Hindawi Publishing Corporation
EURASIP Journal on Applied Signal Processing
Volume 2006, Article ID 34653, Pages 1–14
DOI 10.1155/ASP/2006/34653

An FPGA-Based MIMO and Space-Time Processing Platform

J. Dowle,[1] S. H. Kuo,[2] K. Mehrotra,[1] and I. V. McLoughlin[1]

[1] *Group Research, Tait Electronics Ltd, 535 Wairakei Road, P.O. Box 1645, Christchurch, New Zealand*
[2] *School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada V5A 1S6*

Received 29 November 2004; Revised 23 June 2005; Accepted 30 June 2005

Faced with the need to develop a research unit capable of up to twelve 20 MHz bandwidth channels of real-time, space-time, and MIMO processing, the authors developed the STAR (space-time array research) platform. Analysis indicated that the possible degree of processing complexity required in the platform was beyond that available from contemporary digital signal processors, and thus a novel approach was required toward the provision of baseband signal processing. This paper follows the analysis and the consequential development of a flexible FPGA-based processing system. It describes the STAR platform and its use through several novel implementations performed with it. Various pitfalls associated with the implementation of MIMO algorithms in real time are highlighted, and finally, the development requirements for this FPGA-based solution are given to aid comparison with traditional DSP development.

1. INTRODUCTION

Most papers describing a MIMO-related subject are prefaced by the words "in a richly-scattering environment." Other phrases that can be found include "in the absence of noise" or "assuming perfect synchronization." Still more papers do not even acknowledge such caveats, and yet these phrases have been found to collectively describe some of the major challenges faced when designing a practical working MIMO system. One particular example is the assumption of AWG noise only when performing channel estimation from training data. Generally BER against SNR simulation curves are plotted for data decoded by the channel estimates. In reality, time averaging in a practical implementation is unlikely to be sufficient for the noise power to smooth out, and thus local noise excursions will have an impact on channel estimation accuracy, and that impact is proportional to the noise power. The widely shown BER against SNR curves for such systems (which collectively describe almost any implemented system) therefore ignore an important SNR-dependent factor which can skew performance results.

This paper is primarily concerned with the challenges of MIMO and ST implementation within a baseband signal processing context. A more immediate challenge than the realism of academic MIMO research models is in the very nature of MIMO algorithms themselves; that they comprise some of the more computationally complex problems that face contemporary wireless system designers.

The STAR (space-time array research) platform was designed by Tait Electronics to allow it and its international research partners to explore novel MIMO algorithms, not just through simulation and theory, but through practical working systems. The design team set a task to build a flexible platform that would be capable of a 20 MHz RF bandwidth at a carrier frequency centred on 2.45 GHz, and deliver 12 channels of simultaneous and continuous transmit and receive data, in addition to having baseband signal processing facilities capable of executing MIMO algorithms in real time. The actual algorithms were not specified at the design stage.

Section 2 outlines and analyzes the approach taken to satisfy such open-ended system requirements, whilst Section 3 describes the first three novel algorithms developed for the STAR platform. Section 4 illustrates various implementation issues and their solution within the STAR platform, and Section 5 analyzes the success of the techniques employed through a determination of development, cost, and effort against project deliverables. Section 6 then concludes.

2. THE STAR PLATFORM

Given the requirement to build a platform capable of performing complex MIMO-related processing for up to 12 channels of RF with up to 20 MHz bandwidth, it is evident that the processing scope is unbounded. At the time of design (mid-2002), there was very little published information concerning the complexity of MIMO algorithms. The pragmatic

approach was to source world's largest and world's fastest processing componentry and utilise this in such a way that modular expansion is possible.

2.1. Raw data bandwidth

By contrast, bounds could be placed on sample rate and estimated conversion precision, and this allowed a measure of maximum data throughput in such a system. In fact, a 60 MHz sample rate was adopted with 12/14-bit conversion precision limited by available devices. This meant a peak bidirectional data throughput of 10.8 Gbps for 12-channel I/Q after a decimation-by-two.

It was firstly evident that a single digital signal processor (DSP) would not be capable of meaningfully processing such data flow, and was secondly evident that physical means of transporting such amounts of data are problematic. It therefore becomes necessary to subdivide the problem into smaller blocks. 4-channel blocks were found suitable since the peak data throughput would then be 3.36 Gbps, which is conveyable between modules using paralleled low-voltage differential signalling (LVDS) connections. A single field-programmable gate array (FPGA) was capable of handling the peak data throughput within each 4-channel block, performing a decimation, and supporting data communications at 3.36 Gbps using built-in LVDS drivers. Given bidirectional data communications, a 12-channel system was achieved with oversampled raw data interchange between several FPGAs given the caveat that each data path conveyed no more than 4-channels worth of 30 MHz I/Q data.

This led to the modular and expandable architectural format shown in Figure 1 for a 4-channel variant, and shown in full in Figure 2 with specification shown in Table 1. This system is capable of processing, down to baseband outputs, the data generated by 12 receive channels, and simultaneously generating 12 transmit channels from baseband input. These data chains included MIMO and space-time block-coding algorithms.

2.2. Signal processing

At the time of system design, a very rough estimate of complexity was given for a 2-channel Alamouti [1] implementation of 3 billion multiply-accumulate calculations per second [2]. Given that a 12-channel system was being constructed from three 4-channel modules, and that Alamouti is generally considered to be relatively simple, computational capabilities of each STAR module were required to significantly exceed this if such modules were expected to be able to perform meaningful processing.

Dedicated DSP processors have traditionally been used for wireless baseband processing. A survey of available devices as per [2], updated here, reveals clock rates of up to 1 GHz. Leading edge DSPs contain multiple independent multiply-accumulate (MAC) cores, with Texas Instruments TMS320C6416T series device being capable of up to 8000 16-bit MMACS(million multiply accumulates per second). Analog devices compete with the TS201SABP TigerSHARC capable of achieving 4800 MMACS. The TS210S performs a maximum of eight 16-bit MAC operations per 600 MHz clock cycle. Both were the fastest devices in their class at the time of analysis.

The figures mentioned are for 16-bit calculations only: they are not necessarily representative of the full picture. For example, the ′C64 device mentioned also achieves up to 5760 8-bit MMACS. Both devices have various signal-processing related accelerators built in. However the MMAC and other figures are peak values: whether these are achievable depends very much on software structure, other concurrent operations, and the requirements for external memory. Nevertheless, the figures do indicate a generous upper bound on the fastest processing capability advertised by the two leading DSP manufacturers.

It is evident that both device are capable of a peak processing speed of the approximately required 3 billion calculations per second but do not "sufficiently exceed this." A more detailed analysis reveals problems of memory bandwidth and input-output bus bandwidths that would effectively prevent the devices from handling the large data throughput required without careful design of supporting hardware. Such supporting hardware would probably be best achieved using a reprogrammable device such as an FPGA.

Focussing on FPGA devices revealed the potential for performing all calculations in FPGA. A brief survey of contemporary FPGA devices reinforces this conclusion.

The biggest and fastest FPGA devices currently include the StratixII EP2S180 FPGA from Altera with 179 400 logic elements (LEs) and 96 DSP blocks each capable of 4 MACs at up to 420 MHz when paired to support 18-bit operation.

In this device, use of the DSP blocks alone delivers up to 161 280 MMACS even when none of the built-in logic element resources are reserved for processing. If a proportion of the 179 400 logic elements (LEs, each containing a look-up table and flip-flop) is also used to implement parallel MAC functions, 962 multipliers can be created (given in Altera's data sheets as "soft MACs"). Assuming that these operate at a slower frequency of 180 MHz (which is the practical upper limit observed by the authors for implementation of distributed filters using soft MACs), another 173 160 MMACS are available for use. It is of course unrealistic to assume that the entire FPGA can be utilised as dedicated MACs, but allowing 25% unusable capacity for these would mean that over 290 000 MMACS are available in total.

