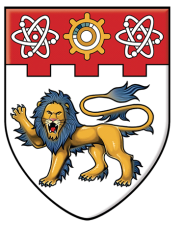

NANYANG
TECHNOLOGICAL
UNIVERSITY

POWER OPTIMISATION FOR ADAPTIVE EMBEDDED WIDEBAND RADIOS

A Thesis submitted to
the School of Computer Engineering
of Nanyang Technological University

by

Pham Hung Thinh

in fulfillment of the requirement for
the Degree of Doctor of Philosophy of
for Computer Engineering

March 25, 2015

List of Figures

List of Tables

Chapter 1

Research Introduction

Chapter 2

Background Literature

Chapter 3

Multiplierless Correlator Design for low-power systems

Chapter 4

A Method for OFDM Timing Synchronisation

Chapter 5

A CFO Estimation Method for OFDM Synchronisation

Chapter 6

A Spectrum Efficient Shaping Method

Chapter 7

An Architecture for Multi-Standard Cognitive Radios

7.1 Introduction

Cognitive radios that support multiple bands, multiple standards and adapt operation according to environmental conditions are becoming more attractive as the demand for higher bandwidth and more efficient spectrum use increases. Traditional implementations in custom ASICs cannot support such flexibility, with standards changing at a faster pace, while software implementations of baseband communications fail to achieve the performance required. Hence, FPGAs offer an ideal platform bringing together flexibility, performance, and efficiency. In this chapter, we show how we can incorporate our contributions in previous chapters into a flexible architecture for multi-standard cognitive radios (MSCR). We show that combining partial reconfiguration (PR) with parameterised modules offers flexibility while minimising reconfiguration time. To the best of our knowledge, we have not encountered work on dynamically reconfigurable OFDM baseband processing for multiple standards.

The work presented in this chapter has also been discussed in:

- T. H. Pham, S. A. Fahmy, and I. V. McLoughlin, "Efficient Multi-Standard Cognitive Radios on FPGAs," PhD Forum Poster in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL), Munich, Germany, September 2014* [1].

- T. H. Pham, S. A. Fahmy, and I. V. McLoughlin, "Efficient OFDM-based baseband processing for Multi-Standard Cognitive Radios on FPGAs," in preparation for submission

to *ACM Transactions on Embedded Computing Systems*.

7.2 Related Work

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. GNU Radio [2] has been a widely used platform in academia. It is a software application that runs on a computer or an embedded ARM processor platform, e.g. on the Ettus USRP E100. 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. Other software based frameworks like Iris [3], have some limited support for FPGAs but suffer from poor bandwidth between software and hardware. Moreover, the compilation time and reconfiguration time of software defined radio systems can be long making them unsuitable for adapting to fast changing conditions.

In an application area with fast moving standards and requiring support for multiple standards, custom ASIC implementation is unlikely to be agile or cost effective enough to cope with fast-changing standards and operating requirements. In order to address this, Delorme et al. [4] presented a heterogeneous reconfigurable hardware platform for Cognitive Radio. It can adapt its hardware structure to support standards like GSM, UMTS, and wireless LAN. Most processing components run as embedded software on the nodes in a network-on-chip processor, while the channel coder and the mapping of the RX chain are implemented inside an FPGA. Partial reconfiguration (PR) is used to switch the channel coder from one context to an another depending on SNR. A processor manages data movement between the different processors, the ASIC, and the FPGA. The need for a large data buffer and inefficient data transfer mechanisms results in increased power consumption and reduced throughput.

