A&A 485, 615–622 (2008)
DOI: 10.1051/0004-6361:200809396

Astronomy & Astrophysics

A new signal processing platform for radio astronomy

A. W. Hotan

School of Maths and Physics, University of Tasmania, Private Bag 21, Hobart, TAS 7001, Australia
e-mail: ahotan@utas.edu.au

Received 15 January 2008 / Accepted 7 April 2008

ABSTRACT

Context. We describe the concept, construction, and testing of TasPGA, a new, versatile digital signal processing device designed for radio astronomy applications.
Aims. A minimalist design philosophy was used to develop a general-purpose radio astronomy instrument based on field programmable gate array (FPGA) technology. The design emphasises reliability and flexibility, facilitating the rapid development of new FPGA firmware for a wide range of experiments.
Methods. Hardware and firmware for TasPGA were designed using software provided by Altium Ltd. and Xilinx Inc. We tested the instrument by implementing a Fourier transform spectrometer capable of dividing 100 MHz of bandwidth into 16 384 channels.
Results. We present data taken with a TasPGA spectrometer at the Mt. Pleasant radio observatory, including the spectra of two astronomical methanol masers.

Key words. instrumentation: miscellaneous – instrumentation: spectrographs – techniques: miscellaneous

1. Introduction

Consumer demand has driven the rapid pace of development in computer and communications technology during the past few decades. Scientific users have often been the primary "early adopters" of these new technological advances. In particular, radio astronomers now have access to instruments that can record a true representation of the electromagnetic fields they observe, with little or no loss of information. Rapid digital sampling of an electromagnetic signal is essential to maximise observable bandwidths. In turn, high data flow rates push the limits of technology and must be handled with powerful signal processing systems. Fortunately, modern electronics are also becoming increasingly versatile. The rapid progression of systems from concept to market and the need for autonomous adaptive devices has led to the development of powerful configurable logic hardware, including field programmable gate arrays (FPGAs).

In many ways, FPGAs fill the gap between application specific integrated circuits (ASICs) and software that can be run on a general purpose processor of the kind found in personal computers (PCs). Historically, radio astronomy instruments required levels of performance or specific functionality that could only be achieved with custom-built electronics. Such instruments typically had a useful lifetime of approximately 5 years. The high development costs associated with these instruments were largely non-transferable to the next generation of hardware, because the available technology evolved too quickly.

By the turn of the millennium, personal computers offered a level of processing power that made the first *software-based* instruments feasible. Software-based instruments proved to be particularly useful for studies of radio pulsars, where a computationally intensive processing technique known as coherent dedispersion could be used to obtain optimal time resolution across an observed pulse profile (Stairs et al. 2000; Liu et al. 2006). Instead of performing complicated signal processing tasks in dedicated hardware, these software-based instruments rapidly transferred digitised data into the memory of a general purpose computer which then performed the necessary operations, using software created with a conventional high-level programming language.

Software-based instruments proved to be highly versatile because their processing algorithms could be changed by editing and re-compiling source code instead of re-building hardware. Unfortunately, to handle reasonable bandwidths in real time, clusters of several dozen computers were required. This led to overheads in data transport and general system administration. However, time spent on the development of software in a high-level language could be readily transferred to the next generation of hardware, making the instrument design process much more efficient.

FPGAs offer an alternative to traditional hardware-based and modern software-based instruments in that they provide compact and efficient hardware-based processing power that can be rapidly re-configured to perform a different task. This is particularly advantageous where space, power consumption, reliability or data transfer capacity exclude the use of general-purpose processors. Modern FPGAs contain dedicated hardware components designed specifically for digital signal processing, along with vast amounts of configurable logic and in some cases, embedded processors. These "system-on-chip" devices are particularly attractive for small projects or in situations where multiple signals must be combined prior to centralised processing (such as LOFAR[1], the MWA-LFD[2] and in the long term, the Square Kilometre Array)[3].

While it is generally accepted that modern FPGA technology will play a crucial role in the development of future radio telescopes, the skills required to make effective use of FPGAs are not widespread within the radio astronomy community. Although programming and software engineering skills are

[1] http://www.lofar.org
[2] http://www.haystack.mit.edu/ast/arrays/mwa/LFD
[3] http://www.skatelescope.org

valuable tools for any scientist, hardware design is often seen as one step too far and a job best left for engineering professionals. The evolution of modern hardware and software technology will continue to blur the boundaries between the two domains and it may not be long before scientists will be able to directly configure a hardware component using high levels of abstraction in much the same way that a computer can be programmed today. Several large-scale radio astronomy projects (including the Australia Telescope Compact Array[4] broadband upgrade) are making use of modern FPGAs. The Center for Astronomy Signal Processing and Electronics Research (CASPER)[5] at UC Berkeley is heavily involved in this field of research. A number of smaller pilot projects (such as the one described in this paper) have also reached the literature (Ferris & Saunders 2004; Stanko et al. 2005). These projects are crucial to encourage the uptake of FPGA technology. They illustrate the power, flexibility and relative affordability of FPGA devices, but they also serve to highlight potential pitfalls.

This paper describes in detail an ongoing technology development project at the University of Tasmania. In Sect. 2 we describe the complete development of a relatively simple FPGA-based instrument platform that provides real-time analysis capabilities for bandwidths up to 100 MHz (and potentially several times more). This system uses a custom-built FPGA circuit board (known as *TasPGA*) that accepts input from one or more analogue to digital (A/D) converters and can transmit processed output to a PC via Ethernet. In Sect. 3, we describe the development of a Fourier transform spectrometer capable of dividing a broad-band signal into 16 384 frequency channels. Section 4 presents some of the first data taken with this spectrometer at the Mt. Pleasant radio observatory. In Sect. 5 we describe some of the most useful experience gained during the project and finally summarise our major findings and key recommendations in Sect. 6.

