SDR (and other radio) futures

Software defined radio's progress to date plus a look at what may be around the corner

INTRODUCTION. Predicting the future is always a risky venture and particularly so when relatively new technologies are involved. However, in the world of software defined radios (SDRs) a number of different architectures are now emerging, each with its own 'pros' and 'cons', giving us a clearer picture of where radios will head in the next decade. In this two-part article we will investigate these architectures and consider individually what they offer the radio amateur who is interested in SDR. In the process, we will also look at how radios using conventional superheterodyne architectures are evolving alongside SDR, assisted by SDR-related digital signal processing techniques. Please note that whilst being a technically based article, the views in this article are those of VK6APH and VK6VZ and may not coincide with other radio 'fortune tellers'.

ARCHITECTURES. At the highest level, the differences between the new SDR architectures and their predecessors – the first generation quadrature sampling detector (QSD)/mixer type and the second generation digital down conversion/digital up conversion (DDC/DUC) – relate to needing a personal computer (or similar device, eg a tablet or mobile phone) so the user can operate the radio as opposed to conventional analogue controls (ie knobs and dials).

Well-known examples of the QSD/mixer architecture are the Flex SDR-1000 [1] and the SoftRock series of kit receivers and transceivers [2], while the DDC/DUC architecture is used by the popular High Performance Software Defined Radio (HPSDR) [3] project (**Photo 1**).

The exciting SDR technologies that are emerging use DDC/DUC, where received signals are digitised 'down' virtually directly at the antenna socket using an analogue-to-digital converter (ADC). The transmitted signal is created digitally and converted 'up' to an analogue signal, at the required output frequency, using a digital-to-analogue converter (DAC) and subsequently amplified in conventional power amplifiers.

Owing to the inherent limitations of the first generation QSD/mixer architecture when it comes to image rejection, the authors consider that DDC/DUC will be the dominant SDR architecture of the future. That being said, the image cancellation techniques being used by the latest QSD/mixer SDRs, in particular the Elecraft KX3, is much more effective than previous techniques. One significant advantage of the QSD/mixer architecture compared with DDC/DUC techniques is the former uses much less power. This can be a consideration where portable operation, for example, is desired, such in the KX3.

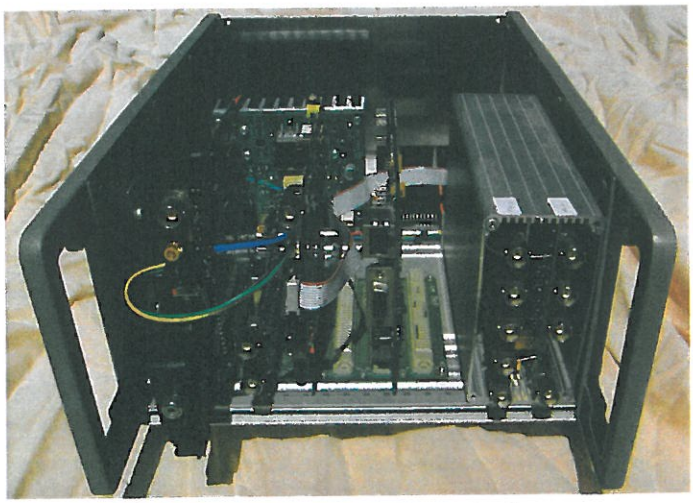

PHOTO 1: VK6VZ's HPSDR transceiver under construction.

ALTERNATIVE ARCHITECTURES. ***Conventional knob-based radios.*** Whilst most of the existing manufacturers of HF amateur radio equipment use DSP techniques, the majority still use an analogue 'superhet' architecture with DSP applied in the final intermediate frequency (IF) stages. However, on the 21st century HF amateur bands, where there are lots of *very* strong signals during contests and major DXpeditions, in order to improve dynamic range there has at least been a move away from using a first IF in the VHF spectrum. Manufacturing crystal filters having steep sides and with a bandwidth of a kHz or two is much easier at HF than VHF.

As a result, Elecraft and Kenwood have gone 'back to the future' by using a HF first IF, in a similar manner to the classic analogue radios of the 1980s such as the Kenwood TS-830S and Drake T4XC/R4C, giving improved close-in dynamic range. However, the DSP that is used in contemporary radios like the Elecraft K3 and Kenwood TS-590S is vastly superior to that implemented in radios of the previous decade.

A radio with a conventional analogue front panel radio but high-grade digital processes appeals to the technically savvy radio amateur who has not embraced the idea of using a personal computer as the sole means of operating his radio. This is a category into which VK6VZ, who reluctantly uses a PC for work every day, still falls.

The Elecraft KX3 [4] is one of the first SDR-based radios to have a traditional analogue 'dial and knob' interface (**Photo 2**) and has sold very well because of its pure SDR sound and good dynamic range. If it had a DDC/DUC architecture instead of a direct conversion one, VK6VZ would probably be using it as his main radio.

As far as the authors are aware the Swiss-designed ADAT ADT-200A [5] is the only commercially available DDC/DUC HF transceiver currently available that uses conventional controls in order to operate it (**Photo 3**). The receiver section uses a 14-bit ADC that offers a claimed signal to noise (S/N) ratio of 74dB (over the half Nyquist bandwith of 36.86MHz). After the subsequent decimation, ADAT claim a blocking dynamic range of 120dB.