Other platforms do not propose specific methods for managing a radio's dynamic behaviour. KUAR [5] is a mature radio platform built around a fully-featured Pentium PC with a Xilinx Virtex II FPGA. The baseband processing is accelerated on FPGA and limited to NC-OFDM waveforms based on the IEEE 802.16 standard. Projects at Virginia Tech [6] have shown dynamically assembled radio structures on FPGAs, where the target radio system is defined at

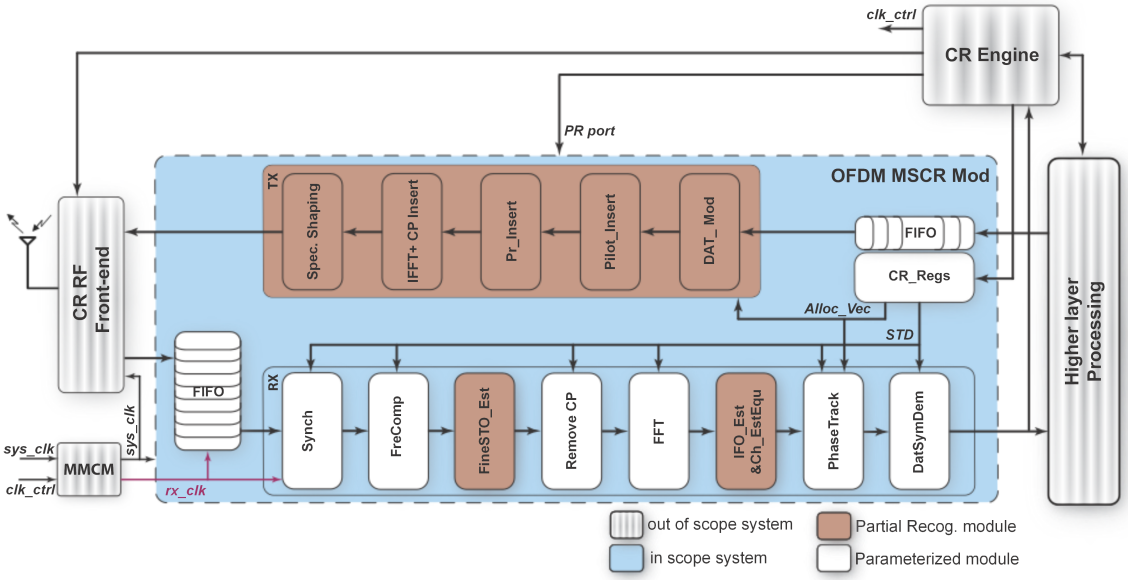

Figure 7.1: The structure of a generic MSCR system

a high-level with datapaths connecting relatively large functional modules. The modules are wrapped, and each of them consists of a PR module with complied partial bit-streams stored in dynamic library. Their online assembly method eliminates the need for run time compilation, thus affording flexibility. A flexible radio controller can insert and remove compiled modules to adapt to current conditions. However, by using low-level reconfiguration circumventing the official PR flow, it is unclear how this approach can be mapped to newer architectures as they appear.

7.3 Proposed OFDM Baseband for MSCR

Fig. 7.1 illustrates the proposed structure of baseband modulation for our OFDM-based MSCR. A mix of partially reconfigurable and parameterised modules make up the baseband implementation. FIFOs are included to help overcome the reconfiguration latency when PR modules are reconfigured. While a sink module is reconfiguring, these FIFOs store samples from the source until the module is ready to process data. Since these modules are now just a small part of the system, buffering is significantly reduced over a more general implementation.

7.3.1 System Description

We developed a prototype MSCR baseband that supports transmitting and receiving non-contiguous OFDM (NC-OFDM) signals. This system can perform with different OFDM symbol lengths and frame formats specified differently according to multiple standards such as IEEE 802.11 [7], IEEE 802.16 [8], and IEEE 802.22 [9]. The main specifications of these standards are summarised in Table 7.1.

Table 7.1: System specifications of three supported OFDM-based standards.

Specifications	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)	64	256	2048
CP Length	16	32	512
Number of data carriers	48	192	1440
Number of pilots	4	8	240

The CR implementation is divided into a control plane and data plane. The data plane performs data processing on the sample data stream. It 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. Modulated sample streams are then transferred to the RF front end to convert to analogue signals and subsequently up-converted before transmission on the RF channel. For the receiver side, the received signals are down-converted to the baseband followed by analogue to digital conversion to form a sample stream. The sample stream is demodulated and processed in the data plane before being transferred to a higher layer.

