An End-to-End Multi-Standard OFDM Transceiver Architecture Using FPGA Partial Reconfiguration

Thinh H. Pham, *Member, IEEE*, Suhaib A. Fahmy, *Senior Member, IEEE*, and Ian Vince McLoughlin, *Senior Member, IEEE*

***Abstract*—Cognitive radios that are able to operate across multiple standards depending on environmental conditions and spectral requirements, are becoming more important as the demand for higher bandwidth and efficient spectrum use increases. Traditional custom ASIC implementations cannot support such flexibility, with standards changing at a faster pace, while software implementations of baseband communication fail to achieve performance and latency requirements. Field programmable gate arrays (FPGAs) offer a hardware platform that combines flexibility, performance, and efficiency, and hence they have become key in meeting the requirements for flexible standards-based cognitive radio implementations. This paper proposes a dynamically reconfigurable end-to-end transceiver baseband that can switch between three popular OFDM standards, IEEE 802.11, IEEE 802.16 and IEEE 802.22, operating in non-contiguous fashion with rapid switching. We show that combining FPGA partial reconfiguration with parameterised modules offers a reduction in reconfiguration time of 71% and a FIFO size reduction of 25% compared to conventional approaches, and provides the ability to buffer data during reconfiguration to prevent link interruption. The baseband exposes a simple interface which maximises compatibility with different cognitive engine implementations.**

I. Introduction and Related Work

The spectral resource demands of wireless telecommunication systems continue to increase [1], while statically allocated spectrum use is close to saturation, leading to what has been termed the "spectrum crunch". Cognitive Radios (CRs) that can adapt to channel conditions to ensure effective spectrum usage are an important technology for addressing this challenge. CRs are designed to transmit dynamically in unused spectral regions without causing harmful interference to primary users (PUs) or incumbent users (IUs) [2]. Apart from the critical issues of spectrum sensing and band allocation, the lower priority of secondary users (SU) raises a challenge in terms of transmission capability and quality of service in cognitive radios. When the spectrum allowed for a CR system is fully occupied by PUs and IUs, CR transmissions can be blocked. Multi-standard cognitive radios are able to operate in multiple frequency bands with different specified standards, representing a more flexible generalisation of CRs.

Most practical CRs are built using powerful general purpose processors to achieve flexibility through software, but they can fail to offer the computational throughput required for advanced modulation and coding techniques and they often have high power consumption. For example, the GNU Radio [3] platform, which is widely used in academia, is a software application that runs on a computer or an embedded ARM processor platform, e.g. on the Ettus USRP E-series. Computational limitations mean that while it has been successful for investigating CR ideas, it is not feasible for implementing advanced embedded radios using complex algorithms, without exploiting extra hardware resources. Meanwhile, custom ASIC implementations are not agile or cost effective enough to cope with the fast-changing specifications and operating requirements of rapidly evolving standards.

Field programmable gate arrays (FPGAs) have long been seen as an attractive middle ground: FPGAs allow hardware designers to build circuits offering full hardware throughput, but with the flexibility of being able to modify the hardware design after deployment. FPGAs have been widely used in military radios, and are widespread in the backbone infrastructure of cellular networks. FPGAs have also found widespread use within demonstration platforms, although they are often limited to fixed function baseband processing. In [4], an FPGA was combined with an ARM-based controller running Linux and a digital signal processor (DSP) based channel equalizer and Viterbi decoder. KUAR [5] was a mature radio platform built around a fully-featured Pentium PC with a Xilinx Virtex II FPGA. The baseband processing was accelerated by FPGA to achieve non-contiguous (NC) OFDM based on the IEEE 802.16 standard and makes use of Linux-based control software. WARP from Rice University [6] is a software and hardware platform that combines an agile RF front-end with an FPGA for the prototyping of radio systems. An extensive library of custom baseband designs was provided for real time implementation, while a software based flow was developed for prototyping. Existing FPGA radio platforms mostly support fixed baseband hardware in the FPGA. To add flexibility, extra hardware must be added to the baseband system, with software control selecting the required hardware portions at runtime. However, as the number of possible baseband configurations increases, this means using a larger FPGA of which only a small portion is active at any one time.

How FPGAs can specifically contribute to CR implementation is through their dynamic programmability. Many FPGAs support a feature called partial reconfiguration, whereby a portion of the hardware can be modified at runtime while the remainder continues to operate unchanged. This allows sections of hardware to adapt to evolving requirements, including real-time channel changes. Exploiting this capability, however, typically requires high levels of FPGA expertise and detailed software/hardware co-design. Some software radio

T. H. Pham is with Nanyang Technological University, Singapore. S. A. Fahmy is with the University of Warwick, United Kingdom. I. V. McLoughlin is with the University of Kent, Medway, United Kingdom.

frameworks like Iris [7], have limited support for FPGA partial reconfiguration but suffer from poor bandwidth between software and hardware components of the system. The heterogeneous platform in [8] adapts its hardware to support standards like GSM, UMTS, and wireless LAN through partial reconfiguration – switching the channel coder from one context to another depending on SNR. Most other processing components run as embedded software on nodes in a network-on-chip processor. This leads to the need for a large data buffer between the FPGA and processor, while inefficient data transfer mechanisms lead to increased power consumption and decreased throughput. Several projects at Virginia Tech [9] have shown dynamically assembled radio structures on FPGAs, where the target radio system is defined at a high-level with datapaths connecting relatively large functional modules. Modules are wrapped, each consisting of a partially reconfigured (PR) module with compiled partial bitstreams stored in a dynamic library. Their on-line assembly method eliminates the need for runtime compilation, thus affording flexibility. A flexible radio controller can insert and remove compiled modules to adapt to current conditions. While this approach appears promising, it is device-specific and is incompatible with standard FPGA design tools.

More recently, effort has been made to better generalise the interface between high layer software and reconfigurable baseband hardware. The work in [10] decomposes a radio into a control plane running in a soft core processor, and a baseband data plane of custom hardware modules in the FPGA fabric. The processor can initiate reconfigurations of the baseband in response to changing conditions through a "cognitive engine". For these platforms, the poor performance of the processor and lack of high level interfaces to the baseband made design difficult.

Some recent hybrid FPGAs include hard processors in the FPGA fabric, such as the Zynq family of FPGAs from Xilinx that include dual core embedded ARM processors. These offer an ideal platform on which to build cognitive radios based on the above model, where the cognitive engine can be implemented in software on a capable processor that can support a full OS stack, while high level APIs offer a low-latency, high-throughput connection to the baseband implemented in the FPGA fabric [11].

This paper explores the design of an OFDM baseband that can be integrated with platforms such as the above. We show how the hardware design can be tuned for multiple standards to minimise reconfiguration time and buffering requirements through a performance-driven combination of partial reconfiguration and parametrised blocks.

A. Contributions

This paper: (1) Develops and outlines an end-to-end FPGA transceiver architecture for multiple wireless standards: IEEE 802.11, IEEE 802.16 and IEEE 802.22. (2) Explores the global trade-off between partial reconfiguration and parametrisation in terms of resource utilisation and reconfiguration latency. (3) Presents an architecture that supports the frequency and timing synchronisation agility necessary for non-contiguous (NC) OFDM operation across different frequency bands on a symbol-by-symbol basis.

The remainder of this paper is organised as follows: Section III discusses the challenges of implementing a cognitive ratio (CR) baseband and approaches able to overcome these challenges. Section IV outlines the proposed architecture for multi-standard CR designs. Section V details the performance analysis for PR-based designs, leading to the optimal solution for the three stated OFDM standards. Finally, Section VI concludes with a discussion of future work.

II. OFDM Cognitive Radio Systems

Orthogonal Frequency Division Multiplexing (OFDM), as a multi-carrier modulation technique, has natural advantages for cognitive radio (CR) implementation [12]. Individual carriers can be enabled or disabled to occupy or free up specific portions of the frequency spectrum, and this can be done on a symbol-by-symbol basis. The method is both spectrally efficient and flexible in time-frequency allocation. In recent years, OFDM has also been adopted as the base modulation scheme for a number of wireless standards in different frequency bands, and has been investigated in terms of spectrum sensing and carrier allocation for CR [13], [14]. A radio capable of supporting different OFDM configurations can thus support a wide range of existing and emerging standards.

A. OFDM Standards

In this paper, we use three different OFDM standards to demonstrate our prototype multi-standard baseband OFDM transceiver architecture. The system supports not just symbol-by-symbol carrier selection within each standard band, but can switch between standards operating in different frequency bands. The system complies with different OFDM symbol lengths and frame formats, according to the three selected standards: IEEE 802.11 [15], IEEE 802.16 [16], and IEEE 802.22 [17]. The relevant specifications for each of these are summarised in Table I. Further standards with parameters within the bounds of those shown are inherently supported, while other standards can be accommodated with minimal design modification.

