 2, 3 or 4 outputs to be generated per filtering cycle. The bottom 4 bits select data samples and coefficients and the upper 2 bits are used to select the phase offsets stored in a 4-entry shift register.

The coefficients are stored in a 1024x18 dual-port ROM and provide a flat passband to 0.4 times the input sample rate. Coefficients are linearly interpolated in the same manner as the DDS when the desired output sample falls between ROM entries. Incoming data is stored in two 32-entry shift registers as it arrives. Multiplexers embedded in the shift register select the required samples. As new samples arrive, the sample counter is incremented and added to the data index to compensate for shifts that occur during computation. Both channels are multiplied by the same coefficient and the results accumulated. The distortion introduced by the resampling process should be less than -96 dBc and is less than –85 dBc on a spectrum analyzer. A more thorough description of the arbitrary resampling process can be found in chapter 7 of reference 5.

When transmitting, a RF compressor is available to reduce the peak to average power ratio (PAPR) of SSB and OFDM signals. It consists of two multipliers and a memory containing 64 coefficients that are selected by the magnitude of the incoming signal. The gain profile may be adjusted for clipping, compression or bypass. Figure 12 shows the logic used.

The magnitude of the input is estimated by the formula 7*(max + min/2)/8. Since compression only occurs at high signal levels only the upper 8 bits of the input are examined. Gain coefficients are stored in a shift register and the upper 6 bits of the magnitude select the appropriate entry. Each 8-bit entry consists of a 4-bit integer and a 4-bit fraction. Low signals levels may be boosted by up to 24 dB while high signal levels can be left unchanged or even attenuated. The look-up table and multipliers ensure that the clipping level or compression profile is independent of signal phase. Processing the I and Q channels independently would have resulted in a 3 dB increase in output level when the phase is 45, 135, 225 or 315 degrees. It would also have altered the phase of the output.

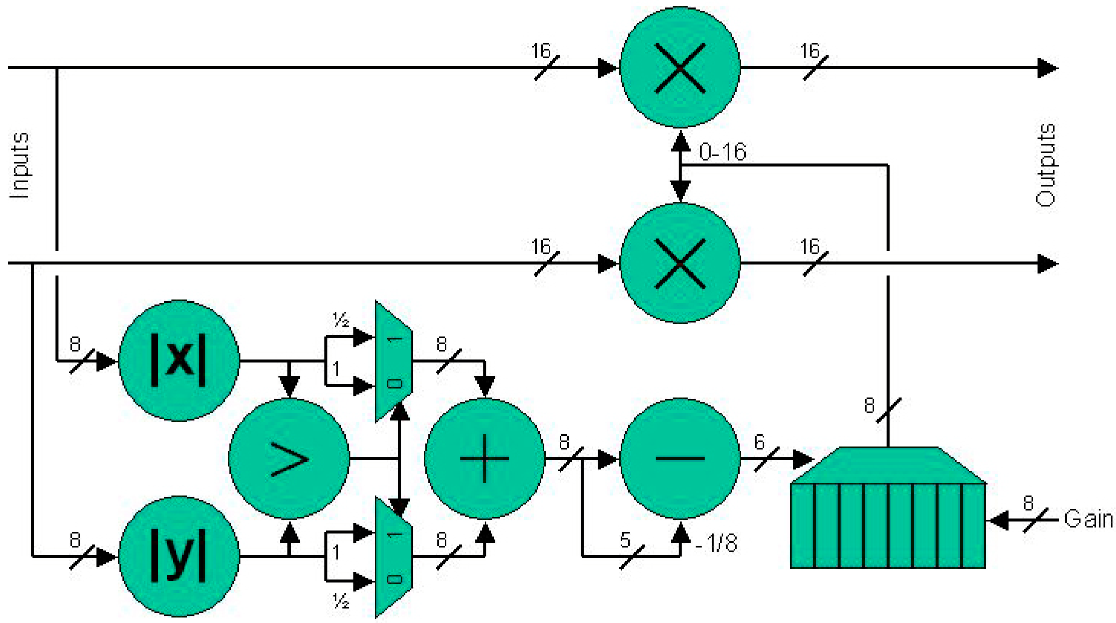


Figure 12 - Compressor/Clipper

The resampler, AGC and compressor are configured via the 8 registers shown in figure 13. The output sample rate is loaded in two 16-bit segments with the lower bits saved in a temporary register until the upper bits are loaded. The output phases are loaded in reverse order and the first N are used as determined by the INT bits.

**Resampler/AGC/Compressor Command**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 60 | Output Sample | LS Fractional Part | | | | | | | | | | | | | | | |
| 61 | Rate Control | Integer Part | | | | MS Fractional Part | | | | | | | | | | | |
| 62 | Output Phases | Integer Part | | | | Fractional Part | | | | | | | | | | | |
| 63 | Rsmpl. Control | | | | | | | | | | | | | INT - 1 | | RSE | VFO |
| 64 | AGC Config. | Attack Gain | | | | Release Gain | | | | | Output Setpoint | | | | | | |
| 65 | Hang Config. | Time (Samples/256) | | | | | | | | | Threshold | | | | | | |
| 66 | AGC Gain Limit | | | | | | | | | Exponent | | | | Mantissa | | | |
| 67 | Comp. Gain | | | | | | | | | Integer | | | | Fraction | | | |

Input sample rate is a fraction of output sample rate ($N = 2^{24} \times F_{IN}/F_{OUT}$)
Output phase has 1-4 entries as specified by the interpolation factor
INT: output interpolation factor (1-4)
VFO: 0) output clock is input clock, 1) enable 32-bit timing DDS for fractional output clock
AGC attack and release gain are powers of two
Compressor has 64 entries loaded in reverse order corresponding to input magnitude

Figure 13 – AGC, Resampler and Compressor Configuration Registers

## 4. Single-Carrier Modem

The CORDIC logic is shown in figure 14. There are three stages – coarse rotation, fine rotation and output correction. The coarse rotation logic rotates the inputs by 90° or 180° so that calculations can be done in the first quadrant. It can negate the X and/or Y inputs and may then swap the inputs. This is determined by looking at the signs of the X and Y inputs or the quadrant of the angle Z. The fine rotation engine consists of the three multiplexers, two 0-15-bit shifters and three 22-bit adder-subtractors in the center of the diagram. The six extra bits ensure the accuracy of the computations. Two extra MSBs allow for magnitude growth during computation. Four extra LSBs limit the loss of resolution as X and Y shift right. 16 fine rotations are performed by adding or subtracting fractions of X from Y and fractions of Y from X. At the same time, the arctangent is subtracted from or added to Z. In the first rotation the multiplexers let in the coarsely rotated data. In the next 15 rotations, the output is fed back to the input and the shift value is incremented.