The largest Xilinx FPGA, the Virtex-4 series XC4VSX55, has 55 295 logic cells, 512 embedded "XtremeDSP" slices each capable of a single 18×18 multiply, and operates at up to 500 MHz (256 000 MMACS). Scaling for density on the same Altera quoted soft-MAC construction density, up to 296 multipliers could be created from the logic cells. If operated at 180 MHz, this provides another 53 280 MMACS. With a 25% assumed overhead, a total of over 290 000 MMACS are available in this device.

Since an FPGA was required for interfacing, and provided a theoretical processing capability far in excess of a

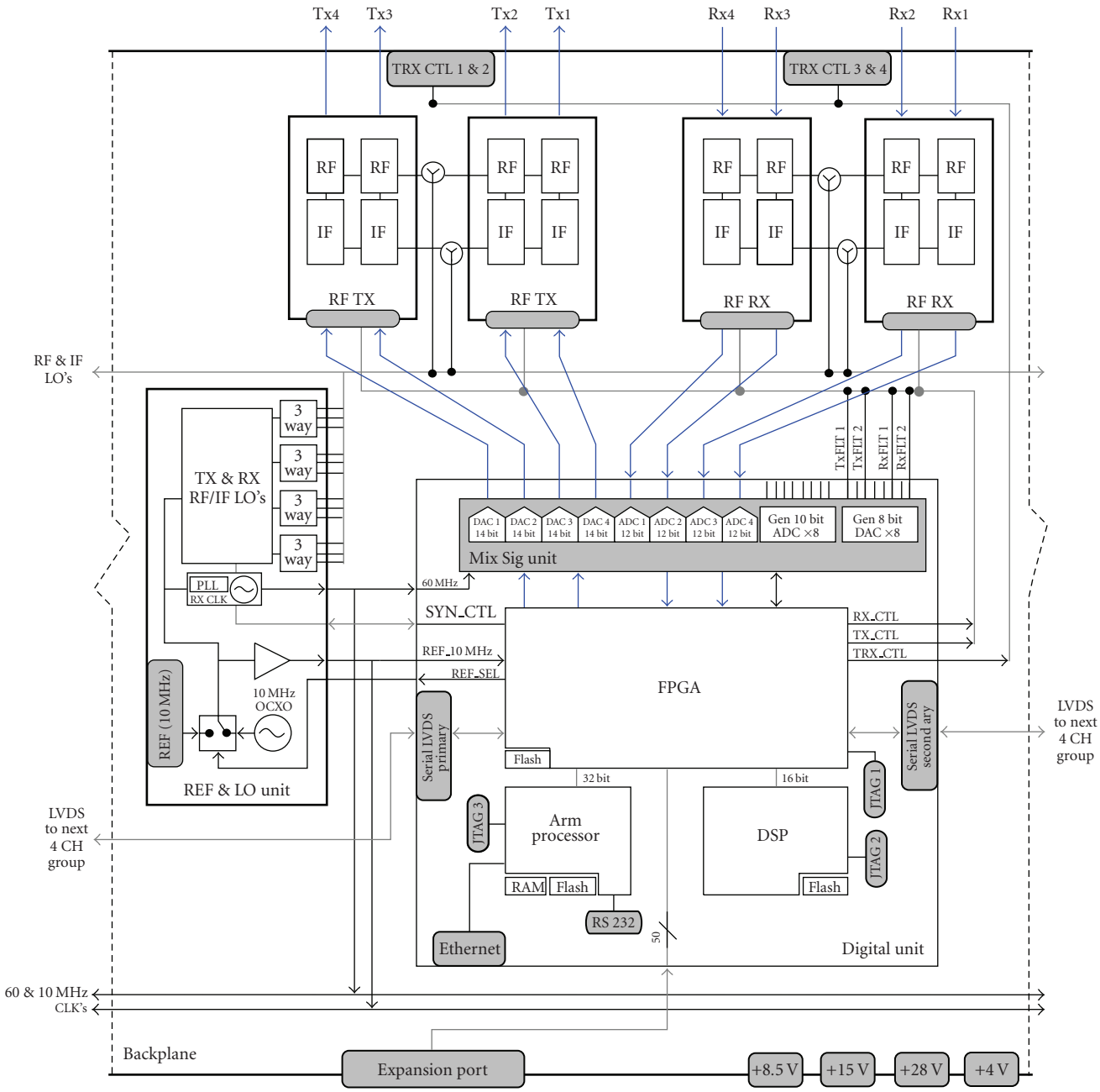

Figure 1: STAR platform in an early 4-channel configuration, showing some of the details of the system architecture.

DSP, the STAR platform was designed such that the majority of baseband processing would be performed by FPGA, with additional FPGA devices provided for front-end sample handling. For experimental and comparative purposes, provision was made for the current fastest DSP processor to be also present on each of the baseband processing boards, although on later board revisions this was removed as unnecessary and replaced with two further FPGAs. There are thus three per PCB, a total of nine FPGAs per 12-channel platform.

2.3. System architecture

A dual conversion approach was chosen for the RF sections of the system and the overall system architecture constructed, as shown in Figures 1 and 2. It can be seen that there are three processing slices each capable of four bidirectional RF channels and a large degree of baseband signal processing.

An oven-controlled crystal oscillator (OCXO) with better than 0.2 PPM (parts per million) drift accuracy provides a stable reference frequency, and a flexible software

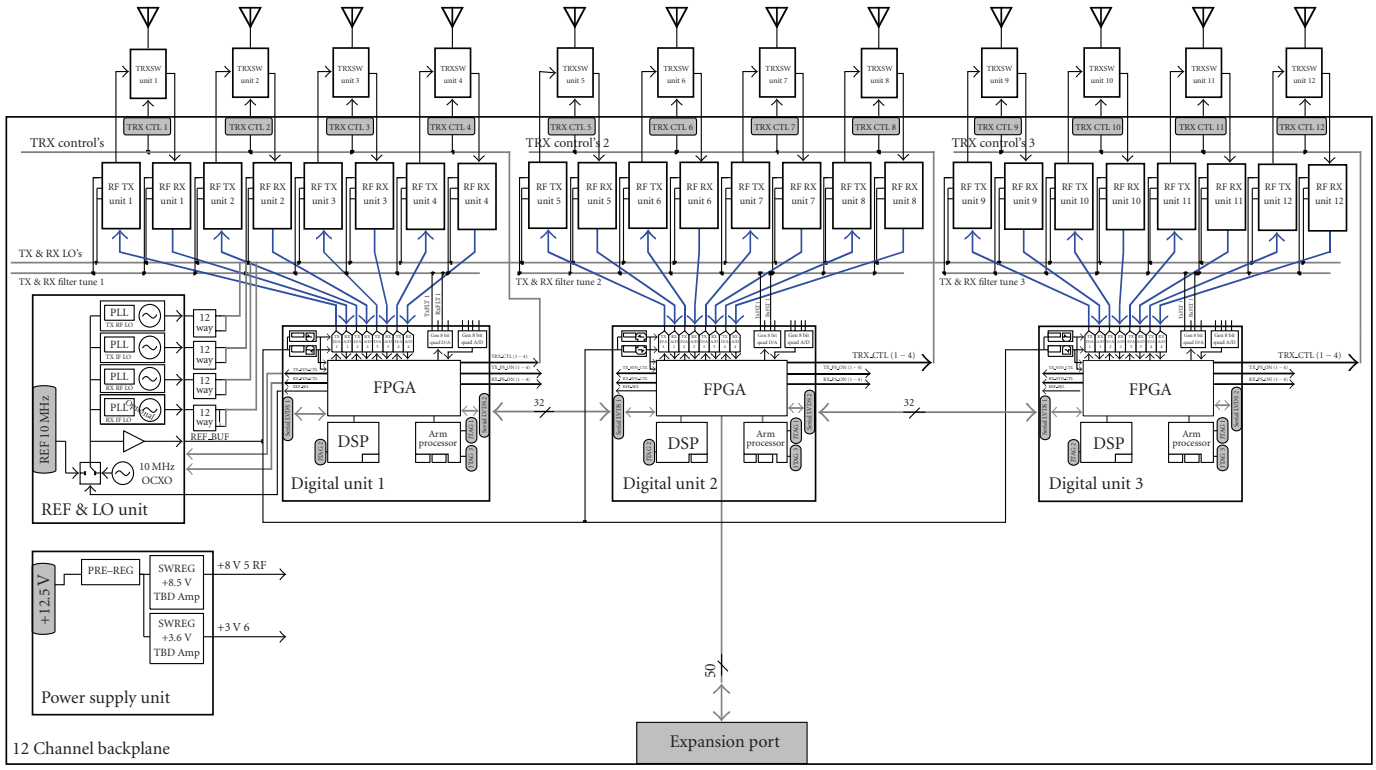

Figure 2: The initial STAR platform system architecture.

