TAPR #107 WINTER 2009

PACKET STATUS REGISTER

President's Corner

Christmas is just fading from recent memory and your BoD is already heavily into the planning stages for the Dayton Hamvention. We plan to have the same amount of booth space as last year and our booths will be back-to-back with our AMSAT brethren. We are trying to set up a common area to allow members to sit and enjoy an eyeball QSO with your board members and friends, and just to catch one's breath.

The current plan is for a two-hour TAPR forum on Friday, starting at 0915. The 15-minute delay to the start should allow folks ample time to get into the building and make their way to the meeting room without having to create a stampede. This is just as well, as I wasn't planning on doing any emergency medicine at the show.

We will again be holding our joint TAPR-AMSAT banquet, with speaker and details to follow. AMSAT's Martha and I are working out some of the logistics while N8UR works with his AMSAT counterpart to ensure the same fine facilities. Changes in overtime policy at the Air Force Museum have precluded a return to that facility, but I'd like to think we could work some magic one day and have a return visit

HPSDR continues to occupy a major part of our time and planning. Mercury (the SDR receiver) has been delivered and we still have a few available. Unfortunately, Penelope (the low power exciter) is out of stock from TAPR, but it appears that there will be a European source of additional units. This will allow TAPR to concentrate on Alexares (AKA Alex), the filter system for HPSDR, consisting of separate boards for receive and transmit. A case to enclose Alex will be available. We are also looking at interim power supplies and enclosures for the project, with details to follow.

Your past president, N8UR, is continuing to be very productive with new time-related projects, including expanded-feature pulse generators and dividers. I'm definitely NOT a "Time-Nut," so I'll let John explain his projects in upcoming issues.

And, even though we will be using the same site as last year for the 2009 DCC, TAPR's president of vice, Steve Bible, is hard at work with your Chicago organizers to expand on the successes of 2008 to bring you an even better 2009 experience. If you missed the 2008 spectacle, you can try to acquire a sense of the experience by watching DVDs of the event, produced by Gary Pearce, KN4AQ, of ARVN – Amateur Radio/Video News. Details will be available on the TAPR web-site, at www.tapr.org.

Once we catch your attention with these DVDs, we hope that you will actually COME to the DCC, because NOTHING can replace the experience of actually being there, and having napkin techie-talks with your colleagues.

In closing, mark your calendars now for the Dayton Hamvention, and plan to spend time with us at the booth, forum, and banquet.

73, Dave VE3GYQ/W8 <ve3gyq@tapr.org>

###

DCC Highlights

By John Koster, W9DDD

The high point of DCC? One would probably have to say the outstanding A/V setup. Two large screens with flamethrower projectors in the main ballroom plus two remote large screen LCD monitors and sound in the hallway and the demo room allowed keeping track of presentations wherever you might be.

Kermit Carlson, W9XA, and Mark Thompson, WB9QZB, spent many hours making arrangements and setting up A/V equipment.

Oh, but how about the roast pig Friday night at the social hour, wasn't that was pretty good too?

In attendance:

Joe Lynch, N6CL, editor of *CQ-VHF* and his XYL Carol, W6CL, were in attendance with sample issues.

Gary Pearce KN4AQ recorded all the sessions and will be offering a DVD of the conference. (Details on how to order, later)

Hap Holly of the R.A.I.N. Report was in attendance Saturday and did interviews with many in attendance.

Larry Wolfgang, WR1B, *QEX* editor represented the ARRL and also presented information on the steps involved in getting an article published.

Of course the usual movers and shakers too numerous to mention were there too.

DCC DVDs For Sale

Did you miss the ARRL and TAPR Digital Communications Conference last September? Did you attend, and now want a record of some, or all of the talks?

Good news! ARVN:Amateur Radio//Video News captured every minute of the presentations on video, and now offers them on six DVDs in high quality video and audio.

The presentations are grouped on the six DVDs in as logical an order as possible. You can buy the DVDs individually ($15/each, plus $3 shipping, $4 international). Or get whole set for $75 (plus $3/$4 shipping). The complete set saves you $30 over buying them individually, after shipping cost. You can order by credit card or PayPal at www.arvideonews.com.

Election Results

Steve Bible, N7HPR, Stan Horzepa, WA1LOU, and Darryl Smith, VK2TDS were re-elected as TAPR Board of Director members at DCC 2008.

President David Toth, VE3GYQ, Vice President Steve Bible, N7HPR, Secretary Stan Horzepa, WA1LOU, and Treasurer Tom Holmes, N8ZM, were re-elected as TAPR officers.

Dues Increase

After many years, we find we must increase the annual dues. In spite of many cost cutting measures implemented over the past 10 years, the overall operating costs continue to increase. So effective January 1st, 2009, the annual dues are $25.00.

###

DCC: "Pretty Interesting!"

The ARRL Letter

Almost 150 aficionados of digital communications came to Chicago for the 27th annual ARRL/TAPR Digital Communications Conference (DCC) the weekend of September 26-28 <www.tapr.org/dcc.html>. This conference is an international forum for radio amateurs to meet, publish their work and present new ideas and techniques. Presenters and attendees had the opportunity to exchange ideas and learn about recent hardware and software advances, theories, experimental results, and practical applications. Not only was the conference technically stimulating, it was a weekend of fun for all who have more than a casual interest in any aspect of amateur digital electronics and communications; introductory sessions were scheduled throughout the conference to introduce new technical topics for both beginners and experts.

One of the attendees was *QEX* Editor Larry Wolfgang, WR1B, representing the ARRL <www.arrl.org/qex>. He gave a talk "Writing for Publication - It's Not Rocket Science (Even if You Are Writing About Rocket Science!)." Retired ARRL Chief Technology Officer Paul Rinaldo, W4RI, also represented the League, speaking on "SDR Outlook." Central Division Vice Director Howard Huntington, K9KM, was present, as was ARRL Contributing Editor H. Ward Silver, N0AX.

Wolfgang, who gave his presentation on Friday morning, spoke with many participants throughout the conference who told him that they were enthused about writing something for *QEX*. "It also seemed like almost everyone there was making a point to stop me in the hallways or at mealtimes to tell me how much they are enjoying *QEX*, and to thank me for the effort I putting into the magazine," he said.

Friday was a full day of technical presentations, including "A Protocol for Multicast Weather Data Distribution over AX25," by Nick Luther, K9NL; "SuitSat-2 Update," by Steve Bible, N7HPR; "EcomScs and GateWayScs AX25 Packet Radio E-Mail," by John Blowsky, KB2SCS, and "Frequency and Other New Initiatives in APRS," by Bob Bruninga, WB4APR. The last presentation of the afternoon was "D-STAR Uncovered," by Peter Loveall, AE5PL.

The DCC provided a separate demonstration room for participants to show off their latest projects. Wolfgang said he had some "play time" in this room after all the presentations on Friday. After dinner that evening, he said there was "a string of D-STAR presentations starting at 7 PM and lasting until after 10 PM. These were all very interesting presentations, including the D-RATS messaging software by Dan Smith, KK7DS. Peter Loveall, AE5PL, presented 'D-PRS Update' about ways the D-STAR data links can be used to send and plot APRS position maps or interface to the APRS system. Robin Cutshaw, AA4RC, gave a presentation about DV-Dongle, a PC add-on that allows connection to D-STAR repeaters around the world through an Internet connection. Pretty interesting!"

In addition to the main technical presentations on Saturday, there was also a full day of introductory sessions, mostly about digital voice and D-STAR.

Also on Saturday, Matt Etttus, N2MJI, gave a report on the Ettus Research USRP2. Tom Clark, K3IO, gave an "AMSAT Update." Paul Wiedemeier, KE5LKY, presented his paper on "Using UDPcast to IP Multicast Data over Packet Radio." Paul Rinaldo, W4RI, gave his "SDR Outlook" that covered the international regulatory front and presented some of the concerns that some world

governments have with regard to the flexibility of SDR. Scott Cowling, WA2DFI, gave an "HPSDR (High Performance Software Defined Radio) Update" during which he described the various circuit boards that make up the HPSDR project. He also demonstrated a complete working radio in the demo room. Jerry Shirar, N9XR, presented a paper called "Clocking the Data," concerning the use of an inverter and a crystal to form a Colpitts oscillator.

Victor Poor, W5SMM, gave an update on Winlink 2000. "He reported that the network presently consists of one Web site, five Common Message Server sites around the world, 150 Radio Message Service Pactor Gateway sites, 800 RMS Packet Gateway sites and over 13,000 registered users," Wolfgang said. "He also mentioned that inactive users are purged from the system from time to time."

Rick Muething, KN6KB, described his "WINMOR Soundcard ARQ Mode for Winlink HF Digital Messaging." Wolfgang said that this was a much anticipated presentation: "Rick reported that WINMOR is 3 to 4 months from beta testing. His preliminary comparisons to Pactor 1, 2 and 3 are based on simulator tests. He indicates that WINMOR will have better throughput than Pactor 1 and comparable to, or perhaps a bit better than Pactor 2, but not quite as good as Pactor 3. He indicated that the efficiency is within about 70 percent of Pactor ARQ."

The TAPR Annual Meeting was held after all the presentations on Saturday. During the meeting, the Board of Directors discussed the idea of coming back to the same hotel for next year's DCC. "The local group that supported DCC this year, including Mark Thompson, KA9MDJ, and Kermit Carlson, W9XA, did a great job of providing AV support," Wolfgang said. "Ron Steinberg, K9IKZ, who owns a local audio/visual company, provided two large projection screens and projectors, so every presentation was shown on both sides of the room, as well as on a wide screen TV outside the room. The audio was very easy to hear, even if you were in the hall outside the room or in the demonstration room; the set-up was left there from the W9DXCC Convention the previous weekend. They offered to provide the same support for next year, if the DCC returned. So, the decision was made to break with tradition, and return to Chicago next year, September 25-27."

Phil Harman, VK6APH, presented the Sunday morning seminar, "Software Defined Radio through the Looking Glass." Wolfgang said he found Harman's presentation to be understandable "at my level of knowledge, and he gave a lot of good insight into many of the design decisions that went into the High Performance Software Defined Radio Project. I found his descriptions of how the hardware and software work to be very interesting. The four hours went by quickly."

###

Microwave Engineering Project (MEP) Update

The Problem is Pointing

By Team MEP [1]

Most microwave stations use dish antennas. Dish antennas at the frequencies of operation of interest to MEP, which range from 3.4GHz to 5.6-5.8GHz, have a very narrow beamwidth. For example, a half-meter dish at 3.4GHz has a half-power beamwidth of 12 degrees. A half-meter dish at 5.6GHz has a half-power beamwidth of 7.5 degrees. These narrow beamwidths mean that unless you are pointing your dish in the right direction and at the correct elevation, you will not hear the other station. Proper pointing is crucial.

Traditional methods of accurately pointing one microwave station at another include setting up a fixed set of antennas, using calibrated setting circles on tripods and knowing the precise bearing to the other stations, guessing and searching by moving the dish around, and by using known locations of beacons to contact stations prepositioned near or at the beacon. Many microwave-band contacts are made during microwave band contest with prior planning and advance coordination.

Station discovery is the process of learning about other stations in range of the searching station in order to establish communications. For MEP, station discovery is intended to be a designed-in distributed method that allows stations to discover their current network environment as automatically as possible. Ideally, stations should also be able to advertise their current and potential configurations and services to other stations.

The methods of station discovery under discussion are

- Increasing the beamwidth
- Using APRS direction finding packet format
- Using omnidirectional in-band beacons and in-band signaling

The discovery function that MEP is working on is exciting because it reveals stations that the operator may not know about, allows for opportunistic contacts, and allows the operator to watch, filter, and monitor activity on the microwave bands without having to personally monitor the station.

For any antenna, the process of increasing the beamwidth usually comes at the expense of gain. This naturally leads to the question of whether or not the increased beamwidth dish's "discovery range" would be too small to be useful. The working range is defined by signal strength required to close the full link at its high bit rate. The discovery range, the range over which a searching station can discover other stations, only needs signal detection and perhaps identification. MEP assumes that there is enough difference between the discovery range and the working range to trade bandwidth for beamwidth in order to enable faster and easier signal searching. Work is ongoing to define and model these ranges.