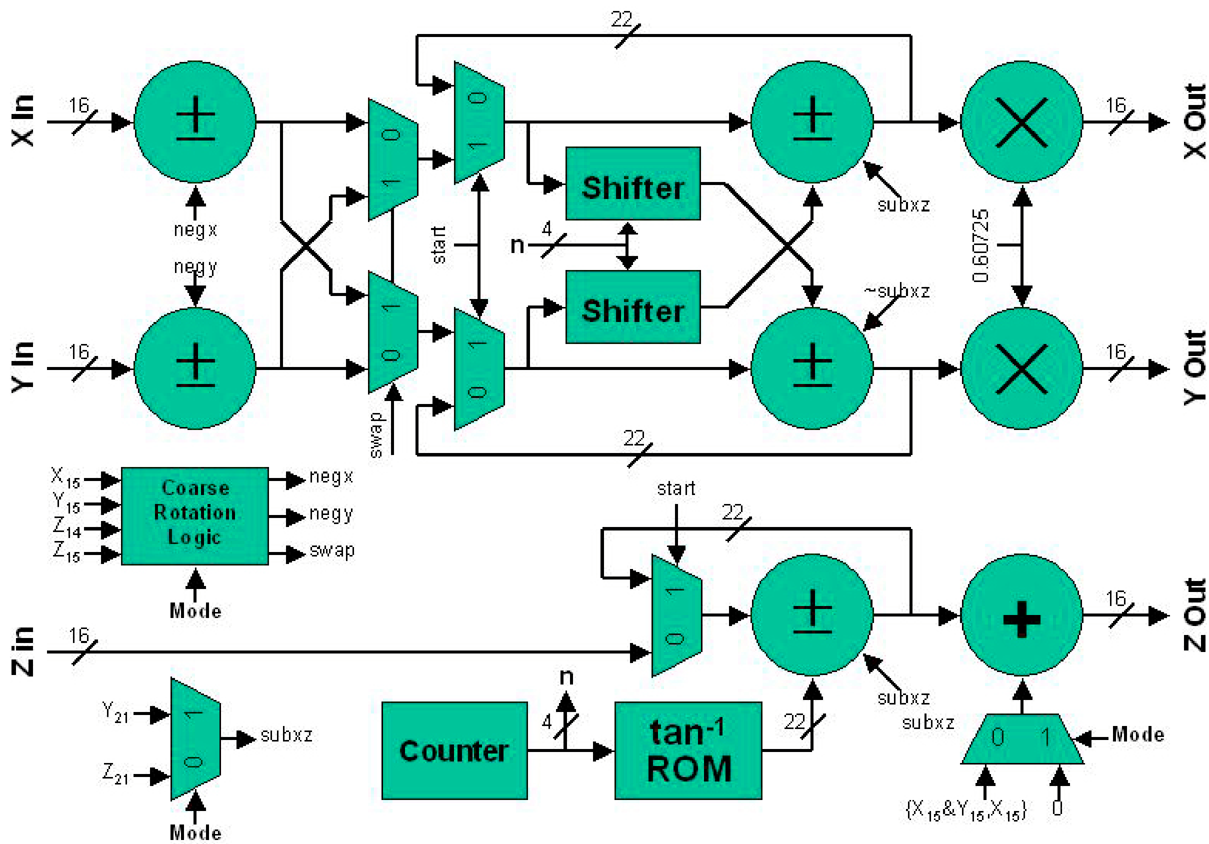


Figure 14 - CORDIC

At each step, the rotation may be clockwise or counter-clockwise as determined by the signal subxz. In the rotate mode Z is driven towards zero by a series of rotations by 45°, 22.5° and ever-smaller increments. In the vector mode, Y is driven towards zero. The length of the signal vector increases by a known value with each rotation. Correction logic multiplies the final X and Y outputs by a constant so that the magnitude remains in the ±1 range. It also alters the Z output to correct for the coarse rotation. The accuracy of the algorithm is 15 bits with full magnitude data and decreases as the magnitude decreases. The AGC system keeps input levels near the maximum to minimize errors. For a full description of the CORDIC algorithm and its computational accuracy, see chapter 25 of reference 6.

The resampling filter may be configured to generate more than one output with a fixed delay between samples. The timing recovery module then processes the phase or frequency for each sample. Early, nominal and late samples are used to measure the slope of the signal prior to and after the sampled data as shown in figure 15. Samples are present when IV is true and the last sample is indicated by setting the final flag. Adjacent samples are stored in registers and subtracted. When the final sample arrives, the clock enable signal goes true for two cycles. This enables the adder/subtracter and the nominal-early and late-nominal signals are accumulated. Subtraction is performed when the difference is negative so the magnitude of the error is recorded. The register following the adder/subtracter stored the first result as the second is being processed. The accumulator, shift register and subtracter form a moving average filter that averages the two error signals over 8 sample periods. The CPU can use this information to alter the time delay and optimize the sampling point for PSK and FSK modems.

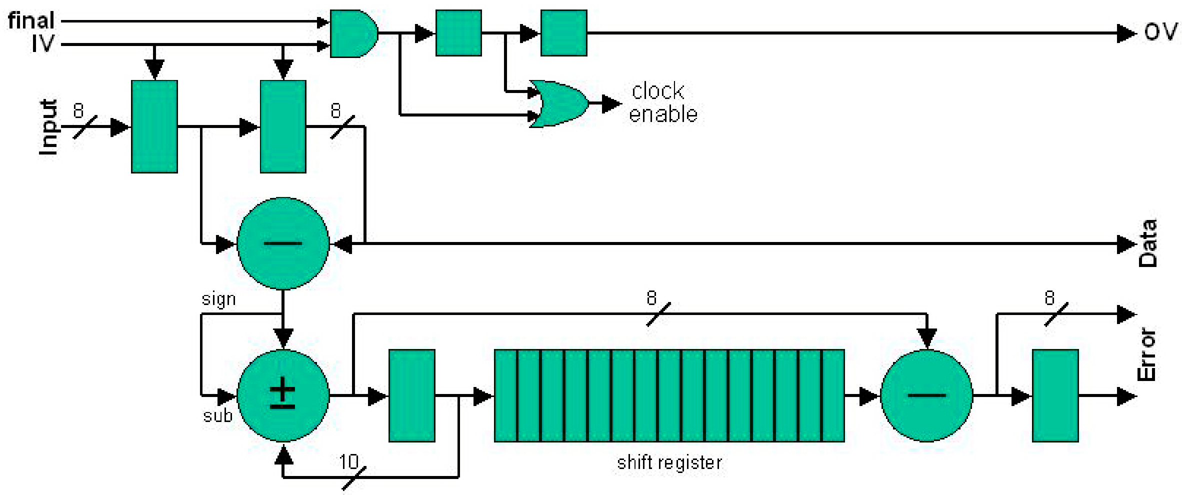


Figure 15 – Timing Recovery

OFDM requires a different type of timing recovery. The null detector, shown in figure 16, is a variable-length moving average filter for the signal magnitude plus additional logic and is used to accurately detect the end of the null symbol. The input is the signal magnitude from the CORDIC module and is added to a 24-bit accumulator. The contents of the RAM are then read and subtracted from the top 18 bits of the accumulator to provide a received signal strength indication (RSSI). Simultaneously, the accumulator value is written into the RAM. The address counter is incremented after each write and is reset to zero when the symbol length is reached. The RAM thus acts as a variable-length time delay.

The RSSI is applied to three comparators and two registers. If the RSSI is greater than the previous maximum, the new maximum is stored in a register and used to set a low signal threshold. While the RSSI is less than this threshold, the minimum signal level register is updated whenever RSSI is less its current value. Each update also loads a counter with a time delay. When the minimum signal level has passed, the count will proceed to zero and the counter will stop. A flip-flop and gate detect this transition and issue a start of frame (SOF) pulse. The FFT may be configured to start automatically when SOF occurs or manually by software intervention. The time delay is usually set to be the cyclic prefix interval.