2. Design considerations

2.1. Guiding principles

TasPGA was designed to be a re-configurable radio astronomy instrument capable of a large number of different tasks. Few specific requirements were placed on the design of the hardware, other than that it should run as fast as reasonably possible and maintain signal coherence throughout the processing pipeline. Our main goal was to develop a device that could demonstrate the capabilities of modern FPGAs and also perform some amount of useful science. TasPGA uses three Xilinx FPGA devices. The most powerful of these is dedicated to signal processing and is connected to a bank of digital I/O ports that have been optimised for rapid data transfer. The board also has a second FPGA with an embedded processor and an array of standard peripheral interfaces, used for administration and control. The third FPGA controls the routing of programming signals to the other two devices, and from one TasPGA board to the next when several are used together. TasPGA does not have a large bank of external memory and although this limits the capabilities of the instrument somewhat, it greatly reduced the complexity of the design.

2.2. Electronic design automation

Both the TasPGA circuit board and all FPGA firmware were designed using version 6 of the Altium Designer (hereafter AD6) suite of electronic design automation tools; a commercial package offered by Altium Ltd.[6]. AD6 integrates the hardware, firmware and embedded software design processes, eliminating many of the pitfalls traditionally associated with migrating from one domain to the next. However, AD6 is not particularly optimised for the design of high-speed digital signal processing pipelines.

The Xilinx ISE Foundation suite was used to synthesise schematic and hardware description language (HDL)-based FPGA logic circuits into firmware. This software was obtained at no cost through the Xilinx university program[7]. It interfaces seamlessly with AD6 to complete the design automation sequence.

2.3. Hardware design

The TasPGA board accepts data from one, two or three external sources along with a number of clock signals that can be used to synchronise all operations with an observatory time standard. A schematic overview of the board is provided in Fig. 1.

The printed circuit board (PCB) itself has 10 layers and was designed to fit inside a standard 19″ rack mount case. It has on-board power regulators and requires an external 5V supply. The PCB houses 3 FPGAs, but the small Xilinx Spartan 3 FPGA is used only as a configurable signal multiplexer/router. This device is responsible for the distribution of signals from an external Joint Test Action Group (JTAG, IEEE 1149.1) interface through to the other two FPGAs for programming, monitoring and debugging purposes.

The primary FPGA is a Xilinx Virtex 4 SX55 (hereafter V4). This device was selected for its large array of dedicated internal DSP48 multiply-and-accumulate blocks. These specialised hardware multipliers can be used for many DSP-related tasks and have been optimised to run at high speed. The V4 also has a large number of general purpose input and output (I/O) pins that can be used for rapid parallel data transfer. This FPGA is connected via matched-length lines to 3 separate 48-bit parallel digital interfaces that use low voltage differential signalling (LVDS). These interfaces each consist of 3 very high density cable interconnect (VHDCI) connectors, one of which carries JTAG and global clock signals while the other two carry 24 differential signal pairs and 8 additional clock inputs and outputs. These interfaces can be connected to an A/D converter or used to cascade several TasPGA boards in series. Boards linked in this way can all be programmed from a single interface.

The V4 is connected to a Xilinx Virtex II pro XC2VP7 FPGA (hereafter V2) via a 64-bit parallel bus. This FPGA has an embedded PowerPC 405 processor and is connected to a DB9-type RS232 serial port, a Universal Serial Bus v 1.1 port and a 100 Mb Ethernet port. Drivers for each of these interfaces can be synthesised out of configurable logic within the FPGA and connected to the embedded processor. Both FPGAs have 4 MB of static random access memory (SRAM) that can be clocked at speeds up to approximately 10 MHz. The V2 is also connected to 64 MB of synchronous dynamic random access memory (SDRAM) which can be clocked at 100 MHz.

[4] http://www.narrabri.atnf.csiro.au
[5] http://casper.berkeley.edu
[6] http://www.altium.com
[7] http://www.xilinx.com/univ