B. System Requirements

The generic structure of an OFDM transceiver is presented in Fig. 1, showing data being modulated, formatted, passed

TABLE I
SPECIFICATIONS OF THREE COMMON OFDM-BASED IEEE STANDARDS.

Specification	IEEE 802.11	IEEE 802.16	IEEE 802.22
Frequency band	2.4–2.5 GHz	5–6 GHz	54–862 MHz
Channel Width	10 MHz	10 MHz	8 MHz
Sampling Frequency	10 MHz	11.52 MHz	9.136 Mhz
FFT size (N_{FFT})	64	256	2048
CP Length	16	32	512
Number of data carriers	48	192	1440
Number of pilots	4	8	240

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2756914, IEEE Access

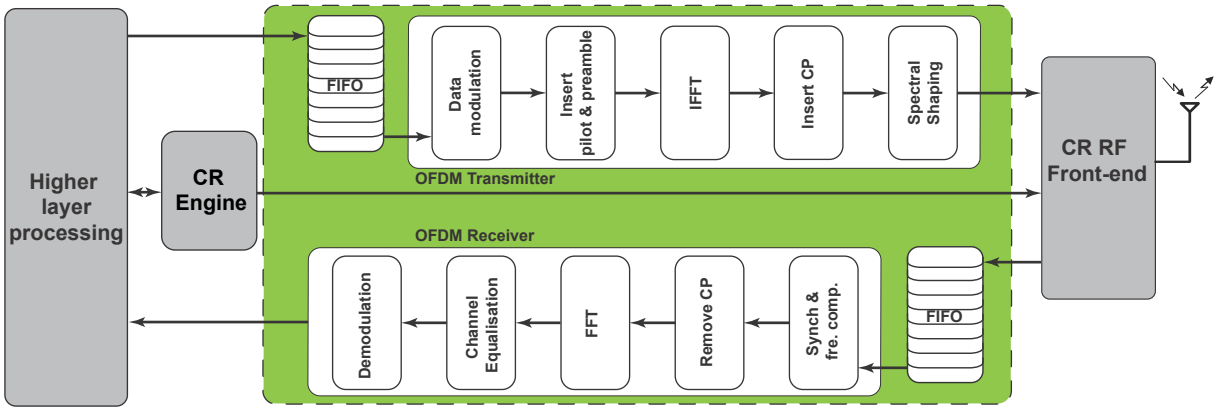

Fig. 1. Generic structure of multi-standard OFDM cognitive radio system.

through an inverse fast Fourier transform (FFT), a cyclic prefix (CP) added, and then being shaped before transmission. The receive chain consists of timing synchronisation and frequency offset compensation, followed by cyclic prefix (CP) stripping, FFT, timing correction, and demodulation. Such an architecture can be made to support the three OFDM standards in Table I by adjusting the baseband operating parameters for each standard (e.g. different sampling frequencies, different sized FFTs, subcarrier modulation, and so on).

Fig. 1 also shows the cognitive radio (CR) engine which senses the spectral environment to control the system. This is used to sense unused spectrum and direct the radio to operate in less crowded frequencies. It should determine the best configuration, then instruct the baseband to adapt by, for example, swapping channel frequency, selecting and deselecting different active subcarriers and sampling at a different rate. This can potentially occur on a symbol-by-symbol basis. The architecture we propose is not limited to any particular CR engine, however we demonstrate one hosted on a processor attached to (or embedded within) the FPGA.

III. Challenges in Multi-Standard Radio Systems

In this section we discuss the implementation of an OFDM system for cognitive radio in general, and then consider the particular challenges encountered in supporting multiple standards, for which solutions will be further discussed and evaluated in Section IV.

OFDM system implementation is known to be relatively simple, low cost, and more effectively parametrised than alternatives such as filter bank multi-carrier (FBMC) radios, hence its wide adoption in current standards [12]. The advantages of implementing OFDM systems using FPGAs are also well known and explored in the literature [4], [5], [9]. This paper goes beyond previous work to explore design of a multi-standard OFDM cognitive radio—a system able to switch between carriers and standards dynamically at runtime. Such a system entails a number of challenges not applicable when building static OFDM radios.

1) Flexibility: The chosen OFDM standards span a wide configuration space. For example, referring to Table I, the FFT/IFFT blocks in each system range in length from 64 to 2048 samples, while the number of carriers ranges from 48 to 1440. Hence, it is necessary to support different FFT window sizes. Traditional static hardware implementation would entail a design containing all possible alternatives and the use of multiplexers to switch between them. But since FPGAs are capable of being reconfigured dynamically at runtime, this does not require all circuits to be implemented simultaneously, hence saving area. Furthermore, some processing blocks, like the FFT, can be designed to be intrinsically flexible, offering faster switching at the cost of marginally increased area consumption. Configurability beyond initial expectations is also something to be considered in the real world; a radio built with flexibility in mind should be tolerant to adding new specifications.

2) Reconfiguration Time: The time taken to react to changes in the channel is a critical metric in CR design. While a part of this is composed of the decision making process in the cognitive engine, the configuration latency, once a switching decision has been taken, is also important. In a static multiplexed system, reconfiguration entails selection using multiplexers and so is very fast. However, as discussed previously, this results in a large area overhead since all circuits must be continually present on-chip. Partial reconfiguration allows parts of the FPGA to be replaced at runtime, and can be used to replace the whole baseband chain or individual blocks. Since reconfiguration time increases as a larger area is reconfigured, careful consideration is required to minimise the size of reconfigurable blocks. Individual parametrised blocks can change mode quickly, but themselves entail area overhead and design complexity to accommodate the parametrisation.

3) Synchronisation: There is a key challenge at baseband related to synchronisation. OFDM systems typically tolerate only a small carrier frequency offset (CFO), leading to strict constraints on the design of the RF front-end. In a multi-standard CR, the RF front-end needs to access a wide bandwidth to cover the supported standards, and be able to switch between frequencies on a symbol-by-symbol basis. Such a precise and yet wide ranging RF front-end can be very expensive to implement precisely in hardware. This can be overcome through improved synchronisation methods in the baseband hardware, as in [18].

4) Spectral Shaping: CRs also demand minimised spectral leakage for both in-band and out-of-band transmitted signals.

This is mandated by standards bodies but is also a requirement for the efficient use (or reuse) of nearby unoccupied carriers. OFDM is known to be susceptible to spectral leakage and thus particular care must be taken to mitigate against this. A fixed spectral mask using RF hardware filtering is not agile in either frequency range or in mask shape, nor is it flexible enough to adapt to changing operating conditions even within a single frequency band, such as selection and deselection of subcarriers and adjustments in transmission power. Hence, enhanced techniques for controlling spectral leakage are required [19].

5) Data Interface: The interface between the baseband and CR engine is another important factor in the implementation of FPGA-based CRs. Many existing hardware radio platforms are extremely difficult to design with, and hence, only hardware experts can use them. While optimisation of low level blocks is necessary, this should be abstracted away from higher layers. A control interface to the cognitive engine needs to provide a high level view of the baseband and allow the higher layers to be designed in software with abstracted APIs supporting feedback from and control of the baseband layer.

IV. Proposed Multi-Standard OFDM Baseband

A. System Outline

The structure of the proposed OFDM system is illustrated in Fig. 2. The baseband modulation implementation must be able to transcieve non-contiguous OFDM (NC-OFDM) signals, as required for a multi-standard radio, supporting different OFDM symbol lengths and frame formats specified according to multiple standards such as IEEE 802.11, 802.16, and 802.22 [17], [16], [15], as summarised earlier in Table I.

The CR architecture is divided into a control plane and a data plane. The data plane tranceives data streams to/from the RF front end through two AXI (Advanced eXtensible Interface) stream interfaces. For transmission, data is sent from higher layers and modulated by the data plane. Then modulated sample streams are transferred to the RF front-end to convert to analogue signals and subsequently be up-converted to RF before transmission. For the receiver, the received signal is down-converted to the baseband followed by analogue to digital conversion to form the incoming sample stream. This is then demodulated and processed in the data plane before the data is transferred to the higher layer for processing.

To support multiple standards, the data plane modules must be flexibly designed to support different standards. In cases where the hardware overhead for parametrisation is below a threshold equivalent to the maximum area of stand-alone modules, it is judged as better parametrised. For those modules that require significant changes (i.e. the parameterised overhead would be greater than this threshold) or are required to support unspecified parameters for future standards (such as the preamble), PR is the preferred option. Separate bitstreams for each implementation are generated for mapping to a PR region. Hence, when switching the entire baseband from one standard to another, only part of the FPGA needs to be reconfigured. AXI-Stream interfaces are used for inter-module communication in the data plane as well as for communication with the higher layers. The AXI-Stream protocol also reduces the requirement for buffering, hence optimising resource usage and total power consumption [20]. Within the processing structure, each module has one slave interface to receive data from the previous module and one master interface to send processed data to the subsequent module.