The ADT-200A has four parallel receivers and eight independent VFOs to provide cross-band and split-band operations. It produces 50W

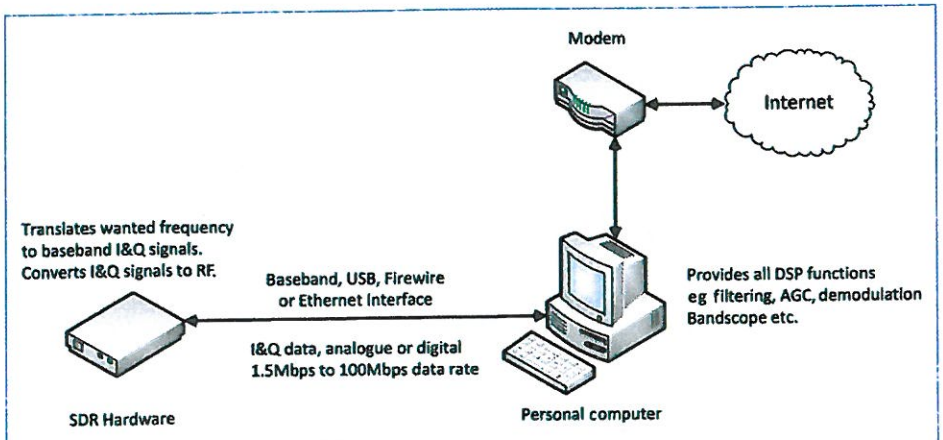

FIGURE 1: SDR connected to a PC.

and also contains an antenna scope that allows measurement complex antenna impedances – like a vector network analyser. The VFOs can all be set to the same or to different bands, independent from its receiver frequencies.

There are two ways to remote control the ADT-200A transceiver: via USB interface, controlled by a PC, or via local area network (LAN) using an optional Web Server Module. This makes it possible to operate the ADT-200A remotely via an internet connection. ADAT says on its website that remote control via LAN is currently only possible between two ADT-200A devices.

It will be interesting to see if traditional radio manufacturers such as Yaesu, Icom, Kenwood, Elecraft and Ten-Tec offer a radio like the ADT-200A in the near future. We anticipate this will be the case since the trend to lower cost and higher performance ADCs means the manufacturing cost of DDC/DUC-based radios will be lower than conventional radios with superheterodyne architectures.

The 'pros' for such a DDC/DUC radio with a conventional 'knobs and dial' front panel are:

- A conventional analogue user interface of this kind is attractive to those who use traditional radios
- Users are drawn to a radio that is contained in a single box for reasons of convenience The radio can be operated without the use of a PC.

In contrast, the 'cons' include:

- Less flexibility in adding highly useful extra features (such as a high definition, wide spectrum bandscope or a wide spectrum Morse decoder/skimmer) than an SDR radio that uses a personal computer to do its DSP
- The radio is likely to cost more than an SDR radio that uses a personal computer to do its DSP.

PC-BASED DSP RADIOS. Presently this architecture (**Figure 1**) represents the largest pool of SDR radios. For some time there have been available a number of commercial radios of this kind in both complete (such as the FlexRadio Systems 3000 and 5000 radios) and kit form (such as the Softrock series). The prime purpose of the SDR hardware in such radios is to reduce the data rate of the analogue-to-digital conversion to a slower rate such that the associated PC can process it.

PHOTO 2: The Elecraft KX3 is a software defined transceiver with traditional front panel controls. Photo courtesy of Elecraft.

For a DDC receiver, if we consider the data rate required for say digitising the entire HF spectrum from 0 to 30MHz with 16 bits per sample, the minimum data rate is: 60,000 x 16 = 960 megabits per second (960Mbps). Whilst this data rate is within the specification of a Gigabit Ethernet interface, the PC needs to process this data at a *sustained* rate. For even a fast personal computer this is akin to 'drinking from a fire hose' and certainly in the near future, if technological development continues at the current rate, is not going to be cost-effective in regards to an amateur radio budget.

In practice, most PC-based DSP SDR hardware reduces the data rate to a more manageable 192,000 samples per second (192ksps), which at 2 x 24 bits per second is approximately 9Mbps. Note the factor of two is because in-phase *and* quadrature samples are generally sent to the personal computer.

Slower sampling rates (eg 96ksps and 48ksps) as well as rates in the low MHz are also common. The higher sampling rates have the advantage that any associated bandscope will be able to display a wider slice of HF spectrum.

At the lower data rates a mid-range PC is capable of providing the necessary DSP for filtering, noise reduction, a bandscope display, etc.

Communications between the SDR hardware and the PC was initially at baseband audio frequencies since the analogue to digital conversion was carried out using a sound card in the PC. For higher performance, the digital conversion was implemented in the SDR hardware, and USB and FireWire used to interface to the PC. However, more recently high speed Ethernet has proved a very popular and reliable interface, such as provided by the Metis and Hermes boards (**Photo 4**) for the HPSDR project [3].