In comparison, weak-signal DX microwave communications are achieved with very little link margin. The discovery range and the working range are assumed to be very similar. In contrast, in what we imagine to be a typical MEP discovery process, we may find that a signal that's loud enough to be demodulated at our data rate is rather easy to detect over a fairly wide range of angles, even with a dish.

Increasing the beamwidth

Increasing the beamwidth is similar in concept to using a spotting scope on a telescope. You have a wider field of view, and less magnification. You orient yourself in the part of the sky you are observing, and then, once you are satisfied that you are pointing at the right object then you switch to the higher-power view through the telescope.

One way to accomplish increasing the beamwidth is by defocusing the dish. Defocusing the dish means moving the feed away from or towards the reflecting surface of the dish. Another way to accomplish increasing the beamwidth is by using a second feed pointing out along the boresight instead of towards the reflector.

While the gain is reduced using these two strategies, it might be possible to match the discovery range to the working range because of the different link requirements of the discovery range compared to the working range.

If discovery works on a narrowband feature of the transmitted signal, it might well be able to detect signals even weaker than the system can demodulate and decode. The narrowband feature could be transmitted solely for that purpose, or it might also be useful as a pilot signal or low-rate data subcarrier.

A disadvantage with this kind of discovery is that it can only find stations that are actively transmitting. This can be mitigated with protocol design, but that leads to problems of power consumption and scalability, as well as concerns about unattended operation.

Using APRS direction finding packet format

As Bob Bruninga describes at www.aprs.org, "Automatic Packet Reporting System (APRS) is a two-way tactical real-time digital communications system between all assets in a network sharing information about everything going on in the local area. On ham radio, this means if something is happening now, or there is information that could be valuable to you, then it should show up on your APRS radio in your mobile. APRS also supports global call sign-to-call sign messaging, bulletins, objects e-mail and Voice because every local area is seen by the Internet System (APRS-IS)! APRS should enable local and global Amateur Radio operator contact at anytime-anywhere and using any device."

Enabling contact is the purpose of the discovery function. The APRS Direction Finding (DF) packet format provides a useful framework for communicating the station discovery information, which includes the position of the station, the direction the station antenna is pointing, the beamwidth of the station, the frequency of the station, and the course and heading if the station is mobile. The information is updated in real time and can be obtained by passively monitoring for station beacons.

In order to accomplish station discovery with APRS, an APRS-equipped transceiver with omnidirectional antenna is attached to or integrated with each MEP station. APRS provides the functions required for station discovery and the results could be input to an automatic pointing subsystem. Because an existing packet format is used, the implementation requires less work than if a custom packet format was developed. The services, modes, or missions of other stations can be determined by APRS query and response, after the station is discovered.

How MEP will use the APRS DF Packet Format

Position coordinates in APRS are a combination of latitude and longitude, separated by a display

Symbol Table Identifier, and followed by a Symbol Code.

For example: 4903.50N/07201.75W-

Latitude is expressed as a fixed 8-character field, in degrees and decimal minutes (to two decimal places), followed by the letter N for north or S for south. Longitude is expressed as a fixed 9-character field, in degrees and decimal minutes (to two decimal places), followed by the letter E for east or W for west. The precision of two decimal minutes of longitude or latitude is about 25 meters.

The / character between latitude and longitude is the Symbol Table Identifier (in this case indicating use of the Primary Symbol Table), and the – character at the end is the Symbol Code from that table (in this case, indicating a "house" icon)[2].

The DF bearing format gives 1-degree pointing precision and also allows for a beamwidth field. Referred to in the specification as Quality (the Q in the NRQ field, explained below), it can express station beamwidth so that other stations could plot an estimate of the antenna pattern. A case for omnidirectional antennas should be reserved. Not all stations will use highly directional dishes.

From page 30 of the APRS Specification [3],

DF reports contain an 8-byte field /BRG/NRQ that follows the CSE/SPD Data Extension, specifying the course, speed, bearing and NRQ (Number/Range/Quality) value of the report. NRQ indicates the Number of hits, the approximate Range and the Quality of the report.

For example, in:

…CSE/SPD/BRG/NRQ…

…088/036/270/729…

course = 88 degrees, speed = 36 knots, bearing = 270 degrees, N = 7, R = 2, Q = 9

BRG is 001 to 360, with 000 indicating an omnidirectional antenna.

The N value is not processed, but is just another indicator from the automatic DF units.

The range limits the length of the line to the original map's scale of the sending station. The range is 2R so, for R=4, the range will be 16 miles.

Q is a single digit in the range 0–9, and provides an indication of bearing accuracy. MEP will use bearing accuracy for beamwidth.

Table 1. Bearing Accuracy

Q	Accuracy
0	Useless
1	<240°
2	<120°
3	<64°
4	<32°
5	<16°
6	<8°
7	<4°
8	<2°
9	<1°

From the APRS Specification page 34,

DF Reports are contained in the Information field of an APRS AX.25 frame. The BRG/NRQ parameters are only meaningful when the report contains the DF symbol (i.e. the Symbol Table ID is / and the Symbol Code is \). If the DF station is fixed, the Course value is zero. If the station is moving, the

Course value is non-zero.

Table 2. DF Report Format without Timestamp

Field	No. of Bytes
1 or =	1
Latitude	8
Symbol Table ID /	1
Longitude	9
Symbol Code \	1
Course/Speed	
Power/Height/	
Gain/Dir	
Radio Range	
DF Signal	
Strength	7
/BRG/NRQ	8
Comment	0 - 28

The next element of information is frequency of operation. This is anticipated to be useful in case MEP employs frequency agility over a large enough portion of the amateur band to where bandwidths can't be assumed to overlap, or to show which band the station is operating on. MEP stations operate in either point-to-point or multiple access modes. Point-to-point operations may occur on a single band. Multiple-access is a dual-band mode. Being able to communicate a frequency may be useful for MEP.

The frequency field, which is part of the proposed APRS Specification 1.2, is expressed as FFF.FFF in MHz. Since MEP stations operate at 3GHz and above, a minor change is needed to express microwave frequencies.

A letter is used to stand for the two numbers that express the thousands and hundreds of MHz. Here are four examples. D stands for 34, E stands for 56, F stands for 57, and G stands for 58.

Table 3: How Codes Enable Microwave Frequency Expressing in Frequency Field

Field	Frequency in MHz
D01.000	3401
E51.000	5651
F60.000	5760
G30.000	5830

Transmitting with one hop via WIDE1-1 once a minute while active has been recommended [4]. If the station is more than one hop away, the odds of a microwave path are thought to be low. Keeping APRS traffic to one hop also helps to keep the man-made interference low.

Here is a template DF packet format with all the fields mentioned so far.

```
AX.25 HEADER:  CALLSIGN>APMEPv,WIDE1-1
DATA: =DDMM.mmN/DDDMM.mmW\000/000/BRG/NRQ/
FFF.FFFMHz ATV-MEP
```

An advantage to using APRS for station discovery is that the information learned about the stations is more detailed than the information that can be learned from scanning for transmitting stations in the microwave band with either a focused or a defocused (wider-angle) beam. Another advantage is that stations need not be actively transmitting on the microwave bands in order to be found by the searching station.

A disadvantage of this method is that it may produce a list of stations that are within VHF range but not within microwave range. This means that the list of stations may include some that are out of microwave range but within VHF range or stations

that are within VHF range but not on a line-of-sight path. Work needs to be done to match, as closely as possible, the VHF discovery range to the microwave working range. Finally, station status must be properly configured and additional VHF equipment reliably working if automatic discovery via APRS is to work. This method is also subject to spoofing, since any APRS station can copy the status of a MEP station and masquerade as a MEP station.

Using omnidirectional in-band beacons and in-band signaling

A third method is to use an omnidirectional in-band beacon or signal. All discoverable stations would transmit an omnidirectional signal at all times, at time intervals as a beacon, or in response to a query. Once other stations are discovered, the directional antenna would be used to communicate.

In the case of continuously transmitted signals, the setting aside of a set of control channels that help stations find each other is a well-known technique in mobile wireless telephony. Many of the practices are applicable to MEP. In the case of an in-band beacon or query/response, APRS could be used as the in-band omnidirectional signaling method as described in the previous section. This eliminates the disadvantage of having extra equipment, as would be the case with requiring a 2m APRS transceiver. While MEP stations would not be transmitting on the 2m APRS channel, they could be placed as objects so that 2m APRS stations would be able to discover them.

Can you suggest another clever way to make station discovery and microwave dish pointing faster and easier? Please join the discussion on our mailing list. Find us on the web at www.delmarnorth.com/microwave

References:

[1] People contributing to this document include Bob Bruninga, Michelle Thompson, Paul Williamson, Timothy Salo, Roger Thompson, and Ken Easton.

[2] A description of display symbols is included in Chapter 20: APRS Symbols of the APRS Protocol Reference. The full Symbol Table listing is in Appendix 2.

[3] www.tapr.org/aprs_working_group.html

[4] Recommended to MEP by Bob Bruninga via e-mail in November 2008.

###

New TAPR Server

By John Ackermann, N8UR

In November, we moved tapr.org to a new server. The move changes the tapr mailing list addresses. "**@tapr.org**" replaces "**@lists.tapr.org**" (the list names themselves remain the same).

Also, the URL for mailing list info is now https://www.tapr.org/mailman/listinfo/<listname> instead of the old lists' tapr.org URL. The archives are now at www.tapr.org/pipermail/<listname> (you'll need to replace "<listname>" with the name of the list).

###

A Soft Processor for Digital Signal Processing

By John B. Stephensen, KD6OZH

Introduction

I've been experimenting with FPGA (field-programmable gate array) based digital signal processing for the past several years. While traditional design methods using Verilog are very efficient for implementing complex algorithms that must be executed in parallel millions of times per second, they are not efficient for implementing much of the control logic. This logic can become very complex but operates at very low rates. Consequently, it can waste a lot of valuable space on the FPGA. It is also very time-consuming to design and debug this logic. Writing assembly language software to execute sequential instructions is much easier than writing Verilog code to do multiple operations in parallel. In the DCP-1, I used a combination of an FPGA and a microprocessor with the microprocessor configuring the FPGA and performing some control functions. However, this has the disadvantage of increasing cost and channeling all of the control lines through a few FPGA pins.

With the availability of FPGAs that can self-initialize from configuration data stored in inexpensive serial flash memory, a better solution is available. Part of the FPGA can be configured as a processor. Since the processor is defined in Verilog and is easily alterable, it is a "soft" processor.

A number of features in modern FPGAs make soft processors easy to implement. Xilinx Spartan-3 Generation FPGAs consist of multiple 4-input look-up tables (LUT4) that can implement logic functions plus dedicated carry logic, multiplexers and registers. Xilinx FPGAs are organized into slices with two LUTs per slice and four slices per configurable logic block (CLB) as shown in figure 1. The CLBs are connected via switches to a routing network. Collections of slices are configured perform complex functions by programming the switches with information stored in on-chip RAM.

Half of the slices in the FPGA can be configured as 16-bit blocks of memory (RAM16) or 16-bit shift registers (SRL16). This distributed RAM is particularly useful as one slice can be configured as a dual-port RAM. Both types of distributed RAM decrease instruction execution time as they eliminate discrete registers and multiplexers. The shift registers are useful when implementing pipelining, where complex operations are broken down into a series of simple operations with storage between them. Different logic paths may have different numbers of operations so shift registers may be used to insert delays to synchronize inputs.

Larger blocks of RAM are also available and can be used to efficiently store large amounts of data or instructions for the soft processor. These can be configured as 512 x 36, 1024 x 18, 2048 x 9, 4096 x 4, 8192 x 2 or 16,384 x 1 bits and have two read/write ports. In addition, there are dedicated 18 x 18 bit multipliers.

There are 8-bit and 32-bit soft processors available for the Spartan-3. The MicroBlaze 32-bit CPU is overkill for a control application as it is designed to use large amounts of external memory. The 8-bit PicoBlaze was designed to be as small as possible and doesn't have an efficient way of accessing memory containing digitized signals. A 16-bit processor is a better solution as it is small but a complete memory address fits into one register and data can therefore be manipulated more quickly.