The baseband data plane is designed to operate in burst mode where it transmits packets as soon as data is available, as well as using flow control to buffer data synchronously. If a data packet is ready to be processed but reconfiguration is in progress, it is stored in a FIFO buffer, then flushed out for processing as soon as reconfiguration is complete. Since the resource requirements for the transmitter are significantly less than the receiver, reconfiguration time is also much shorter, and there is consequently no loss of transmitted packets in the case of PR. Hence, for the transmitter, a single monolithic PR region is adopted to support the different standards. The much larger receiver subsystem, by contrast, needs to continuously receive and process data packets. The longer reconfiguration time would cause packet loss unless mitigated by buffering. Hence a large FIFO would be needed to store received samples from the RF front end during reconfiguration, but this is problematic since it results in significant additional resource usage. Combining PR and parametrisation, allows us to trade-off buffer requirements against parametrisation overheads and select a desired balance point.

The control plane contains a cognitive radio (CR) engine that is required to perform adaptation based on the requirements of the application, ultimately deciding which baseband standard to use at any point in time, and which sub-carriers to enable for each timeslot. An AXI-Lite register interface between the control and data planes allows status to be interpreted and parameters to be set using registers in the hardware fabric. The CR engine also includes the PR controller that is responsible for loading bitstreams stored in DRAM into the corresponding PR regions through the ICAP (Internal Configuration Access Port) interface [21] when necessary. Such a control plane can be implemented in a number of ways. It can be standalone software running on a soft processor core, or implemented in hardware in a separate part of the FPGA. By ensuring that symbol data is processed through a data plane independently of the control plane, we are able to achieve high throughput as well as abstract the data processing away from the control software. Note that the control plane implementation is out of scope for this paper, though the interface defined is generic enough to support different implementations. In the simulation, we have manually recreated the control pathways that would be provided by a full control plane, and generated the same control plane messaging to instruct and modify the baseband and module parameters.

The flexible OFDM baseband designed here differs from standard basebands in a number of ways. First, an allocation vector determines which sub-carriers to enable, allowing the radio to respond to varying channel occupancy conditions. When the frequency band of the current operating mode is mostly occupied by PUs and IUs, the CR engine can instruct the baseband to switch to another standard or frequency band

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2756914, IEEE Access

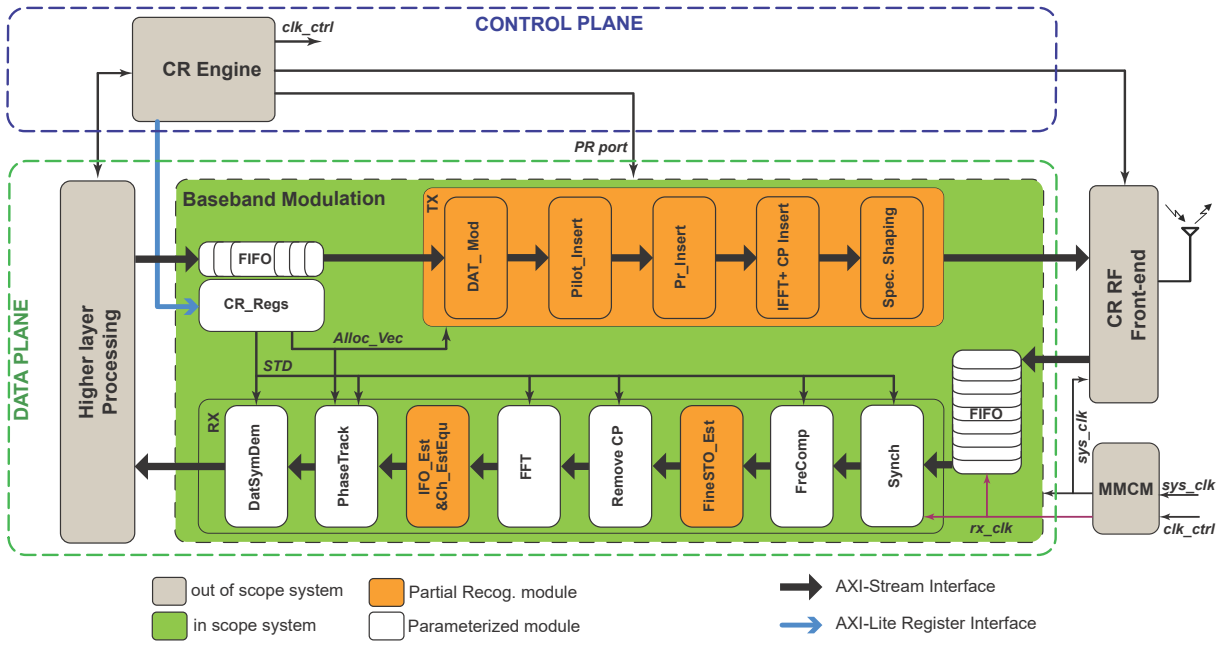

Fig. 2. Structure of the reconfigurable and parameterised multi-standard CR system.

that is currently (or will soon be) free for transmission. The PR controller inside the CR engine downloads the bitstreams of the PR modules required for the new standard. In the meantime, the CR engine configures parametrised modules and sets relevant parameters in the CR_Regs register such as allocation vectors ($Alloc\_vec$), symbol modulation type (MOD), and standard (STD).

The transmission hardware is much simpler than the receiver so will not be detailed here, save to mention incorporation of the advanced spectral shaping approach [19] used to minimize interference around selected subcarriers as needed for a multi-standard CR. We now discuss the receiver baseband modules briefly, indicating how they are designed to be flexible.

B. Synchronisation

To support burst mode, the CR system must detect the presence of a frame and estimate the frequency offset required based only upon the available burst data, i.e. just using the received frame preamble. The module therefore performs estimation based on the timing parameters, defined in [22],

$$\begin{aligned} P(d) &= \sum_{m=0}^{L-1} (r^*_{d+m} r_{d+m+L}) \\ R(d) &= \sum_{m=0}^{L-1} |r_{d+m+L}|^2 \end{aligned} \quad (1)$$

where d denotes a time sequential index of received samples r, L is the periodic length of the short preamble, * denotes complex conjugation.

In the proposed approach, length L is parametrised so that the module can effectively support the three standards as well as other combinations that might be needed in future standards.

TABLE II
SUPPORTED PARAMETER SPACE, WITH THE CONFIGURATIONS OF THE THREE SUPPORTED STANDARDS SHOWN.

NFFT	L=16	32	64	128	256	512
64	802.11					
128						
256			802.16			
512						
1024						
2048						802.22

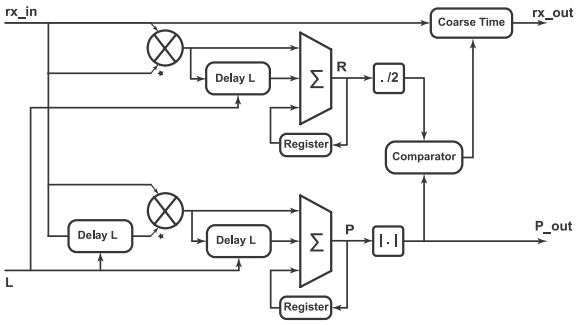

Fig. 3. Block diagram of Synchronisation module.

This is shown in Table II, which maps the configuration space in terms of FFT sizes and lengths and identifies the positions of the three target standards. Fig. 3 shows a block diagram of its parametrised implementation. The timing metrics are calculated using auto-correlation on received samples. *Coarse Time* detects the new frame and roughly estimates the start of a frame using blind estimation that provides generality to support multiple standards.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2756914, IEEE Access

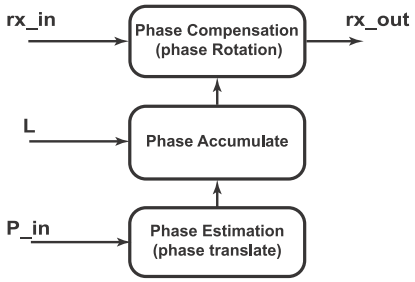

Fig. 4. Block diagram of frequency compensation module.