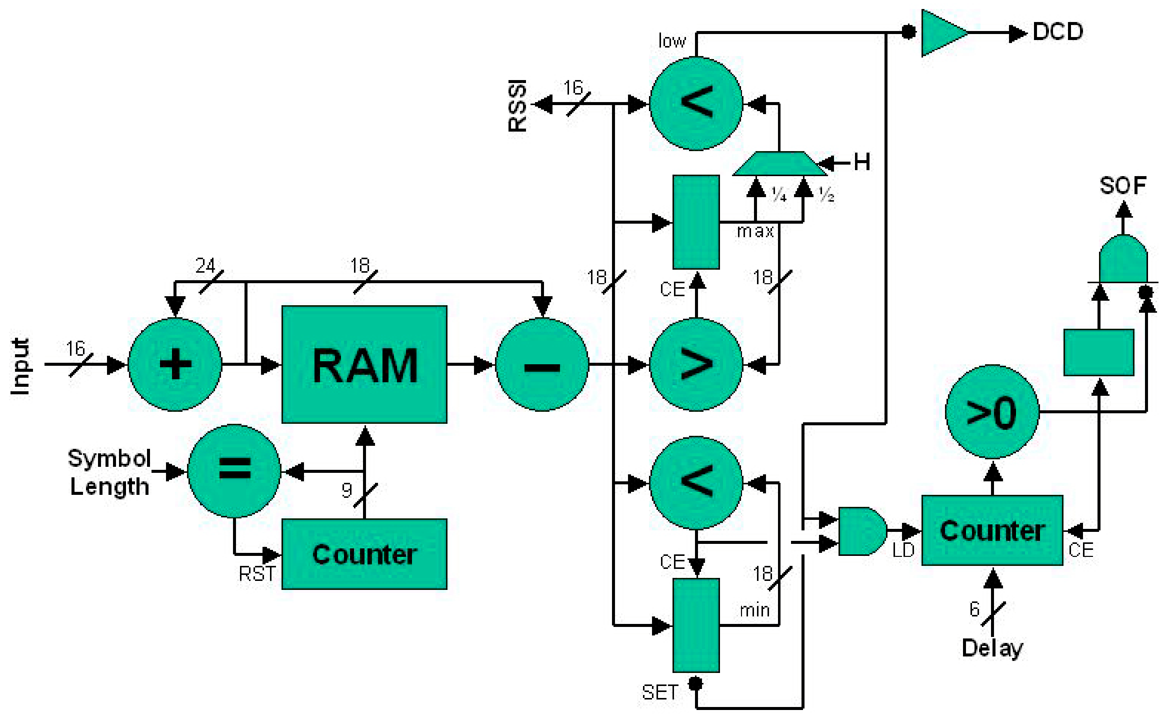

Figure 16 – Null Detector

The phase correlator, shown in figure 17, compares the cyclic prefix of each received symbol with the end of the symbol from which it was copied. The input is the top 8-bit of the phase output of the CORDIC module. As each sample arrives, the 1024 x 8 RAM is read and subtracted from the input. At the same time, it is written to the RAM and the address counter is incremented. The RAM acts as a variable-length shift register with the length determined by resetting the counter when the FFT size is reached. The phase difference is added to a 12-bit value when positive or subtracted from that value when negative and the magnitude of the phase difference is accumulated. This is applied to a variable-length shift register whose output is subtracted from the current accumulated value to give the average phase error over that time interval. When the average phase error is less than 90° (¼ full scale), the minimum value of the phase error is tracked by updating the minimum value register whenever it is greater than the current average. An update also loads a down counter with a time delay. Once the minimum phase error has passed the counter proceeds to zero, stops counting and a sync pulse is emitted. The FFT module uses the sync pulse to save the current sample counter and this information may be used to monitor the placement of the OFDM sampling window.

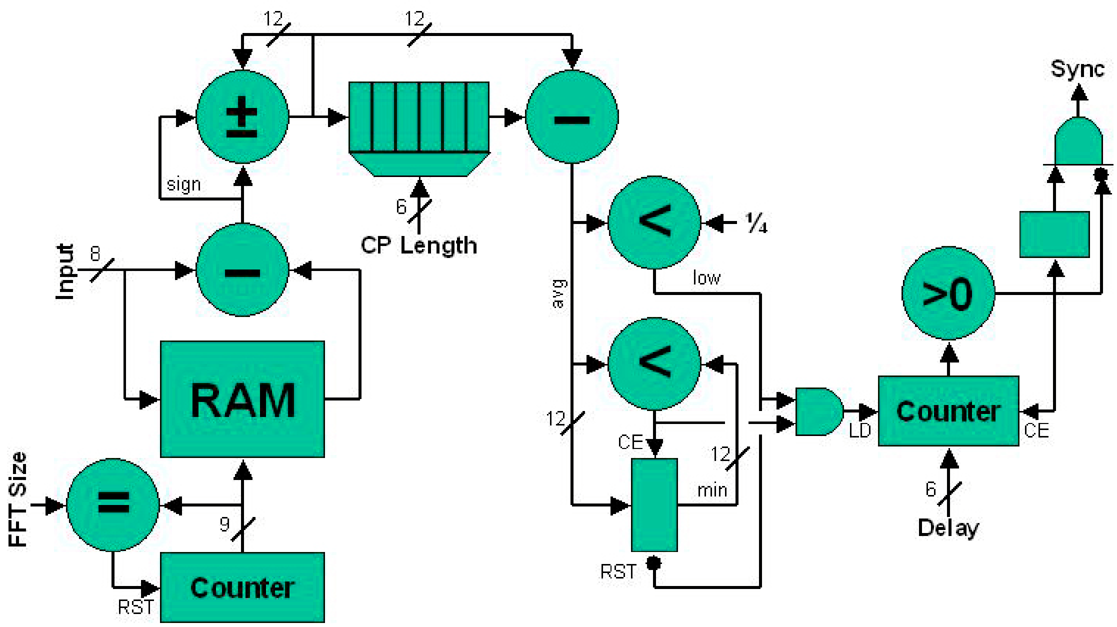


Figure 17 – Phase Correlator

Eight registers, shown in figure 18, configure the modem. The first two registers are used to load baseband data, with the X register loaded first and the 2 samples placed in the FIFO when the Y register is written. There are also 4 status registers. The X and Y registers should be read in sequence and the FIFO entry will be updated after the Y input is read. The E and F bits indicate that the FIFO is empty or full. RSSI is updated after every sample is processed, but is not buffered as it changes slowly.

**Modem Command**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | X Input | In-phase or Magnitude Data | | | | | | | | | | | | | | | |
| 41 | Y Input | Quadrature or Phase Data | | | | | | | | | | | | | | | |
| 42 | Configuration | | | | | | | | TRE | FM Delay (1-16) | | | | | | SSB | FM |
| 43 | | | | | | | | | | | | | | | | | |
| 44 | Phs. Correlator | CP Length - 1 | | | | | | FFT Size | | | | | | | | | |
| 45 | Null Detector | SOF Output Delay | | | | | | H | Symbol Length | | | | | | | | |
| 46 | BFO | BFO Frequency | | | | | | | | | | | | | | | |
| 47 | Control | | | | | | | | | | | | | | INI | RST | XMT |

SSB: 0) AM/PM/FM (CORDIC vector mode) or 1) LSB/USB/ISB transceive (CORDIC rotate mode)
FM: 0) AM/PM or 1) AM/FM transceive (only when SSB = 0)
TRE: 0) analog 16-bit receiver outputs or 1) enable resampler and clock recovery for digital modes
H: 0) -12 dB threshold or 1) -6 dB threshold for null detector
RST: flush FIFO
INI: initialize null detector by making current magnitude the maximum magnitude

**Modem Status**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | X Output | In-phase or Magnitude Data or Recovered Symbol Value | | | | | | | | | | | | | | | |
| 41 | Y Output | Quadrature or Phase Data or Timing Error | | | | | | | | | | | | | | | |
| 42 | RSSI | Average Magnitude | | | | | | | | | | | | | | | |
| 43 | FIFO Status | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | E | F |

Figure 18 – Modem I/O Registers

The FM bit enables phase accumulation on transmit and phase differentiation on receive. The FM delay determines the number of samples over which differentiation is done. There is always one output per sample. The SSB bit enables the BFO for receive and transmit.

An additional FIR filter is provided that may be used for audio filtering, sideband separation, pilot carrier tracking or FSK waveform shaping. It may be configured to implement 2 to 16 filters of 31-255 taps and may downsample by any integer factor and upsample by a factor of 1 to 7. The CPU writes data to the filter, initiates a convolution and then reads the result(s). The filtering occurs at a rate of one tap per clock cycle and in parallel with any program that the CPU is executing.