On this basis, the 'pros' for such architecture include:

- The SDR hardware is often relatively simple and hence low cost (ie Softrock series of receivers and transceivers).
- The user may be able to use an existing PC that is already used in the shack/household.
- There is a wide range of suitable SDR application software to run on a personal computer (such as *PowerSDR* and *KISS Konsole*) and support (including suitable drivers) for different operating systems (such as Windows XP, Windows 7 and Linux).
- The PC software is frequently 'open source', meaning that the user with some programming skills can add new features themselves or via a group of similarly interested radio amateurs (such as openhpsdr.org).
- A suitably skilled individual, or group of enthusiasts of this kind, can also provide code updates, bug fixes, user support and new features.

In contrast, the 'cons' include:

- Since all the DSP is being performed in a PC, then the user experience is dominated by how well the PC performs. Simultaneously loading the PC with several other tasks (ie downloading from the internet whilst listening to your favourite music CD and keeping your eye on HF using an SDR application) on a low performance PC is a sure recipe for a poor user experience.
- Give the myriad of possible hardware, operating systems and installed software combinations this architecture can present a very difficult, time consuming and expensive, user support load for a commercial SDR manufacturer. Often user support issues are not related to the SDR

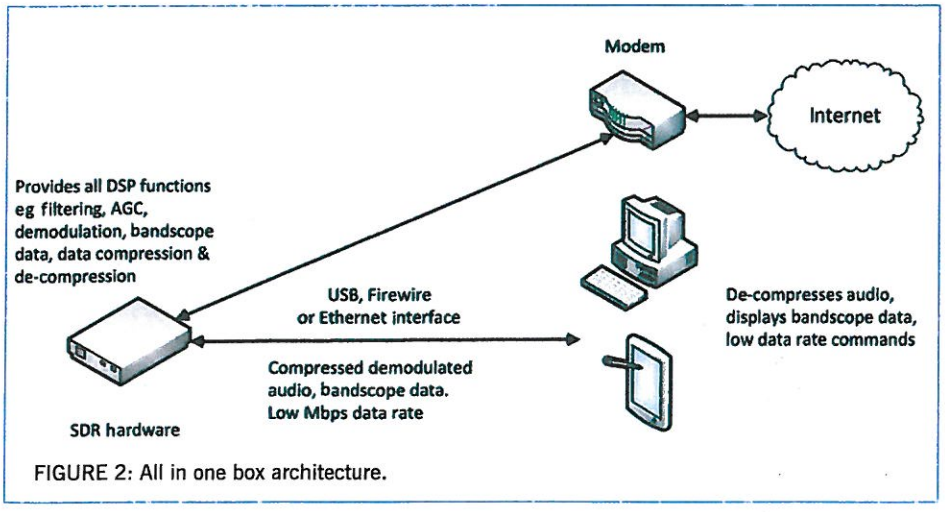

FIGURE 2: All in one box architecture.

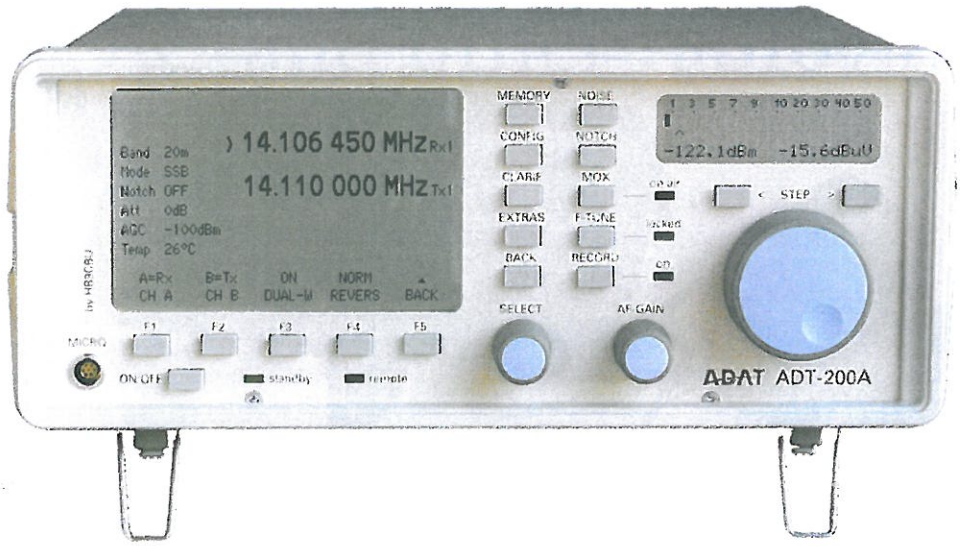

PHOTO 3: ADAT ATC-200A DDC/DUC HF transceiver. Photo courtesy of www.adat.ch.

hardware or software but other applications or device drivers un-associated with the components provided by the SDR manufacturer.

- For the user who wishes to use his SDR in a remote station configuration, a local PC will be required in order to provide the necessary interface between the SDR hardware and the network used to connect it to the PC (eg the internet).

SDR-BASED DSP RADIOS. In this new architecture the bulk of the DSP is undertaken in the SDR hardware (**Figure 2**). In addition to reducing the data rate, the SDR hardware undertakes *all* the DSP required in terms of filtering, noise reduction, automatic gain control (AGC), demodulation, etc.

This kind of architecture appears to have been adopted by FlexRadio Systems in its FLEX-6000 'Signature' range **[6]** of SDRs, which were announced last year. Since virtually no DSP is undertaken in the associated PC, the processing demands on the latter are reduced enormously.