I didn't find what I wanted in available intellectual property, so I designed a soft CPU for my application and it should also be useful for general signal

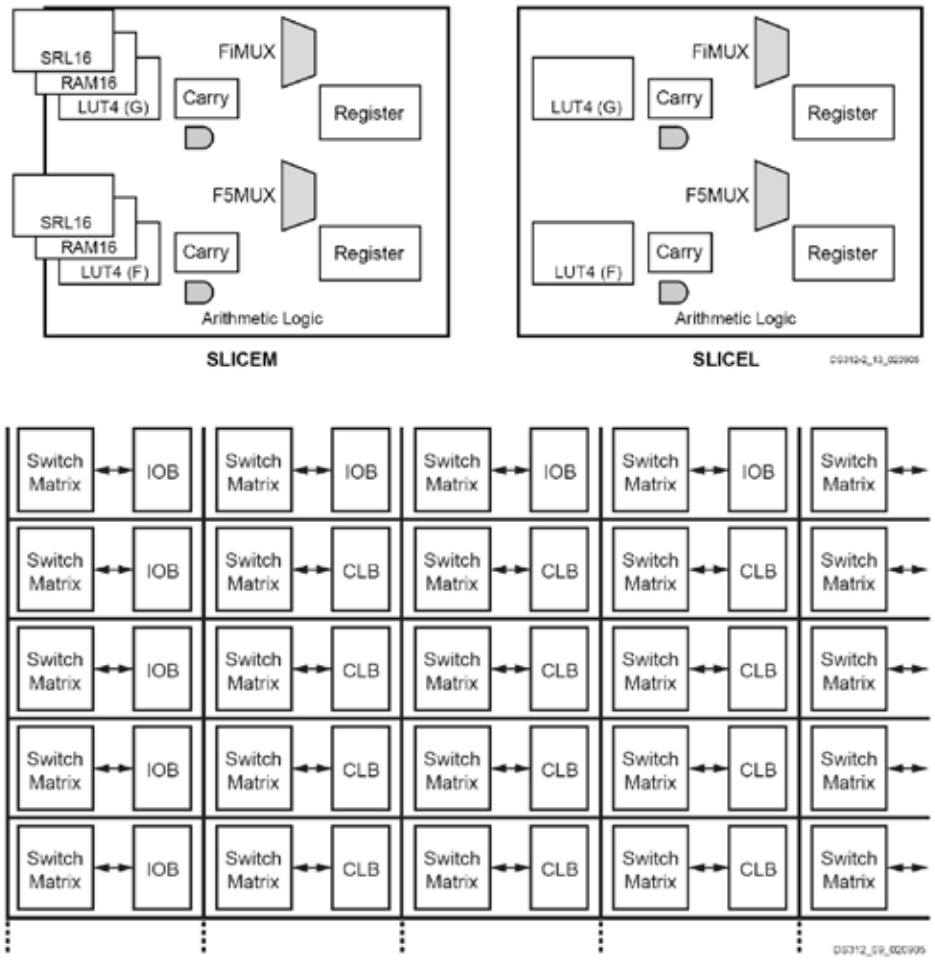


Figure 1 – FPGA Architecture

processing applications. It is written in Verilog and will fit in the smallest Spartan-3 series FPGA. All instructions execute in one cycle at 75 MHz. It can also be customized to add instructions that speed up the application.

Basic Processor Architecture

This soft processor, called CPU16, implements a Harvard architecture using separate instruction and data paths in order to allow simultaneous fetching and execution of instructions. Consequently, the instruction and data address spaces are separate. This is faster than the Von Neuman architecture that has a single memory holding both program and data and uses separate clock cycles to fetch and execute instructions. However, most programs are full of constants that are more convenient to embed into the program. Consequently, the Harvard architecture is modified to allow instructions and data to cross paths in a few cases. Constants may be encoded into instructions in program memory and loaded into registers. In addition, any location in the program memory may be written from the registers. This allows the program to be modified after FPGA configuration.

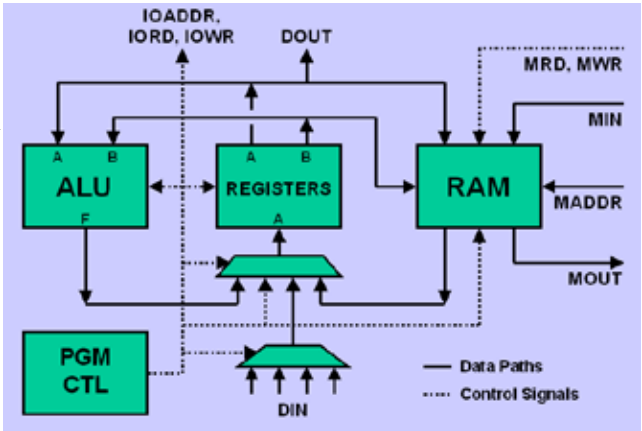


Figure 2 – CPU and Memory

The CPU16 processor has a RISC (reduced instruction set computer) architecture with 16 general-purpose registers as shown in figure 2. All arithmetic and logical operations occur between registers. Data is transferred between registers and memory or I/O ports using dedicated instructions that do not modify memory addresses. This speeds up the processor as both data manipulation and memory access instructions can execute in one clock cycle.

This register file is implemented by sixteen 1-bit wide dual-port distributed memories with 1 read/write port (A) and 1 read-only port (B). The A and B fields of the instruction select particular registers for port A and port B of the register file. A multiplexer on input port A allows writing data from the ALU, immediate data instructions, input ports or random-access memory (RAM) as determined by the operation field of the instruction. Note that the memory is dual-port and direct memory access (DMA) may occur simultaneously with programmed access.

The CPU16 is designed for low-cost FPGAs where the amount of program and data memory is limited. Since the FPGA will be attached directly to an analog to digital converter (ADC) it's advantageous to avoid external memory as it can introduce extraneous noise. Memory comes in 18 kilobit blocks so the largest memory that avoids the use of speed-robbing data multiplexers on the output is sixteen 16k by 1-bit block RAMs. The processor implements 16-kiloword program and data address spaces. The largest Spartan-3E FPGA has 36 kilowords of block RAM so almost all of the memory can be utilized. Since the CPU can modify the program memory, paging can be used to implement larger programs. Subroutines specific to a particular mode of operation can be loaded as needed. They can be stored in the same serial flash memory that is used for FPGA configuration and need only be accessed during mode switching. If necessary, data memory can be increased to 64 kilowords, as 16 address bits are available from the general-purpose registers.

The arithmetic and logic unit (ALU) manipulates data and addresses. All operations occur between registers using two-address instructions where one register is both the source and destination. This is much faster than single-address machines with a dedicated accumulator as it eliminates much of the data movement instructions that would otherwise be required. A 3-address machine (specifying the locations of 2 operands and the result independently) would have been slightly faster but programs would be significantly larger. This trade off works well in a 16-bit architecture as 8 of the 16 bits in arithmetic and logic instructions are consumed specifying operands and 8 are available to specify operations. Fewer than 256 instructions are implemented because some need more space for operands.

A lot of effort was spent on implementation to ensure that the processor operates as fast as possible. The logic was optimized to ensure that propagation delays are roughly equal in the various parts of the processor. For example, the delays incurred in comparing two registers, adding two registers, incrementing the program counter and reading memory are similar. Estimated post-synthesis clock rates have been above 100 MHz and timing after integration with other logic and placement and routing has allowed clock rates above 75 MHz.

The instruction set uses 5 basic formats as shown in figure 3. The type field is 2 bits where 0 specifies call instructions. Type 1 specifies other program control instructions. Type 2 specifies immediate data and ALU instructions that don't write to registers, including comparisons, multi-cycle arithmetic operations and

I/O port output instructions. Type 3 specifies ALU instructions that store data in registers, including I/O input and memory access instructions.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALL	0		Absolute Address													
JMP/LOOP/RET	1		Op.		Condition			Relative Address								
Immediate Data	2		Op.		Data								A			
I/O	2/3		Operation			Port							A			
ALU	2/3		Operation			Modifier			B/C				A			

Figure 3 – Basic Instruction Format

Program-control instructions use two address formats. Call instructions use a 14-bit absolute address and can access the entire 16,384 words of program memory. Jump instructions use a 9-bit relative address and can access the previous 255 or next 256 instruction locations. The jump is taken if the condition specified by a 3-bit field is satisfied. Otherwise instruction execution continues in sequence. The full program address space can be accessed by loop instructions that use a 14-bit absolute addresses stored by the mark instruction. This allows backward jumps to any location.

Immediate data instructions allow constants to be loaded into any register. 16-bit constants with values between –128 and +127 are loaded with one instruction, as the 8-bit value is sign-extended to 16 bits. Values outside this range require two instructions, with the first holding the least significant byte and the next holding the most significant byte of the word.

I/O instructions use a 7-bit direct address to identify up to 128 16-bit wide I/O ports. Memory access instructions use 14-bit indirect addresses stored in register B. Register A contains the data to be written to memory and is the repository for any data read from memory.

ALU instructions may operate on 16-bit words, two 8-bit words or 1-bit. Most use register A as the source for one operand and the destination for the result. The second operand may be register B or a 4-bit constant, C, specifying the bit in register A to be modified.

Figure 4 shows how instructions are allocated within the space provided by the type, operation and modifier fields. About 10% of the instruction space is unused so there is room for future enhancement.

CPU16 Operation Code Matrix

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	CALL															
1																
2																
3																
4	JMP		NOP		JL		JNL		JZ		JNZ		JC		JNC	
5	RET	LOOP	NOP	NOP	RL	LL	RNL	LNL	RZ	LZ	RNZ	LNZ	RC	LC	RNC	LNC
6	MARK								RPT							
7																
8	LDL															
9	LDH															
A	UMSB	UMLN	UMAC	UMUL	MSUB	MULN	MAC	MUL	FDIV	IDIV			BTST		PGM	
B	CMP				CMPC				OUT							
C	IN								NOP	WR	RD	XCHG				
D	SUB	ADD	INC	DEC	SUBC	ADDC	INCC	DECC	CSUB	CADD	INCB	DECB				
E	MOV	AND	OR	XOR	NOT	RST	SET	INV	MLL	MLH	MLLS	MHLS	SWPB	REV	SHR*	SHL*
F	LPL	LPH	LQ	LR												

*Shift instructions include variants ROR, ROL, RRC, RLC, LSR, LSL and ASR.

Figure 4 – Operation Code Matrix

The individual instruction formats are described in detail in the following sections.

Program Flow Control

The program control unit controls program execution. It contains the program counter (PC), RAM for the instructions in the program, the stack

pointer (SP) and RAM for the return address stack. A simplified block diagram is shown in figure 5.

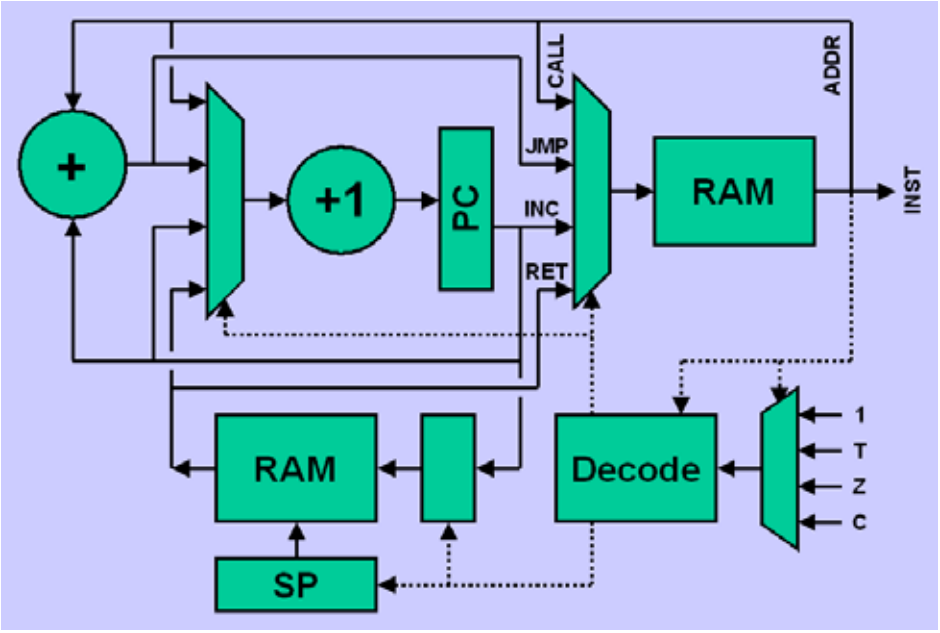


Figure 5 – Program Control Unit

The address of the next instruction to be fetched from RAM is selected by a 4-input multiplexer. In the case of a call instruction, this is a 14-bit address. In the case of a jump instruction where the condition is true, the address is the 9-bit relative address added to the program counter. In the case of a return instruction, it is obtained from the RAM holding the stack. Otherwise, the program counter is used. Note that there are two address multiplexers to account for the fact that the program counter is incremented in parallel with RAM access.