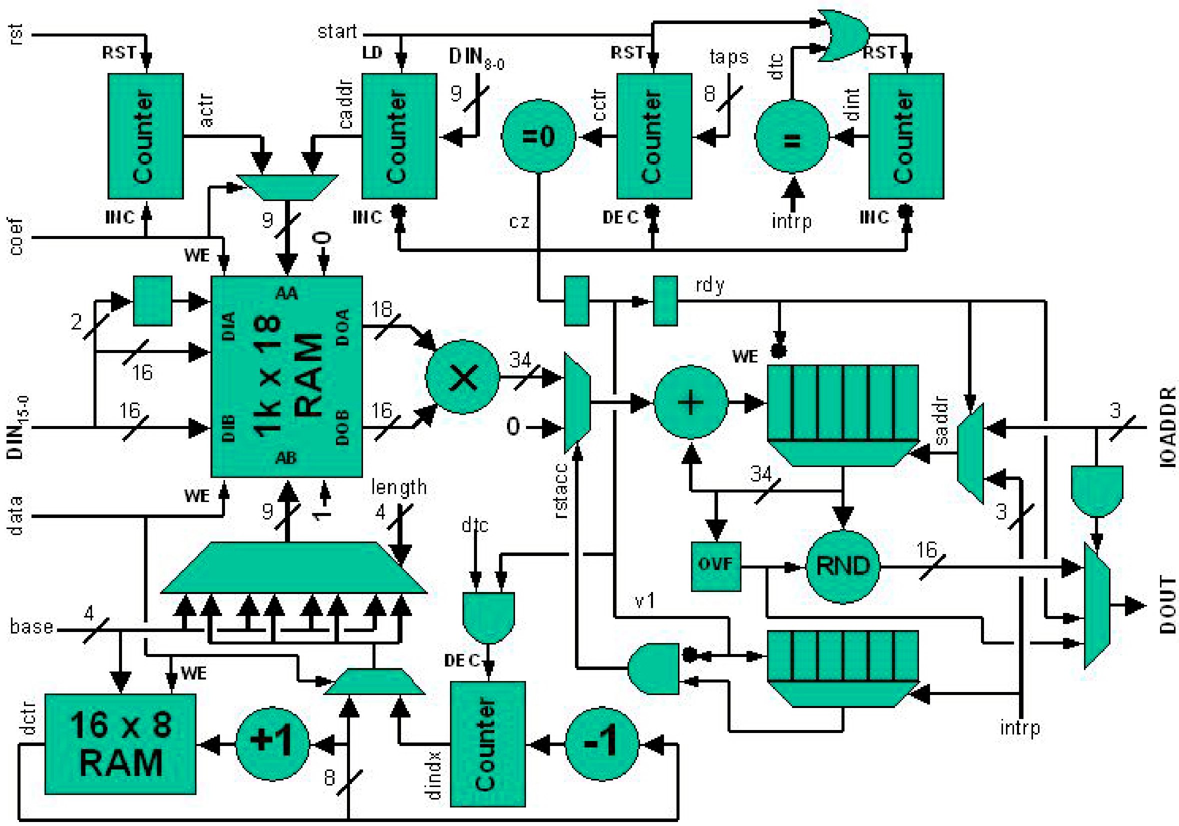


Figure 19 – FIR Filter

A single 1024-word dual-port RAM is used to store 16-bit data and 18-bit coefficients. Coefficients are initialized using two output ports. The most significant 2 bits are stored in a temporary register at one port and then combined with the least significant 16 bits written to the second port. All coefficients for all filters must be written in sequence and the address counter (actr) increments on the LSW write. Data is written to another output port and is stored at an address supplied by one of 16 address counters (dctr) that are implemented with a 16 x 8 RAM. The counter used is selected by the base address register. The base address is combined with the counter (actr or dctr) in a 4:1 multiplexer. The length configuration register determines which RAM address bits are obtained from the counters and the base address register. This allows the creation of 32, 64, 128 or 256-word data buffers. The base address used must be on a 32, 64, 128 or 256-word boundary, respectively.

The filtering process starts by writing the base address of the coefficient set to a special output port. This address loaded into the coefficient address counter (caddr) where it is incremented as coefficients are retrieved. The base address of the current data buffer is decremented and loaded into the data index counter (dindx). The index decrements as data is retrieved from the one port of the RAM and multiplied by coefficients from the other port. The number of filters taps is determined by the taps configuration register and is load into a down counter (cctr) when the filter starts. The filter engine runs until that counter reaches zero. The data/coefficient products are added to an accumulator implemented in a 34-bit wide shift register. The length of that shift register is one if no interpolation is done. The first product must be loaded into the shift register so a multiplexer switches the feedback from the shift register output to zero for the first sample in the filter.

Loading multiple data words and then starting the filter provides downsampling. Upsampling is more complicated and uses the data interpolation counter (dint). Normally this counter is zero and does not change value. When the intrp configuration register is non-zero the counter increments until that value is reached and resets to continue incrementing again until the coefficient counter (cctr) goes to zero and the signal cz is true. Intrp also controls the length of the accumulator shift register and the length of the rstacc signal that causes loading instead of adding. A 1-bit wide shift register implements a delay line that controls the length of rstacc.

**FIR Filter Command**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **28** | Input Sample | Data | | | | | | | | | | | | | | | |
| **29** | Load Filter | LSW | | | | | | | | | | | | | | | |
| **2A** | Coefficients | | | | | | | | | | | | | | | MSB | |
| **2B** | Reset Config. | | | | | | Interpolation-1 | | | Filter Length (Taps) | | | | | | | |
| **2C** | Configure | | | | | | | | | | | Length | | Data Base Address | | | |
| **2D** | Start Filter | | | | | | | | Coefficient Base Address | | | | | | | | |
| **2E** | Rst. Coef. Addr. | | | | | | | | | | | | | | | | |

Length is power of two minus five (0=32, 1=64, 2=128, 3=256)
Data base address is in 32-word segments

**FIR Filter Status**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **28** | Result 0 | Data | | | | | | | | | | | | | | | |
| **29** | Result 1 | Data | | | | | | | | | | | | | | | |
| **2A** | Result 2 | Data | | | | | | | | | | | | | | | |
| **2B** | Result 3 | Data | | | | | | | | | | | | | | | |
| **2C** | Result 4 | Data | | | | | | | | | | | | | | | |
| **2D** | Result 5 | Data | | | | | | | | | | | | | | | |
| **2E** | Result 6 | Data | | | | | | | | | | | | | | | |
| **2F** | Status | OVF | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RDY |

Figure 20 – FIR Filter Configuration and Status Registers

The I/O ports used by the filter coprocessor are shown in figure 20. Note that the first 7 status registers map to the accumulator shift register. Only the first is used when there is no downsampling. The last status register contains flags. The RDY flag is 0 when the coprocessor is busy and 1 when the coprocessor is idle. The OVF flag is set if the currently selected accumulator has overflowed.

## 5. Fast Fourier Transform Module

The FFT module is shown in figure 21 and uses free intellectual property from Xilinx. The Xilinx IP is not shown in detail but a simplified version of the surrounding logic is shown and it is useful to understand how the FFT is controlled. A good description of the FFT algorithm can be found chapter 4 of reference 7.

The leftmost dual-port RAM is 512 entries by 32-bits and is split in half with one half containing the data in the process of being received or transmitted and the other half containing the data being loaded or unloaded by the FFT. The swap bit is inverted to switch halves when a new sample period starts. The counter (sctr) on the left-hand side of the diagram controls sampling. It is reset when not receiving or transmitting and enabled when transmission starts or when the SOF (start of frame) input is pulsed. It counts when RIV is true (reception) or TOE is true (transmission). The output is the address for reading data from (TDO) or writing data to (RDI) the RAM and counts up until the number of samples in the FFT (fftlen) is reached. At that point it is loaded with the inverted cyclic prefix length (cplen) and counts up from that negative number. Writing to the RAM is inhibited during the cyclic prefix. The upper 1, 2 or 3 bits of the RAM address are removed by the mask logic for 128, 64 and 32-point FFTs, respectively. During reception, the FFT starts when the data carrier detect (DCD) input is true and the counter is loaded with cplen. During transmission, DCD is ignored.