Table 1: STAR platform specifications.

Channels	Selectable 1–12 channels TDD or FDD
Frequency band	2.0–2.7 GHz (to include ISM 2.4–2.5 GHz)
Bandwidth	RF 3 dB bandwidth 4 & 17 MHz supported by switchable SAW filters in 2nd IF stage
Conversion	Dual up/down 14 bit DACs, 12 bit ADCs
Sampling rate	Direct IF 15 MHz sampling up to 64 MHz
Gain adjustment	20 dB switch at ADCs/DACs
Power adjustment	1 dB compression of 15 dBm (32 mW)
Noise floor	−130 dBm/Hz at ambient on receiver
Receiver	Input IP3 approx. −19 dBm

programmable synthesizer generates all derivative clocks and frequencies from this.

Custom switched mode power regulators followed by low-noise low-drop-out linear voltage regulators provide power supplies with very low-noise component to each subsystem within the STAR platform.

2.4. System control

Whilst there is a strong MMACS argument for the use of FPGA in baseband signal processing, it is still recognised that control software is easier and quicker to develop using high-level language and scripting tools [3]. For this reason, the platform incorporates a small ARM processor running Linux [4].

The embedded Linux system, connected by ethernet to a company internet or intranet, allows storage and transmission of very large volumes of data (over 10 Gb have been transferred during various tests), albeit not at speeds that would always be suitable for real-time data transfer.

The embedded Linux control processor has been dedicated to low-speed control and monitoring applications, and integrated with a highly novel web-based management interface [4] for ease of control, setup, and analysis of system operation.

3. ALGORITHMIC DEVELOPMENT

The STAR platform has hosted implementation of a number of MIMO and space-time algorithms comprising several

published methods from the academic research community and several nonpublished methods. Three are presented in this paper. In each case, the published algorithm described a theoretical approach evaluated through some form of simulation. In such cases, the gap between the evaluation and a real-world real-time implementation is large. In the extreme case, this may include discrete time sampling, but otherwise may include one or more issues such as self-generated noise (including inter-symbol interference), non-Gaussian additive noise, Doppler shift and spreading, timing mis-synchronization, and fixed-point word length effects including rounding errors.

The algorithmic development process used with the STAR platform would begin with a defined algorithm implemented in Matlab or Octave [5]. As much as possible, the effects of noise and errors, Doppler shift or spreading, and timing mis-synchronization would be included in the simulation [6].

3.1. Simulation refinement

This simulation must then be extended to cater for the effects of binary word length and rounding error. Unlike a DSP or general purpose microprocessor, computations performed in FPGA are relatively independent of word length. For example a 16-bit DSP would likely be confined to performing calculations, using 16, 32, 48, or 64 bits fixed point, or constructed floating point using separate mantissa and exponent [7]. By contrast, an FPGA could perform one part of a calculation with 17-bit logic and another part with 23-bits, or indeed whatever is necessary to maintain system performance.

Octave provides a good framework for the investigation of such word length effects, although such an investigation is generally time consuming since it generally precludes the use of many inbuilt accelerator functions in Octave which assume floating point throughout.

3.2. Example development process

Figure 3 outlines an example of an algorithmic module development process for channel estimation on FPGA starting from a fixed-point Octave simulation. Test vector files are generated, using Monte-Carlo style simulation inputs, that are time aligned to describe inputs and outputs of the module. These files contain a sequence of fixed-point numbers with the bit precision required for each input and output. These are used to derive various testbeds.

In the example shown, VHDL modules are authored and simulated functionally in ModelSim before being moved to Quartus II for full timing simulation and logic synthesis. In each case, the VHDL design is intended to be bit-exact with the Octave source. Since the actual implementation can involve unusual number-theoretic transformations or novel numerical tricks, it is common that bit-exactness will be broken during the process, in which case the implementation technique is folded back into the Octave source code and the simulation testbed is repeated to again ensure continued bit-exactness. It is therefore important to acknowledge that the design flow is a two-way process—and this has an impact on development team dynamics.

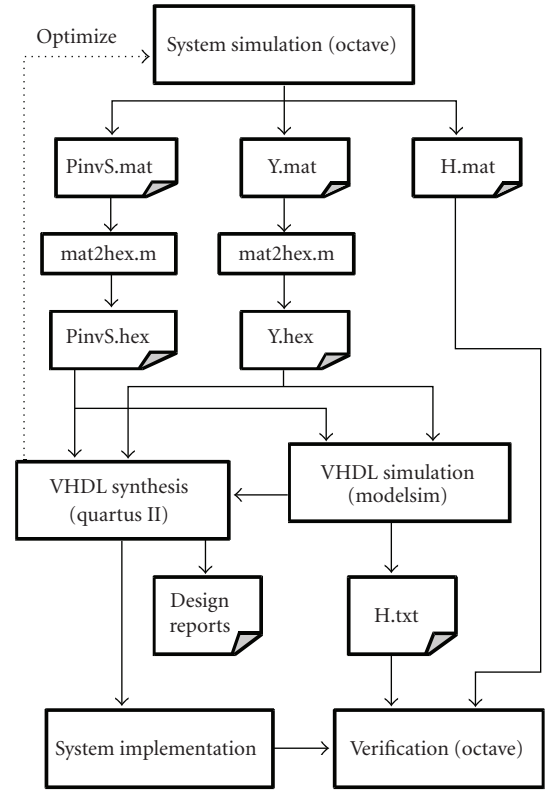

Figure 3: Implementation process for verifiable algorithm translation between Octave/Matlab and full VHDL.

3.3. Human resource requirements

The experience of the team developing the STAR platform has been that a multidisciplinary multi-talented team is required for system implementation. Successful results are unlikely where development is split along the lines of (i) theory, (ii) simulation, (iii) VHDL coding, (iv) hardware. The development process is highly coupled, much more than for a traditional specification-bound DSP development.

It is more desirable to split a multidisciplinary team along the boundaries of module requirements such as (i) digital front-end, (ii) channel estimator, (iii) equaliser... and so forth, where each module team has the responsibility to move that module from a set of equations, through simulations that are incrementally increasing in reality, through VHDL simulations to final code.

Given a floating point overall system simulation, fixed-point modules can be substituted into this when available, and interfacing requirements checked and fixed. The final result will be two-fold: a working VHDL implementation and a bit-exact system simulation. The simulation is invaluable in tracking down implementation problems and will aid with diagnosing issues identified in field testing.

Table 2: Data transmission format.

Antenna no.	Burst 1	Burst 2
Antenna 1	S_1	$-\widetilde{S}_2$
Antenna 2	S_2	$\widetilde{S}_1$

The STAR platform was used in such a way to develop three separate systems designed to explore interesting spaces within the multidimensional multiantenna, MIMO, and block coding algorithm continuum. These three systems are now introduced before particular implementation issues are identified in Section 4 and results and analysis from these are presented in Section 5.