C. Frequency Compensation

The *FreComp* module performs fractional carrier frequency offset (CFO) estimation based on the value of P passed from the *Synch* module. Fractional CFO estimation and compensation are defined as,

$$\begin{aligned} \widehat{\Delta f} &= \frac{\angle P}{2\pi L} \\ \widehat{r(d)} &= r(d)e^{-j2\pi\widehat{\Delta f}d} \end{aligned} \tag{2}$$

where $\widehat{\Delta f}$ is the estimated fractional CFO, $\angle P$ denotes the angle of P, and N is the FFT length. In hardware implementation, a phase rotation sub-module is used to compensate fractional CFO by rotating the received sample phase by the correct angle. This is calculated and accumulated based on estimated fractional CFO.

$$\begin{aligned} \phi(d) &= \phi(d-1) + \frac{\angle P}{L} \\ \widehat{r(d)} &= r(d)e^{-j\phi(d)} \end{aligned} \tag{3}$$

According to (3), the computation of *FreComp* depends on the periodic length of the short preamble that is used to calculate P. Assuming that L is normally defined with a power of two value, the division by L can be computed using a right shift. Fig. 4 illustrates the proposed block diagram of parametrised module for frequency compensation. This module can be effectively implemented to support the required standards by using a shifter that supports these value of L.

D. Fine STO Estimation

FineSTO_Est estimates the starting sample for each OFDM symbol. Supporting multiple standards leads to a potentially large CFO that includes integer as well as fractional (or fine) CFO. The *FineSTO_Est* module uses the algorithm in [18] that is robust to high integer CFO. The metric for fine STO estimation is:

$$S(d) = \sum_{m=0}^{L-1} |r_{d+m+L}|^2 |a_m|^2, \tag{4}$$

when $|a_m|$ denotes the normalised amplitude of the preamble at the transmitter. Fig. 5 presents the proposed block diagram of the fine STO estimation module. *Peak Detect* finds the maximum value of the correlation that is employed to accurately estimate the STO and *Fine Time* determines the exact first sample of the next OFDM symbol (long preamble symbol). The metric defined in (4) is calculated based on a

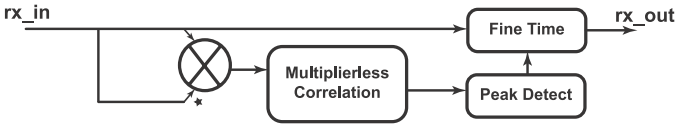

Fig. 5. Block diagram of fine STO estimation module.

multiplierless correlation between received samples and the transmitted preamble [23] since cross correlation using full multiplication would be extremely costly in terms of resources. Multiplierless correlation cannot support multiple standards in its most efficient form since the hardware implemented depends entirely on the fixed preamble, which is different for each standard. Thus fine STO Estimation is implemented using a PR module. Each supported standard results in a separate partial bitstream that must be reconfigured in the PR region at runtime by the *CR engine* when the underlying standard is switched. Future standards can be supported after deployment by simply generating the required partial bitstream based on the defined preamble.

E. Remove Cyclic Prefix

RemoveCP removes the cyclic prefix attached to each OFDM symbol, and this module depends on the length of CP L_{CP} that is different for each standard. *RemoveCP* consists of a counter to count from the beginning of each symbol and remove the CP samples if the counted value is smaller than L_{CP} This module is parametrised by adjusting L_{CP} to support multiple standards.

F. FFT

We use the highly efficient Xilinx FFT/IFFT IP core which supports modification of the FFT length at runtime to cater for different standards. When the length is changed, *FFT* is modified using the relevant input and the change completes within a few clock cycles. The module always occupies an area sufficient for the largest FFT size required, but its flexibility means minimal reconfiguration time.

G. IFO Estimation and Channel Equalisation

IFO_Est&Ch_EstEqu corrects IFO and performs channel equalisation. IFO results in a cyclic shift in the frequency domain. The IFO can be determined with robustness to frequency selective channels using correlation on the second (long) preamble [24], [25],

$$\hat{\epsilon} = \underset{\tilde{\epsilon}}{\operatorname{argmax}} \left| \sum_{k=0}^{N_{FFT}-1} Y^*(k-1)Y(k)X^*(k-\tilde{\epsilon})X(k-1-\tilde{\epsilon}) \right| \tag{5}$$

where ϵ denotes the value of IFO, $\hat{\epsilon}$, $\tilde{\epsilon}$ are estimated and trial values of ϵ, respectively, $Y(k)$ and $X(k)$ denote the k^{th} frequency symbol index of the received subcarriers and the known transmitted preamble, respectively, and the OFDM symbol size N_{FFT} is equal to the FFT size. Fig. 6 presents the proposed block diagram of the IFO estimation and channel equalisation module. *IFO Correction* is performed by cyclically shifting OFDM symbols corresponding to the estimated

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2756914, IEEE Access

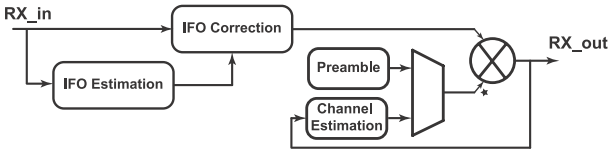

Fig. 6. The block diagram of IFO estimation and channel equalisation

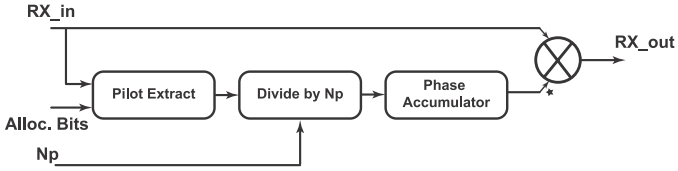

Fig. 7. Block diagram of phase tracking module.

IFO. After compensating for IFO, the effect of the channel can be estimated using information from the second preamble symbol. The estimation and compensation of channel and residual effects can be expressed as:

$$\begin{aligned} Y(k) &= X(k) * H(k) + N(k) \\ \hat{H}(k) &= \frac{Y(k)}{X(k)} \qquad (6) \\ \hat{R}(k) &= \frac{R(k)}{\hat{H(k)}} \qquad (7) \end{aligned}$$

$H(n)$ represents the channel effect and $N(k)$ is the AWGN. The equalization taps are estimated in (6), and the compensation for received data carriers is given in (7) in which $R(k)$, $\hat{R}(k)$ denote received and compensated data carriers, respectively. Since QPSK sub-carrier modulation is used for this proof of concept, amplitude is not a concern. Thus the complex division of channel estimation and compensation can be equivalently performed by multiplying by the conjugation of $X[k]$ and $\hat{H[k]}$, respectively.

This module depends on the second preamble that is specified differently for each standard. Hence, the *IFO_Est&Ch_EstEqu* module is implemented as a PR module to obtain effective standard-specific implementations. Again, this requires that the CR engine load the required partial bitsream at runtime.

H. Phase Tracking

PhaseTrack estimates the residual common phase error in each OFDM symbol after channel equalisation [26]. The estimation is computed on pilot symbols inserted in the OFDM symbol. The transmitted pilots are typically assigned the values $\{\pm 1\}$. The residual phase error causes a phase rotation on received pilots and is computed as

$$P_{k,l} = cos\theta_l - \alpha.k.sin\theta_l + j(sin\theta_l - \alpha.k.cos\theta_l), \qquad (8)$$

where $P_{k,l}$ denotes the phase of the received pilot which has frequency index k in the l^{th} OFDM symbol. $cos\theta_l + jsin\theta_l$ is the residual common phase error of the l^{th} OFDM symbol, and α is the slope of the phase distortion. The residual common phase error is generally estimated for the supported standards using,

$$\begin{aligned} cos\theta_l &= \frac{1}{N_P} \sum_{k \in S_P} \Re\{P_{k,l}\}, \qquad (9) \\ sin\theta_l &= \frac{1}{N_P} \sum_{k \in S_P} \Im\{P_{k,l}\}, \end{aligned}$$

where N_P denotes the number of received pilots employed for estimation, S_P is the set of used pilot frequency indices.

According to (9), N_P can be parametrised to support multiple standards. Fig. 7 presents the proposed block diagram of the parametrised PhaseTrack module. *Pilot Extract* finds the employed pilots for phase tracking in the OFDM symbol based on the allocation vector ($Alloc.\ Bits$). *Phase Accumulator* computes the residual common phase error. The phase error is compensated for by multiplying the data carriers by the complex conjugate of the estimated common phase error.

I. Data Symbol Demodulation (DatSymDem)

In the final step, the received bits are extracted from the data symbol by a demodulation block named *DatSymDem*. In the present implementation, this only supports QPSK modulation but can be extended to support different data symbol modulations such as 16-QAM or 64-QAM in future, using the same basic interface and parametrisation. All data symbols go through this module, and 2 bits are assigned to the output according to the signed bits of the real and imaginary parts of the data symbol.