To support multiple standards, the data plane must provide the flexibility of switching baseband modules to the parameters specified by different standards. AXI Stream interfaces are used for inter-module communication in the data plane as well as for communication with software. With partial reconfiguration, modules that that swap into the same region must share the same interfaces, so this is a suitable abstraction. The AXI Stream protocol also reduces the

requirement for buffering, hence optimising resource usage and total power consumption [10]. 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 control plane is a cognitive radio (CR) engine that is required to perform adaptive based on the requirements of the application, ultimately deciding which base standard to use, and which sub-carriers to enable. A bus register interface between the control and data planes allows them to communicate. 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 [11] when necessary.

The control plane can be implemented in a number of ways. It can be standalone software running on a processor core. It can also be hardware in a separate part of the FPGA. Alternatively, for maximum flexibility and programming support, it can run on top of an operating system on the processor. By ensuring that symbol data is processed and moved through the data plane independently of the processor in the control plane, we are able to achieve high throughput. Normally, the data plane processes data streams based on a specified standard. 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 instructs the baseband to switch to another standard or frequency band that is currently (or will soon be) free for transmission. The PR controller inside the CR engine is required to download the bitstreams of PR modules according to the new standard. In the meantime, the CR engine configures the system by writing relevant parameters to registers in CR_Regs such as allocation vectors (Alloc_vec), symbol modulation type (MOD), and standard (STD).

The scope of this chapter focuses on the managing the baseband adaptation in a way that avoids long configuration time. We do not explore the cognitive functions, nor discuss sensing, which can be implemented in a variety of ways. The system takes advantage of combining PR with parameterised modules to offer flexibility while minimising reconfiguration time. The CR system is designed to be capable of operating in burst mode in which it transmits packets as soon as data is available [12]. If a data packet is ready to be sent but reconfiguration is in progress, it is buffered in a FIFO. It is then flushed out of FIFO for transmission after reconfiguration is completed. Since the resource requirements for the transmitter are significantly

less than the receiver, reconfiguration time is also shorter. We discuss the resource utilisation and reconfiguration time of the transmitter in Section 7.4. Short reconfiguration time means there is no critical disruption of the processing chain and loss of transmitted packets. Hence, for the transmitter, we use a single PR region with the whole transmission chain for simplicity and flexibility.

The receiver subsystem, by contrast, is required to continuously receive and process data frames. Furthermore, the hardware usage of the receiver subsystem is much larger than the transmitter, resulting in longer reconfiguration time. One way to avoid losing data frames, is to use a large FIFO to store received samples from the RF front end during reconfiguration, but this can be problematic as this results in significant resource usage. Our aim here is to reduce reconfiguration time in the receive chain to mitigate the need for such large FIFOs.

We propose making some modules parameterised, while others are reconfigured using PR. We explore how to optimise this mix in Section 7.3.2. Now smaller FIFOs can be used and reconfiguration can be applied at a finer granularity to minimise the impact of reconfiguration on buffer storage requirements. After reconfiguration is complete, it is necessary to flush the FIFOs and "catch up" with the received samples. This can be done by increasing the processing clock rate until the FIFOs are no longer full. A Mixed-Mode Clock Manager (MMCM) is used for this purpose, modifying the processing clock (rec_clk) to a higher frequency. It is then reduced back to the sampling rate (sys_clk) to reduce power consumption. The MMCM module is also required to change the processing rate of data plane according to the different sampling rates specified in different standards, shown in Table 7.1.

7.3.2 Module Description

In this section, we detail the design of each functional module shown in Fig. 7.1, showing how to build a multi-standard implementation. There is a tradeoff between the simplicity, flexibility, but long reconfiguration latency of a PR module and the increased hardware overhead, complexity, but faster configuration of a parameterised module. The comparison between the hardware usage of parameterised modules for multiple standards and that of individual specified standard modules is evaluated. In cases where the hardware overhead for parameterisation is a threshold of the maximum area of a stand-alone module, it is judged as better parameterised. For those modules that require significant changes (i.e. the parameterised overhead is

greater than this threshold) or are required to support unspecified parameters for future standards (such as the preamble), PR can be used on a per-module basis. Separate bitstreams for each implementation are generated to be mapped to a PR region. Hence, when switching the entire baseband from one standard to another, only part of the FPGA needs to be reconfigured.