3.4. Time-reversal space-time block coding

Recently, an Alamouti [1] inspired, but computationally simpler, time-domain block processing scheme [8–10] was developed. Named time-reversal (TR) space-time block coding (STBC), this lends itself to decoupled and parallel equalisation schemes and is particularly suitable for FPGA-based implementation [11]. In particular, the receive decoding process is simplified through the ordering and coding of transmit sequences.

As part of the STAR implementation work, the equations were first reordered into simplified time-domain formulations [6] and then investigated in the presence of channel error effects and timing synchronization errors [11].

In principal, TR-STBC is a 2×1 system where formatting and processed repetition of transmitted data ensure dual diversity across two timeslots, but obviously provide no capacity gain. Data transmission format is shown in Table 2, where S_1 and S_2 are transmit data blocks each comprising multiple data words as shown for the case of S_1:

$$S_1 = \{d_1(0), d_1(1), \ldots, d_1(N)\}. \tag{1}$$

In blocks $\widetilde{S}_1$ and $\widetilde{S}_2$, the individual data symbols themselves are time reversed and each is complex conjugated denoted for simplicity by D as is

$$\begin{aligned}\widetilde{S}_1 &= \{d_1^*(N), d_1^*(N-1), \ldots, d_l^*(0)\}\\ &= \{D_1(0), D_1(1), \ldots, D_1(N)\}.\end{aligned} \tag{2}$$

If the channel impulse response from Antenna 1 to the receive antenna is g_0, g_1, g_2, and g_3 assuming a 4-tap channel response, and the channel impulse response from Antenna 2 to the receive antenna is p_0, p_1, p_2, and p_3, then the received signal for the first data burst can be expressed as

$$\begin{aligned}r_1(t) = {} & g_0 d_1(t) + g_1 d_1(t-1) + g_2 d_1(t-2) + g_3 d_1(t-3)\\ & + p_0 d_2(t) + p_1 d_2(t-1) + p_2 d_2(t-2) + p_3 d_2(t-3)\\ & + n_1(t) \quad \text{for } t = 0, \ldots, N,\end{aligned} \tag{3}$$

where $n_1(t)$ is assumed to be white noise with zero mean. We have made the assumption that the channel is stationary over a symbol block and during both bursts, and in practice, this is generally achievable by judicious choice of symbol block length.

Similarly, the received signal for the second burst, when time-reversed and complex conjugated by the receiver, is

$$\begin{aligned}r_3(t) = r_2^*(N-t) = {} & -g_0^* d_2(t) - g_1^* d_2(t+1) - g_2^* d_2(t+2)\\ & - g_3^* d_2(t+3) + p_0^* d_1(t) + p_1^* d_1(t+1) + p_2^* d_1(t+2)\\ & + p_3^* d_1(t+3) + n_2^*(N-t) \quad \text{for } t = 0, \ldots, N.\end{aligned} \tag{4}$$

With some simplification, it is then possible to form a matrix using the q notation of [8] as

$$\begin{bmatrix} r_1(t) \\ r_3(t) \end{bmatrix} = \begin{bmatrix} g(q^{-1}) & p(q^{-1}) \\ p^H(q) & -g^H(q) \end{bmatrix} \begin{bmatrix} d_1(t) \\ d_2(t) \end{bmatrix} + \begin{bmatrix} n_1(t) \\ n_3(t) \end{bmatrix}. \tag{5}$$

This can then be solved in one of several ways and linear combining in this case is used to extract a single stream of decoded data from the equations.

The architecture of the receiver is shown in Figure 4, where all operations apart from the Viterbi equaliser and ARM control processor were performed in FPGA. The finite state machine (FSM) controller was replaceable in the STAR platform by a custom flexible embedded processor for ease of programmability [3]. Although there is a single receive antenna, there are two streams of data to be decoded post matched filtering, and the second of these is denoted by the grey blocks in the figure. The debug buffer shown could accept data from, or inject given data into, any major position in the data flow path. This was an invaluable means of applying test-vector stimulus (as in Figure 3) to the implemented system in order to perform real-time black-box testing of individual implemented modules in situ.

3.5. Adaptive multivariate (AMV) DFE-MIMO

There are many MIMO schemes ranging from the simplest linear equaliser through to complicated maximum-likelihood (ML) solutions which require exponentially increasing amounts of computational resources when scaled. Despite the dramatic continuous improvements in computational technology, suboptimal but realizable MIMO solutions are more likely to be implementable with current technology. BLAST [12] is one such family of algorithms without the computational load of a full ML solution, but aimed at better performance than linear equalisation. Similarly, the decision feedback equalizer (DFE) was chosen as a candidate for investigation on the STAR platform in the

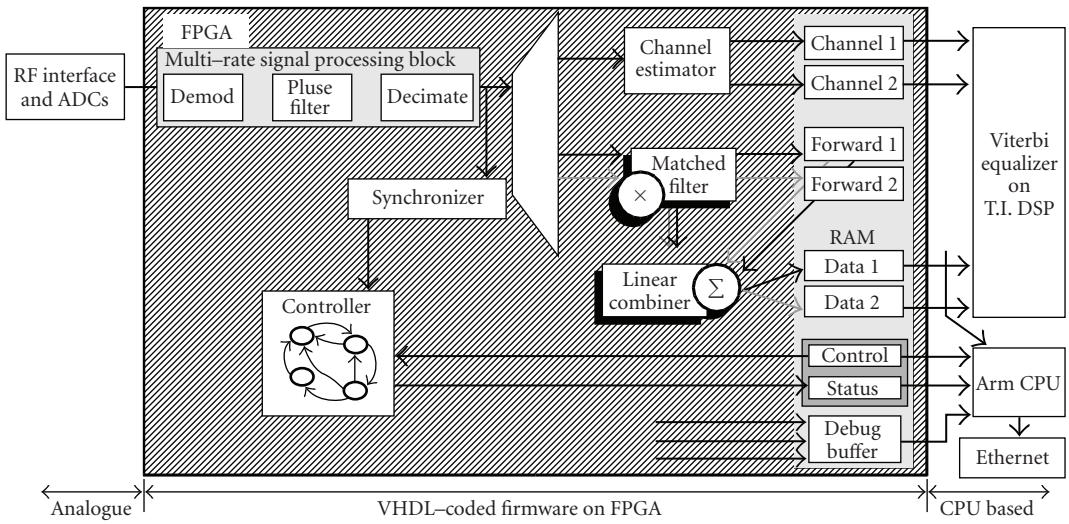

Figure 4: Implementation architecture for TR-STBC decoder.

hope that it could provide a good reduced complexity equalisation solution—less then a full maximum-likelihood sequence estimator (MLSE), but with similar performance levels. It also provides a continuous path for improvement through delayed decision-feedback sequence estimation [13] to full MLSE.

Multivariate DFE is based upon the standard single-thread DFE as presented in most undergraduate textbooks. For a given sample instance t, a soft decision input $z(t)$ is a scalar quantity represented by

$$z(t) = w^{ff}y(t) - w^{fb}\hat{x}(t), \tag{6}$$

where w^{ff} and w^{fb} are row vectors representing complex FIR filter tap weights, and $y(t)$ and $\hat{x}(t)$ represent the state of the shift registers shown in Figure 5 at time t. There are multiple ways of extending the single-thread DFE to the MIMO equivalent [14] generally differing in feedback filter specifics [15]. MUD-DFE [14] was the variant chosen for implementation on the STAR platform.