The data rate between the SDR hardware and the PC (which may be geographically distant) only needs to be sufficient to pass the receiver audio, microphone audio, bandscope data and control signals (eg PTT and CW key closures).

Applying readily available data compression techniques **[7]** to the receiver audio can result in a data rate between the SDR hardware and PC as low as 400kbps. This is because the associated PC only needs to decompress the data and pass it to its local sound card plus format the bandscope data for display.

As this latter data is at only a few tens of updates per second (thanks to the persistence of our vision) then this similarly requires little in the way of bandwidth. Assuming the bandscope display will make full use of any graphics acceleration (eg DirectX or OpenGL) available on the PC, then again there is little loading of the PC processor.

On this basis, the 'pros' for such architecture include:

- The performance of the associated PC has much less impact on the overall user experience with this flavour of SDR architecture. As a result, the user has much more freedom in regard to the power of the personal computer being used and what other software it runs at the same time.
- Changes to the personal computer hardware and software have little effect on the overall performance of the SDR system. As you can imagine, this is a very attractive scenario to commercial radio manufacturers, as it potentially requires only a relatively small amount of user support.
- The low PC processing requirements means lower-end computing devices (eg tablets, netbooks and smartphones) could potentially be used to control the SDR hardware.
- Given the processing power available in the SDR hardware, it should be possible to connect it (via a suitable modem) directly to the internet. This would enable remote operation of the station from any location worldwide.
- The protocol used between the SDR hardware and PC may be publically available, perhaps even an application programming interface (API), in which case a user with some programming skills could add new features of a cosmetic (ie graphical user interface) nature.

In contrast, the 'cons' include:

- Substantially more complex SDR hardware is required, resulting in higher design, development and product costs.
- Generally the tools used to develop the complicated firmware necessary for the SDR are likely to be specialist in nature. This means that for commercial providers, the code is either likely to be proprietary or if 'open' very hard for a user to amend. As a result, the user ends up being totally dependant on the manufacture to provide all future code updates, bug fixes, digital modes and new features – a situation that radio amateurs interested in experimentation are unlikely to be comfortable with.
- If wideband in-phase and quadrature (I & Q) data is required to be sent to the user's PC (eg for digital modes, real-time recording of a wide bandwidth of signals or the CW Skimmer application **[8]**) then the low data rate advantages of the architecture are mostly lost.

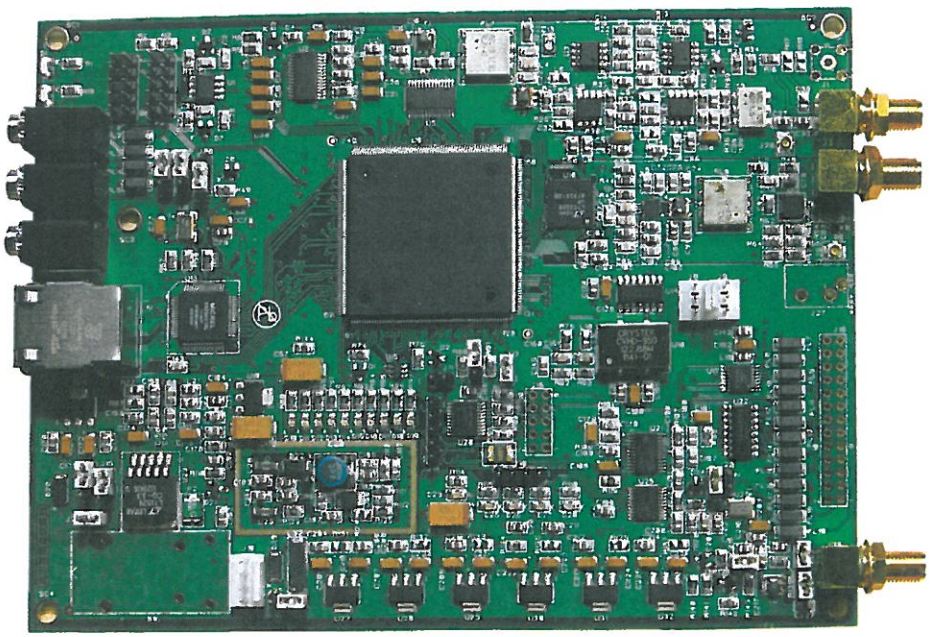

PHOTO 4: Prototype of Hermes openHPSDR DUC/DDC transceiver board.

In the second part of this article, Phil and Steve will describe the new SDR Server architecture, look at how cheap, small, single board computers (such as the Raspberry Pi) can be used with this architecture and conclude their investigation into the future of SDR.

WEBSEARCH

[1] www.flex-radio.com/Products.aspx?topic=sdr1k_details
[2] http://fivedash.com
[3] http://openhpsdr.org
[4] www.elecraft.com/KX3/kx3.htm
[5] www.adat.ch/index_e.html
[6] www.flex-radio.com/
[7] See 'What is Opus?' at www.coolutils.com/Formats/Opus
[8] www.dxatlas.com/cwskimmer/

SDR (and other radio) futures part 2

Using the SDR Server architecture with cheap, small computers such as the Raspberry Pi – and a look at the future of SDR

SDR SERVER. In this architecture the software defined radio (SDR) hardware provides the absolute minimum signal processing necessary in order to present the digital data to an associated personal computer, which we will refer to as an 'SDR Server' (see **Figure 3**).