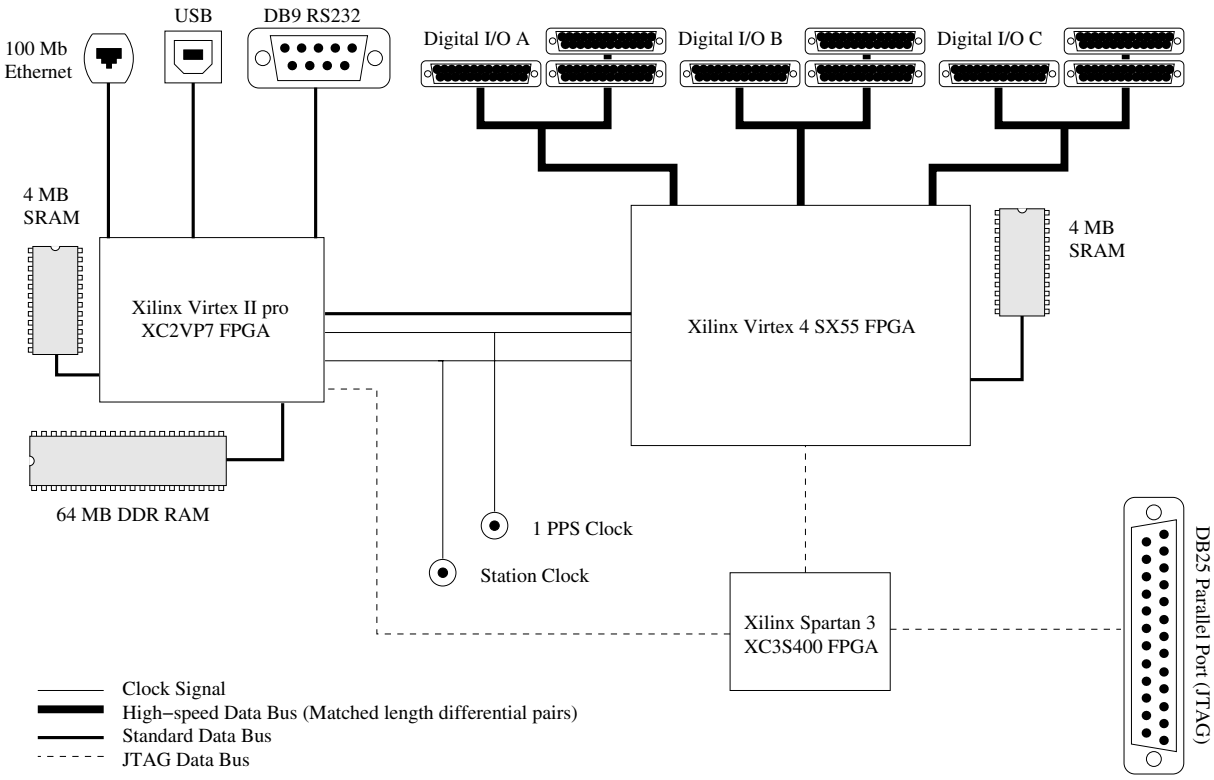

Fig. 1. Block diagram of all major components on the TasPGA circuit board.

Components within each FPGA can be clocked from a number of sources. The primary DSP components will typically be clocked from a signal that arrives via the same cable as the data (to ensure phase coherence). Two additional external clocks can be connected to dedicated on-board drivers via SMA-type connectors. These clock signals fan out to both the V2 and V4. One is intended to carry a "1 Pulse Per Second" signal that governs the timing of all events on the board. The other accepts any clock signal with a frequency up to 200 MHz. The PCB also has a tunable crystal oscillator that can be used to clock on-board components when no external clock signals are available.

The PCB was designed with the help of an engineer and fabricated by a third-party sourcing firm[8] that oversaw the purchase of all components. This company also arranged for fabrication of the PCB itself and the final assembly of each unit. 5 boards were manufactured in the first run and only a few, very minor problems were found on the top layer of the board after construction. These were easily rectified without any need for re-manufacturing.

2.4. Firmware design

A Typical FPGA firmware design flow starts with basic planning, followed by design entry (describing the required logic in a machine-readable form), simulation and synthesis. Design entry typically involves a mix of schematic-based input and the development of code written in one of several standard hardware description languages. This is usually followed by simulation of the design and iterative improvement until the firmware is ready to be synthesised and tested in hardware. Simulation of DSP networks is particularly challenging, because many operations are time-critical and it can be difficult to accurately predict the behaviour of the hardware. Expensive device-specific simulation tools are usually provided by FPGA vendors to overcome these problems, but we chose to experiment with an approach that bypassed the simulation stage entirely. AD6 provides the ability to insert "embedded instruments" into FPGA firmware. These instruments perform basic monitoring and control tasks (frequency counting, logic analysis, etc.) and their output can be read back in real-time from within the design environment. Using these embedded instruments to assist with the debugging process, we moved straight from the design entry stage to firmware synthesis and in-situ testing. Although this increased the number of design iterations required, it also reduced the cost and complexity of the EDA tool chain.

As mentioned before, AD6 is highly suited to the development of a complete system. It provides many useful driver modules for standard components and makes embedded software development very swift. However, this emphasis on system-wide development means that the precise optimisation of specific aspects of an FPGA logic path can sometimes be performed better outside of AD6. In order to optimise the performance of the signal processing firmware, some speed-critical components of our designs were synthesised specifically for the V4 FPGA using the Xilinx "Core Generator" software and then imported into AD6.

2.5. Selecting an A/D converter

TasPGA operates purely in the digital domain and must be coupled to a high-speed A/D converter. Atmel[9], National Semiconductor[10], Analog Devices[11] and Maxim[12] all make