J. FIFO buffer (FIFO)

FIFOs are needed in the transmitter and in the receiver to buffer data received from the higher layer and the RF front end, respectively. The FIFO buffers are implemented using Xilinx FIFO IP cores that efficiently use BlockRAMs, with 2 port AXI stream interface configuration. During normal operation, one data word is written to the buffer by the higher layer/RF front end, while one is read out from the buffer by the transmitter/receiver in each system clock cycle. Therefore, the FIFO buffers normally operate in an almost empty state. When reconfiguration is required to switch baseband standard, transmit and receive processing are temporarily suspended and the FIFOs store incoming data streams. The buffer for the transmitter is less critical than the receiver since transmission operates in burst mode with gaps between frames, however the receiver must process continuously in order to detect incoming frames, perform synchronisation, and correction. This means that there is little spare time to flush data that has been buffered in the receive FIFOs during reconfiguration. Therefore, the receiver FIFO is configured with independent clocks as shown in Fig. 8 to allow the clock manager to increase the receive processing rate (rx_clk) to be higher than the sampling rate of the RF front end to help empty the FIFO faster. Once the receiver FIFO is almost empty the processing rate is returned to the sampling rate to reduce power consumption. Since the FIFO IP cores support independent write and read clocks, this functionality is seamless to the stream processing.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2756914, IEEE Access

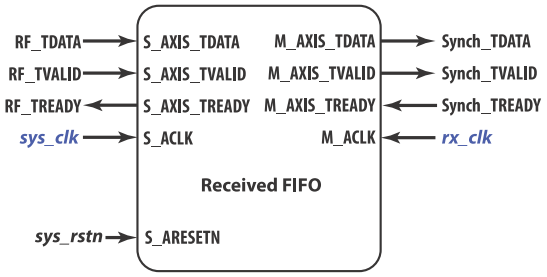

Fig. 8. Receiver FIFO module.

K. Mixed-Mode Clock Manager (MMCM)

To support this flushing of FIFOs and "catch up" with the delayed received samples, the clock rates need to be adjusted. A Mixed-Mode Clock Manager (MMCM) is used for this purpose, which is able to temporarily increase the processing clock (rec_clk) frequency. The sampling clock rate is also adjusted as required to switch between standards. Increasing operating speed is done by instructing the MMCM module to adjust the frequency of the input clock of the receiving modules, *rx_clk*. Because these modules, when implemented on FPGA fabric, are able to operate at a frequency higher than $2\times$ the operating clock, no additional resources are consumed in supporting this increased frequency. We benefit from continual improvements in FPGA architecture that mean higher frequencies are achievable than typically required for baseband processing, often even greater than $2\times$ normal rate.

Doubling the operating clock rate after changing the standard allows the samples stored in the FIFO during reconfiguration to be processed. The duration between two mode switches must be longer than the sum of the halting time and the system reconfiguration time. The results presented in Section V show that the sum of the halting time and the system reconfiguration time for the proposed approach is less than 3 ms. Practically speaking, this is more than sufficient for the target standards (as well as for many other OFDM standards) and thus the proposed approach is sufficiently agile for a multi-standard baseband.

Once the FIFOs are empty, the clock speed is reduced back to the sampling rate (sys_clk) to reduce power consumption. As mentioned, the MMCM module also needs to change the processing rate of the data plane according to the various sampling rates specified by different standards, shown in Table I.

V. Performance Analysis

The individual modules and baseband configurations were tested for functional correctness through detailed simulation and comparison with MATLAB prototypes. These served to provide the implementation definition as well as a source for both random and non-random test vectors for each module. Common input test vectors were used as inputs to OFDM modulation in both implementations and testbench scripts stored resultant output vectors that were compared in MATLAB. The performance of the specific algorithms used for synchronisation was evaluated in MATLAB with both AWGN and frequency selective channels, as discussed in [23], [18], [25].

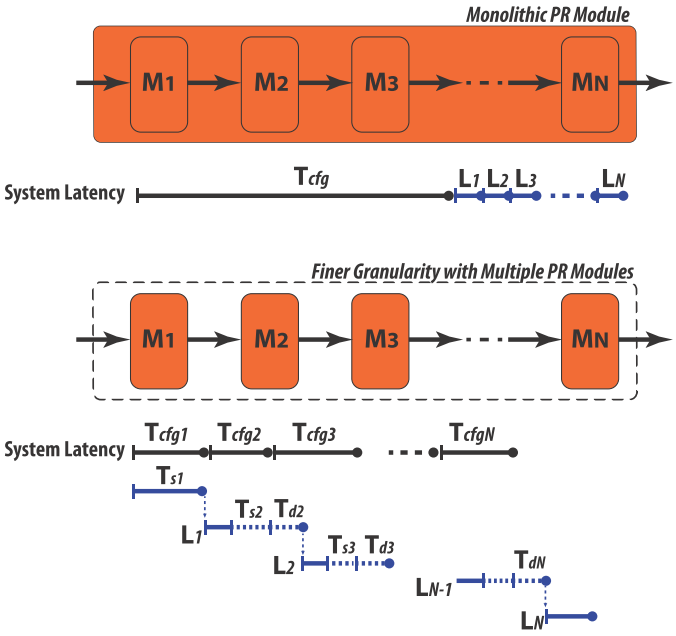

Fig. 9. Block diagram illustrating the effect of reconfiguration size on system latency.

The focus of this paper is on the reconfiguration capability, so we will now explore this in detail. We are interested in trade-offs between reconfiguration time and area consumption, and how this affects the characteristics of the flexible baseband.

A. Analysing Latency and Halting Time of PR Modules

One crucial challenge when implementing CR systems on reconfigurable hardware is long reconfiguration time when modifying the baseband configurations. PR can take a significant amount of time, especially for a large monolithic module. In the case of a monolithic CR receiver, the system would have to stall during reconfiguration, potentially causing the loss of data packets and possibly even loss of synchronisation. A large FIFO would be needed to store the stream of received samples long enough to prevent data loss. A longer reconfiguration time would demand a larger FIFO, resulting in significantly increased hardware resource and power consumption. Another approach often taken is to leave frame losses to be dealt with by higher layers, however this makes sense only in situations where the transmitter and receiver are following compatible protocols.

We analyse the system to evaluate the latency for a monolithic PR design, as well as for a system employing a finer granularity with multiple PR modules, and a mix of PR and parametrised modules. Fig. 9 illustrates the system latency for a system with a monolithic PR module and for one with finer granularity having multiple smaller PR modules. We consider a system consisting of N modules. T_c refers to the reconfiguration time. The system or module cannot process data during its reconfiguration time. L_i is the computation latency of the n^{th} module. Received data can, of course, be

processed during the computation latency. In the case of a large monolithic PR module, the latency, L_{sys}, and halting time, T_{hlt} that require a FIFO to buffer the received data which would otherwise be lost, is calculated as

$$\begin{aligned} L_{sys} &= T_c + \sum_{i=1}^{N}(L_i) \\ T_{hlt} &= T_c, \end{aligned} \tag{10}$$

Finer granularity is possible by dividing the system into multiple sub-modules, each of which is implemented in a separate PR region. When a module completes configuration, it can process received data while the following module is configured. Note that only one module can be configured at a time due to the presence of only one configuration interface. Therefore, the system reconfiguration latency and halting time for the case of multiple PR modules can be calculated as

$$\begin{aligned} L_{sys} &= \sum_{i=1}^{N}(T_{ci}) + T_{dN} + L_N \\ T_{hlt} &= \sum_{i=1}^{N}(T_{si}), \end{aligned} \tag{11}$$

where T_{di} refers the processing delay of the following module and T_{si} is the stalling time to wait for configuration of the following module. If the computation latency of a module, L_i, is greater than the reconfiguration time of the following module, T_{ci+1}, the following module must delay operation by a duration T_{di} before it receives input data for processing. Otherwise, the previous modules are halted for duration T_{si} until the following module is completely configured. The following module begins processing data just after its configuration is done ($T_{di} = 0$).

$$T_{di} = \begin{cases} Max(T_{ci}, T_{di-1} + L_{i-1}) - T_{ci} & i = 2..N, \\ 0 & i = 1 \end{cases} \tag{12}$$

$$T_{si} = \begin{cases} T_{ci} - min(T_{ci}, T_{di-1} + L_{i-1}), & i = 2..N, \\ T_{c1} & i = 1 \end{cases} \tag{13}$$

Substituting the above into (11),

$$\begin{aligned} L_{sys} &= \sum_{i=1}^{N}(T_{ci}) + L_N + \\ &\quad (Max(T_{cN}, (Max(...) - T_{cN-1}) + L_{N-1}) - T_{cN}) \\ T_{hlt} &= \sum_{i=1}^{N}(T_{ci}) - \sum_{i=2}^{N}(min(T_{ci}, T_{di-1} + L_{i-1})), \end{aligned} \tag{14}$$

We can see that system reconfiguration latency and halting time in the case of multiple PR modules is theoretically reduced thanks to being able to overlap the reconfiguration and data processing periods. Practically, the reconfiguration times are usually significantly greater than the processing latencies. This leads to $T_{di} = 0$ and $min(T_{ci}, T_{di-1} + L_{i-1}) = L_{i-1}$ resulting in an approximated equation for (14):

$$\begin{aligned} L_{sys} &= \sum_{i=1}^{N}(T_{ci}) + L_N \\ T_{hlt} &= \sum_{i=1}^{N}(T_{ci}) - \sum_{i=1}^{N-1}(L_i), \end{aligned} \tag{15}$$

In addition, because of optimisations in hardware compilation, the overhead of partitioning into multiple PR modules leads to the fact that $\sum_{i=1}^{N}(T_{ci})$ is greater than T_c, in (10). Therefore, the system latency, L_{sys}, and halting time, T_{hlt} in (14) may be greater than that in (10). Generally, the finer granularity approach can only achieve efficiency in terms of the system latency and halting time, if the gain of overlapping the reconfiguration and data processing is greater than the overhead of partitioning into multiple PR modules.