In an $n \times m$ MIMO DFE receiver, let the m received signals be denoted by $y_i(t)$ and n decisions $x_j(t)$. In MIMO-DFE, there are $n \times m$ feed forward filters $w_{i,j}^{ff}$ and $m \times m$ feedback filters $w_{i,j}^{fb}$ with the input to the jth decision device $z_j(t)$ written as

$$z_j(t) = \sum_{\alpha=1}^{m} w_{\alpha,j}^{ff} y_\alpha - \sum_{\alpha=1}^{n} w_{\alpha,j}^{fb} \hat{x}_\alpha, \tag{7}$$

where it is obvious that all $z_j(t)$ are dependent on all m received signals and all n previous decisions together. This can be visualised as the sum of the output of $m + n$ independent FIR filters, and is shown diagrammatically connected to other processing blocks in Figure 6.

To calculate the tap weights adaptively, we take (7) and write in the form of a normal equation

$$\begin{aligned} z_j(t) &= \left[w_{1,j}^{ff}, \ldots, w_{m,j}^{ff} \mid w_{1,j}^{fb}, \ldots, w_{n,j}^{fb} \right] \begin{bmatrix} y_1(t) \\ \cdots \\ y_m(t) \\ \hat{x}_1(t) \\ \cdots \\ -\hat{x}_n(t) \end{bmatrix} \\ &= \left[w_j^{ff} \mid w_j^{fb} \right] \left[-\frac{y(t)}{\hat{x}(t)} \right]. \end{aligned} \tag{8}$$

For calculating filter weights, $[w_j^{ff} \mid w_j^{fb}]$ must be found such that the decision error be minimized:

$$\left[w_j^{ff} \mid w_j^{fb} \right] = \operatorname{argmin} \left[\left[w_j^{ff} \mid w_j^{fb} \right] \left[-\frac{y}{\hat{x}} \right] - x_j \right], \tag{9}$$

where the form of this equation follows that for the single-thread DFE case. At this point a recursive least squares (RLS) solution could be found although there are several operations in this process that are undesirable from an implementation point of view; namely, the complex number inverse lookup table and the operations that result in an $L \times L$ square matrix. An alternative to the matrix inverse approach is the stochastic or steepest decent family of adaptive algorithms which are generally slower to converge [15] but less complicated to process. For this reason, the initial STAR implementation, centred around the LMS algorithm, which updates the filter weights according to

$$W(k+1) = W(k) + \mu\epsilon \begin{bmatrix} y \\ -\hat{x} \end{bmatrix}^H \tag{10}$$

and requiring only L, multiply and accumulate operations.

The initial system utilised 4 transmitting antennae each transmitting independent data streams with an air

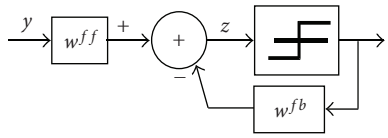

Figure 5: SISO DFE block diagram showing feed forward and feedback filters.

modulation format of $\pi/4$ DQPSK for its immunity to frequency drift.

In addition to the DFE processing, the receiver FPGA comprised modules for IF to baseband demodulation, root raised cosine matched filtering, and synchronization. The DFE filter weights were calculated for every packet based on training. A separate module performed weight updates and allowed effective algorithmic experimentation.

A 1 MHz pulse shaping root raised cosine filter with 100% roll-off receive filter and 60 MHz baseband sampling rate was used with a 120 MHz processing clock [15].

For efficiency, the sum of the multiple FIR filters was implemented with a single high-speed multiply and accumulate circuit by concatenating all inputs and tap weights in the right order without resetting the accumulator in between. In other words, the sum of the FIR filters can be implemented as one larger FIR filter:

$$\sum_{i=1}^{4} w_i y_i = WY \quad \text{for } W = [w_1, w_2, \ldots, w_4], \quad Y = [y_1, y_2, \ldots, y_4]^T. \tag{11}$$

Figure 7 shows a single DFE decision device building block. Four instances of this block were used to construct a 4×4 DFE receiver [15]. The feedback filters could similarly be merged into a single block multiply and accumulate operation. However, one of the benefits of DFE is that the feedback filter only operates from a finite set of constellation points and thus eliminates the need of a multiplier in some instances. In the STAR implementation, a better resource utilisation was thus to keep the feedback filters separate. Using built-in FPGA memory, it is very convenient to construct block RAM to store filter weights as well as the shift register states. The filters shown in Figure 7 are built from RAM blocks to correspond directly to $[y^T \mid -\hat{x}^T]$.

With filter weights stored in RAM, the adaptive algorithm simply updates those weights through a single write interface, while the DFE uses the read interface provided that the DFE modules do not need to access the memory location that the adaptive algorithm module is currently writing—which is a timing issue. In the case of the LMS algorithm, weight updates are independent for every tap and can be written as

$$W \text{ (new)} = W \text{ (old)} + \mu \text{ data error}, \tag{12}$$

and each filter coefficient is updated by adding a scaled version of the variable that the coefficient is multiplying for that instant in time. This allows the adaptive algorithm to integrate very closely with the filters, although RLS was found to be less optimal in this respect [15].

3.6. OFDM-MIMO

Orthogonal frequency division multiplexing (OFDM) is a multi-carrier-based digital modulation technique, in which a number of orthogonal waves are multiplexed in one symbol waveform, aiming to mitigate ISI in a frequency selective fading channel. It is advantageous both in terms of absolute data rate and in terms of spectral efficiency (bps/Hz). OFDM-MIMO is a particularly attractive combination since it combines the advantages of both OFDM and MIMO technology. MIMO is inherently capable of providing high spectral efficiency limited theoretically only by the minimum of the number of transmit or receive antennae, while OFDM provides high spectral efficiencies and effective ISI mitigation. The OFDM implementation transforms a frequency selective fading channel response into single tap flat fading channels in the frequency domain.

For these reasons, OFDM-MIMO was chosen for implementation on the STAR platform, with similar rationale to published implementations by other authors [16, 17]. Discrete matrix multi-tone modelling was chosen to reduce the complexity in a frequency selective fading system implementation, and this holds good for both flat and frequency selective fading channels. In our model, K data symbols are transmitted from each antenna per block, and a cyclic prefix added to the beginning of the data sequence such that the last $(L-1)$ symbols are transmitted before the full block of K symbols. This is true of sequences from each of M_T transmit antennae. There are M_R receive antennae with a multi-path length L. The architecture is shown in Figures 8 and 9 for transmit and receive processing elements, with the algorithm that was implemented also described in [18]. Timing-critical elements were implemented in VHDL but offline channel estimation, fine timing synchronization, and frequency correction and detection were implemented in Matlab. This demonstrated the underlying principles of implementation, but provided a very rapid path to evaluation of OFDM-MIMO under real channel conditions but without lengthy development requirements. Other authors [16, 17] have implemented similar systems, demonstrating that the FFT, IFFT, and backend processing could easily be performed in FPGA if required.

Let the $M_R \times M_T$ impulse response matrix describing the channels be $G[l]$ for the lth tap for $l = 0, 1, \ldots, L-1$. The i, jth element of $G[l]$ are represented by $g_{i,j}(l)$ denoting the channel impulse response from jth transmit antenna to the ith receive antenna for the lth tap. $s_j[k]$ is the signal prior to IFFT: K symbols to be transmitted on antenna j at time (or tone) k for $k = 0, 1, \ldots, K-1$.

Similarly, $y_j[k]$ is a block of symbols received after the FFT on antenna i for time (or tone) k for $k = 0, 1, \ldots, K-1$.