[8] http://www.sourceman.com.au

[9] http://www.atmel.com

[10] http://www.national.com

[11] http://www.analog.com

[12] http://www.maxim-ic.com

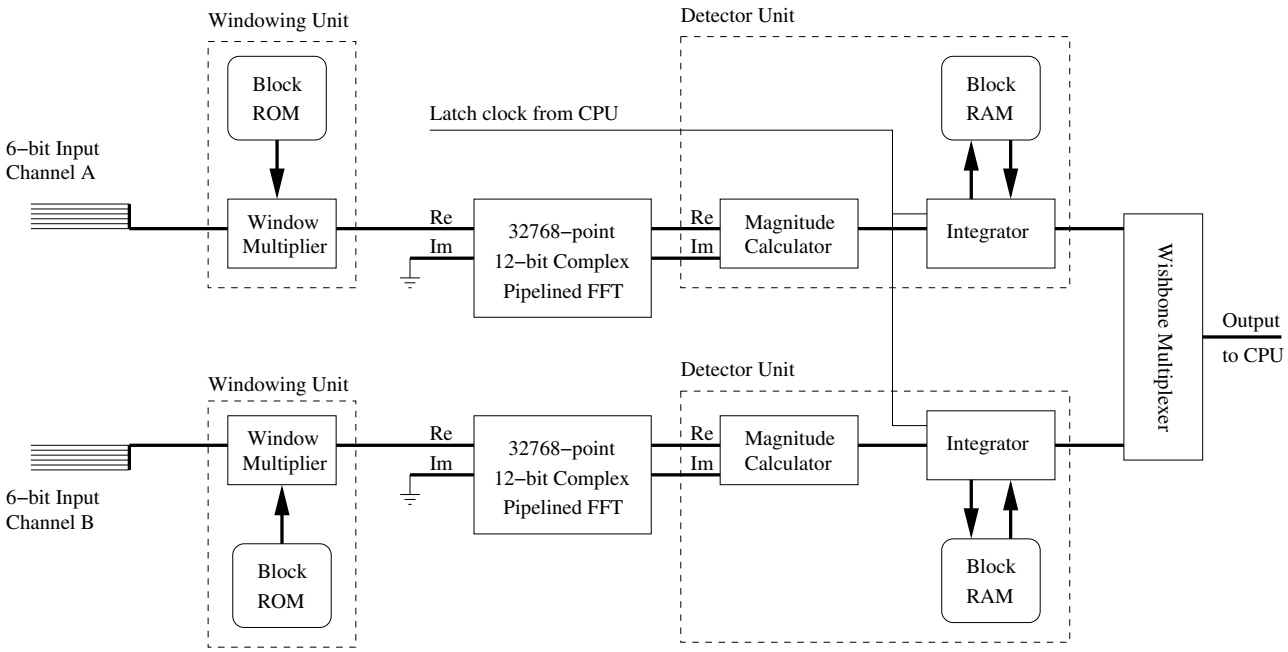

Fig. 2. Block diagram of the data path within the Virtex 4 FPGA.

suitable devices but we chose the MAX105 from Maxim. This device can operate at sampling rates of 800 Msps, providing a maximum bandwidth of 400 MHz at the Nyquist rate. The MAX105 has 6-bit quantisation precision and two analog input channels, allowing dual-polarisation observations with a single A/D converter. 6-bit precision should offer better dynamic range than the 2-bit precision typically used in radio astronomy instruments and can be purchased for a reasonable cost. A custom interface cable was constructed by dividing and re-terminating one end of a standard VHDCI cable with connectors that could be attached to the MAX105 evaluation kit output pins.

2.6. Embedded processing

One advantage of the TasPGA design is its ability to integrate seamlessly with existing computer systems. By using industry standard protocols like Ethernet and RS232, we avoid the need for any additional hardware or device drivers for a PC to communicate with the board. External commands can easily be sent to the embedded processor in the V2, which has access to any other sub-system on the TasPGA board. The first simplistic version of the TasPGA embedded software receives all incoming configuration commands via RS232, making use of a direct connection between the board and a host PC. Given appropriate software for the embedded processor, TasPGA is capable of operating autonomously in a networked environment, removing the need for a host PC altogether. Presently, outgoing data is packaged into User Datagram Protocol (UDP) packets for transport via Ethernet. These packets are fully routable and it is possible to connect TasPGA to a network switch and simply specify the internet protocol (IP) address of the destination PC using the embedded software. For simplicity, the Ethernet device is currently dedicated to outgoing data. This link could easily be made bi-directional with some additional software development.

Because the V2 is likely to perform similar tasks regardless of the signal processing algorithms implemented on the V4, we have developed a generic V2 firmware module that instantiates a 32-bit embedded processor and connects it to an array of sensors, I/O interfaces and user-defined control lines that fan out to inputs on the V4. In this way, it is usually possible to leave the V2 firmware in place and make small changes to the code running on the embedded processor whenever a new signal processing engine is installed on the V4. This greatly reduces the work required to develop new signal processing algorithms and makes the system easier for new developers to approach.

3. A Fourier transform spectrometer

3.1. Data pipeline

The power spectrum of an astronomical signal is often of interest and is relatively easy to compute with digital hardware using the Discrete Fourier Transform (DFT) or, with greater computational efficiency, the Fast Fourier Transform (FFT) described by Cooley & Tukey (1965). To demonstrate the capabilities of TasPGA, we developed an FFT spectrometer capable of dividing up to 100 MHz of bandwidth into 16 384 individual channels. The firmware was designed using a minimalistic approach and incorporated only those components required to build a functional spectrometer. All signal processing tasks are performed by the V4 and the number of frequency channels is fixed to the largest number that could be stored in the V4. It is therefore necessary to trade frequency resolution for bandwidth when planning an observation. A schematic diagram of the data path through the V4 is shown in Fig. 2.

Samples are clocked into the FPGA by the A/D converter's "data ready" signal, which is transmitted along the same cable as the data themselves. Within each channel, the MAX105 outputs two consecutive samples on every "data ready" rising edge at half the actual sampling rate. In theory, this enables parallel processing by trading an increase in bit width for a reduction in clock speed. Our present firmware is not able to take advantage of this feature and we run the A/D inputs through a multiplexer within the V4 to generate sequential time samples at twice the input clock rate (effectively stitching the signal back together).

To avoid contamination of the output spectrum by edge-effects resulting from the finite FFT length, it is essential to

apply a window function to the raw data. By down-weighting the points close to the edge of each Fourier transform "frame", unwanted harmonic distortion can be minimised. We used FPGA block memory to store a 32768-point "Hann" window with 6 bits of precision per point. The input index of the FFT is passed as an address to the window storage buffer and the retrieved value is then multiplied by the current sample each clock cycle, resulting in 12-bit wide inputs to the FFT unit. The FFT is computed using a 32 768-point complex pipelined streaming FFT core. We used version 4.1 of the Xilinx FFT Core Generator to produce a "black box" netlist for the FFT engine. This was incorporated into AD6 as a custom library component, thus bypassing the need to develop our own FFT engine.

The FFT core is clocked at the (Nyquist) sampling rate and produces one output for every input sample. Because the signal is real-sampled, the imaginary inputs of the complex FFT are tied to ground. At the present time, the Xilinx Core Generator software cannot produce an FFT core designed to accept purely real-sampled data. The use of a full complex-to-complex FFT results in the waste of some FPGA resources because half of the output points contain redundant information and are therefore discarded.