Hence, we propose to mix PR modules and parametrised modules in our flexible baseband to obtain a significant reduction in system reconfiguration latency and halting time. Parametrised modules have the benefit of much faster switching between modes than with PR operation, but if the operating modes are very different, can result in a large area overhead. For each module in the processing chain, commonalities across different operation modes are analysed and for modules requiring only minor modifications to the datapath, parametrised versions are created. If the i^{th} module is parametrised, the configuration time of this module can be eliminated because the it can switch operating mode within a few clock periods. This approximately results in the following simplified equations:

$$\begin{aligned} T_{ci} &\approx 0 \\ T_{si} &\approx 0 \\ T_{di} &\approx T_{di-1} + L_{i-1}, \end{aligned} \tag{16}$$

The above equations show the increasing efficiency of overlapping reconfiguration and data processing leading to significant reduction in the system latency and halting time.

B. Full OFDM Baseband Analysis

We analyse the results of applying this method to the full receiver baseband of the proposed system, when implemented on a Xilinx Virtex 6 FPGA (XC6VLX240T). We compare a large monolithic PR module, finer granularity PR, and the proposed mixture of PR and parametrisation. Slot based PR is widely used and the only method supported by Xilinx and Altera tools, and hence we employ it. One if its limitations is that all resources in the slot are consumed by any module occupying the slot, regardless of whether it actually uses these components or not. The configuration time for a module also depends entirely on the slot size, even if it is using only a fraction of the slot's resources. There has been some research on alternative methods that reduce resource wastage by allowing more fine-grained reconfiguration, hence also reducing reconfiguration time [27]. However, these approaches support a limited number of FPGA devices, require significant engineering effort and expertise to port, and remain unsupported in official tool flows. Furthermore, these improvements may still not benefit overall reconfiguration latency because

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2756914, IEEE Access

this depends on worst case latency (i.e., the reconfiguration time of the largest components).

To determine the reconfiguration time of a PR module, we generated bitstreams for all modules required for our baseband design. The area of a PR region must satisfy the needs of the largest mode it will host. For the monolithic PR module, it is required that the PR module be able to contain the full 802.22 OFDM baseband implementation, which is the largest receiver implementation among the three target implementations.

Similarly for the fine-grained approach, the size of the PR modules is determined based on the module configurations in the 802.22 OFDM-based implementation. Table III reports the hardware resource usage for each sub-module and the full transmitter and receiver systems for 802.22 on the Virtex 6 device. M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8 denote the functional modules of the OFDM based system: synchronisation, frequency compensation, fine STO estimation, remove CP, FFT, IFO estimation and channel equalisation, phase tracking, data symbol demodulation, respectively. M_R, M_T are the monolithic receiver and transmitter sub-systems, respectively.

We determine the bitstream size for each functional block according to the number of occupied CLB, DSP, BRAM columns that provide sufficient required resources for the block in a rectangular region on the FPGA floorplan. This granularity is necessary to satisfy the requirements of the PR toolflow [28]. Fig. 10 illustrates the bitstream sizes of each PR module, which determines reconfiguration time. The bitstream sizes of the sub-modules are relatively small compared to the monolithic PR module for the receiver sub-system. The M_3 bitstream is the largest among the sub-modules and the bitstream size of the receiver is almost three times the size of the transmitter.

TABLE III
RESOURCES FOR 802.22 OFDM BASEBAND.

Module	Slices	DSPs	BRAMs
M_1(*Synch*)	498	5	0
M_2(*FreComp*)	474	4	0
M_3(*FineSTO_Est*)	2,414	0	0
M_4(*RemoveCP*)	23	0	0
M_5(*FFT*)	1,179	15	11
M_6(*IFO_Est&Ch_EstEqu*)	1,249	6	0
M_7(*Phasetrack*)	523	3	0
M_8(*DatSymDem*)	4	0	0
M_R(Receiver)	6,363	33	11
M_T(Transmitter)	1,668	15	11

The overall latency of the transmit and receive chains for the different standards is shown in Table IV and demonstrates the benefits of an FPGA hardware baseband. Such latencies cannot be achieved using a software baseband running on top of an operating system or even real-time operating system.

Processing latencies for the individual modules are shown for the three standards in Fig. 11. We can see that 802.11 has the shortest latency because this standard uses the shortest FFT length, and hence the shortest symbol length for OFDM modulation.

It should be noted that during this latency the module still receives input data for processing. The processing chain must be halted when the latency elapses but the reconfiguration

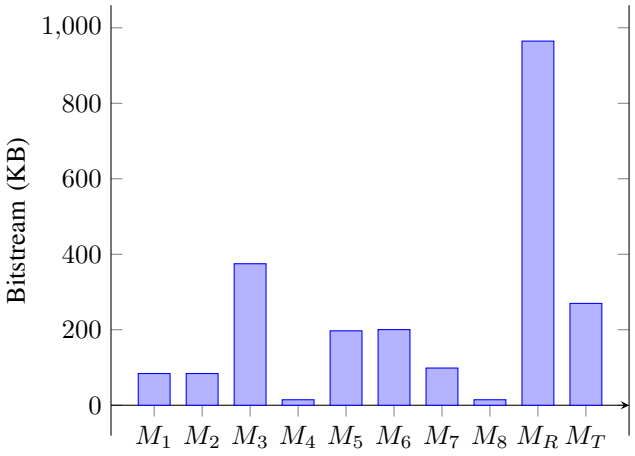

Fig. 10. Bitstream sizes for the PR modules.

TABLE IV
BASEBAND CHAIN LATENCIES IN µs

Baseband chain	802.11	802.16	802.22
Transmitter	22.5	54.3	463.6
Receiver	71.9	153.5	1446.6

of the subsequent module has not yet been completed, as discussed in Section V-A. The configuration time of a PR module is determined based on the size of PR region regardless operating standard. The PR region must be able to contain the largest module among the three target modules for three standards. Therefore, the worst case halting time occurs for the case of switching baseband to an operating standard with functional modules with the shortest latency.

Partial reconfiguration is performed using a high throughput ICAP controller that supports a data rate of 380 Mbps, close to the theoretical limit of the FPGA [21]. We use a sampling frequency of 10 MHz (i.e. clock period of 0.1 µs that is typically defined for the 802.11p standard. Compute latency is calculated for the 802.11 standard, as shown in

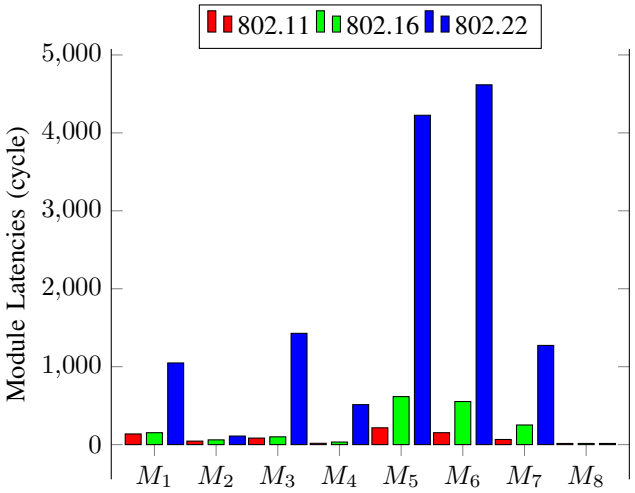

Fig. 11. The latency of sub-modules for the three standards.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2756914, IEEE Access

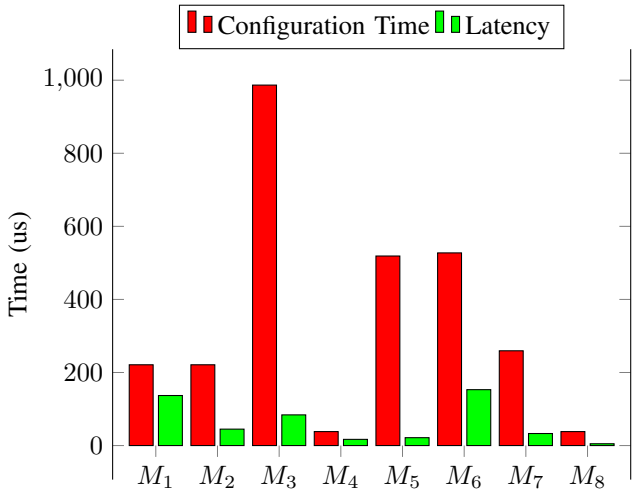

Fig. 12. The configuration time and latency of sub-modules for OFDM-based CR.