The sequence of symbols to be transmitted over each antenna is first inverse Fourier transformed (IFFT) and a cyclic prefix (CP) of length $(L-1)$ is added before the K symbols. Thus $K+L-1$ symbols are transmitted from each antenna. At the receiver, the CP is stripped off and then an FFT is taken of the remaining K symbols from each antenna. The signal at the ith antenna (after FFT) for the kth time (or tone) is given

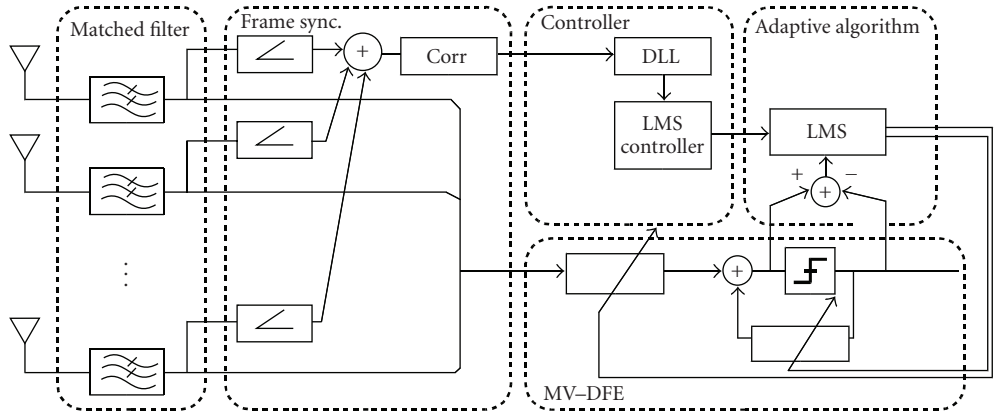

Figure 6: Architectural structure of the AMV-DFE-MIMO receiver showing the data path from transmitters through the MIMO DFE structure and adaptive algorithm. This is entirely implemented in FPGA.

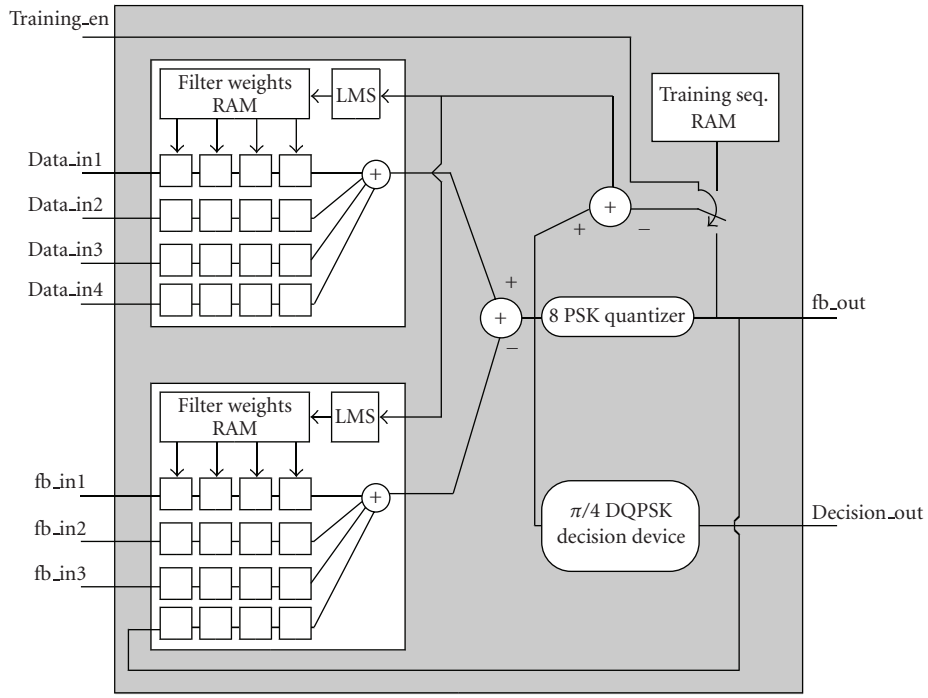

Figure 7: DFE multiplier block.

by

$$y_i[k] = \sum_{j=1}^{M_T} \omega_{i,j}[k]s_j[k] + n_i[k] \quad \text{for } i = 1, 2, 3, \ldots, M_R, \tag{13}$$

where $n_i[k]$ designates additive noise and $\omega_{i,j}[k]$ is the FFT of the channel impulse response:

$$\omega_{i,j}[k] = \sum_{l=0}^{L-1} g_{i,j}[l]e^{-j(2\pi ld/K)} \quad \text{for } k = 0, 1, 2, \ldots, (K-1). \tag{14}$$

If we now define

$$H[k] = \sum_{l=0}^{L-1} G[l]e^{-j(2\pi ld/K)} \tag{15}$$

as the MIMO channel impulse response matrix for the kth tone computed from the FFT of the time domain channel impulse response matrix for the L taps, so

$$H[k]_{i,j} = [\omega_{i,j}[k]]. \tag{16}$$

So $H[k]$ is an $M_R \times M_T$ matrix, $y[k]$ and $n[k]$ are M_R element vectors, and $s[k]$ is an M_T element vector. The MIMO model

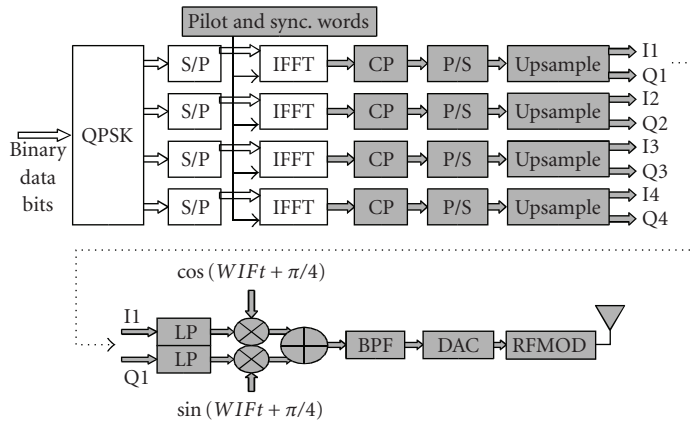

Figure 8: OFDM-MIMO transmit structure showing those elements that had been implemented in FPGA (shaded) and those offline in Matlab (unshaded), but with only a single RF chain reproduced for clarity. For some tests, the Matlab/FPGA interface was actually moved up to the BPF rather than at the CP insertion block for convenience. S/P and P/S are serial-to-parallel and parallel-to-serial converters, respectively.

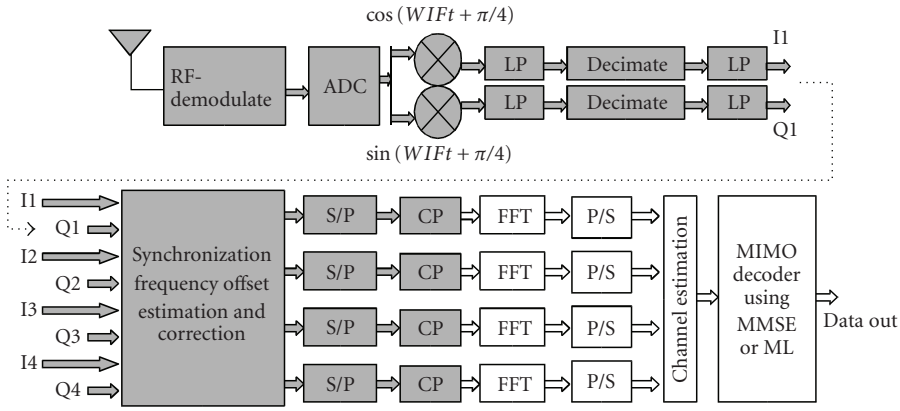

Figure 9: OFDM-MIMO receive structure showing those elements that had been implemented in FPGA (shaded) and those offline in Matlab (unshaded), but with only a single RF chain reproduced for clarity. For some tests, the Matlab/FPGA interface was moved to the decimator rather than the CP block for convenience. S/P and P/S are serial-to-parallel and parallel-to-serial converters, respectively.

equation now becomes

$$y[k] = H[k]s[k] + n[k] \quad \text{for } k = 0, 1, 2, \ldots, (K-1). \quad (17)$$

In summary, the MIMO-OFDM method configures the frequency selective channel of bandwidth B into K orthogonal flat fading channels, each of B/K bandwidth.