The amplitude of the complex output of the FFT core is computed every clock cycle and sent to a pipelined integrator unit that retrieves the accumulated total for the given channel from memory, adds the current amplitude and then writes the new total back into memory. All output indices greater than half the transform length are simply ignored to avoid the storage of redundant information. Dual-port memory within the FPGA is used to store the integrated values as it supports simultaneous read/write operations. Each integrated value is stored as an unsigned 32-bit number, allowing several seconds of data to accumulate before numerical overflow occurs. An external latch clock is used to drive a reset system that synchronously copies the integrated values into a secondary storage buffer for safekeeping while initialising the working memory for the next integration cycle. When using the dual-port memory buffer in read-before-write mode, there is no dead-time associated with the latch-reset operation as each total can be copied out and overwritten with the next input sample during the same clock cycle. The secondary storage buffer for each channel is connected via a multiplexed Wishbone[13] bus to the embedded processor on the V2. The embedded processor receives an interrupt at the end of every integration cycle, causing it to execute a routine that reads out the accumulated data from the storage buffers on the V4, then formats the values into UDP packets which are transmitted to a specified IP address via 100 Mb Ethernet.

In order to monitor the performance of the A/D converter, the V4 also accumulates a histogram of the number of samples falling in each of the $2^6 = 64$ A/D levels. One in every 4 samples are skimmed from the incoming data and the total number falling into each A/D level is accumulated in a small amount of on-chip memory. The same latch clock that governs the integration time of the power spectrum is used to reset the histogram accumulator after transferring the result to a secondary storage buffer. The accumulated histograms are read by the embedded processor on the V2, which transmits the statistical data to the host PC for display. In this way, the power level of the input signals can be adjusted in real time for optimal digitisation.

[13] http://www.opencores.org.uk/projects.cgi/web/wishbone

3.2. User interface

UDP packets transmitted by the spectrometer can be routed through a local area network (LAN) or even the Internet. Currently, only the spectra (and sampler histograms) are transmitted in this way but there is no reason why other status information and control commands could not also be sent or received via Ethernet in the future. This would allow the instrument to be driven entirely autonomously or controlled from a remote location.

Presently, the observer interacts with the spectrometer via a graphical user interface (GUI) that runs on a host PC which is connected to the board via an RS232 serial line. The GUI was written in C++ using Trolltech's Qt library[14] and the Qwt scientific plotting package[15]. It displays the current integration time (which can be set via a programmable clock divider within the V4) and the FPGA core temperatures. The GUI also opens a network socket that accepts packets from the spectrometer and plots the data in real-time. Each integration is written to an SDFITS (Garwood 2000) file along with header parameters that record crucial information. These can be specified either from the GUI itself or via another network socket for automatic control. The SDFITS files can then be processed using the ATNF Spectral Analysis Package[16] (ASAP) or any other software compatible with the SDFITS standard.

3.3. Limiting factors

The maximum bandwidth of the input signal is limited by the speed at which data can flow through the processing pipeline. Without manual editing of the V4 floor plan, the maximum bandwidth is just over 100 MHz. It may be possible to improve this by optimising the placement of core components within the FPGA but this has not yet been attempted. It is possible to run the A/D converter (and hence the V4) much more slowly and thus increase the spectral resolution by sacrificing bandwidth. However, at present it is necessary to present a band-limited signal to the A/D converter to prevent aliasing. In future it would be desirable to always run the A/D converter at the maximum rate and apply a digital filter to band-limit the signal.

The rate at which all data from one integration cycle can be transmitted over Ethernet introduces a lower bound on the integration time. The embedded processor must transmit $2 \times 16\,384 \times 32$ bits (plus the A/D histograms) over an Ethernet interface theoretically capable of 100 Mb/s. This should be possible in roughly one hundredth of a second but lab tests show that the whole transmission process actually takes slightly more than one second. The embedded processor in the V2 is only clocked at 10 MHz to allow direct use of the SRAM connected to the FPGA and this may contribute to the poor Ethernet performance.

The amount of dual-port block RAM (BRAM) available on the V4 directly limits the achievable spectral resolution and is the only FPGA resource that is almost fully utilised by the spectrometer firmware. The data accumulators use the majority of the available BRAM, but each FFT core also requires a significant amount. In total, roughly 80% of the available BRAM is used in the current firmware. The available memory currently limits the maximum number of frequency channels. It would be possible to double the number of channels by doing away with the secondary storage buffer used to latch the integrated data. However,

[14] http://www.trolltech.com/qt
[15] http://qwt.sourceforge.net
[16] http://www.atnf.csiro.au/computing/software/asap

this would introduce an amount of dead-time equal to that required for the embedded processor to read in the accumulated data at the end of the integration cycle. In future, if additional channels are required, it may be possible to link several boards and divide the observing band in stages, rather than using one large transform.

Given the inherent linearity of the Fourier transform, it should be possible to greatly increase the bandwidth of the spectrometer by utilising parallel processing, especially given that the current firmware uses only a few percent of the total logic available on the V4. Incoming data can be divided into two or more parallel pipelines on input to the V4 by switching the destination of each successive sample and wrapping back to the first pipeline after the last one has been reached. These parallel streams can then be transformed independently, after which the results can be combined to form an overall spectrum. Since each parallel pipeline has an effective sampling rate that is reduced by a factor equivalent to the number of pipelines, the most complicated elements of the firmware can be clocked much more slowly than the incoming data rate. This in turn should allow the processing of larger bandwidths and will therefore be an important area for future development.