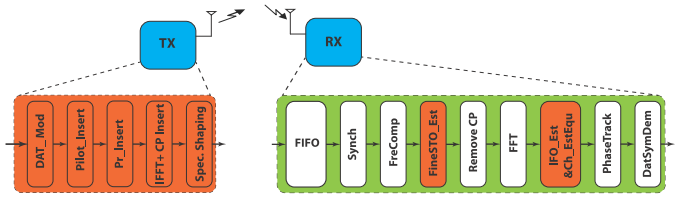

Fig. 13. Transmission scenario requiring reconfiguration.

Fig. 12. As can be seen, the module latency is very small in comparison to configuration time in all cases. It is thus clear that overlapping reconfiguration and data processing is not sufficient to overcome reconfiguration delay completely, and so the finer granularity approach may not improve significantly over a monolithic PR module.

The system halting time is an accumulated value of the halting time in each module described in (13). During the halting time, the processing chain is halted and a FIFO is required to buffer input samples. Because the halting time of the synchronisation module depends on the time when a new frame is detected, the timing offset must be taken into account. Given the transmission scenario shown in Fig. 13, when a standard switch is required, both the transmitter and receiver take time to reconfigure the system to the new operating standard. In the proposed receiver, the synchronisation module is a parametrised module, so reconfigures quickly – within a few clock periods – and hence quickly process input samples. However, a new frame cannot be sent quickly because the transmitter is still being reconfigured, resulting in a timing offset in the receiver. It is thus reasonable that the minimum timing offset can be chosen as the configuration time of the transmitter with hardware characteristics reported in Table III.

C. Comparison with Conventional Approaches

We have investigated the proposed approach in terms of halting time, FIFO capacity requirements, and reconfiguration latency compared to conventional approaches such as using

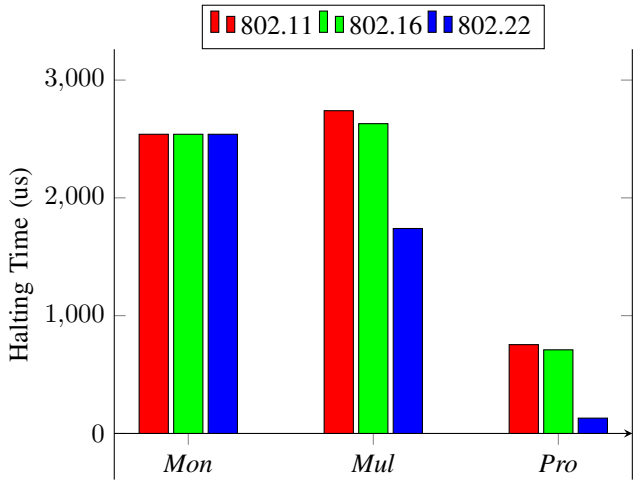

Fig. 14. Halting time comparison of the three approaches.

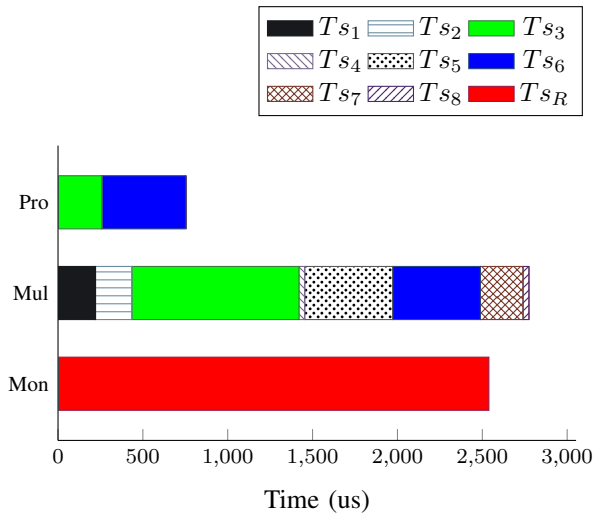

Fig. 15. Halting time breakdown for the three approaches.

multiple PR modules and monolithic PR for the whole chain as typically used in FPGA-based dynamic radios [8], [11]. It is worth noting that even these PR approaches remain somewhat specialist due to the design challenge associated with PR design on FPGAs. *Mon*, *Mul*, *Pro* denote the results for the monolithic PR approach, the multiple PR approach, and the proposed approach, respectively.

Fig. 14 shows the system halting time of the three approaches when the baseband changes operating standard to 802.11, 802.16, and 802.22. The longest halting time of the three approaches is when the baseband is switched to operate in 802.11 standard. Fig. 15 presents the halting time of functional modules in the three approaches for the worse case (i.e. the longest halting time). Ts_n is the halting time of the corresponding M_n functional module. Ts_R is the halting time of the monolithic receiver sub-system.

We can see from Fig. 15 that the halting time of the multiple PR baseband is in fact greater than that of the monolithic PR baseband, because the gain achieved by overlapping the reconfiguration and data processing is less than the area

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2756914, IEEE Access

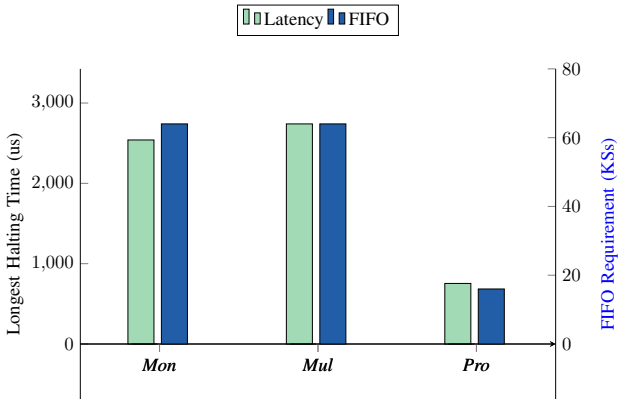

Fig. 16. Comparison of three approaches in terms of the longest halting time and FIFO buffer capacity requirements.

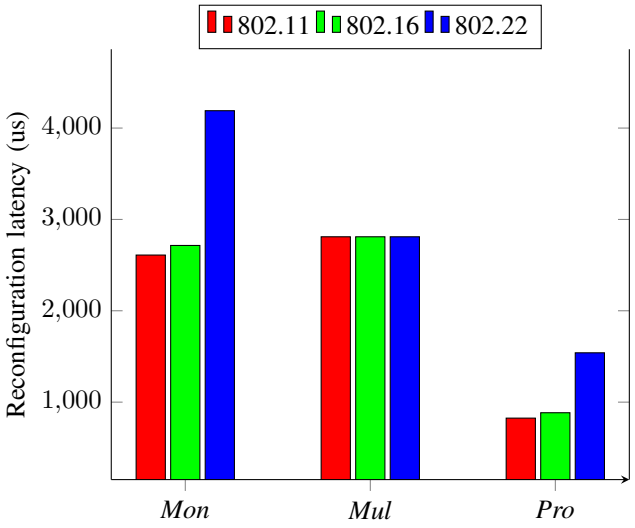

Fig. 17. System reconfiguration latency for the three approaches.

overhead of partitioning into multiple PR modules.

This PR area overhead is due to the use of multiple PR regions. Each functional module in the processing chain is assigned to one PR region, even when it is occupies just part of that region. Functional modules can be configured to one of three choices for the three operating standards. The PR region is defined by the largest module among the three. This leads to the sum of multiple PR regions being larger than when using a monolithic PR region. Hence, the halting time of the fine-grain PR approach can actually end up longer than that of the monolithic PR approach. This comparison is mathematically expressed in (10) and (15).

In the proposed approach, some functional modules in the processing chain are implemented by parametrised modules instead of PR modules, leading to elimination of the reconfiguration time for these PR modules. This is mathematically illustrated in (16), and is important in ensuring that the proposed approach improves on the alternatives. In fact, the proposed approach significantly reduces halting time, to less than a third of that of the monolithic PR module approach. This results in a significant reduction in FIFO buffer requirements to cover the halting time.