In the FPGA implementation, an over-air frame structure as shown in Figure 10 was formatted, controlled, and synchronized in the FPGA, with ten consecutive data words transferred in each packet. For experimental purposes, random or Matlab-generated data was uploaded to FPGA and used in transmission continuously until such time as the data was adjusted. This obviously differs from the implementation required in a production implementation, but does allow repeatable tests to be performed with static data when necessary and allow as well a range of different data packets to be tested as required.

In terms of packet data structure, since receive data is four times oversampled, there are 640 synchronization chips and 2560 training chips (multiplexed between antennas as shown in Figure 10 and including CP), followed by 10 data words comprising 3200 OFDM chips (again including CP). It was found that the ring time of the combined analogue RF filters extended 96 chips beyond the total 6400 structured chips in a packet, and thus a guard time was inserted between packets to accommodate this.

Time synchronization was performed by correlation between synchronization words—gross synchronization was implemented in FPGA, whilst fine oversampled alignment performed in Matlab using standard techniques.

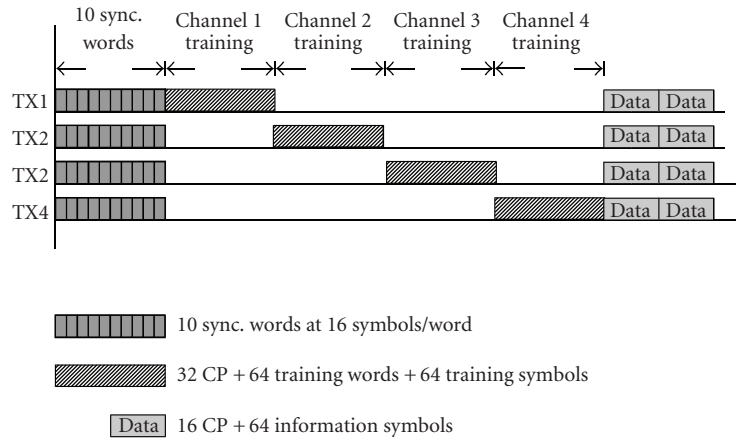

Figure 10: OFDM-MIMO on-air packet structure, including synchronization, training, and data words, formatted, and controlled in FPGA, for 4×4 experimental test setup. There were a total of 10 data words transmitted per antenna per packet.

The frequency offset, f_{off}, at a function of the sampling instant, for T_c training duration (N_c symbols) at f_s sampling frequency, was estimated by determining the phase angle of the timing detection metric:

$$f_{\text{off}}(\tau_{\text{sync}}) = \frac{\theta(\tau_{\text{sync}})}{2\pi T_c} = \frac{f_s \angle\{\lambda(\tau_{\text{sync}})\}}{2\pi N_c}, \tag{18}$$

where $\theta(\tau)$ is the phase angle of the sum of the correlations of training symbols (which can be calculated unambiguously within a range equal to half the subcarrier spacing).

Experience revealed that whilst the system was highly tolerant to timing synchronization errors, frequency offset estimation was the single most critical factor in the OFDM-MIMO performance. Bearing in mind that this was a QPSK system, it is expected to be significantly more critical when utilising higher density constellations.

Back-end Matlab processing allowed a comparison of ML and MMSE decoding. Although currently uncorroborated, preliminary indications show that ML, whilst normally providing higher performance than MMSE, tends to perform worse when frequency offset estimation errors occur. It is also evident that, at high receive power levels, MMSE and ML estimates tended to converge.

4. IMPLEMENTATION ISSUES ASSOCIATED WITH THE ALGORITHMS

Each of the three implemented systems followed the implementation methodology of Section 3.2 and resulted in working systems that allowed the investigation of algorithm operation under various real operating scenarios. The test platforms were mobile, and antenna construction modular such that various geometries could be explored. Table 3 compares the implementations, and although far from an exhaustive list of possible MIMO and space-time algorithm options, the chosen methods covered a wide span of possibilities. This was a deliberate approach to build expertise in the design team, and test as wide a variety of algorithm types and modulation formats as possible. Note also that although 12-channel sounding tests have been performed to prove platform integrity, at the time of writing, a full 12-channel MIMO implementation has not been completed using the platform.

5. ANALYSIS OF STAR DEVELOPMENTS

The STAR platform development began with a very brief initial exploratory phase followed closely by simultaneous platform development and academic search and evaluation to determine suitable algorithmic approaches. The hardware platform comprises enclosure, power supplies, high-precision clocks synthesizer, RF, mixed-signal, and digital components on 23 printed circuit boards (PCBs). The total development time for the first working non-MIMO system, a multichannel channel sounder, was 10 months.

5.1. Development phases

The first algorithm implementation was TR-STBC, and utilised most of the 12 engineers for approximately 4 months, although evaluation and testing continued with fewer engineers for longer. At the close of the TR-STBC subproject development, a decision was made to continue on with AMV-DFE-MIMO and OFDM-MIMO developments in parallel since sufficient STAR platforms existed.

The same 12-member engineering team cooperated on both implementations. The OFDM-MIMO system FPGA component was limited to pulse shaping of stored transmit data, receiver front-end, decimation, simple filtering, and data capture subsystems. Actual OFDM decoding was performed offline using Matlab. This system was thus sufficient to explore the implementation of the high-bandwidth OFDM-MIMO front end and the effects of different

Table 3: Parameters for three STAR-platform implementations.

Name	TR-STBC	DFE-MIMO	OFDM-MIMO
Configuration	2×1	4×4	4×4
Bandwidth	2 MHz	1 MHz	15 MHz
Modulation	BPSK	$\pi/4$ DQPSK	64 carrier QPSK
Data rate	2 Mbps	8 Mbps	107 Mbps
LEs[a] used Tx/Rx	3500/23500	4500/36000[b]	< 10000 each
DSP blocks Tx/Rx	0/16	0/48	4/4
References	[2, 6, 8–11, 19]	[15]	[16–18]

[a] An LE, the basic processing unit in an Altera FPGA, comprises combinational logic, a flip-flop, lookup table, and input/output.
[b] The Rx used a 3-FPGA solution: 2 Cyclone FPGAs performed dedicated front end processing, using 29000 of the 36000 total.

channels, antennae, and gains, but was not encumbered by channel estimation, FFT design, and data reconstruction issues. However these final three issues have been demonstrated as FPGA implementations by other authors, most notably Wouters et al. [17] in the 2×2 PICARD demonstrator, and discussed by Kaiser et al. in [16], as well as in the DFE-MIMO and TR-STBC systems here (excluding the FFTs).

Approximately 3 months were required to deliver the final two working MIMO systems, with only two engineers allocated to constructing the OFDM-MIMO implementation. Again indoor and outdoor evaluative testing programmes followed the developments using a reduced team.

5.2. Development team

The full engineering team comprised 3 recent graduates, 4 engineers with 1 to 3 years experience, 3 senior engineers, one principal engineer, and a project manager. One of the senior engineers was devoted to Matlab simulations and none of the team had experience of FPGAs or VHDL, although several had experience porting algorithms to DSP.

The timescales indicate that, although the initial investment in equipment and the learning-curve for FPGA development were large, given such a newly experienced team, the time required to utilise the hardware in different ways to explore three diverse ST/MIMO algorithms was not excessive.

5.3. Experimental conditions

Channels test environments for all implementations included interior office space, university campus, parkland, urban street-scape, and building-to-building link. Distances ranged from approximately 3 m to 500 m with the majority of indoor tests confined to below 40 m [2]. Channel rank problems were endemic, with the DFE-MIMO system [15] being particularly susceptible to low-rank effects. In 100 m tests across a car park and between buildings, average BER achieved with misaligned antennae was observed to significantly exceed that from aligned antennae. Timing synchronization and (especially for OFDM) frequency offset were significant issues, reinforcing published work in that field. Research on these effects is ongoing, but with relevance to compensation algorithms more so than to a discussion of implementation platform. Antennae were in a proprietary steerable multielement patch arrangement to be published separately.