4. Observations

4.1. Susceptibility to interference

The radio frequency emissions generated by modern communications infrastructure often impact on radio astronomy. In particular, it can be necessary to observe an astronomical spectral feature in the presence of strong, locally generated narrow-band interference. In these cases, it is crucial to ensure that the interfering signal does not contaminate the (much weaker) astronomical signal. One factor to consider is the amount of instrumental isolation between each spectrometer channel. Traditionally, auto-correlation spectrometers have been used in radio astronomy because fast correlators could be implemented relatively easily in hardware. An auto-correlation spectrometer computes the power spectrum of a signal using the Wiener-Khinchin theorem (Bracewell 1965), which states that the power spectrum is the Fourier transform of the auto-correlation function. The auto-correlation function can be computed more easily in hardware and integrated for several seconds before being passed to a computer that can perform the Fourier transform at a much-reduced rate. In the simplest case, auto-correlation of a single-frequency (sinusoidal) signal produces a sinusoidal output, which generates a corresponding peak in the output of the Fourier transform. However, if any harmonic distortion or other systematic errors are introduced into the auto-correlation function, an interfering signal can be inadvertently spread throughout the entire observing band.

Using modern electronics, it is possible to perform a Fourier transform in hardware without the intermediate auto-correlation stage. For reasons of simplicity, a pure FFT spectrometer should be less susceptible to strong, narrow-band interference than an auto-correlation spectrometer. One of the main improvements that can be made is in the number of bits used to quantise each sample during the analogue to digital conversion stage. TasPGA has been tested with a 6-bit sampler, but would work equally well with 8 or more bits per sample (although more resources would be used on the V4). The additional dynamic range associated with multi-bit sampling is especially valuable in the presence of interference as it helps to isolate the effects of a strong signal. It is possible to isolate individual channels to a greater

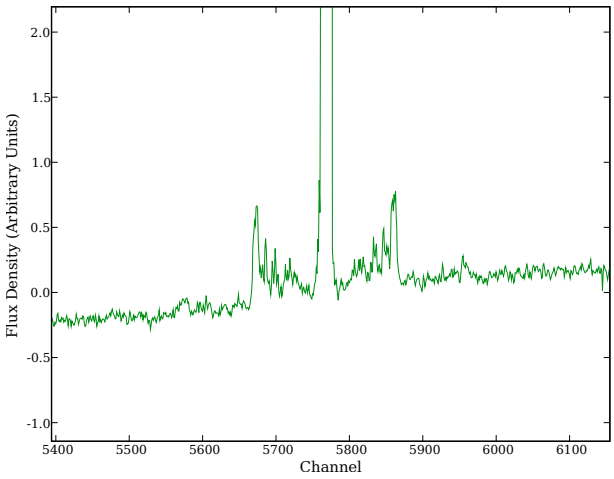

Fig. 3. Spectrum of a broad-band noise source with a strong single-frequency tone injected. The axes have been restricted to the region surrounding the base of the tone where some amount of leakage can be observed.

degree using a polyphase filterbank (Vaidyanathan 1990) instead of a pure FFT, but the extra complexity would likely reduce the maximum number of frequency channels available. This trade-off will be an important consideration for future instrument design.

To test the dynamic range and channel isolation of the TasPGA FFT spectrometer in the presence of strong interference, we generated flat-spectrum noise in the laboratory and introduced a strong, single-frequency tone using a radio frequency signal generator. The strength of this tone was increased until it just started to distort the A/D conversion statistics. After integrating for one hour, the tone extended 65 dB above the root mean square (rms) noise level. Figure 3 shows a closeup of the spectrum near the base of the artificial tone. Clearly, the channels on either side of the tone have been contaminated by leakage that is approximately 14 dB above the noise rms at worst. We conclude that even in the region surrounding a very strong source of interference, the level of contamination should be at least 50 dB below the level of the interference after an hour of integration. Further from the artificial tone, there was no noticeable contamination of the spectrum.

As an additional test, we obtained a sky spectrum covering 100 MHz centred on 1.45 GHz (Fig. 4). This spectrum was recorded while the telescope was pointed at the zenith.

The Mt. Pleasant radio telescope is located 12.5 km from the city of Hobart and only 6.5 km from Hobart Airport. Consequently, there are many strong sources of interference in the spectrum, the most powerful of which is at least 30 dB above the rms of the baseline noise. After two hours of continuous integration, the baseline of the spectrum is almost completely dominated by interference and leakage. It is highly unlikely that the few small remaining sections of relatively clear bandwidth would be useful to astronomers. Given the number of interfering sources, it is unclear whether the situation could be improved simply by increasing the isolation between individual channels.

4.2. Deep integration tests

Because most signals of interest to radio astronomers are very weak, it is often necessary to observe for several minutes or hours and average the data to help extract a signal from background noise. If we assume that background noise is generated

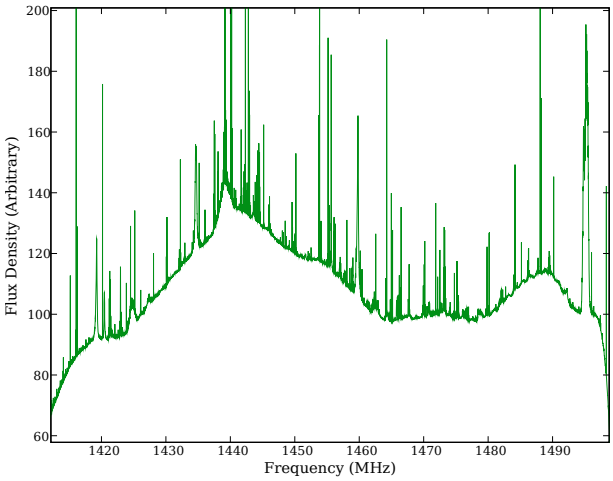

Fig. 4. Zenith spectrum centred on 1450 MHz and spanning 100 MHz, integrated for 2 h with dual orthogonal polarisations combined. The strongest sources of interference extend well above the vertical range of the plot, which has been condensed to better display the baseline.