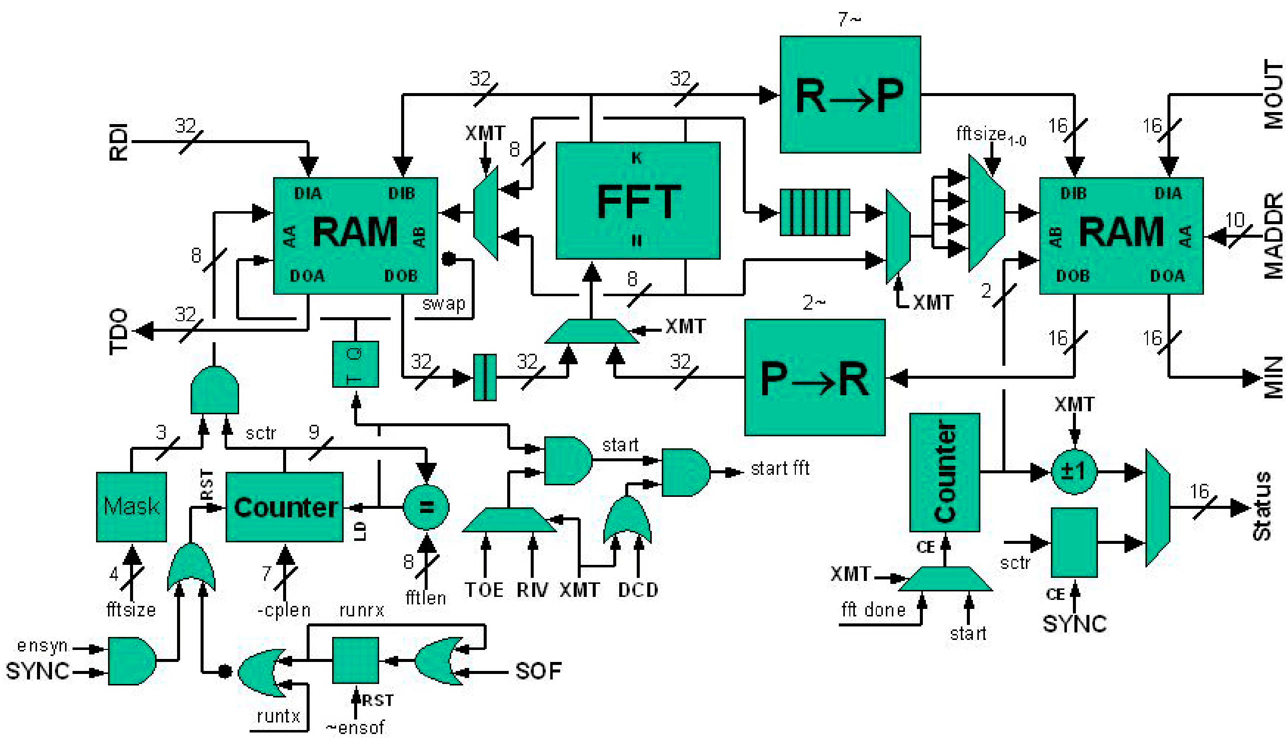


Figure 21 – FFT and Buffer Memory

The Xilinx FFT module provides 5-8-bit addresses when reading data into the N input or writing data from the X output. These addresses are multiplexed into the port B RAM address port for time-oriented data. Three clock cycles are provided for reading data so there is a 2-clock delay in the data path from the leftmost RAM. The rightmost RAM stores data ordered by frequency and is connected directly to the CPU on the port A side. The other side is accessed by the FFT via logic that converts between rectangular and polar coordinate systems. Conversion to polar coordinates (phase and magnitude) takes time so there are six pipeline registers in the address path. The address is modified by a 4-input multiplexer to center the zero-frequency sample in each buffer with positive and negative frequency samples on either side. The RAM is 1024x16-bits but is divided into four 256-sample by 16-bit buffers. The upper 2 address bits are provided by a base address counter which is incremented when the FFT is complete (receive) or the FFT is started (transmit). This value is incremented or decremented by one and provided as status to the CPU.

The other status port is the sample phase register that saves the sample counter value when SYNC pulses are received. When the ensof configuration bit is true these pulses will also reset the sample counter. There are four other status bits that are not shown in the logic diagram but are shown in the I/O port diagram, below.

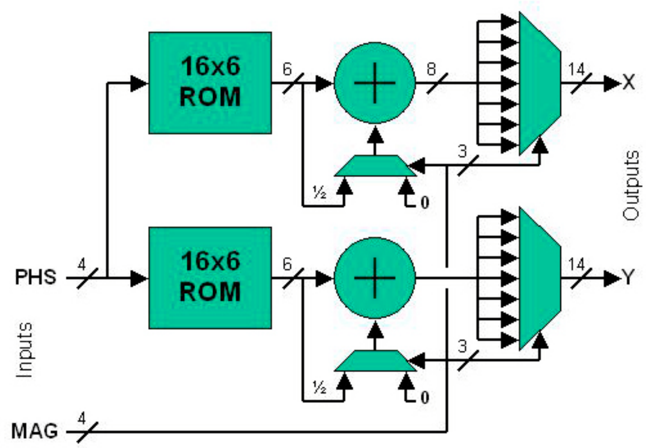


Figure 22 – Polar to Rectangular Conversion

The input to the IFFT consists of a 4-bit phase and a 4-bit magnitude. These are converted to Cartesian coordinates by the circuit in figure 21. Two 16 entry by 6-bit ROMs contain sine and cosine look-up tables. The outputs can be increased by approximately 3 dB by multiplying by 1.5. 6 dB steps are obtained by using two 8-input multiplexers as shifters. One multiplexer input is zero to allow disabling the subcarrier. This gives a 39-dB range.

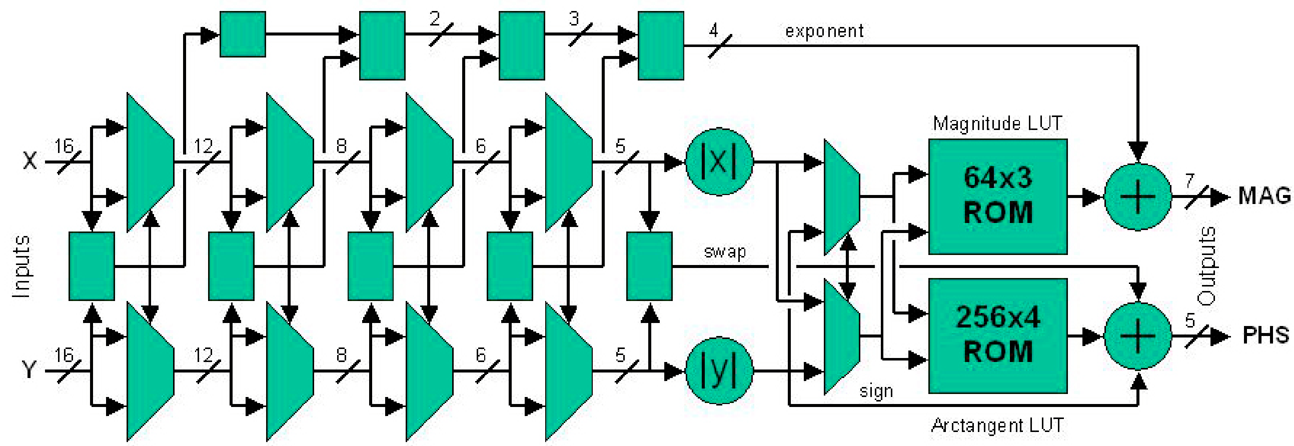


Figure 23 – Rectangular to Polar Conversion

The output of the FFT is converted from 16 x 16-bit Cartesian coordinates to 5 x 7-bit polar coordinates by the circuit in figure 23. CORDIC is not used as it would require more resources given such limited precision. The inputs are first converted to a floating-point format with a 4-bit exponent, two sign bits and two 4-bit fractions. One X/Y pair is converted per clock cycle by using four 2-input multiplexers and making a decision at each stage to shift by 8, 4, 2 or 1 bit positions as the samples travel through the circuit. The 5 x 5-bit output is then rotated into the first quadrant by taking the absolute value and swapping X and Y when necessary. The arctangent is obtained from a 256x4 ROM and the output is adjusted back to the correct quadrant by an adder. This process reduces the size of the ROM by a factor of 4. The output is 5 bits of phase that is suitable for soft-decision Viterbi decoding of 8PSK. The logarithm of the magnitude is obtained from the 64x3 ROM and added to the exponent. This gives a 7-bit logarithmic magnitude output.

**FFT/IFFT Command**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 48 | FFT Reset | FWD | | | | | | | | | | | | FFT Size ($2^x$) | | | |
| 49 | FFT Configure | | Cyclic Prefix Length | | | | | | | FFT Length - 1 | | | | | | | |
| 4A | FFT Start | | | | | | | | | | | | | SYN | SOF | RST | RUN |
| 4B | FFT Transmit | | | | | | | | | | | | | | | | |
| 4C | FFT Scaling | 7 | | 6 | | 5 | | 4 | | 3 | | 2 | | 1 | | 0 | |

SYN = synchronize FFT to phase correlator output
SOF = start FFT when start of frame (null symbol) detected
RST = flush FIFO
RUN = manually start IFFT
Scaling factors are 0-3 right shifts per iteration.

**FFT/IFFT Status**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 48 | Current Addr. | 0 | 0 | 1 | 0 | 0 | 0 | Base Addr. | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 49 | Status Flags | INT | BSY | EOF | RUN | SOF | 0 | 0 | Synchronization Detector Phase (samples) | | | | | | | | |

INT = FFT/IFFT complete interrupt (reset when read)
BSY = FFT/IFFT in progress
EOF = end of received frame (RSSI below threshold when sampling complete – no FFT started)
RUN = sampling
SOF = start of frame (null symbol detected)

**FFT/IFFT I/O Buffer Format**

| Use | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TX | Magnitude (3 dB steps) | | | | | | | | Phase (22.5° steps) | | | | | | | |
| RX | Magnitude (1.5 dB steps) | | | | | | | 0 | Phase (11.25° steps) | | | | | 0 | 0 | 0 |

**IFFT Magnitude Encoding**

| SC | F0 | E0 | D0 | C0 | B0 | A0 | 90 | 80 | 70 | 60 | 50 | 40 | 30 | 20 | 10 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -15 | -18 | -21 | -24 | -27 | -30 | -33 | -36 | -39 | -42 | -45 | -48 | -51 | -54 | -∞ | -∞ |
| 32 | +15 | +12 | +9 | +6 | +3 | 0 | -3 | -6 | -9 | -12 | -15 | -18 | -21 | -24 | -∞ | -∞ |
| 64 | +21 | +18 | +15 | +12 | +9 | +6 | +3 | 0 | -3 | -6 | -9 | -12 | -15 | -18 | -∞ | -∞ |
| 128 | +27 | +24 | +21 | +18 | +15 | +12 | +9 | +6 | +3 | 0 | -3 | -6 | -9 | -12 | -∞ | -∞ |
| 256 | +33 | +30 | +27 | +24 | +21 | +18 | +15 | +12 | +9 | +6 | +3 | 0 | -3 | -6 | -∞ | -∞ |

Figure 24 – FFT Configuration and Status Registers

The FFT module is configured via the 5 registers shown in figure 24. The FFT should be reset before changing other parameters. The current address register contains the address of the 256-word buffer to be used to prepare the next symbol to be transmitted or the location of the buffer with the last received symbol. The INT bit is set when the buffer address changes. The SOF and EOF bits can be used to determine the start and end of the frame. SOF is set when a null symbol is detected and EOF is set when RSSI drops by 6 or 12 dB depending on the setting of the H bit in the modem configuration.