The program control logic determines the maximum clock rate of the CPU. Figure 6 is a table of the delays in the PCU as estimated by ISE 10.1 after logic synthesis. The numbers are calculated prior to placement and routing so they represent the fastest possible speed of this logic. When implemented, the delays are likely to increase by about 25%. However, this information can be used to optimize the CPU early in the design. Data path delays should be equal or faster so that the CPU clock rate does not decrease. Slow processes, such as multiplication, can be broken up into two or more instructions or spread over multiple clock cycles. Common operations, such as addition and memory access should be optimized to occur in one clock cycle.

Cell:in->out	fanout	Gate Delay	Net Delay	Logical Name (Net Name)
RAMB16_S9_S9:CLKA->DOA1	166	0.012	1.340	pfcu/rom/pram1 (pdata<9>)
LUT4_L:I2->LO	1	0.704	0.000	pfcu/ccmux/lut1 (pfcu/ccmux/y1)
MUXF5:I1->O	4	0.321	0.622	pfcu/ccmux/mux (pfcu/fv)
LUT3:I2->O	56	0.704	1.305	pfcu/sel_0_or00001 (pfcu/sel<0>)
LUT2_L:I2->LO	1	0.704	0.000	pfcu/pcimux/mux1/lut0 (pfcu/pcimux/mux1/y0)
MUXF5:I0->O	1	0.321	0.595	pfcu/pcimux/mux1/mux (pfcu/pcin<1>)
LUT1:I0->O	1	0.704	0.000	pfcu/Madd_pc_add0000_cy<1>_rt (pfcu/Madd_pc_add0000_cy<1>_rt)
MUXCY:S->O	1	0.464	0.000	pfcu/Madd_pc_add0000_cy<1> (pfcu/Madd_pc_add0000_cy<1>)
MUXCY:CI->O	1	0.059	0.000	pfcu/Madd_pc_add0000_cy<2> (pfcu/Madd_pc_add0000_cy<2>)
MUXCY:CI->O	1	0.059	0.000	pfcu/Madd_pc_add0000_cy<3> (pfcu/Madd_pc_add0000_cy<3>)
MUXCY:CI->O	1	0.059	0.000	pfcu/Madd_pc_add0000_cy<4> (pfcu/Madd_pc_add0000_cy<4>)
MUXCY:CI->O	1	0.059	0.000	pfcu/Madd_pc_add0000_cy<5> (pfcu/Madd_pc_add0000_cy<5>)
MUXCY:CI->O	1	0.059	0.000	pfcu/Madd_pc_add0000_cy<6> (pfcu/Madd_pc_add0000_cy<6>)
MUXCY:CI->O	1	0.059	0.000	pfcu/Madd_pc_add0000_cy<7> (pfcu/Madd_pc_add0000_cy<7>)
MUXCY:CI->O	1	0.059	0.000	pfcu/Madd_pc_add0000_cy<8> (pfcu/Madd_pc_add0000_cy<8>)
MUXCY:CI->O	1	0.059	0.000	pfcu/Madd_pc_add0000_cy<9> (pfcu/Madd_pc_add0000_cy<9>)
MUXCY:CI->O	1	0.059	0.000	pfcu/Madd_pc_add0000_cy<10> (pfcu/Madd_pc_add0000_cy<10>)
MUXCY:CI->O	1	0.059	0.000	pfcu/Madd_pc_add0000_cy<11> (pfcu/Madd_pc_add0000_cy<11>)
MUXCY:CI->O	0	0.059	0.000	pfcu/Madd_pc_add0000_cy<12> (pfcu/Madd_pc_add0000_cy<12>)
XORCY:CI->O	1	0.804	0.000	pfcu/Madd_pc_add0000_xor<13> (pfcu/pc_add0000<13>)
FDRE:D		0.308		pfcu/pc_13
Total		9.556ns	(5.695ns logic, 3.861ns route)	
			(59.6% logic, 40.4% route)	

Figure 6 – PCU Delays

Note that routing delays make up about 40% of the propagation delay in the PCU. The programmable switches and the capacitance of the lines that interconnect the slices cause this delay. The Spartan-3E is implemented using a 90-nanometer minimum feature size. As semiconductor fabrication processes improve, the slices will have to become more complex so that the amount of time wasted moving bits from one location to another decreases. The newer Virtex-5 FPGA is manufactured with a 65-nm process and uses 6-input LUTs and a 4 LUT slice to minimize interconnect delays.

The program control instructions are listed in figure 7. Calls are unconditional and the call uses absolute addresses. The jump, loop and return instructions may be conditional or unconditional. Jump instructions use addresses that are relative to the program counter.

	15 14	13 12	11 10 9	8 7 6 5 4 3 2 1 0
CALL	00	Absolute Address		
JMP	01	00	Condition	Relative Address
RET	01	01	Condition	0
LOOP	01	01	Condition	1
MARK	01	10	0	
REP	01	10	1 *000*	Count

Figure 7 – Program Control Instructions

Three condition flags may be tested as shown in figure 8. The C and Z bits are the carry and zero flags for comparison operations and for 16-bit arithmetic operations, including add, subtract, increment and decrement. The Z bit is necessary for loop control and the C bit is necessary for software division routines. The L bit is shifter link bit and is also used to test bits in registers.

Condition	Description
000	Always
001	Never (NOP)
010	L set by previous shift or bit test instruction
011	L reset by previous shift or bit test instruction
100	Z set by previous arithmetic or comparison instruction
101	Z reset by previous arithmetic or comparison instruction
110	C set by previous arithmetic or comparison instruction
111	C reset by previous arithmetic or comparison instruction

Figure 8 – Jump Conditions

JMP loads the program counter (PC) with the specified address and continues program execution from that location. JL, JNL, JZ, JNZ, JC and JNC jump if the L bit is true, the L bit is false, the Z bit is set, the Z bit is reset, the C bit is set or the C bit is reset. Otherwise, execution continues with the next instruction in sequence.

CALL pushes the next instruction address onto the return address stack and then jumps to the specified address. The address is stored temporarily in a register while the stack pointer is incremented and then written into the dedicated RAM holding the return address stack. It contains up to 16 return addresses to allow subroutine nesting up to 16 levels. There is no overflow or underflow indication.

The RET (return) instruction restores the address in the address stack to the program counter and continues execution from that location. RL, RNL, RZ, RNZ, RC and RNC return if the L bit is true, the L bit is false, the Z bit is set, the Z bit is reset, the C bit is set or the C bit is reset. The stack pointer is decremented after the address is read.

The MARK instruction pushes the next instruction address onto the return address stack without jumping. It is used with the LOOP instructions for long backward jumps.

The LOOP instruction restores the address in the return address stack to the program counter and continues execution from that location. LL, LNL, LZ, LNZ, LC and LNC loop if the L bit is true, the L bit is false, the Z bit is set, the Z bit is reset, the C bit is set or the C bit is reset. The stack pointer is decremented if the loop is exited.

The REP (repeat) instruction causes the next instruction to execute count + 1 times. It inhibits incrementing the program counter so the same instruction is issued multiple times with no additional overhead.

Program Memory and I/O Write Instructions

Figure 9 shows the format of the program memory write and I/O port write instructions. They do not alter the flags or the general-purpose registers. The logic implementing these instructions generates a write enable strobe when the correct type and operation field values are encountered.

Figure 9 – I/O and Memory Access Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PGM	10		110			*000*			B				A			
OUT	10		111			Port							A			

The PGM instruction writes an instruction in register A to program memory at the address in register B. The upper 2-6 bits of register B are ignored depending on the size of the program memory (1-16 k). The program memory is a true dual port memory with two read/write ports. The current instructions are fetched from one port at one address while new instructions are loaded through the other port at a different address.

Output instructions use direct addressing to access up to 128 ports up to 16 bits wide. OUT copies the contents of register A to the selected port by placing the port address on IOADDR, the data on DOUT and asserting IOWR.

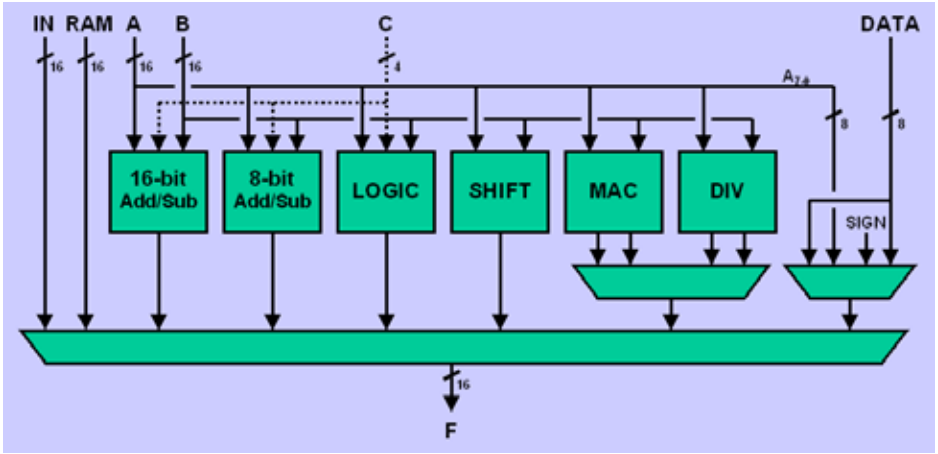


Figure 10 – Arithmetic Logic Unit (ALU)

Arithmetic and Logic Unit

Arithmetic and logical operations are implemented in the ALU shown in block diagram form in figure 10. A and B are data inputs from 2 selected

registers, C selects a particular bit in a word and F is the output. There are actually 6 functional units with parallel inputs and one output is selected by a multiplexer controlled by the operation field in the instruction. The same multiplexer also appears in figure 2.

Data from the register file flows through the ALU to be written back into the register file in the same clock cycle so the ALU must be as fast as possible. A common 8-input multiplexer composed of four LUTs, two F5MUX and one F6MUX is used. As shown in figure 11, there are dedicated connections between the sections of the multiplexer that completely avoid the routing network delays associated with interconnecting LUTs.

Note that the multiplier-accumulator (MAC) and divider (DIV) outputs share one port. They are combined in a subsidiary 4-input multiplexer that selects the upper product, lower product, quotient or remainder based on the value of the modifier field in the instruction. The additional delay can be tolerated because the multiply-accumulate and divide operations are started by one instruction and the results are stored within the functional unit. The results are then read by another instruction.

Figure 12 shows the format of the immediate data, input and memory access instructions. Note that the operation field normally selects the multiplexer port. However, immediate data instructions require 8-bits to specify the value and 4-bits to specify the destination register so port 7 of the ALU output multiplexer is selected whenever an instruction begins with 011.

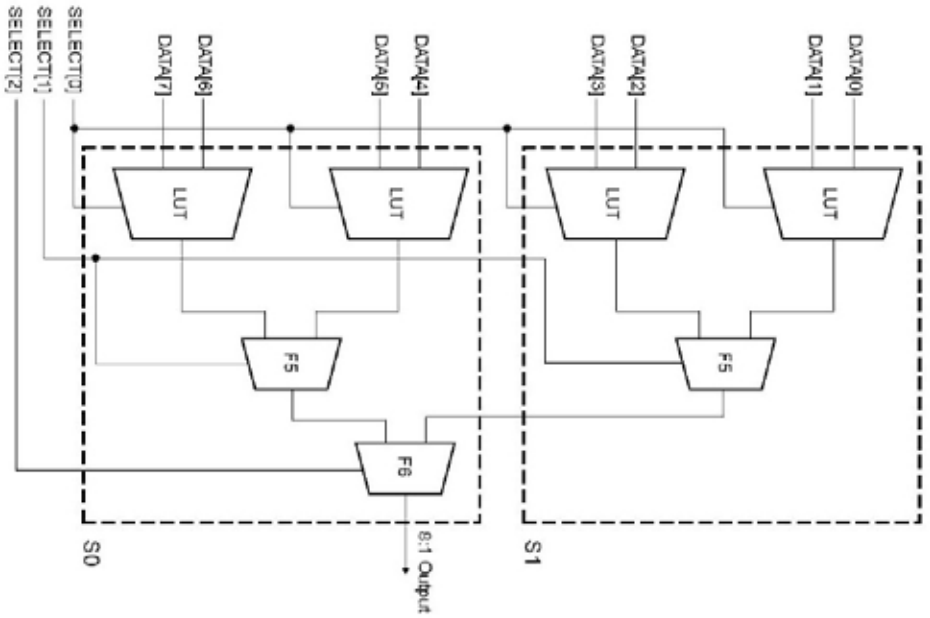


Figure 11 – 8-Input Multiplexer Logic

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IN	11		000			Port							A			
WR	11		001			*0*	0	1	B				A			
RD	11		001			*0*	1	0	B				A			
XCHG	11		001			*0*	1	1	B				A			
LDL	10		00			Data							A			
LDH	10		01			Data							A			

Figure 12 – I/O and Memory Access Instructions

Input instructions use direct addressing to access up to 128 ports up to 16 bits wide. IN copies the contents of the selected port from the DIN bus to register A and asserts IORD.