Fig. 16 compares the three approaches in terms of longest halting time and FIFO capacity requirements. The longest halting time is the halting time when the baseband is switched to the 802.11 standard. The FIFO is required to buffer received data in the duration of the longest halting time to avoid losing data. The FIFO capacity requirement is calculated based on multiplying the sampling frequency by the longest halting time, followed by rounding up to the next power of two, as required for the FIFO buffer IP resources on Xilinx FPGAs. Table V reports required resources for 32-bit AXI4 interface FIFO implementation with some different available size configurations. The FIFO requirement for the proposed approach is only 16 kilo-samples (KS) while the two other approaches require a FIFO which must store up to 64 KS.

Fig. 17 compares the three approaches in term of reconfiguration latency in cases of switching operating standard to 802.11, 802.16, and 802.22. As can be seen, the reconfiguration latencies of *Mon*, *Pro* for the case of 802.22 is longest compared to the case of 802.11, and 802.16 because the 802.22 standard has the longest OFDM symbol (which requires long processing latencies). However, for *Mul*, the reconfiguration latencies are the same for the three cases because the processing latency of the functional modules is smaller than their reconfiguration time. In particular, the proposed approach decreases reconfiguration latencies in comparison to others approaches. For 802.22, the reconfiguration latency is reduced by 0.63% and 0.45% compared to *Mon* and *Mul*, respectively. In the case of 802.11, reconfiguration latency is significantly reduced by 68% and 71% compared to *Mon* and *Mul*, respectively.

VI. Conclusion and Future Works

This paper proposed a design for an efficient multi-standard OFDM baseband for FPGA based cognitive radios. Individual blocks in the transmit and receive chains are designed to support different standard requirements, and improved synchronisation and transmission shaping are incorporated. The proposed system combines parametrized modules and partially reconfigured modules to achieve flexibility while minimizing reconfiguration time. We show that this mixture results in a significant reduction of 71% compared to conventional approaches in terms of system reconfiguration latency. To avoid data loss, FIFO buffers are used to store data during reconfiguration, and we show that the proposed approach reduces storage requirements to 25% of other PR approaches. The interface to the higher layer processing was also discussed and shown to be compatible with different implementations

TABLE V
Memory resources for 32 bit AXI4 interface FIFOs.

FIFO size (KSs)	18Kb BRAMs	36Kb BRAMs
8	1	7
16	1	14
32	0	29
64	0	58
128	0	116

of cognitive engines. In future work, we aim to incorporate this baseband design into a full CR platform to enable radio designers to take advantage of this flexibility in dynamic radio experiments.

REFERENCES

[1] Y. Liao, L. Song, Z. Han, and Y. Li, "Full duplex cognitive radio: a new design paradigm for enhancing spectrum usage," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 138–145, May 2015.

[2] S. K. Sharma, T. E. Bogale, S. Chatzinotas, B. Ottersten, L. B. Le, and X. Wang, "Cognitive radio techniques under practical imperfections: A survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 1858–1884, Fourthquarter 2015.

[3] Free Software Foundation, Inc. (2009) GNU Radio - the GNU software radio. [Online]. Available: http://www.gnu.org/software/gnuradio/

[4] J. Dowle, S. H. Kuo, K. Mehrotra, and I. V. McLoughlin, "FPGA-based MIMO and space-time processing platform," in *EURASIP Journal of Applied Signal Processing special issue on MIMO implementation*, no. 34653, Jan. 2006, pp. 1–14.

[5] G. J. Minden *et al.*, "KUAR: A flexible software-defined radio development platform," in *Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, Dublin, Ireland, 17-22 Apr. 2007.

[6] K. Amiri, Y. Sun, P. Murphy, C. Hunter, J. R. Cavallaro, and A. Sabharwal, "WARP, a unified wireless network testbed for education and research," in *Proceedings of the IEEE International Conference on Microelectronic Systems Education*, 2007, pp. 53–54.

[7] P. Sutton, J. Lotze, H. Lahlou, S. Fahmy, K. Nolan, B. Özgül, T. Rondeau, J. Noguera, and L. Doyle, "Iris – an architecture for cognitive radio networking testbeds," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 114–122, Sep 2010.

[8] J. Delorme, J. Martin, A. Nafkha, C. Moy, F. Clermidy, P. Leray, and J. Palicot, "A FPGA partial reconfiguration design approach for cognitive radio based on NoC architecture," in *Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference (NEWCAS-TAISA)*, June 2008, pp. 355–358.

[9] P. Athanas, J. Bowen, T. Dunham, C. Patterson, J. Rice, M. Shelburne, J. Surıs, M. Bucciero, and J. Graf, "Wires on Demand: run-time communication synthesis for reconfigurable computing," in *Proceedings of International Conference on Field Programmable Logic and Applications*, 2007.

[10] J. Lotze, S. A. Fahmy, J. Noguera, B. Ozgul, L. Doyle, and R. Esser, "Development framework for implementing FPGA-based cognitive network nodes," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, 2009.

[11] S. Shreejith, B. Banarjee, K. Vipin, and S. A. Fahmy, "Dynamic cognitive radios on the Xilinx Zynq hybrid FPGA," in *Proceedings of the International Conference on Cognitive Radio Oriented Wireless Networks*, 2015, pp. 427–437.

[12] H. A. Mahmoud, T. Yucek, and H. Arslan, "OFDM for cognitive radio: merits and challenges," *IEEE Wireless Communications*, vol. 16, no. 2, pp. 6–15, 2009.

[13] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 1, pp. 116–130, 2009.

[14] Y. Zhang and C. Leung, "Resource allocation in an OFDM-based cognitive radio system," *IEEE Transactions on Communications*, vol. 57, no. 7, pp. 1928–1931, 2009.

[15] *IEEE Standard 802 Part11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE std.802.11-2005.

[16] *IEEE Standard for Local and Metropolitan Area Networks Part16: Air Interface for Fixed Broadband Wireless Access Systems*, IEEE std.802.16-2009.

[17] *IEEE Standard for Wireless Regional Area Networks Part22:Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Policies and Procedures for Operation in the TV Bands*, IEEE std.802.22-2011.

[18] T. H. Pham, I. V. McLoughlin, and S. A. Fahmy, "Robust and Efficient OFDM Synchronization for FPGA-Based Radios," *Circuits, Systems, and Signal Processing*, pp. 1–19, 2014.

[19] T. H. Pham, S. A. Fahmy, and I. V. McLoughlin, "Spectrally efficient emission mask shaping for ofdm cognitive radios," *Digital Signal Processing*, vol. 50, pp. 150–161, 2016.

[20] Q. Liu, G. Constantinides, K. Masselos, and P. Cheung, "Data-reuse exploration under an on-chip memory constraint for low-power FPGA-based systems," *IET Computers & Digital Techniques*, vol. 3, no. 3, pp. 235–246, 2009.

[21] K. Vipin and S. Fahmy, "A high speed open source controller for FPGA Partial Reconfiguration," in *International Conference on Field-Programmable Technology (FPT)*, Dec 2012, pp. 61–66.

[22] T. Schmidl and D. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613–1621, Dec. 1997.

[23] T. H. Pham, S. A. Fahmy, and I. V. McLoughlin, "Low-power correlation for IEEE 802.16 OFDM synchronisation on FPGA," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 8, pp. 1549–1553, August 2013.

[24] M. Park, N. Cho, J. Cho, and D. Hong, "Robust integer frequency offset estimator with ambiguity of symbol timing offset for OFDM systems," in *Proceedings of the Vehicular Technology Conference (VTC)*, 2002, pp. 2116–2120.

[25] T. H. Pham, S. A. Fahmy, and I. V. McLoughlin, "Efficient integer frequency offset estimation architecture for enhanced OFDM synchronization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 4, pp. 1412–1420, 2016.

[26] A. Troya, K. Maharatna, M. Krstic, E. Grass, U. Jagdhold, and R. Kraemer, "Efficient Inner Receiver Design for OFDM-Based WLAN Systems: Algorithm and Architecture," *IEEE Transactions on Wireless Communications*, vol. 6, no. 4, pp. 1374–1385, April 2007.

[27] A. Sohanghpurwala, P. Athanas, T. Frangieh, and A. Wood, "OpenPR: An Open-Source Partial-Reconfiguration Toolkit for Xilinx FPGAs," in *IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, May 2011, pp. 228–235.

[28] K. Vipin and S. A. Fahmy, "Architecture-aware reconfiguration-centric floorplanning for partial reconfiguration," in *Reconfigurable Computing: Architectures, Tools and Applications: Applied Reconfigurable Computing (ARC)*, 2012.