## 6. Error Control Logic

Many communications protocols use cyclic redundancy checks (CRCs) for error detection. Calculations on a general-purpose processor are time consuming, as it requires bit manipulation. The CRC hardware shown in figure 25 provides a means to generate and check the two most commonly used CRCs quickly. The CPU writes bytes or words in parallel to the two shift registers. This sets a counter to the number of dibits to be processed and enables the CRC-16 and CRC-32 logic. The shift register outputs are multiplexed at twice the CPU clock rate so the CRC logic operates at 160 Mbps. Thus a word can be processed in 8 clock cycles or a byte in 4 clock cycles.

CRCs can be computed in parallel with data copying by the CPU. The block copy routine is modified to output a word between the memory read and memory write. After the transfer is complete, the CPU inputs one of the accumulated CRCs and appends it to the data. CRC-32 is the algorithm used by Ethernet and CRC-16 is the algorithm used by HDLC and AX.25.

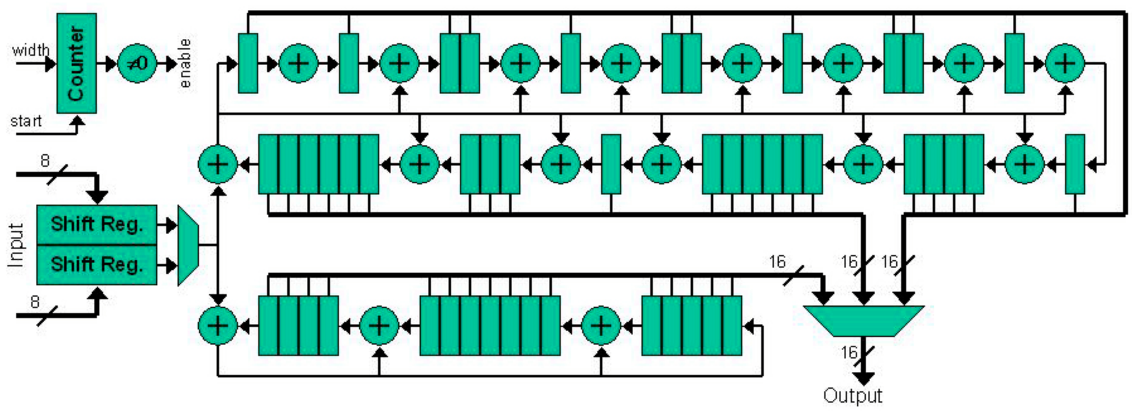


Figure 25 – Cyclic Redundancy Check Logic

**CRC Command**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 38 | Byte Input | | | | | | | | | Data | | | | | | | |
| 39 | Word Input | Data | | | | | | | | | | | | | | | |
| 3B | Initialize CRC | | | | | | | | | | | | | | | | |

CRCs initialized to all one's

**CRC Status**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 38 | CRC32L | LSW CRC-32 (~x[16:31]) | | | | | | | | | | | | | | | |
| 39 | CRC32H | MSW CRC-32 (~x[0:15]) | | | | | | | | | | | | | | | |
| 3A | CRC16 | CRC-16 (~x[0:15]) | | | | | | | | | | | | | | | |

Figure 26 – CRC Configuration and Status Registers

CRCs only allow error detection, but convolutional codes, including trellis-coded modulation, allow error correction. They may be implemented using the programmable convolutional encoder and Viterbi decoder modules.

The encoder exclusive-ORs delayed versions of the uncoded data bits to generate coded outputs. Configuration registers select the bits to be used. Figure 27 shows the logic for encoding one bit. The configuration register, AND gates and exclusive-OR gate are replicated 4 times. Two to four outputs may be generated for each input so that BPSK, QPSK and 8PSK can be accommodated. Either natural or gray coding may be selected.

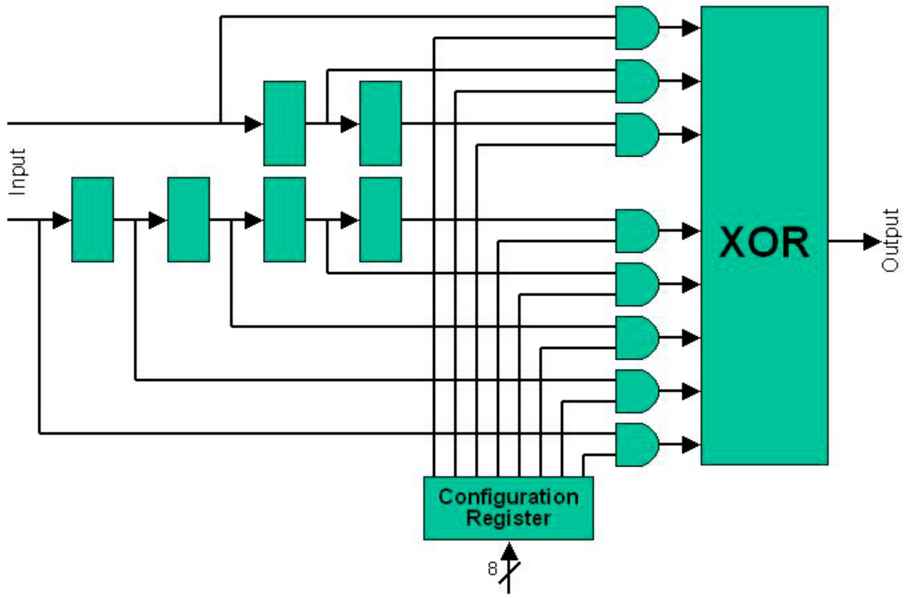


Figure 27 – Convolutional Encoder

The data input port is used to start encoding and the result may be read from a status port with the next instruction. Two bits may be encoded into 8PSK or one bit may be encoded into QPSK. Alternatively, one bit may be encoded into two BPSK channels.

**Convolutional Encoder Command**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | Data Input | | | | | | | | | | | | | | | Data | |
| 11 | Output Mag. | | | | | | | | | | Magnitude | | | | | | |
| 14 | Chan. 0 8PSK | | | | | | | | | U1 | S10 | S11 | U0 | S00 | S01 | S02 | S03 |
| 15 | Chan. 0 QPSK | | | | | | | | | U1 | S10 | S11 | U0 | S00 | S01 | S02 | S03 |
| 16 | Chan. 0 BPSK | | | | | | | | | U1 | S10 | S11 | U0 | S00 | S01 | S02 | S03 |
| 17 | Chan. 1 BPSK | | | | | | | | | U1 | S10 | S11 | U0 | S00 | S01 | S02 | S03 |

**Convolutional Encoder Status**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | Binary Chan. 0 | Magnitude | | 0 | 0 | 0 | 0 | 0 | 0 | | 8PSK | | 0 | 0 | 0 | 0 | 0 |
| 11 | Binary Chan. 1 | Magnitude | | 0 | 0 | 0 | 0 | 0 | 0 | BPSK | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | Gray Channel 0 | Magnitude | | 0 | 0 | 0 | 0 | 0 | 0 | | 8PSK | | 0 | 0 | 0 | 0 | 0 |
| 13 | Gray Channel 1 | Magnitude | | 0 | 0 | 0 | 0 | 0 | 0 | BPSK | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 28 – Encoder Configuration and Status Registers

The Viterbi algorithm is an efficient method to decode convolutionally encoded data. It keeps track of the distance between symbols in a sequence in order to estimate the most-likely transmitted symbol in a noisy environment. The encoder assumes a variety of states as it transmits data and the decoder determines the probability of it being in a particular state by comparing distances.