Memory access instructions use register indirect addressing of up to 65,536 words of memory. Both reads and writes complete in one clock cycle. Write (WR) copies the contents of register A to the memory location at the address in register B. Read (RD) copies the contents of the memory location at the address in register B to register A. Exchange (XCHG) copies the contents of memory location B to register A while writing the content of register A to memory location B.

LDL and LDH use a subsidiary 2-input multiplexer to convert 8-bit to 16-bit data. The LDL instruction replaces the lower 8 bits of register A with immediate data and the sign is extended to the upper 8 bits. The LDH instruction replaces the upper 8 bits of register A with immediate data and does not affect the lower 8 bits.

Arithmetic Operations

The add/subtract functional units each utilize two full-adders per bit to implement the add, subtract, increment and decrement operations. One unit adds 16-bit words and the other adds two 8-bit words. This partitioning minimizes propagation delays for the carry bits.

A 3-input LUT implements a half-adder (exclusive-or) and carry propagation is performed by a dedicated multiplexer and exclusive-or gate that follows the LUT in each logic unit. This forms a full-adder as shown in figure 15. The third input (S) inverts the B input for subtraction.

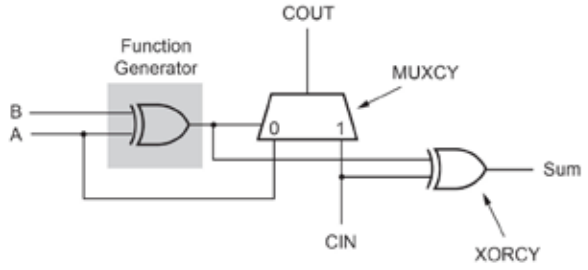


Figure 13 – Full Adder

A 16-bit adder is formed from 16 full adders as shown in figure 14. The carry outputs are wired to the carry inputs in the adder for the next most significant bit forming a ripple-carry chain. This chain is implemented with dedicated connections in Spartan-3 generation FPGAs so each bit adds only 59 picoseconds of propagation delay.

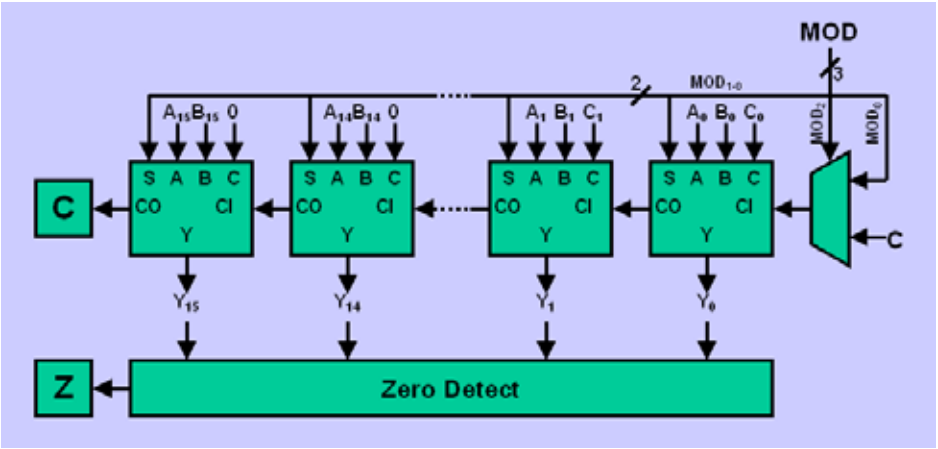


Figure 14 – 16-bit Arithmetic Unit

There is also zero-detection logic for the arithmetic unit. To minimize propagation delay, it is created from 16 MUXCY carry multiplexers where each multiplexer propagates the carry bit when the data input is 0 or propagates a 0 if the data input is 1. The initial input is 1 so when all data bits are 0 the zero flag (Z) is set. If any data input is 1, Z is reset. This is useful for loop control where a decrement instruction is followed by a JNZ instruction.

After synthesizing the arithmetic unit, it's useful to look at the delays in that logic. The total delay is about equal to the PCU and the internal delays are distributed as follows:

Program Memory	13%
Register File	29%
Arithmetic Logic	37%
Output Multiplexer	21%

Figure 15 – Arithmetic Unit Delays

The delay caused by fetching instructions is minimal. The output multiplexer contributes more delay even though it has been optimized – using LUTs would have added 1.9 ns and slowed the CPU down by 20%. The circuitry that does the work, the register file and arithmetic logic, make up 2/3 of the delay so the design is fairly optimal.

Two arithmetic units are needed to provide both 8 and 16-bit math without interrupting the dedicated carry propagation logic, which would reduce the speed of the processor. The two 8-bit units operate independently on bits 15-8 and bits 7-0. They have no carry registers and no zero detection. Operation on two 8-bit words simultaneously is useful for signal processing where one byte is the phase and the other byte is the magnitude in logarithmic form. Addition then maps into multiplication and subtraction maps into division.

The 8 and 16-bit arithmetic instructions are encoded as shown in figure 16. The carry, overflow and zero flags are updated for 16-bit arithmetic operations but not for 8-bit operations.

	15 14	13 12 11	10 9 8	7 6 5 4	3 2 1 0
CMP	10	1100	000	B	A
CMPC	10	1100	100	B	A
SUB	11	010	000	B	A
ADD	11	010	001	B	A
INC	11	010	010	*0000*	A
DEC	11	010	011	*0000*	A
SUBC	11	010	100	A	A
ADDC	11	010	101	A	A
INCC	11	010	110	*0000*	A
DECC	11	010	111	*0000*	A
CSUB	11	011	000	B	A
CADD	11	011	001	B	A
INCB	11	011	010	*0000*	A
DECB	11	011	011	*0000*	A

Figure 16 – Single-Cycle Arithmetic Instructions

CMP subtracts register B from register A and sets the zero and carry flags. No register is modified. CMPC is the same, but propagates the carry bit.

ADD adds register B to register A and leaves the result in register A (A=A+B). SUB subtracts register B from register A and leaves the result in register A (A=A-B). ADDC and SUBC are the same, but propagate the carry bit.

INC adds 1 to register A (A=A+C) and DEC subtracts 1 from register A

(A=A-C). INCC and DECC are the same, but propagate the carry bit. INCC and DECC usually use a constant with a value of 0.

CADD and CSUB independently add or subtract the corresponding two bytes in each word (AL = AL ± BL and AH = AH ± BH).

INCB and DECB increment or decrement the lower 8-bits without affecting the upper 8 bits. The carry, overflow and zero flags are not used.

There are 4 unused bits in the increment and decrement instructions. In the future, these could be used with 6-input LUTs in Virtex-5 FPGAs to allow incrementing and decrementing by 1-16.

Logical Operations

The logic unit operates on two 16-bit inputs (A and B) or on one bit from input A as specified by input C and uses three 4-input LUTs per bit. No flags are affected. Two of the LUTs are located in one slice and joined by a dedicated multiplexer (F5MUX) as shown in figure 17. The third LUT generates a mask for bit selection.

One LUT receives a bit from register A and a bit from register B plus the two least significant bits of the modifier field. It implements register to register and, or, exclusive-or and move operations. The other LUT receives one bit from register A, one bit from a 1 of 16 decoder driven by the C input and two modifier bits. It implements register-wide bit inversion (not) and single-bit set, reset and invert. The operation performed by each LUT is determined by the lower two bits in the modifier field and the most significant bit selects the output used. The encoding is shown in figure 18.

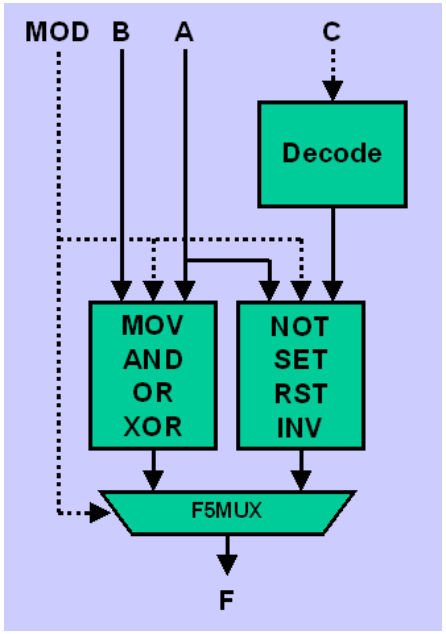


Figure 17 – Logic Functions

Four instructions perform logical operations between registers. MOV copies register B to register A. OR sets bits in register A if either of the corresponding bits in register A or register B are 1. XOR sets bits if only one of the two corresponding bits is 1. AND sets bits in register A if the corresponding bits in register A and register B are both 1.

The bit manipulation instructions modify register A. RST and SET change the value of the bit selected by C to 0 and 1, respectively. INV complements the value of the selected bit. NOT complements the value of all bits.

	15 14	13 12 11	10 9 8	7 6 5 4	3 2 1 0
MOV	11	100	000	B	A
AND	11	100	001	B	A
OR	11	100	010	B	A
XOR	11	100	011	B	A
NOT	11	100	100	*0000*	A
RST	11	100	101	C	A
SET	11	100	110	C	A
INV	11	100	111	C	A

Figure 18 – Logical Instructions

Shift, Rotate and Sign Extension Operations

The shift unit is an 8-input multiplexer (DMUX) with the input selected by the modifier field as shown in figure 19. Different operations are implemented by changing the order of the bits on the input ports. This allows shifting by one bit in either direction, rotating by 8 bits, bit order reversal and byte to word conversion. It also implements sign extension.

There is also a link register and two additional multiplexers, driven by bits 4-7 of the instruction (C). A 16-input multiplexer (BMUX) loads the link register with any bit selected from the A input. Any bit may be selected during a bit test instruction. During shifts it is restricted to the least or most significant bit of the A input. A 4-input multiplexer (SMUX) determines the value of the end bits in single-bit shifts. The end bits can be set to zero or one, to the value of the least or most significant bit or to the value of the link bit from the previous shift. The link bit allows shifts to be extended to 32 or more bits in order to allow implementation of software division and other operations.

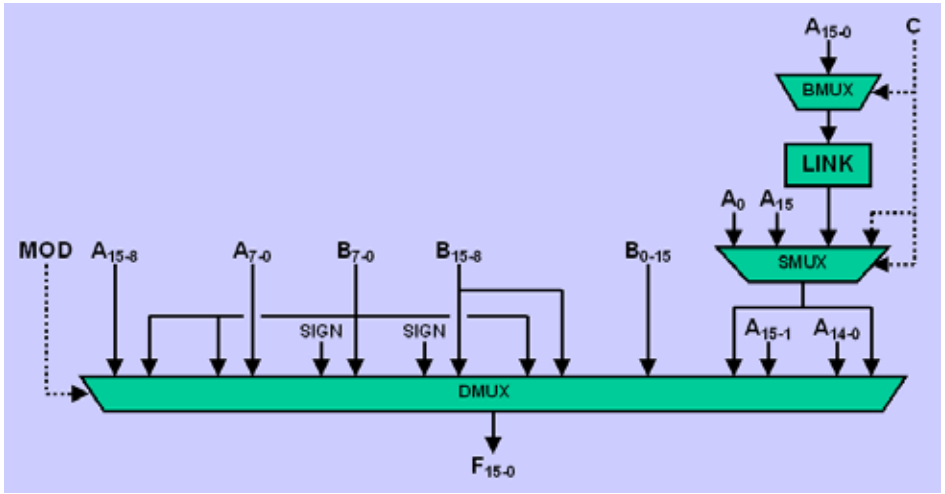


Figure 19 – Shift Unit

Figure 20 shows how these instructions are encoded. Note that additional operations are available by using different values for bits 7-4 in the instruction. The most useful instructions are documented here.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BTST	10		101			100			C				A			
MLL	11		101			000			B				A			
MLH	11		101			001			B				A			
MLLS	11		101			010			B				A			
MHLS	11		101			011			B				A			
SWAP	11		101			100			B				A			
REV	11		101			101			B				A			
ROR	11		101			110			0000				A			
ROL	11		101			111			0101				A			
RRC	11		101			110			1000				A			
RLC	11		101			111			1001				A			
LSR	11		101			110			1100				A			
LSL	11		101			111			1101				A			
ASR	11		101			110			0100				A			

Figure 20 – Shift and Rotate Instructions

BTST copies bit C in register A to the link register.