The SDR Server is a PC whose dedicated function is to undertake all the digital signal processing (DSP) required by the associated SDR and provide suitable interfaces to the SDR hardware and the PC (or laptop or tablet, etc) that controls the SDR. The latter computer will be referred as the SDR Client.

The SDR Server only runs the software necessary to process the data from the SDR hardware (plus any necessary housekeeping tasks) and present it in the required format to the SDR Client PC. The SDR Server is not intended to run *any* non-SDR related user applications and is totally dedicated to its SDR data processing task. As a result, the SDR Server does not require a display, keyboard or mouse to be connected to it in order to operate. When necessary, the SDR Server can be managed via Ethernet from another PC (say, the home desktop PC or laptop) or remotely via the internet.

Whilst we have used the term 'PC' for the SDR Server, this conjures up a rather grander image than the kind of hardware that is actually used. In practice, the PC can be a small, low-cost, single-board computer (SBC), costing only a few tens of pounds.

Just how far the size and prices of SBCs are likely to fall in the future is an interesting question, but currently experimenters are using the sub-£30 credit card size Raspberry Pi [9] to implement an SDR server [10].

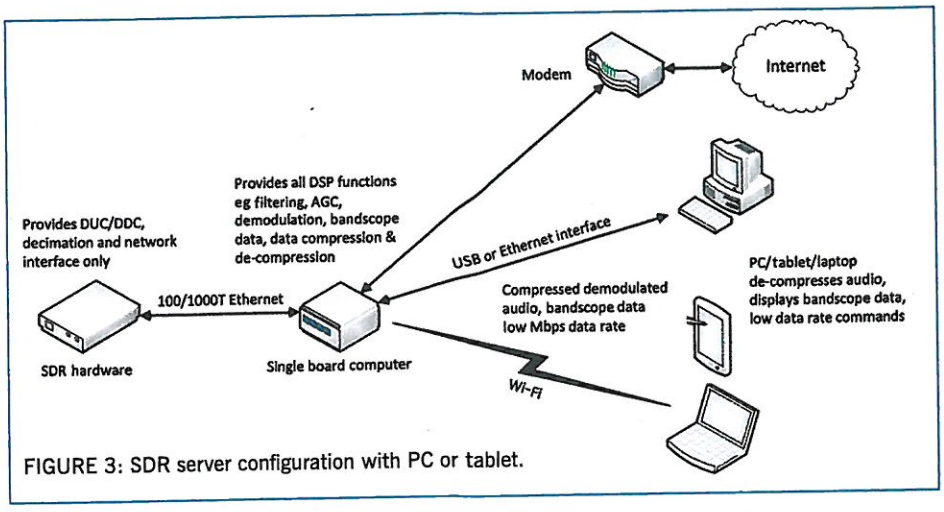

FIGURE 3: SDR server configuration with PC or tablet.

Finding a suitable SBC to act as an SDR Server is very easy since there are lots of suitable candidates appearing on the market. A popular choice is the recently released ODROID-X2 [11].

In general, the SBC uses the Linux operating system and the SDR Server application code is open source. This means that an individual or group of enthusiasts can provide code updates, bug fixes, user support and new features.

The use of Linux enables a large variety of user support tools to be accessed. Hence, if remote access to a SDR Server is enabled by its owner/user, assistance in troubleshooting can be provided by others who directly access the SDR Server that requires maintenance.

Another feature is that since the SDR Server is undertaking all the DSP and data reduction services, as in the case of the previously discussed SDR-based DSP architecture, the associated demands on the main PC (in this case, the SDR Client) are greatly relaxed.

In addition to the 'pros' of the SDR-based DSP architecture, the 'pros' of the SDR Server architecture in summary are:

- The processing undertaken by the SDR hardware is the minimum necessary to communicate with the associated SDR Server, which means hardware costs are minimised.
- As the SDR Server uses a small, low-cost mass-produced SBC, upgrading the latter to use a more powerful featured SBC in the future is likely to be relatively low cost. This scenario is assisted by the SDR Server running the free Linux operating system.
- The associated application software is frequently open source, so users with some programming skills can add new features themselves or via a group of similarly minded enthusiasts.
- A suitably skilled company, group of enthusiasts or individual can provide code updates, bug fixes, user support and new features.
- Remote access to the SDR Server, via the internet, can be used by a skilled company, group or individual to assist a user who has technical issues.
- An SBC that has Wi-Fi support would enable access to the associated SDR from anywhere within Wi-Fi range.

In contrast, the 'cons' include:

- The SDR consist of two hardware components – the SDR itself and the SDR Server. This increased the complexity