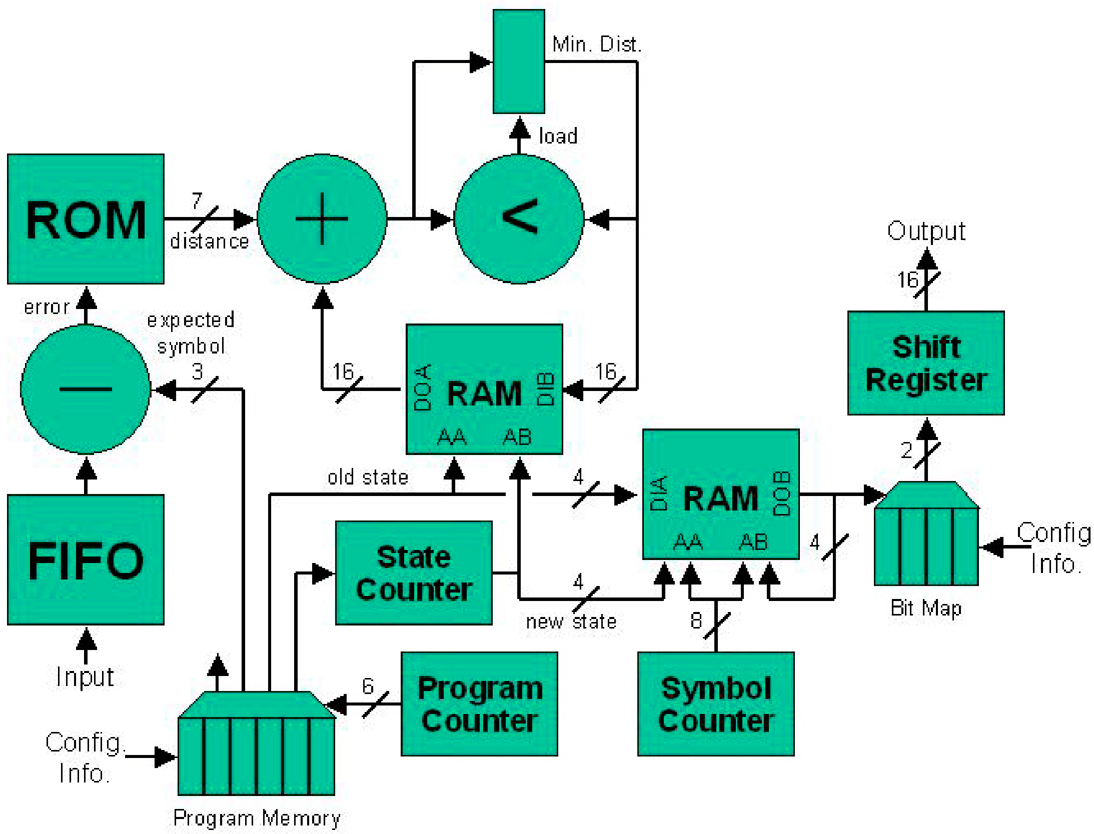


Figure 29 – Viterbi Decoder

Received symbols are written to a 16-entry FIFO in the Viterbi decoder. As symbols are extracted from the FIFO, they are processed once for each possible new state of the transmitting convolutional coder. The received symbol is compared to the expected symbol for a particular previous state and the phase error is converted to a distance. This new distance is added to the cumulative distance to the previous state retrieved from the 32-entry dual-port metrics RAM and compared to the current minimum distance. If it is less than the minimum it replaces the minimum and the previous state is saved. This process generates a set of metrics measuring the probability that each input symbol matches each possible transmitted symbol. The likely previous state for each possible new state is stored in a dual-port block RAM.

The block RAM can store path information for up to 256 symbols. After all symbols have been processed, the contents of the ROM are scanned in reverse order, tracing back over the most likely path that the transmitting encoder took. The traceback logic requires that the last state be zero. As traceback occurs, the states are looked up in a bit map and the decoded data bits are shifted into an eight-dibit register. The bit values corresponding to each state are loaded in reverse order via the bit decoding configuration register. The traceback process stops on every eight symbol and the OV bit is set to allow the CPU to read the decoded byte or word via the first two status registers. The number of symbols remaining is contained in the lower 8 bits of the third status register.

**Viterbi Decoder Command**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 18 | Input Signal | | Symbol 1 | | | | | | | | | Symbol 0 | | | | | |
| 19 | Configuration | Two | | | | | | | | | | | | Paths to evaluate | | | |
| 1A | Path List Input | | | | | | EOS | LP | FP | Symbol 0 | | | Sig1 | | Previous State | | |
| 1B | Bit Decoding | | | | | | | | | | | | | | | Value | |
| 1C | Start Traceback | | | | | | | | | | | | | | | | |
| 1D | Reset | | | | | | | | | | | | | | | | |

**Viterbi Decoder Status**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 18 | Byte Output | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | Data | | | |
| 19 | Word Output | | | | | | | | Data | | | | | | | | |
| 1B | Symbol Counter | OV | IE | IF | 0 | 0 | 0 | 0 | 0 | | | | | Symbols in Buffer | | | |

OV: data output valid
IE: input FIFO empty
IF: input FIFO full

Figure 30 – Viterbi Decoder Configuration and Status Registers

The states and symbols assumed in the decoding process are stored as a program in a 64-entry by 11-bit shift register. The decoder can be programmed for any forward error-correcting code with less than 16 states using 2, 4 or 8-level signaling. Instructions are loaded via the path list input register shown in figure 30. The number of paths to evaluate is also specified and determines the number of clock cycles required for decoding. Each path requires 4 clocks and 4 additional clocks are required for pipeline delays.

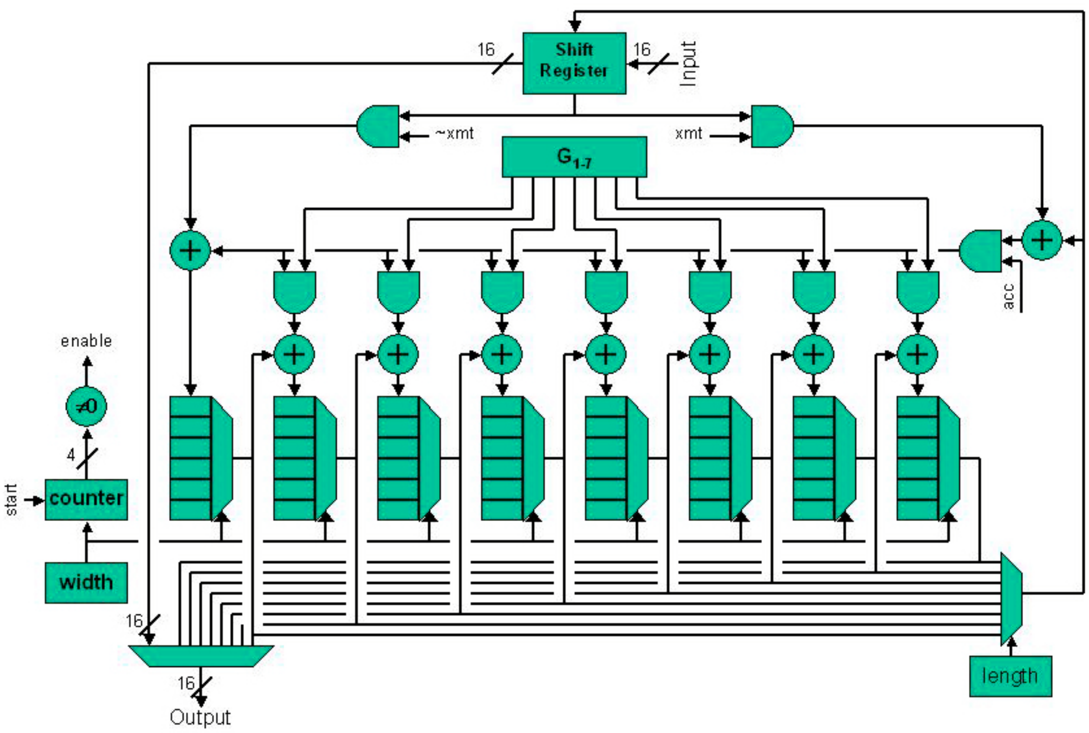


Figure 31 – BCH Encoder/Decoder

A Bose Chaudhuri Hocquenghem (BCH) CODEC module supports block codes with 8-247 data bits and 3-8 parity bits. It may be configured for specific codes by specifying the length of the shift register and placement of the feedback taps ($G_{1-7}$). The word width (N) is variable so that encoding or decoding of 1 to 16 data bits may occur in parallel at one clock cycle per bit.