FIFO Buffer (FIFO)

There are two FIFOs at the transmitter side and the receiver side to buffer data sent from the higher layer and the RF front end, respectively. The FIFO buffers are implemented using Xilinx FIFO IP cores with 2 port AXI stream interface configuration. In the normal operation of data stream, one data word is written to the buffer by the higher layer/the RF front end, while another one is read out from the buffer by the transmitter side/receiver side in each system clock cycle. Therefore, the FIFOs normally operate in an almost empty state. When reconfiguration is required to switch the baseband, the transmitter processing and receiver processing are suspended. The FIFO buffers store incoming data to avoid losing frames. Because the system performs in burst mode, the transmitter streams are not continuous, therefore, the transmitter FIFO can flush stored data during gaps between bursts. The data buffer is not a critical issue for the transmitter side.

The receiver must, however, process continuously to detect incoming frames, and so, there is no spare time to flush data that has been buffered in the FIFOs during reconfiguration. Therefore, the receiver FIFO is configured with independent clocks as shown in Fig. 7.2. In order to flush stored data in the received FIFO after reconfiguration, the MMCM increases the receiver 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 minimise power consumption. Since the FIFO IP cores support independent write and read clocks, this functionality is seamless to the stream processing.

Synchronisation (Synch)

Our CR system communicates in burst mode. Hence, is must detect the presence of a frame and estimate the frequency offset required, based upon the preamble of the received frame. The *Synch* module performs estimation as discussed in Chapter 4. Fig. 7.3 shows a block diagram of its implementation.

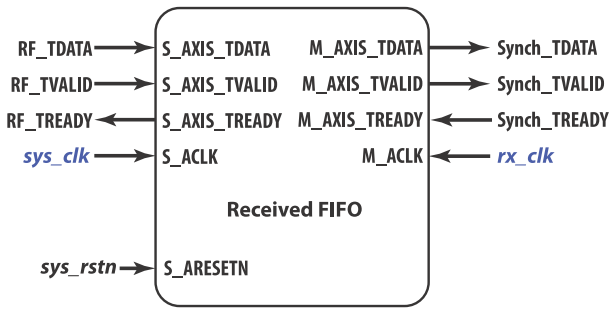

Figure 7.2: The receiver FIFO module.

The timing metrics are calculated by 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 for application to multiple standards. By parameterising the length of L, the synchronisation module can effectively perform for three current supported standards as well as being extensible for future standards.

This module is implemented as a parameterised version with parameter L whose values are defined together with the length of FFT (N), shown in Table 7.2. Combinations of L, and N allow it to support multiple standards. The parameterised values consist not only of the required combinations for 802.11, 802.16, and 802.22, but also support other combinations for future standards.

Table 7.2: Parameterised values according to supported standards

N \ L	16	32	64	128	256	512
64	802.11					
128						
256			802.16			
512						
1024						
2048						802.22

Frequency Compensation

The *FreComp* module performs fractional CFO estimation based on the value of P passed from the previous module *Synch*, as described in Chapter 4. Fractional CFO estimation and