6. CONCLUSION

Firstly, the use of programmable FPGA logic for performing MIMO and space-time baseband signal processing has been demonstrated. The claim is that this required less effort, and resulted in a more stable system than a similar DSP-based implementation, and that certainly follow-on developments would undoubtedly benefit in this way.

Secondly, that the enormous processing capability of a platform like STAR is sufficient to implement several varieties of space-time algorithm, that these can be developed rapidly and accurately using only FPGAs for baseband signal processing. Details of three example implementations have been presented, with performance data published elsewhere. Each implementation was the first-known implementation of the relevant technique, either in real time, or using an FPGA-based system.

ACKNOWLEDGMENTS

This work was partially funded by the NZ Foundation for Research, Science, and Technology. Thanks are due to Andrew Jones for his RF and platform design and the diagrams of Figures 1 and 2. Finally, the efforts of the entire Tait Electronics Ltd. Group Research STAR team are gratefully acknowledged.

REFERENCES

[1] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, 1998.

[2] A. M. Baghaie, S. H. Kuo, and I. V. McLoughlin, "FPGA implementation of space-time block coding systems," in *Proceedings of IEEE 6th Circuits and Systems Symposium on Emerging*

Technologies: Frontiers of Mobile and Wireless Communication (MWC '04), vol. 2, pp. 591–594, Shanghai, China, May–June 2004.

[3] R Shadich and I. V. McLoughlin, "A modular computational engine for communications processing," in *Proceedings of Australian Telecommunications, Networks and Applications Conference (ATNAC '03)*, Melbourne, Australia, December 2003.

[4] I. V. McLoughlin and T. Scott, "Space-time processing—Linux style," *Linux Journal*, vol. 2004, no. 125, pp. 8–8, 2004.

[5] Octave homepage: http://www.octave.org, November 2004.

[6] K. Mehrotra and I. V. McLoughlin, "Time reversal space time block coding with channel estimation errors," in *Proceedings of 4th International Conference on Information, Communications & Signal Processing and 4th Pacific-Rim Conference on Multimedia (ICICS-PCM '03)*, vol. 1, pp. 617–620, Singapore, December 2003.

[7] A. B. Premkumar, A. S. Madhukumar, and C. T. Lau, "MAC units for matched filters in DS-CDMA systems," *IEEE Transactions on Broadcasting*, vol. 48, no. 1, pp. 52–57, 2002.

[8] E. Lindskog and A. Paulraj, "A transmit diversity scheme for channels with intersymbol interference," in *Proceedings of IEEE International Conference on Communications (ICC '00)*, vol. 1, pp. 307–311, New Orleans, La, USA, June 2000.

[9] P. Stoica and E. Lindskog, "Space-time block coding for channels with intersymbol interference," in *Proceedings of 35th Asilomar Conference on Signals, Systems and Computers (ACSSC '01)*, vol. 1, pp. 252–256, Pacific Grove, Calif, USA, November 2001.

[10] E. G. Larsson, P. Stoica, E. Lindskog, and J. Li, "Space-time block coding for frequency-selective channels," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '02)*, vol. 3, pp. 2405–2408, Orlando, Fla, USA, May 2002.

[11] K. Mehrotra and I. V. McLoughlin, "Time reversal space time block coding with channel estimation and synchronization errors," in *Proceedings of Australian Telecommunications, Networks and Applications Conference (ATNAC '03)*, Melbourne, Australia, December 2003.

[12] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Technical Journal*, vol. 1, no. 2, pp. 41–59, 1996.

[13] A. Duel-Hallen and C. Heegard, "Delayed decision-feedback sequence estimation," *IEEE Transactions On Communications*, vol. 37, no. 5, pp. 428–436, 1989.

[14] C. Tidestav, "The multivariable decision feedback equalizer: Multiuser detection and interference rejection," Ph.D. dissertation, Uppsala University, Uppsala, Sweden, 1999.

[15] S. H. Kuo, J. Dowle, and I. V. McLoughlin, "A reconfigurable platform for MIMO research realtime implementation of 4×4 adaptive multi-variate DFE," in *Proceedings of Virginia Tech's 14th Symposium on Wireless Personal Communications*, Blacksburg, Va, USA, June 2004.

[16] T. Kaiser, A. Wilzeck, M. Berentsen, and M. Rupp, "Prototyping for MIMO-systems: an overview," in *Proceedings of 12th European Signal Processing Conference (EUSIPCO '04)*, Vienna, Austria, September 2004.

[17] M. Wouters, P. Van Wesemael, R. Vandebriel, A. Dewilde, and M. Libois, "Real time prototyping of broadband wireless LAN systems," in *Proceedings of IEEE 15th International Workshop on Rapid System Prototyping (RSP '04)*, pp. 226–231, Geneva, Switzerland, June 2004.

[18] K. Mehrotra and I. V. McLoughlin, "Low complexity detection algorithms for a MIMO-OFDM system," in *Proceedings of Virginia Tech's 14th Symposium on Wireless Personal Communications*, Blacksburg, Va, USA, June 2004.

[19] S. H. Kuo, I. V. McLoughlin, and K. Mehrotra, "Reconfigurable processing framework for space-time block codes," in *Proceedings of Australian Telecommunications, Networks and Applications Conference (ATNAC '03)*, Melbourne, Australia, December 2003.

J. Dowle received his B.S. of Engineering degree in electrical and electronic engineering from the University of Canterbury, Christchurch, New Zealand, in 2001. Since completing his Bachelors, he has been working in Group Research at Tait Electronics Ltd., Christchurch as a Design Engineer. He has worked in digital signal processing using field programmable gate arrays (FPGA), multiple-input multiple-output (MIMO) communications systems, and electronic hardware design.

S. H. Kuo received his M.A. of Engineering degree from the University of Canterbury, Christchurch, New Zealand, in 2000, working in the field of chaotic cryptography. He then joined Tait Electronics Ltd. as a Design Engineer, working on space-time and MIMO algorithms for digital wireless communications. In 2005, he began working towards a Ph.D. at Simon Fraser University in Vancouver, Canada, on multi-user detection. Howie holds a number of patents on areas related to digital wireless communications, and is a Member of the Golden Key Honour Society.

K. Mehrotra received his B.S. of Technology degree in electrical engineering from the Indian Institute of Technology, Kanpur, in 1979, and Ph.D. from the Indian Institute of Science, Bangalore, India, in 1994. He worked in the Avionics Design Bureau of the Hindustan Aeronautics Limited, Hyderabad, India, from 1979 to 1996, and in the Switchtec Power Systems, Christchurch, from 1996 to 1998. He served as an Assistant Professor in the Department of Aerospace Engineering in the Indian Institute of Science, Bangalore, India, from 1998 to 1999, where he taught a part of the course in navigation, guidance, and control. Presently, he works in the Group Research of Tait Electronics Ltd, Christchurch, as a Senior Design Engineer. He has worked in the areas of radar tracking, digital signal processing, power electronics, control systems, hardware design, orthogonal frequency division multiplexing (OFDM), multiple-input multiple-output (MIMO) communication systems, and power amplifier linearization. He holds patents in the areas of adaptive time division duplexing, network timing protocol, and digital communication system.

I. V. McLoughlin completed his Ph.D. in audio signal processing from the School of Electronic & Electrical Engineering at the University of Birmingham in 1997, funded by Simoco Telecom, where he worked in the Advanced Technology Group. Prior to returning to the university for his Ph.D., he worked for around 5 years for the British Government and for GEC Research Ltd. (Hirst Research Centre). In 1998, he emigrated from the UK to lecture at Nanyang Technological University, School of Applied Science (now School of Computer Engineering) in Singapore, and from there came to Christchurch, New Zealand, to take up a position as Principal Engineer in Tait Electronics Group Research. He holds patents in speech intelligibility improvement and distributed ad hoc wireless networking. He is also the director of a small electronics company and charitable trust.