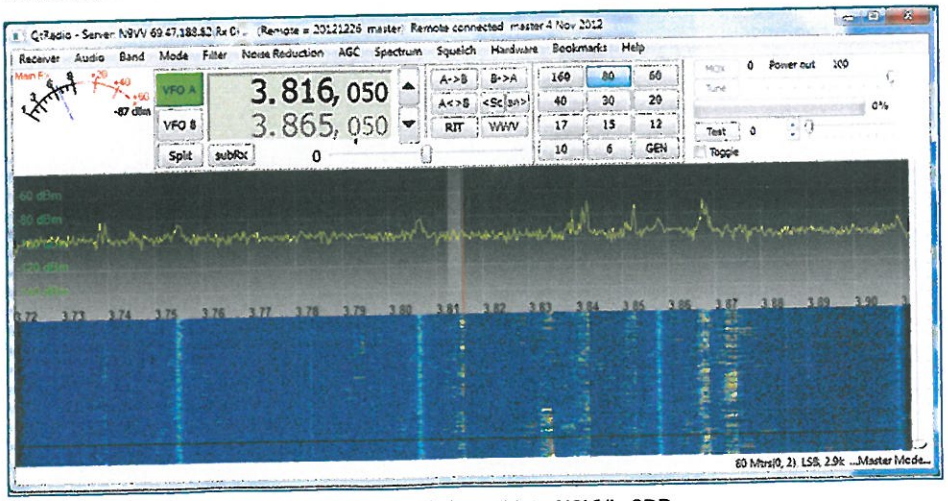

FIGURE 4: Screen shot by VK6APH of *QtRadio* logged into N9VV's SDR.

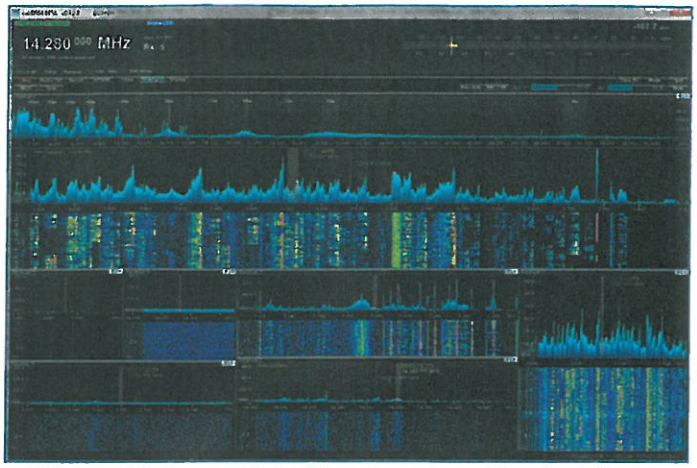

FIGURE 5: *cuSDR* screen shot with seven receivers. Note the CPU usage is only 11%. (Image courtesy of Hermann, DL3HVH).

and need for space, which may affect compact remote installations.

- Assuming that Ethernet is used to communicate between the SDR and the SDR Server then, unless a SBC with dual Ethernet ports is available [12], a small Ethernet switch will be required to connect to the local PC (SDR Client) and/or modem. However, a combined switch and modem could be used here. Note that low-cost Ethernet switches can be a source of receiver noise and/or spurs.
- Those who are uncomfortable with using PCs for SDR purposes may feel intimidated by the need to use a second PC running an operating system such as Linux, despite the former costing only a few tens of dollars and the latter being free.

SINGLE BOARD COMPUTER TRENDS.

The SBC market is developing very rapidly because of the commercial need for low-cost, high-performance embedded processing. Similar to the conventional PC market, such SBC often feature multiple-core processors and high-performance graphics processing units.

The production of software that supports multiple cores and graphics processing units (ie parallel processing) is a rapidly developing field with a number of alternative approaches. The well known graphics card manufacturer Nvidia is promoting a technique called CUDA™.

Nvidia explains CUDA as follows: "CUDA is a parallel computing platform and programming model that makes using a graphics programming unit for general purpose computing simple and elegant. The developer still programs in the familiar C, C++, FORTRAN, or an ever expanding list of supported languages, and incorporates extensions of these languages in the form of a few basic keywords.

"These keywords let the developer express massive amounts of parallelism and direct the compiler to the portion of the application that maps to the GPU." More information relating to CUDA is available [13]. Note the use of CUDA is limited to Nvidia hardware."

An alternative approach to the Nvidia CUDA is used in the Open Computing Language (OpenCL) [14]. OpenCL is an open standard that enables parallel computing on heterogeneous platforms consisting of central processing units, graphics processing units, digital signal processors and other kinds of processors. It has been adopted by Intel, Advanced Micro Devices, Nvidia and ARM (formerly known as Advanced RISC Machines).

CURRENT SDR SERVER SOFTWARE.

Members of the internationally-supported openHPSDR project [15], in which VK6APH and VK6VZ have a long-standing involvement, are actively developing software for the previously described SDR Server architecture. Both of us have operational HPSDRs, based on the Mercury and Penelope digital down conversion and digital up conversion receiver and transmitter boards, and run a variant of the *PowerSDR* software.

There are two software developments in particular which we think readers will find of interest – *ghpsdr3* (originated by John Melton, G0ORX/N6LYT and *cuSDR* (currently under development by Hermann von Hasseln, DL3HVH). *ghpsdr3* [16] is a software defined radio server/client or hardware server/DSP server/client format program written specifically for HPSDR. G0ORX/N6LYT is developing the software on the Ubuntu version of Linux (specifically, version 9.10). The good news for those users of Microsoft Windows is that John has ported the server and DSP server software to run on it and has produced written instructions on how to set up multiple independent receivers using Windows [17]. Whilst John's code is presently 'receive only', the variant *ghpsdr3-alex* [18] provides transmit capabilities.

The current version of *ghpsdr3* allows for the SDR Server and SDR Client software to be run on the same PC or on separate PCs. John and a number of collaborators are working on a full set of client software to run on multiple PCs, connecting to the servers using TCP/IP protocols. You can follow the development of this code on John's blog [19].