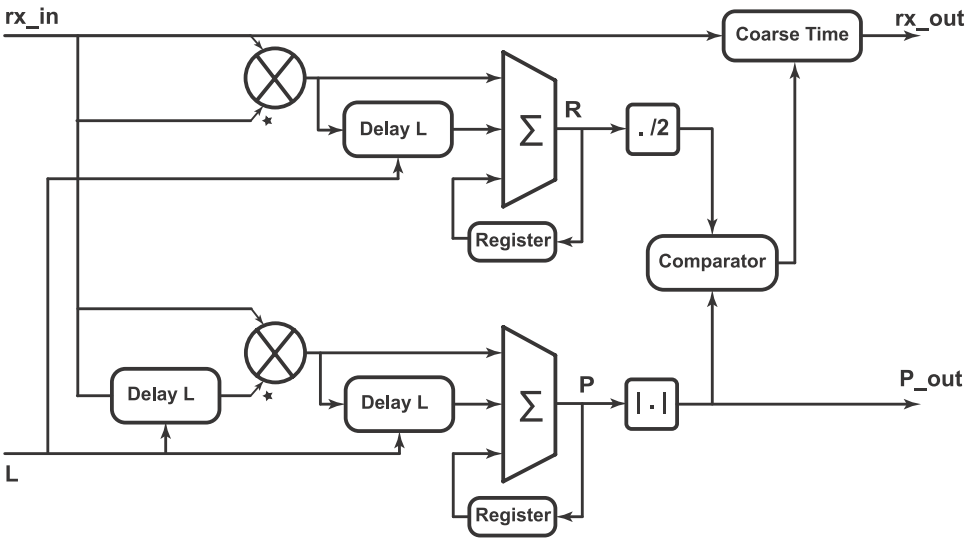

Figure 7.3: Block diagram of Synchronisation module.

compensation are mathematically expressed as:

$$\begin{aligned} \widehat{\Delta f} &= \frac{\angle P}{2\pi\frac{L}{N}} \\ \widehat{r[d]} &= r[d]e^{-j2\pi\widehat{\Delta f}\frac{d}{N}} \end{aligned} \tag{7.1}$$

where $\widehat{\Delta f}$ is estimated fractional CFO, $\angle P$ denotes the angle of P. Fig. 7.4 illustrates the process for frequency compensation. 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.

$$\begin{aligned} \phi[d] &= \phi[d-1] + \frac{\angle P}{L} \\ \widehat{r[d]} &= r[d]e^{-j\phi[d]} \end{aligned} \tag{7.2}$$

According to (7.2), the computation of *FreComp* depends on the periodic length of short preamble that is used to calculate P. Assuming that L is normally defined as a power of two value, the division by L can be effectively computed by a right shift. Therefore, this module can be effectively implemented to support multiple standards by parameterising a right shifter according to the value of L.

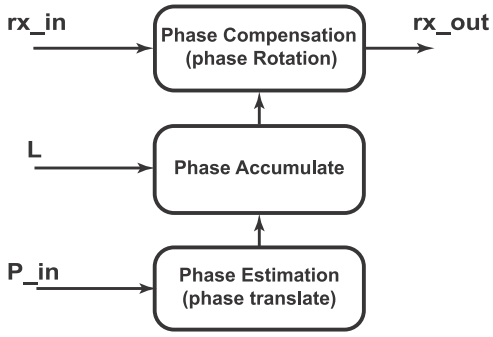

Figure 7.4: Block diagram of frequency compensation module.

Fine STO Estimation

FineSTO_Est estimates the starting sample of each OFDM symbol. The RF front-end for MSCR need to access a wide range of frequencies, shown in Table 7.1. Depending on the standard being used, the CFO may be large, resulting in the presence of IFO. Therefore, the implementation of *FineSTO_Est* is based on the algorithm presented in Chapter 5 that is also robust to IFO. The metric for fine STO estimation is expressed as:

$$S[d] = \sum_{m=0}^{L-1} |r[d+m+L]|^2 |a[m]|^2, \tag{7.3}$$

when $|a[m]|$ denotes the normalised amplitude of the preamble at the transmitter.

Fig. 7.5 shows the block diagram of fine STO estimation. The metric is calculated based on a multiplierless correlation between received samples and transmitted preamble. *Peak Detect* finds the maximum value of 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). However, multiplierless correlation is not flexible and the implementation depends on the preamble which is different for each standard. Therefore, we employ PR module for the FineSTO_Est module to obtain flexibility. Each supported standard is implemented separately to a bitstream to be reconfigured at runtime by the *CR engine* when the underlying standard is switched. The PR region must be large enough to house the largest bitstream among the supported standards.