The BCH algorithm is executes simultaneously on multiple bit streams where each bit in a word is part of an independent stream. When transmitting, the CPU writes data to an output port with ACC set to 1. This loads the shift register and on each of the following clock cycles one data bit passes through the exclusive-OR on the extreme right side of the diagram where it is combined with the currently selected bit in each shift register. On each clock cycle, all shift registers advance and the next bit is processed. Setting ACC to 0 causes the written data to be ignored but the next accumulated parity bits are moved into the shift register where they can be accessed at an input port and transmitted serially. When receiving, ACC is set to 1 and both data and parity words are written to the output port. The accumulated syndrome for each bit can be read by setting the width register to the bit number. The CPU then uses any non-zero syndrome for error correction.

**BCH CODEC Command**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 50 | Data | Data | | | | | | | | | | | | | | | |
| 51 | Configuration | | Length - 1 | | | | | | | G7 | G6 | G5 | G4 | G3 | G2 | G1 | |
| 52 | Configuration | | | | | | | | | | | | | Width - 1 | | | |
| 53 | Control | | | | | | | | | | | | | | | | ACC |

**BCH CODEC Status**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 50 | Parity | Parity | | | | | | | | | | | | | | | |
| 51 | Syndrome | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Syndrome | | | | | | | |

Figure 32 – BCH Configuration and Status Registers

## 7. I/O Ports

Several methods are provided for the DCP-3 to communicate digitally with other devices. The internal circuitry is not described in this document, as it concentrates on the signal-processing hardware.

**Ethernet Command**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Receive Done | | | | | | | | | | | | | | | | |
| 1 | Start Transmit | | | | | | Length (bytes) | | | | | | | | | | |
| 2 | MDIO Bits | | | | | | | | | | | | | | DIR | CLK | DAT |
| 3 | Control | | | | | | | | | | | | | | | | ENA |

DIR: PHY management data (DAT) direction (0=receive, 1=transmit)
ENA: enable (1) or disable (0) MAC and PHY

**Ethernet Status**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Status | MDI | INT | TXE | RXR | RXF | Length (bytes) | | | | | | | | | | |

MDI: PHY management data input
INT: PHY management interface interrupt input

Figure 33 - Ethernet Configuration and Status Registers

The Ethernet port uses a single transmit buffer and double-buffered receive. 2 kB of dual-port RAM is provided to store the frame to be transmitted and 4 kB of dual-port RAM is provided to buffer received frames. The CPU may access one received frame at the same time another is being received. The hardware supports only full-duplex operation at 100 Mbps at this time. Transmission is accomplished by writing a frame and its preamble to the transmit buffer and then outputting the length to port 1 as shown in figure 33. The CPU then checks the TXE bit in the status register for completion of transmission before writing another frame. The CPU may also check the RXR bit for received frames and obtain the length of the received frame. It then accesses the receiver buffer memory to retrieve the frame. One half of the buffer is visible which contains the first frame received. When the CPU signals done by writing to port 0, the next frame is presented.

A universal asynchronous receiver/transmitter (UART) is provided for low-speed communication. It is used by the loader firmware for downloading software and may be used for other purposes, such as implementing a TNC. Signaling rates between 110 and 460,800 baud are supported. The baud rate divisor is set to generate the appropriate clock at 16 times the symbol rate. 15-byte receive and transmit FIFOs are provided and may be read and written via port 20. Status flags indicate when the transmit FIFO is empty (TXE), the transmit FIFO is ready for data (TXR), the receive FIFO is full (RXF) and received data is available (RXR). The receive FIFO has a framing error bit for each entry. The UART supports only 8-bit characters without parity.

**UART Command**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | Data | | | | | | | | | Data | | | | | | | |
| 21 | Baud Rate | Baud Rate Divisor - 1 | | | | | | | | | | | | | | | |

**UART Status**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | Data | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FE | Data | | | | | | | |
| 21 | FIFO Status | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | TXE | TXR | RXF | RXR |

Figure 34 - UART Configuration and Status Registers

Note that the Ethernet interface and UART do not implement CRC generation and checking. This may be done by using the CRC hardware described previously.

There are two SPI ports. One is used to operate the low-speed DAC and the other is connected to the onboard serial flash memory. Writing a byte to the SPI transmit data port sends a command to the flash memory. Simultaneously, data is transmitted from the flash memory and appears in the received data port. The data transmission rate is fixed at 20 Mbps and the CPU must wait 32 clock cycles for operations to complete. Ports A and B control the slave select line, which must be active when sending commands to the flash memory.

**SPI Command**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | Transmit Data | | | | | | | | | Data | | | | | | | |
| A | SS Off | | | | | | | | | | | | | | | | |
| B | SS On | | | | | | | | | | | | | | | | |

**SPI Status**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | Received Data | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Data | | | | | | | |

Figure 35 - SPI Configuration and Status Registers

The lower portion of the serial flash memory contains the FPGA configuration that is loaded automatically when power is applied so it must not be disturbed. However, the upper portion may be used for data and program storage.

The DAC port is 16-bits wide and is write-only. The lower 12 bits control the DAC and the upper 4 bits must be set to zero. 64 clock cycles are required after the write for data to be transferred to the DAC.

**Low-Speed DAC**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 68 | LS DAC | 0 | 0 | 0 | 0 | Value | | | | | | | | | | | |

Figure 36 – Low-Speed DAC Output Register

An I$^2$S port is provided to attach an external 1 or 2-channel 16-24-bit audio CODEC operating at 32 ksps. Data for the left and right channels is written to separate pairs of output ports. The least significant byte (LSB) is written first and is zeroed after every write to the most significant word (MSW). The MSW write causes data to be transmitted. Use of the left and right channels must be alternated for stereo CODECs. Left and right channel data is read from a common pair of input ports and the "Left" status flag indicates the active channel. Both ports have a 15-entry FIFO. Status flags indicate FIFO status and input sample source (left or right channel).

**I$^2$S Command**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 30 | Right Channel | LSB | | | | | | | | | | | | | | | |
| 31 | Data | MSW | | | | | | | | | | | | | | | |
| 32 | Left Channel | LSB | | | | | | | | | | | | | | | |
| 33 | Data | MSW | | | | | | | | | | | | | | | |

**I$^2$S Status**

| | Register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 30 | Right and Left | LSB | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 31 | Chan. Data | MSW | | | | | | | | | | | | | | | |
| 33 | Status Flags | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | TXE | TXR | 0 | Left | RXF | RXR |

Figure 37 - I$^2$S Configuration and Status Registers

## 8. Conclusion

The combination of a fixed hardware configuration stored in flash memory plus downloadable software has proven to be very flexible. Much development time was spent optimizing the use of FPGA resources. For example, dedicated multipliers are used in the FIR filters where high speed is necessary and serial multipliers are used in the AGC logic after the sample rate has been reduced. Logic such as the CORDIC module was designed for reuse when switching from transmit to receive so little standard IP was used other than the FFT module. That hardware can now be leveraged for multiple applications by changing the software. The current program implements a transceiver for AM, FM and SSB plus a high-speed OFDM modem. The resampler and timing recovery hardware should allow implementation of a high-speed AX.25 TNC in the future.

Several PC-based programs were developed for use with the DCP-3, including an assembler for the CPU and utilities to format configuration information for loading into flash memory. These programs and the complete Verilog source code for the FPGA are available on the TAPR web site. The source code and it derivatives are provided for personal non-profit educational use in the Amateur Radio Service and are not licensed for redistribution.

Since the hardware is defined in Verilog it can be ported to newer and denser FPGAs. The XC3S500 was first shipped in 2006. An XC3S1400A FPGA can now provide more than twice the logic and 60% more memory for little additional cost. This would allow use of ADCs and DACs up to 160 Msps, faster filters and more complex signal constellations for higher data rate OFDM modems. At a somewhat higher price, Spartan-6 series FPGAs can eventually provide even more memory and logic. I plan to make a new version of the board in the near future. In the mean time, I'll be using this one on the air.

**9. References**


1. John B. Stephensen, "Software Defined Radios for Digital Communications", November/December 2004 QEX.

2. John B. Stephensen, "A Soft Processor for DSP", Winter 2009 Packet Status Register.

3. Andreas Chrysafis, "Digital Sine-Wave Synthesis using the DSP56001/2", Motorola Signal Processing Division, 1988.

4. William Sabin, "Automatic Gain Control for CW Reception", QST, July 1963.

5. Fredric J. Harris, "Multirate Signal Processing for Communication Systems", ISBN 0-13-146511-2, Prentice-Hall, 2004.

6. Scott Hauck, Andre DeHon, "Reconfigurable Computing – The Theory and Practice of FPGA-Based Computation", ISBN 978-0-12-370522-8, Elsevier, Inc., 2008.

7. Richard G. Lyons, "Understanding Digital Signal Processing", ISBN 0-201-63467-8, Addison-Wesley, 1997.