Connecting an SDR to a PC or a SBC running *ghpsdr* Server software enables the user to access and control the radio via the internet anywhere in the world that an internet connection is available. The access can be limited to the owner of the system or advertised for public access. A website [20] is available that shows a real-time list of *ghpsdr* servers that may be openly accessed by anyone using the free *QtRadio* software [21].

The screen shot in **Figure 4** is of VK6APH using the *QtRadio* software to access the Apache Labs [22] ANAN-10 SDR belonging to Ken, N9VV, in Chicago, USA. Versions of *QtRadio* are available that will run on a Windows PC, Android tablet or mobile phone. VK6APH often listens to the early evening 3.8MHz nets in the USA during lunchtime in Australia using *QtRadio* on an Android tablet.

As its name suggests, *cuSDR* is intended to use CUDA and OpenCL in a Client/Server architecture. The code is presently available for beta testing as a single application (where the Client and Server code are combined into one program).

A screen shot of *cuSDR* simultaneously processing seven receivers from a Hermes [23] SDR is shown in **Figure 5**. Note how low the processor loading is, despite the very large amount of digital signal processing that the program is doing.

In case you have any concerns about the processing capability of a SBC, then the following example should hopefully put your mind at rest. It should also give you a mind-bending taste of what is currently possible with the SDR Server architecture.

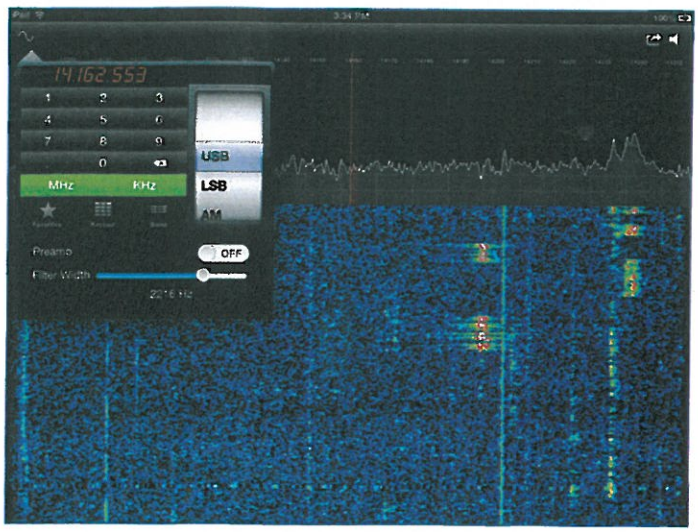

FIGURE 6: Screen shot of an iPad controlling a Hermes SDR.

One of the highest performance SDRs presently on the market is the ANAN-100D manufactured by Apache Labs. This contains two independent 16 bit analogue to digital converters. Each ADC is capable of providing seven fully independent receivers, each with a bandwidth of 384kHz – which means the total number of available receivers is 14. This means that, theoretically, you could have separate receivers covering the entire width of the 1.8, 3.5, 7, 10, 18 and 21MHz amateur bands (a total of 11 receivers), plus three 384kHz wide receivers on 28MHz or on 50MHz. With all 14 receivers providing 384kHz of 24 bit in-phase and quadrature (I &Q) data, the bit rate to the host PC is a minimum of 260Mbps.

The recently announced SECO CARMA SBC **[24]** provides a remarkable amount of processing power for a small computer. Based on the Nvidia Tegra3 ARM central processing unit and Nvidia CUDA-enabled graphics processing unit, the board provides 270 Single Precision GFlops ($270x10^9$ floating point operations per second). Even more impressive is the latest Nvidia GeForce GTX690 graphics processor **[25]** which incorporates 3072 CUDA cores, providing 5.6 Single Precision TFLOPS ($5.6x10^{12}$ per second). Such performance leaves even the most high performance digital signal processing chip in its dust!

Whilst the SECO board is certainly not at the low end of the market, the central processing unit and graphics processing unit combination used on the CARMA is likely to find its way into low-end SBC boards in the near future. Hermann, DL3HVH, is currently porting his *cuSDR* code to use this board.