MLL copies the least significant byte of register B to the least significant byte of register A without affecting other bits in register A. MLH copies the least significant byte of register B to the most significant byte of register A without affecting other bits in register A.

MLLS copies the least significant byte of register B to the least significant byte of register A and then extends the sign to fill 16 bits. MHLS copies the most significant byte of register B to the least significant byte of register A and then extends the sign to fill 16 bits.

SWAP copies the upper byte of register B to the lower byte of register A and the lower byte of register B to the upper byte of register A. REV reverses the order of the bits while copying from register B to register A. Bits 15 and 0 are

swapped, bits 14 and 1 are swapped, etc.

ROL and ROR rotate the contents of register A left or right by one bit with bit 15 replacing bit 0 or bit 0 replacing bit 15, respectively. RLC and RRC rotate the contents of register A left or right by one bit with the link bit replacing bit 0 or bit 15, respectively. The link register contains the previous value of bit 15 after ROL or RLC and bit 0 after ROR or RRC.

LSL shifts the contents of register A left by one bit and clears bit 0 while setting the link bit to the previous value of bit 15. LSR shifts the contents of register A right by one bit and clears bit 15 while setting the link bit to the previous value of bit 0.

ASR shifts the contents of register A right by one bit without affecting bit 15. The link register contains the previous value of bit 0.

Optional Multiply-Accumulate Instructions

The MAC unit, shown in figure 21, is a 17-bit signed multiplier followed by a 32-bit accumulator where products may be added to or subtracted from the accumulating sum. The two most significant bits of the A and B inputs may be set to zero or to the value of bit 15 of the inputs to enable unsigned and signed multiplication. The output of the multiplier is latched in the product register and may be read

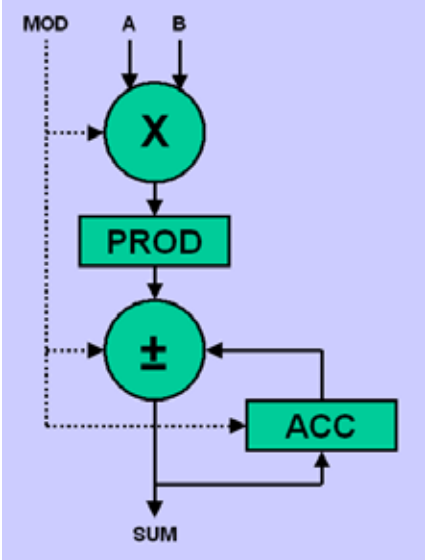


Figure 21 – Multiply-Accumulate Unit

on the next clock cycle by resetting the accumulator register. Individual bits in the modifier field select signed or unsigned inputs, select addition or subtraction and whether to reset the accumulator. If the accumulator is not reset, it accumulates the sum of products for a convolution operation.

There are 10 multiply-accumulate instructions, as listed in figure 22. Type 2 instructions are used to start multiplication with options selected by the modifier field. The instructions complete in 1 cycle but results require 2 instruction cycles to become available. Type 3 instructions are used to copy the results into general-purpose registers.

MUL performs a 16-bit by 16-bit signed multiply and leaves the result in a 32-bit accumulator. UMUL performs an unsigned multiply. MULN and UMLN perform signed and

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UMSB	10		100			000			B				A			
UMLN	10		100			001			B				A			
UMAC	10		100			010			B				A			
UMUL	10		100			011			B				A			
MSUB	10		100			100			B				A			
MULN	10		100			101			B				A			
MAC	10		100			110			B				A			
MUL	10		100			111			B				A			
LPL	11		110			000			*0000*				A			
LPH	11		110			001			*0000*				A			

Figure 22 – Multiply Instructions

unsigned multiplies and negate the result.

MAC and UMAC perform signed and unsigned multiplies and add the result to the existing accumulator contents. MSUB and UMSB perform signed and unsigned multiplication and subtract the result from the accumulator. Operands may be introduced at a rate of one pair per instruction cycle and the results will be available on the next instruction cycle after the final operands are loaded.

LPH provides the upper 16-bits of the accumulator and LPL provides the lower 16 bits of the accumulator.

The MAC unit is also useful for complex multiplies. CI = AIBI - AJBJ can be implemented with a MUL instruction followed by MSUB, LPH and LPL instructions. CJ = AIBJ + AJBI can be implemented with a MUL instruction followed by MAC, LPH and LPL instructions. Only 8 instructions are needed to generate a 64-bit product from two 32-bit arguments.

Optional Division Instructions

The DIV unit, shown in figure 23, is the most complicated with multiple shift registers and a subtracter. It implements division of two unsigned integers by repeated subtraction. The divisor is loaded into a 16-bit register and the dividend into the least significant bits of the 32-bit remainder register. The upper half of the remainder register is cleared. If the output of the subtracter is positive, its output is written to the most significant bits of the remainder register and the register is shifted left. Otherwise, only the shift occurs. This repeats for all 16 bits in the dividend. The quotient register is also shifted left and it accumulates ones for successful subtractions and zeros for unsuccessful subtractions.

The remainder can also be divided by the divisor to form a 16-bit fractional result for a 32-bit long quotient. Signed division may be implemented in software by converting the operands to positive numbers and then adjusting the results if the operands had different signs.

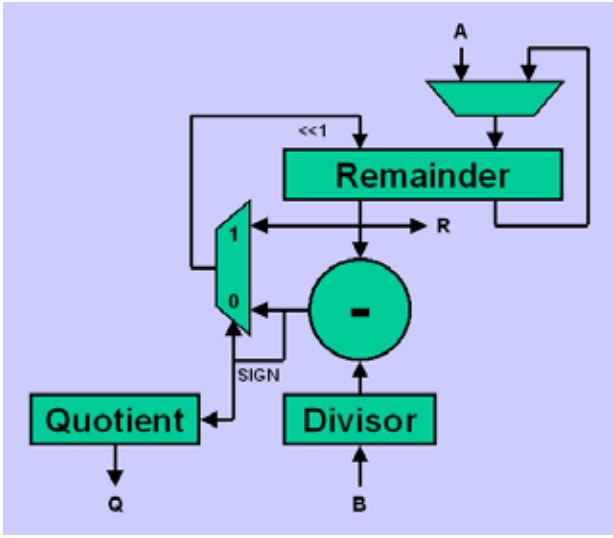


Figure 23 – Division Unit

There are 4 division-related instructions that are formatted as shown in figure 24. The division operations consume many clock cycles, but may occur in parallel with other operations. The programmer must insure that a new division operation is not issued until the previous one has been completed.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDIV	10		101			000			B				A			
FDIV	10		101			001			*0000*				*0000*			
LQ	11		110			010			*0000*				A			
LR	11		110			011			*0000*				A			

Figure 24 – Division Instructions

IDIV performs a 16-bit by 16-bit unsigned integer division leaving a 16-bit quotient and 16-bit remainder. A is divided by B and the result is left in the quotient and remainder registers after 18 clock cycles.

FDIV continues the calculation in order to generate the fractional portion of the quotient in Q after 17 more clock cycles.

The LQ and LR instructions read the quotient and remainder registers. The upper bit of R may be monitored to determine when the calculation is complete. It is 1 during computation.

Assembler

It's inconvenient to code machine-language instructions by hand so a mnemonic assembly language was defined and an assembler implemented in Delphi. The assembler software operates on a PC and reads a text file containing assembly language statements. It then generates a Verilog file for incorporation into the FPGA. The language syntax is fairly traditional with a label field followed by a mnemonic operation code and one or more named registers and an optional numeric value. Figure 25 shows a sample of the compiler output listing.

The instructions described previously are included along with a number of pseudo-instructions described here. Several of these pseudo-instructions are

```
Line   Addr   Inst   Source
   1                 ;
   2                 ; Test program for 16-bit soft CPU
   3                 ;
   4   0000: 6340          LDL     R0,$34   ;initialize LSB R0
   5   0001: 7120          LDH     R0,$12   ;initialize MSB R0
   6   0002: 6011          LDL     R1,1     ;sign extend 8-bit value
   7   0003: 6022          LOAD    R2,2     ;load 16-bit values
       0004: 7002
   8   0005: 6FFD          LOAD    R13,255
       0006: 70D
   9   0007: 6FFE          LOAD    R14,+255
       0008: 70E
  10   0009: 601F          LOAD    R15,-255
       000A: 7FFF
  11   000B: B003          MOV     R3,R0    ;copy to R3
  12                 MAIN:
  13   000C: BB03          OUT     R3,$30   ;output R3
  14   000D: C303          IN      R3,$30   ;modify R3
  15   000E: BB03          OUT     R3,$30   ;output R3
  16   000F: D200          INC     R0       ;increment counter
  17   0010: D300          DEC     R0       ;decrement counter
  18   0011: B300          CLR     R0       ;clear R0 and R3
  19   0012: B333          CLR     R3
  20   0013: D010          SUB     R0,R1    ;decrement by 201h
  21   0014: D423          SUBC    R3,R2
  22   0015: D110          ADD     R0,R1    ;increment by 201h
  23   0016: D523          ADDC    R3,R2
  24   0017: D810          CSUB    R0,R1    ;decrement by 1
  25   0018: D910          CADD    R0,R1    ;increment by 1
  26   0019: DB00          DECB    R0       ;decrement by 1
  27   001A: DA00          INCB    R0       ;increment by 1
  28   001B: B300          CLR     R0       ;clear R0
  29   001C: B210          OR      R0,R1    ;test logical operations
  30   001D: B310          XOR     R0,R1    ;test logical operations
```

Figure 25 – Assembly Language Program Listing

simplified versions of instructions defined previously in this document. They are included to simplify programming.

LOAD generates a pair of LDL and LDH instructions to load a 16-bit value into a register.

CLR zeroes all bits in a register. It actually exclusive-OR's the selected register with itself.

NOP does nothing by OR-ing register 0 with itself. It is generally used in timing loops.

Other pseudo-instructions modify the state of the assembler as follows:

EQU assigns a label to a numeric constant and REG assigns a label to a register. They may be used to improve the readability of assembly language programs.

ORG sets the program address for the next instruction. It defaults to 0 at the beginning of assembly.

DW assigns a label to the current data address and increments the data address by a specified number of words. It is used to reserve memory space.

Conclusion

The speed of the processor is about twice that of the PicoBlaze and processes twice as many bits per instruction so it is a big improvement over the free IP (intellectual property). Complex logic blocks provided in the Xilinx IP library, such as a CORDIC module for calculating arctangents must be clocked at a similar rate so the CPU16 is a good match for my application. Since the processor requires only 297 slices, it fits in the smallest Spartan-3 Generation FPGA (XC3S50A) using only 42% of the slices. In the medium-sized FPGAs that would be used for signal processing, it occupies 2½ to 6% (XC3S1400A and XC3S500E) with the rest available for dedicated DSP logic, such as filtering, FFT and CORDIC modules.

The number of slices used in various portions of the CPU is interesting. Only 1/4 are used for program control. 2/3 are used for processing data with 1/3 of that for multiplication and division. The register file and miscellaneous logic occupy the last 7%.

PCU	24%
ALU	42%
MUL/DIV	27%
Other	7%

If more processing power is required, it's reasonable to implement multiple processors per FPGA. This is a better approach than just increasing memory size as the amount of processing power is increased. It also allows processors to be dedicated to specific functions. For example, an XC3SD3400A FPGA could support 30 processors – each with 4 kB of data memory and 4 kB of program memory.