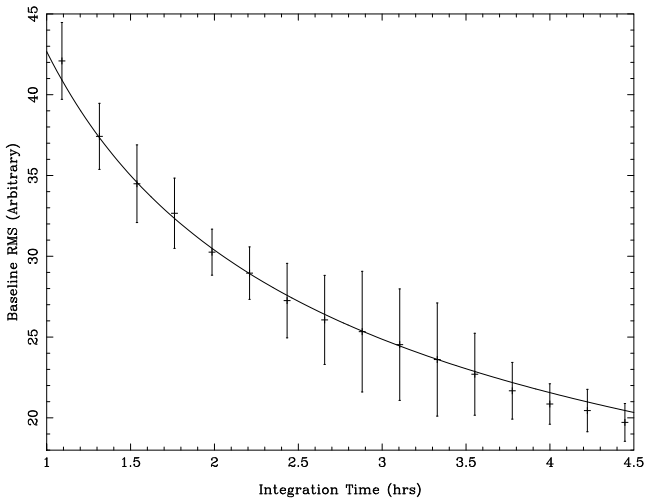

Fig. 5. Ensemble average baseline rms as a function of time for a deep integration of an artificial noise source. Each individual sub-integration was 1.7 min in duration and 8 were averaged to form each data point. The total integrated time for each point is shown on the horizontal axis. Each point and its uncertainty were calculated from an ensemble of 4 test observations, each 4.5 h in duration. The ideal theoretical trend is drawn as a solid line and tracks the measured points to within the error estimates.

by random processes, its amplitude should increase as the square root of time (with continued integration) and our signal amplitude should increase linearly (if the signal is persistent). Overall, the S/N ratio should increase as the square root of time. However, if there are any non-random components present in the background noise, this will not be the case. In order to test the noise floor of the TasPGA FFT spectrometer, we observed an approximately flat-spectrum noise source for a period of 20 h, recording an integrated spectrum every 1.7 min. We calculated the rms of a portion of the baseline noise in the first integration and then repeated the process after averaging successively more integrations. In order to obtain an estimate of the uncertainty in this process, our data were broken into 4 equal-length segments, each approximately 4.5 h in duration. Baseline rms points were calculated for each data segment and averaged. The standard deviation of equivalent data points in each of the four segments was used as an estimate of the uncertainty in the measurement.

Figure 5 shows that the data do indeed follow the expected trend of baseline rms decreasing with the square root of time, up to almost 5 h of continuous recording. Treating the ensemble as a single continuous trial, we confirm that the noise integrates down as expected over periods in excess of 18 h (Fig. 6).

In the case of Fig. 6, the data are dominated by a random walk above and below the expected trend. Performing the same ensemble averaging technique as used for Fig. 5 would provide a better estimate of the true system performance, along with the uncertainty in each data point. Unfortunately, the amount of recording time required to perform such an ensemble average is prohibitive. For most sources (except those that are circumpolar) and telescopes, 16 h can be considered a reasonable upper limit on the time available for a single continuous observation.

4.3. Maser observations

As a final test, we used the TasPGA spectrometer to observe two astronomical methanol masers with the Mt. Pleasant 26 m radio telescope at a sky frequency of 6.668 GHz, where the system temperature is roughly 1000 Jy. In both cases, a total bandwidth of 8 MHz was divided into 16 384 spectral channels for a frequency resolution of 488.3 Hz. The first maser, G263.25+0.52, has a measured peak flux density of 57.3 Jy (Schutte et al. 1993) and was easily detected in a 4 min observation (see Fig. 7).

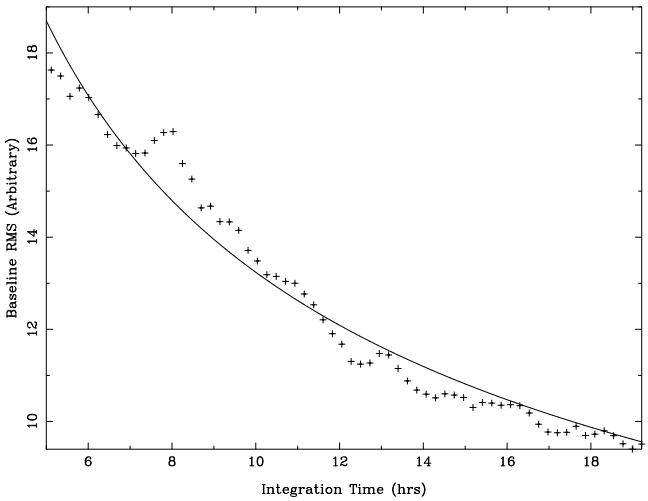

Fig. 6. Baseline rms as a function of time for a deep integration of an artificial noise source. Each individual sub-integration was 1.7 min in duration and 8 were averaged to form each data point. The total integrated time for each point is shown on the horizontal axis. The ideal theoretical trend is drawn as a solid line.

Although no explicit flux calibration was performed at the time of the observation, we can estimate the measured flux of the source using the radiometer equation and a typical value of the system temperature of the Mt. Pleasant 6.7 GHz receiver (approximately 1000 Jy). The peak of the maser emission corresponds to a flux density of 53 Jy, which is similar to the measurement made by Schutte et al. (1993).

The second maser, G285.35+0.00, has a measured peak flux density of 10.0 Jy (van der Walt et al. 1995) and was similarly well detected in an 11 min observation (see Fig. 8). In this case, our estimated peak flux density is 8 Jy, similar to the measurement made by van der Walt et al. (1995).