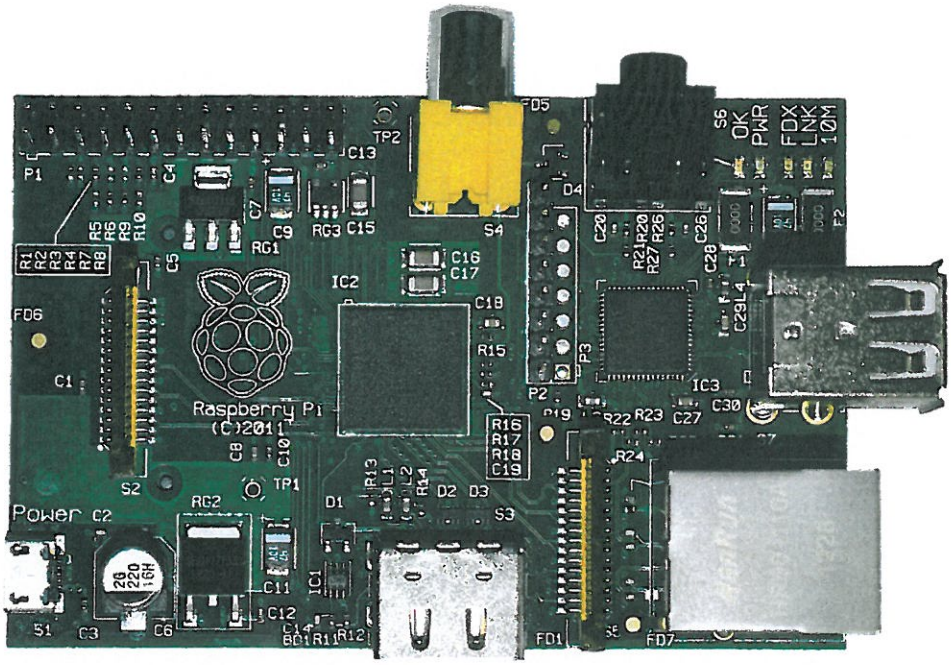

PHOTO 5: The Raspberry Pi single board computer.

CONCLUSIONS. For amateur-developed and supported SDR projects, the power and flexibility of the SDR Server architecture has many benefits – which have also been recognised by commercial SDR manufacturers. A recent announcement by RF-Space **[26]** for their SDR-IQ server software supports this view.

However, the performance of the household PC, tablet and mobile phone are also increasing at a rapid rate, giving a fresh boost to the established second generation digital down conversion/digital up conversion (DDC/DUC) SDR architecture. A fine use and example of the processing power of the Apple iPad™ is the software developed by Jeremy McDermond, NH6Z, called *Heterodyne* **[27]**. This enables a user to connect to an openHPSDR SDR (eg the Hermes DDC/DUC transceiver) via a Wi-Fi enabled Ethernet switch, with all the digital signal and display processing necessary being carried out within the iPad. A screen shot of an iPad controlling a Hermes SDR is shown in **Figure 6**.

The early years of the transition from analogue radio to SDR was dominated by the need to develop new hardware. As the DDC/DUC architecture becomes the *de facto* standard for radios of all kinds – everything from mobile phones to handheld VHF/UHF amateur FM transceivers now have a DDC/DUC SDR as their heart – in our view the SDR hardware will move to 'black box' status.

What differentiates one radio from another will be focused on the graphical user interface and the features it provides, rather than on the hardware. Finally, Software Defined Radios will truly live up to their name. This is going to require a major change of perception by radio amateurs and professionals in what they see as a 'radio'.

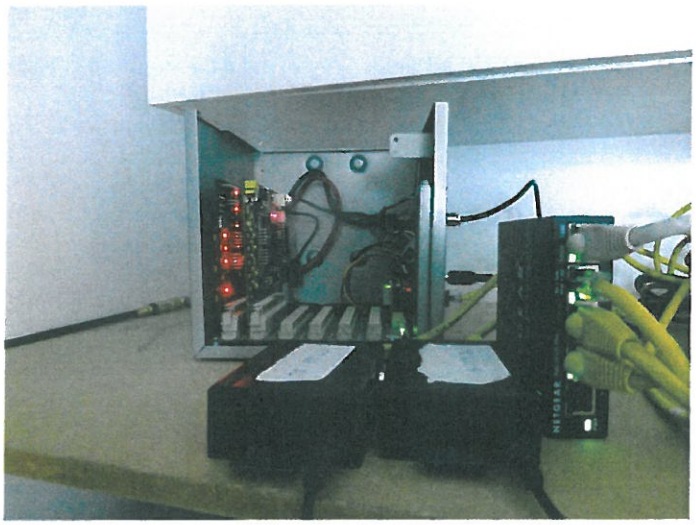

PHOTO 6: John Melton's *ghpsdr* development system using two Raspberry Pi boards (one as a SDR hardware server and the other carrying out the DSP). John points out that on the latest Raspberry Pi operating system that supports hardware floating point operation, he can run all the software on a single board, including running an Apache web server.

WEBSEARCH

[9] www.raspberrypi.com
[10] See John Melton's blog at http://g0orx.blogspot.com.au/
[11] ODROID-X2 – see www.hardkernel.com/
[12] Axiomtek – see CAPA111 embedded 3.5 inch SBC – www.axiomtek.com/
[13] CUDA – http://en.wikipedia.org/wiki/CUDA
[14] OpenCL – http://en.wikipedia.org/wiki/OpenCL
[15] www.openHPSDR.org
[16] http://openhpsdr.org/wiki/index.php?title=Ghpsdr3
[17] http://openhpsdr.org/wiki/index.php?title=Multiple_independent_receivers_-_how_to_set_up_on_Windows
[18] http://napan.ca/ghpsdr3/index.php/Main_Page
[19] http://g0orx.blogspot.com.au/
[20] The list of ghpsdr servers can be found at http://qtradio.napan.ca/qtradio/qtradio.pl
[21] You can download the latest Windows version of *QtRadio* from http://napan.ca/ghpsdr3/index.php/QtRadio_on_Windows
[22] https://apache-labs.com/
[23] To follow cuSDR development, join Google+ - see https://plus.google.com/107168125384405552048#107168125384405552048/posts
[24] SECO web site – www.seco.com/en
[25] Nvdia GeForce GTX690: www.geforce.com/hardware/desktop-gpus/geforce-gtx-690
[26] www.rfspace.com
[27] www.nh6z.net/Amatuer_Radio_Station_NH6Z/My_Software.html