This soft processor simplifies the implementation of high-speed digital modes for Amateur Radio and reduces the cost. It may also be useful for other applications. Verilog source code for the CPU16 and Delphi source code for the cross-assembler are available on the TAPR Web site at ftp://www.tapr.org/software_lib/Soft_Processor_for_DSP/. Note that there are licensing restrictions, but they allow personal experimentation.

The assembler is written to generate code for a 2 k x 16-bit program memory and will have to be modified for other sizes. The processor is configured to use 2048 words of program memory and 2048 words of data memory formed from four 1024 x 16 bit block RAMs. The source code can be altered to accommodate different amounts of memory.

###

Asterisk Provides an Interoperability Platform

By Steve Lampereur, KB9MWR

What is Asterisk?

Asterisk is an Open Source PBX & Telephony Platform. It's often labeled as the future of telephony. It has been around since about 1999 and the platform is open source - Linux operating system based. It can support a variety of signaling protocols, but by far SIP, the session initiation protocol, for VOIP and other text and multimedia sessions, has become a standard.

For those scratching their head a bit... PBX stands for private branch exchange. It is a machine that handles many businesses telephones calls for you. Its main functions are to transfer calls to different individual phones; play music when somebody is put on hold; to play automated voice responses when a call is received; to provide an options menu for the caller etc.

Asterisk allows one to build a phone system. It adds features, functionality and reduces deployment costs in ways which; at first are a little difficult to understand.

How does this relate to Amateur Radio?

Very simple, the future of two way radio is digital. As of writing, TV broadcasters are required to be full digital and shut down their analog transmitters in Feb. 2009. The only spectrum broadcasters are required to vacate are channels 64 thru 69 that will become the new "700 MHz band" that is being auctioned off by the FCC. The vacated areas of this spectrum will be utilized for: Public Wireless deployment (Cellular/PCS); a wide-band private data network that will be shared between public safety and paying customers; and new spectrum for public safety that will butt right up to the re-located NPSPAC National Public Safety Planning Advisory Committee band being moved to 806-809/851-853 by Sprint/Nextel.

Public safety also has guidelines to migrate to APCO-25 digital. The future of two way radio is digital, and we must also advance in this direction. The digital premise is that it generally allows more use in a more efficient/flexible use of band space.

Most present day government communication centers that use analog systems happen to have a VOIP based dispatch console. This analog to VOIP patching is something that we are presently also embracing in ham radio with IRLP, EchoLink, Yaesu WIRES II, and the like.

You're seeing the digital migration in the commercial world as I pointed out; and the only analog part left of traditional telephone is the "last mile" drop to your home. Time Warner and now AT&T are providing digital phone service to close that up too.

As of writing there aren't any 100% directed approaches to tie this to the hobby that I can point you to. There are a number of open ended ideas from a variety of different people. What I'm saying is there is no one entity steering the ship, so to speak. These ideas are still in development. This makes it precisely the time to jump aboard and get our hands in it and see what we can do with it. So in light of that I suggest a Google search for more info...

Digital Voice Interoperability Software Strides

Perhaps some of you remember the ARRL "It Seems to Us: Interoperability" statement from October last year regarding emerging digital voice.

Well the good news is here are some of the strides

I've run across:

-OpenP25 Project <http://openp25.org/index.php/category/project-status/>

"We've determined that the open source Asterisk PBX appears to be a good framework on which to build a P25 ISSI (Inter RF Subsystem Interface) switch. Asterisk has a mature SIP stack and already has the ability to transparently pass RTP frames between SIP channels. The National Institute of Standards and Technology (NIST) has made an open source program for ISSI testing freely available to the P25 community. A full-featured open source P25 ISSI switch is clearly achievable."

The Project 25 Inter RF Subsystem Interface (P25 ISSI) is a non-proprietary interface that enables RF subsystems (RFSSs) built by different manufacturers to be connected together into wide area networks. Apparently consideration is being given to enhancing Asterisk app_rpt to support such a low-level P25 radio interface.

The open source project25 interface is a good idea. Unfortunately the problem they're facing is that most of the manufacturers don't bother following the ISSI spec, nor does the ISSI spec call out hardware interface details. So basically the "plugs" on the back of say, a Quantar... don't match the "plugs" on the back of a Mastr III.

Open D-Star Project

http://opendstar.org/

The OpenDSTAR group has released several software tools which build on existing commercially available repeaters and Internet gateways to extend functionality.

And this interoperability work in progress attempt has been around longer than the P25 one. They too are working a SIP stack into their DPLUS gateway add-on daemon.

A project is lead by Scott, KI4LKF expand the Asterisk capabilities with his RtpDir bridge software (Real Time Protocol Director) software package for VoIP/RF Gateways. <http://tech.groups.yahoo.com/group/rtpDir/>

It can bridge transmit and receive from/to Asterisk/app_rpt, IRLP, Echolink, Speak-Freely and he is working on D-Star. There is support for all link interfaces, sound mode or ASCII mode, VA3TO, WB2REM, ULI, Rigblasters, MFJ, SignalLink, etc.

RadioGrid RG-4 Radio Gateway

www.radiogrid.com/

The guys from NHRC who make repeater controllers have come up with a commercially targeted (and priced) controller called the RadioGrid RG-4 Radio Gateway. It's built around Asterisk open source PBX technology. It's specifically for VOIP linking designed with interoperability applications in mind. It uses a 500 MHz Blackfin processor, 64 MB RAM, 4 MB Flash memory, 1 GB SD Card

A key component in all these digital voice systems is the vocoder. P25 uses the IMBE vocoder from Digital Voice System Inc. (DVSI). It costs $150K to get the rights to play with that mode plus $5 a seat. There is no off the shelf IC to do it. Fortunately D-Star uses AMBE from DVSI they do offer a single chip solution for $20 single quantity and are happy to sell to hams.

The DV Dongle

http://dvdongle.com/

The DV dongle is an important development. It was started by Moe Wheatley, AE4JY and Robin

Cutshaw, AA4RC. It contains the hardware ability to process AMBE full duplex. Presently software applications exist to use this to communicate from a computer to a D-Star gateway. Further development is in the works so that it can be interfaced to a radio's packet radio port that has the necessary discriminator connections. This may be a huge milestone. The ability to retrofit an existing repeater could be possible with this. Not only that, but you may be able to retrofit it in such a way that it can be usable in analog and for D-Star.

It would be great to see an a D-Star radio that supports something like this on the market possibly before the P25 interface idea ever makes it to the market. We should encourage D-Star manufactures to come up with a similar style non-proprietary interface for D-Star. As unfortunately at this time there are no known interfaces to the Icom D-STAR radios that allow access to the on-air data stream.

It would be nice to see some agreement and standards in place amongst the various guys working on software solutions and the manufactures, to help make things more stream-lined. Either way people are working on interoperability solutions, toward convergence with open standard codecs like G.711 using real word protocols like SIP and RTP protocols so one can connect the radios together into wide area networks.

Then one would be able to have a D-Star radio in their shack also interconnected to various Asterisk powered applications in their house or abroad. Where if you weren't around to take a directed D-Star call, it could be configured as a DID to a system and use a ring group / follow me list to let the radio caller ring a home phone or leave a voice mail. Calls could be route to/through the Public switched telephone network and so on. i.e., a very powerful ability for EMCOMM and personal use.

###

Write Early and Write Often

Packet Status Register (*PSR*) is looking for a few good writers, particularly ham radio operators working on the digital side of our hobby, who would like to publicize their activities here.

You don't have to be Shakespeare to contribute to *PSR* and you don't have to use Microsoft Word to compose your thoughts. The *PSR* editorial staff can handle just about any text and graphic format, so don't be afraid to submit whatever you have to wa1lou@tapr.org .

The deadline for the next issue of *PSR* is March 31, so write early and write often.

###

Ham Radio Text Messaging/Contact Initiative www.aprs.org/aprs-messaging.html

By Bob Bruninga, WB4APR

This article is a call to action for TAPR Digital, Software and Network Engineers. We are not just Hams with single focus interests, we are communicators with experience, resources, intuition and initiative to help establish communications anywhere at any time. Although we have our own frequencies and radios, our diversity of frequencies is not just our best asset, it is also our worst curse. With 1000 frequencies on HF and 2000 channels on VHF/UHF below 500 MHz, how can we find each other when needed? And its not just the dimension of frequency, its space and time as well.

Ham radio needs a Universal Text-messaging system to tie all of its disparate messaging systems together so that text-messages can easily reach any ham radio operator by call sign alone, anywhere at any time on any device!

After 9/11 and Katrina it was clear that Amateur Radio needs immediate responsive communications to simply locate and establish initial communications. Although APRS provides an excellent tool for mutual exchange of location information what is needed for emergent contact anywhere, anytime is a Local and Global Text-Messaging -by call sign- capability that makes it possible to connect people independent of frequency and time as well. Fortunately, this instant contact across the dimensions of space, time and frequency has been in APRS since day one, and it is so powerful because it is all done on a single local/national APRS calling frequency; anytime, anywhere. We just need to link in all other ham radio texting systems.

A previous article[1] summarized all of the capabilities of specialized mobile and handheld APRS radios as shown in figure 1, but this article shows how mobile and handheld text messaging at least for the purpose of initial local/global contact can extend not only to all ham radios, but to all manner of personal electronic systems, cell phones, blackberry's and more.!

Surely, every Ham radio operator can find at least one of the following techniques, devices or systems to establish communications with other hams anywhere, anytime.

Organic APRS Messaging

As discussed in the previous article, APRS client software and APRS built-in radios are designed specifically to send and receive messages on their front panel as shown in figure 2... In this case, it is sending an e-mail. Just press the MSG button, select INPUT on the MSG Menu, enter the call sign of any APRS ham radio operator anywhere in the world and your message will be delivered in real time via the local/global APRS system.

Figure 2. This shows the message menu and a typical APRS message. In this case, it is an e-mail to A3XYZ@AMSAT.ORG

Normal APRS messaging is just like text messaging in that it is sent and received in real time. Fortunately, APRS also provides many links between live messaging and e-mail as well. Simply address the message to the word EMAIL instead of a call sign, and make the first text of the message be an e-mail address followed by your message. In the example in figure 3, above, an EMAIL to W3XYZ@AMSAT.ORG saysOK in OceanCity with HT & whip! If your e-mail is digipeated, then you will see MY MESSAGE flashed on the screen indicating success.

Paging Radio Messaging

But there's more than that! In addition to the APRS radios, the FT-51R family and TH-78A radios (from the 1990's) as shown in figure 3 have a built-in TEXT messaging and paging function that can also be used for messaging. Messages are entered from the Keypad and displayed on the radio front panel. The signaling method of these radios is being integrated into the APRStt (TouchTone) system as detailed in the following paragraphs.

Messaging on All Other DTMF Ham Radios

Since 2001, APRS has defined a communication method using only DTMF that can be used from all other radios called APRStt (touchtone)[2]. This DTMF Messaging extends basic information exchange to all radio amateurs, not just those that are APRS equipped. Even the old curmudgeon that shows up with his venerable IC-2AT as shown in figure 4, for example, should be able to participate.

Text-Messaging with Any Radio

The HAMHUD device is an add-on to the speaker mic of any radio. It gives a plug-n-play solution that brings full APRS functionality including text-messaging to any radio. In addition, the new Kenwood RC-D710 APRS display head can be purchased alone and plugged into the external audio interfaces of any radio. All of the APRS hardware and functionality is in the display head itself! Here it is shown plugged into an inexpensive $88 Alinco HT to provide front-panel APRS and text-messaging capabilities to the operator.

Messaging on Palm Pocket and Laptop Devices

OLPC APRS INTERFACE: Many Amateur Radio operators carry a variety of personal computer systems that also provide personal communications whenever the device is in range of a WiFi network. These stations may also maintain Amateur Radio connectivity and text messaging via a variety of application programs. Shown here in Figure 5 is the application called APRSXO which gives full APRS message capability to the One-Laptop-per-child XO laptop. This interface not only provides the APRS text-messaging capability but also provides an approximate geographic position into the APRS system as well. This activity was written by Jack Zielke, KG4GJY. See the web page [3] for installation and operation.