In both cases, the morphologies and velocities of the spectral components match well with the data presented in the respective

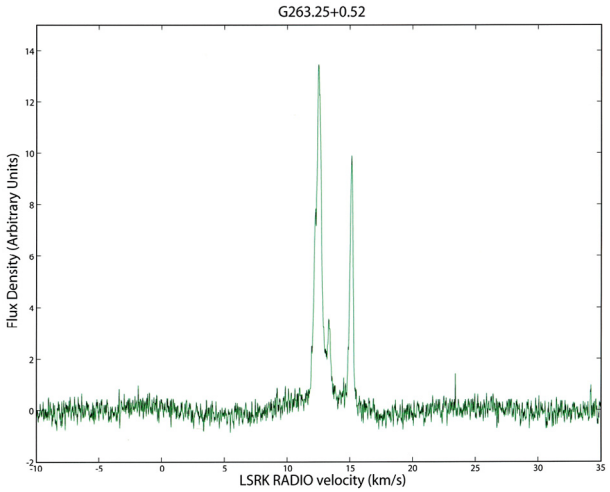

Fig. 7. TasPGA spectrometer output from a 4 min observation of the Methanol maser G263.25+0.52, with a peak flux density of 57.3 Jy (Schutte et al. 1993).

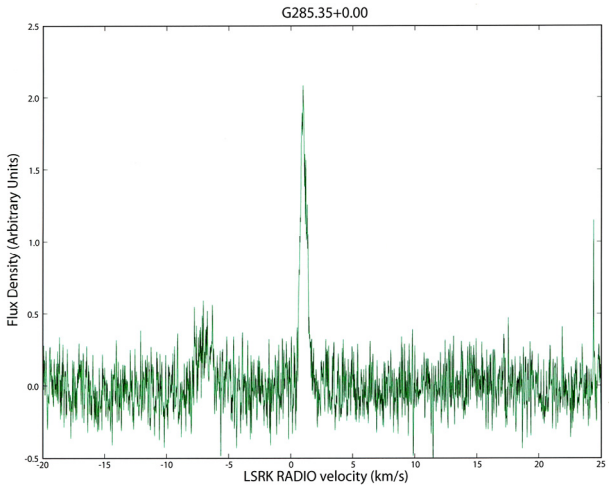

Fig. 8. TasPGA spectrometer output from an 11 min observation of the Methanol maser G285.35+0.00, with a peak flux density of 10.0 Jy (van der Walt et al. 1995).

discovery papers (Schutte et al. 1993; van der Walt et al. 1995), leading us to believe that the TasPGA spectrometer is functioning correctly.

5. Discussion

Although the complete evolution of TasPGA from concept to deployment took approximately 2 years (including hardware development time), the development of the FFT spectrometer occupied less than 500 person-hours of time. Once examples of functional firmware for the V2 and V4 had been developed, new signal processing algorithms could be implemented in a matter of weeks or months, depending on their complexity. In an age of readily available and powerful consumer electronics, we have shown that it is possible to design and build a new instrument with only limited engineering support. Once the initial hardware design phase is complete and a set of "template" firmware containing key device drivers has been developed, FPGA-based instruments can be modified remarkably quickly. Even when the FPGA hardware becomes obsolete, it is likely that firmware already developed could be transported to the next generation of hardware with minimal modification. It can still be difficult to convert a signal processing algorithm into an equivalent logic circuit, however modern electronic design automation tools are constantly improving and this also reduces the amount of time spent in the development phase.

Although the skills required to write FPGA firmware are typically taught during an engineering degree, they are also highly complementary to the basic electronics knowledge gained during many experimental physics courses. In the coming years it will be important to spread these skills throughout the scientific community to foster a better understanding of the capabilities of the next generation of instruments.

6. Conclusion

We have used TasPGA to demonstrate that good results can be obtained with a simple FFT-based spectrometer, especially when sampling with 6- (or more) bit resolution instead of the typical 2-bit resolution used in many radio astronomy instruments. Useful spectra have been obtained in wide-band (100 MHz total bandwidth) and narrow-band (8 MHz total bandwidth) modes, allowing better characterisation of the radio frequency interference environment at the Mt. Pleasant radio telescope, and the re-detection of two astronomical Methanol masers. Although TasPGA was designed to be capable of new science, it is also a useful platform for students and researchers wishing to experiment with FPGA-based signal processing methods.

Acknowledgements. The author would like to thank Brett Muir for his work on the PCB design and, along with John Russell and Chris Weimann, for offering useful logic design advice. Eric Baynes and Brett Reid contributed to the integration of TasPGA into the Mt. Pleasant radio observatory and Sally Long contributed to firmware development and testing. We would also like to thank Altium Ltd. and Xilinx Inc. for their donation of EDA tools and prototype development boards to the University of Tasmania. This research was also supported by an Australian Research Council grant (DP 0559613) to the University of Tasmania.

References

Bracewell, R. 1965, The Fourier, Transform and its Applications (New York: McGraw-Hill)
Cooley, J. W., & Tukey, J. W. 1965, Math. Comput., 19, 297
Ferris, R. H., & Saunders, S. J. 2004, Experimental Astron., 17, 269
Garwood, R. W. 2000, in Astronomical Data Analysis Software and Systems IX, ed. N. Manset, C. Veillet, & D. Crabtree, ASP Conf. Proc., 216, 243
Liu, L.-Y., Ali, E., & Zhang, J. 2006, Chin. J. Astron. Astrophys. Suppl., 6, 53
Schutte, A. J., van der Walt, D. J., Gaylard, M. J., & MacLeod, G. C. 1993, MNRAS, 261, 783
Stairs, I. H., Splaver, E. M., Thorsett, S. E., Nice, D. J., & Taylor, J. H. 2000, MNRAS, 314, 459
Stanko, S., Klein, B., & Kerp, J. 2005, A&A, 246, 391
Vaidyanathan, P. P. 1990, Proc. IEEE, 78, 56
van der Walt, D. J., Gaylard, M. J., & MacLeod, G. C. 1995, A&AS, 110, 81