APRS E-mail Messaging to Anywhere

In addition to conventional APRS text messaging, any APRS station can use this capability to send brief e-mail messages via APRS to any cell phone or Internet user as well. The technique is extremely simple. Just address your APRS message to one of the e-mail engines below and make the first word of the message be the intended e-mail address followed by the message. This is automatically picked off the APRS Internet System (APRS-IS) by the e-mail engine and sent via conventional e-mail. The sender receives an APRS ack for the message.

* WU2Z e-mail Engine: The original APRS e-mail system. Send to "EMAIL"
* AE5PL e-mail Engine: Send to "AE5PL-EM".
* APRSlink on WLNK-1: Send to "WLNK-1" using the "SP" command [4]

Internet to APRS Messaging:

Although sending e-mail from any APRS client to any other ham radio operator anywhere in the world is easy as noted in the previous paragraph, there are different issues when sending general e-mail from the internet back to APRS. There are a few methods for sending text-messages from on-line APRS Internet (APRS-IS) systems to APRS mobiles and handheld operators. Because of the issues of security and license integrity, we hope that such systems implement the full APRS message mechanism to assure reliability and accountability listed in the reference[5]. Here are some existing systems:

*FINDU.COM MESSAGING, The original APRS on-line global information system called FINDU.COM written by Steve Dimse, K4HG now supports web based messaging. His page provides a link [6] that allows you to send a message from your browser to any APRS user.

*OPENAPRS.NET – Another global APRS on-line resource was written by Greg, NV6G[7]. Greg reports that OpenAPRS.net has full messaging support since August 08. Just sign up for an account (which is free), messaging is under

the Tools window which is accessed from the right icon bar. OpenAPRS includes a "Friends List" that will display when OpenAPRS has detected one of your friends sending APRS messages to let you know when they are online.

*WINLINK EMAIL - An APRS interface has been added to the Winlink system so that any APRS radio can be used to send and receive e-mail via the Winlink system. This interface requires no software or special interface, it is simply a technique that any APRS radio operator can use to log in and send/receive e-mail. This system is called APRSlink and was designed by Lee, K0QED.

E-mail Messaging from Cell Phones

To complete the full integration of APRS into the global e-mail system, we also need techniques to receive e-mail via APRS. The objective of this web page is to summarize all of the global messaging capabilities that can be used by APRS operators to send and receive text-messages no matter what the device and no matter where they are. This includes cell phones, palm-tops, blackberry's, pagers, and any other portable device.

APRS Messages to Your Cell Phone

N3FLR Frank Rossi reports that he receives all his APRS messages via his cell phone. First he set up the RSS capability [9] at FINDU.COM that his computer watches using YahooAlert[10]. He set up Yahoo Alert for a pager, and used his phone's text e-mail address (such as xxxxxxxxxx@txt.att.net). This will work with a text pager also. When FINDU sees a message to him on APRS it generates an RSS Feed that Yahoo Alerts is watching. Yahoo Alerts then forwards the RSS Message as Text to his cell phone. Although this is only one way communications, it still lets him receive his APRS messages at any time on his cell phone. He can customize the RSS feeds from FindU for weather alerts, or APRS users X amount of miles from him.

APRS Messages and the iPhone

NV6G, Greg has announced a beta-application for the iPhone[11] that has implemented full APRS messaging support through OpenAPRS's DCC interface. It also enables ham radio iPhones to be tracked using their GPS through the internet network to OpenAPRS's servers and out to the APRS-IS. Both systems follow OpenAPRS's license verification system. He expects to release the software sometime in late November.

APRS Messaging on Windows-Mobile 5&6

Lynn, KJ4ERJ, author of APRSISCE mentions that he has an APRS-IS client (beta)[12] running for Windows Mobile 5 and 6 specifically tested on the AT&T Tilt and maybe the SmartPhone (Motorola Q, I believe)... See his mobile KJ4ERJ-12 on FINDU.COM or APRS.FI. This can bring APRS to every ham with these cell phone devices!

Other APRS Tactical Situational Awareness

Remember that global APRS messaging is just a small part of the overall local and Global APRS objective of providing local real-time situational

awareness and connectivity for the exchange of pertinent information among Amateur Radio operators. Here are those other dimensions of information available to the Amateur Radio operator:

* Local situational awareness [the global APRS system]

* Local/Global Message capability by call sign (this article)

* Mobile display of locally recommended voice frequency for travelers.[13]

* Ability to check-in and do basic functions from any DTMF radio using APRStt [2]

* Ability to use thousands of DTMF Messaging radios (FT-51R and TH-78A) [14]

Travelers Voice Repeater Frequencies

One of the most useful bits of text-message available to the Amateur Radio traveler is the display of the locally recommended travelers voice repeater frequency. This information can be pushed to the front panel of any of these texting devices when the operator enters the footprint of that repeater. Since 2004 we have been encouraging this Local Info Frequency Initiative [13]. It displays the best recommended voice repeater and other RF assets of value to the traveler such as the local IRLP, EchoLink, and Winlink frequencies, or NET times or meetings in progress, etc as shown below on the TH-D7 displays.

Notice how the IRLP and EchoLink nodes identify not only their node numbers and call signs, but also their Tone, Range and status. (Bsy, Rdy, Lnk etc)... By pressing the OK button to see the POSIT screen, you can see that the EchoLink node is 17.1 miles to the Southwest. [These photos were taken before we noticed that the Frequency on the second line was missing!]

APRS is a two-way local Information Distribution and Communication System (not just a vehicle tracking system). Please see the web page that corrects these misconceptions [14].

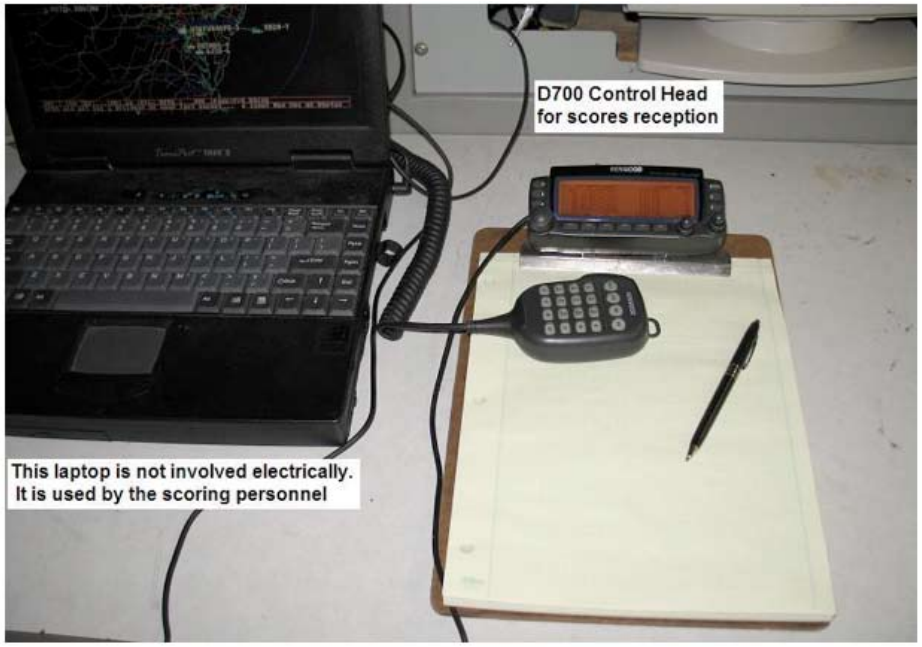
The objective is Human-to-human local real-time info and communications! Universal Text-Messaging among all Ham radio clients is our new initiative. Let the APRS Internet System (APRS-IS) be the backbone. Let all those creative ham radio code writers write the gateways to all available products!

GPS Trackers Are Two-Way-Too

Even transmit-only APRS trackers should be configured to facilitate two-way human communications. The RECEIVER of any tracker should be tuned to a desired voice communication calling channel with this frequency information placed into the beacon text of the tracker. This way, all who see the tracker position packet can also see his frequency and establish contact with the operator. . Often this can simply be the APRS Voice Alert frequency which is automatically included in every APRS radio [16].

Handheld Eventdata Entry Via Messaging

While we are talking about overlooked capabilities,... Not only can these radios, cell phone and laptops convey messages and frequencies, but they also make excellent data entry devices in the field for Amateur Radio at special events (and not just position and messages!)[17]. As shown in the figure above, a few simple text messaging key strokes can report numbers, scores, times, ID's and all manner of data at events supported by ham radio. These error-free transmissions are far more efficient than voice reporting in most cases.



Conclusion

Every teenager in the world now takes text messaging for granted and APRS has had handheld text messaging now for over ten years. Some other ham radios have had DTMF text message for over 15 years. It is time we tie all this together into a universal HAM radio text messaging system. The global and local APRS backbone exists. All we need are a few authors to write the interfaces and gateways to the variety of devices. The objective is universal Ham Radio Text Messaging via CALLSIGN addressing.

References:

[1] Maximizing the Mobile Motorist Mission, *QST*, September 2008

[2] www.aprs.org/aprstt.html – APRS TouchTone System

[3] www.aprs.org/aprsxo.html

[4] www.activeham.com/pmwiki.php?n=Main.NewAPRSLinkSPCommand –E-mail via the WLNK-1

[5] www.aprs.org/messages/message-issues.txt – Message issues from the internet to the APRS system.

[6] www.findu.com/cgi-bin/entermsg.cgi? – Provides a .CGI link for message entry

[7] www.openaprs.net – Provides an on-line message capability via the openaprs.net system

[8] www.winlink.org/aprslink – Provides a winlink APRS packet radio EMAIL capability

[9] http://rss.findu.com – An RSS output capability on FINDU.COM

[10] http://alerts.yahoo.com – An e-mail alert system on YAHOO

[11] The iPhone APRS application by NV6G, Greg [site TBD]

[12] APRS on the AT&T Tilt phone by KJ4ERJ [site TBD]

[13] www.aprs.org/localinfo.html –Displays local repeater info on travelers APRS radios

[14] www.aprs.org/APRS-tactical.html –The misconceptions about APRS

[15] www.aprs.org/FT51-TH78l.html – Using the FT-51R and TH-78A for text messaging

[16] www.aprs.org/VoiceAlert3.html – The APRS Voice Alert back-channel for instant contact

[17] www.aprs.org/aprsevent.html –Using APRS messaging for data reporting at scout and other events

###

New TAPR Badge Available

The brand new TAPR Badge with your name, call sign, and the TAPR logo is now available.

The price for the badge is $14 US for TAPR members and non-members plus shipping and handling. Visit https://www.tapr.org/tapr_addorder.php?add=316 to order online.

###

Packet Status Register

#107 Winter 2009, ISSN: 1052-3626

Published by
TAPR
phone 972-671-TAPR (8277)
e-mail taproffice@tapr.org
URL www.tapr.org
TAPR Office Hours
Monday – Friday, 9 AM – 5 PM Central Time

Postmaster: Send address changes to TAPR, P. O. Box 852754, Richardson, TX 75085-2754. *Packet Status Register* is published quarterly by TAPR. Membership in TAPR, which supports the electronic publication of the *Packet Status Register*, is $25.00 per year payable in US funds.

TAPR is a community that provides leadership and resources to radio amateurs for the purpose of advancing the radio art.

Submission Guidelines

TAPR is always interested in receiving information and articles for publication. If you have an idea for an article you would like to see, or you or someone you know is doing something that would interest TAPR, please contact the editor (wa1lou@tapr.org) so that your work can be shared with the Amateur Radio community. If you feel uncomfortable or otherwise unable to write an article yourself, please contact the editor for assistance. Preferred format for articles is plain ASCII text (Microsoft Word is acceptable). Preferred graphic formats are PS/EPS/TIFF (diagrams, black and white photographs), or TIFF/JPEG/GIF (color photographs). Please submit graphics at a minimum of 300 DPI.

Production / Distribution:

Packet Status Register is exported as Adobe *Acrobat* version 5 and distributed electronically at www.tapr.org

PSR Packet Status Register Editor:

Stan Horzepa, WA1LOU
One Glen Avenue, Wolcott, CT 06716-1442 USA
phone 203-879-1348
e-mail wa1lou@tapr.org

TAPR is a not-for-profit scientific research and development corporation [Section 501(c)(3) of the US tax code]. Contributions are deductible to the extent allowed by US tax laws. TAPR is chartered in the State of Arizona for the purpose of designing and developing new systems for digital radio communication in the Amateur Radio Service, and for disseminating information required, during, and obtained from such research.