Remove Cyclic Prefix

RemoveCP removes the cyclic prefix attached to each OFDM symbol. The performance of this module depends on the length of CP L_{CP} specified differently in each standards. *RemoveCP*

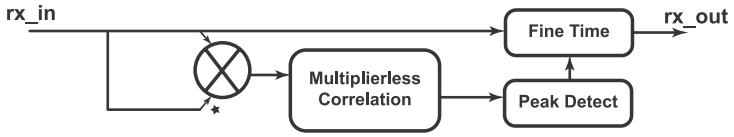

Figure 7.5: Block diagram of fine STO estimation module.

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 can be parameterised by adjusting L_{CP} to support multiple standards.

FFT

FFT is based on the Xilinx FFT/IFFT IP core, which supports modification of the FFT length at runtime to support different standards. When the standard is changed, the length of the FFT is modified using the relevant input. This reconfiguration completes within a few clock cycles.

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. IFO is determined based on the method in Chapter 5:

$$\hat{\epsilon} = \underset{\tilde{\epsilon}}{\operatorname{argmax}} \left| \sum_{k=0}^{N-1} Y^*[k-1]Y[k]X^*[k-\tilde{\epsilon}]X[k-1-\tilde{\epsilon}] \right| \tag{7.4}$$

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 is equal to the FFT size.

Fig. 7.6 illustrates the block diagram of IFO estimation and channel equalisation. *IFO Correction* is performed effectively by cyclically shifting OFDM symbol corresponding to the estimated IFO estimation.

After compensating for IFO, the effects of the channel and residual STO must be taken into account to compensate the received sub-carriers. Using information of the second preamble symbol, the effect of the channel can be estimated. The estimation and compensation of

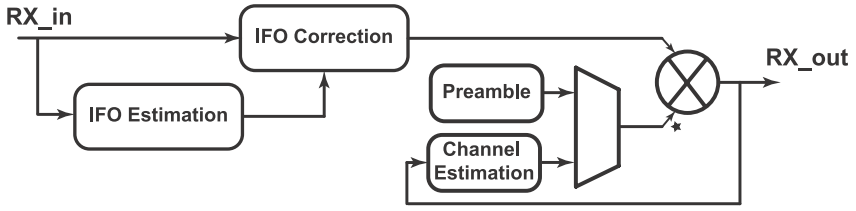

Figure 7.6: The block diagram of IFO estimation and channel equalisation

channel and residual effects can be expressed as:

$$\begin{aligned} Y[k] &= X[k] * H[k] + N[k] \\ H[k] &= \frac{Y[k] - N[k]}{X[k]} \\ \hat{H}[k] &= \frac{Y[k]}{X[k]} \qquad (7.5) \\ \hat{R}[k] &= \frac{R[k]}{\hat{H[k]}} \qquad (7.6) \end{aligned}$$

where $X[k]$, $Y[k]$ are the transmitted and received carriers in the preamble, respectively. $H[n]$ represents for the channel and residual STO effect and $N[k]$ is the AWGN. The equalization taps are estimated in (7.5), and the compensation for received data carriers is given in (7.6) in which $R[k]$, $\hat{R}[k]$ denote received and compensated data carriers, respectively.

Since we use QPSK sub-carrier modulation, amplitude is not a concern. So, 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. Therefore, PR is used for the *IFO_Est&Ch_EstEqu* module to obtain effective standard-specific implementations.

Phase Tracking

PhaseTrack estimates the residual common phase error in each OFDM symbol after channel equalisation. The *PhaseTrack* implementation is based on an algorithm presented by Troya et al. [13]. The estimation is computed on the pilot symbols inserted in the OFDM symbol. The transmitted pilots are typically assigned the values ± 1. The residual phase error causes a phase rotation on received pilots.

$$P_{k,l} = cos\theta_l - \alpha.k.sin\theta_l + j(sin\theta_l - \alpha.k.cos\theta_l), \qquad (7.7)$$

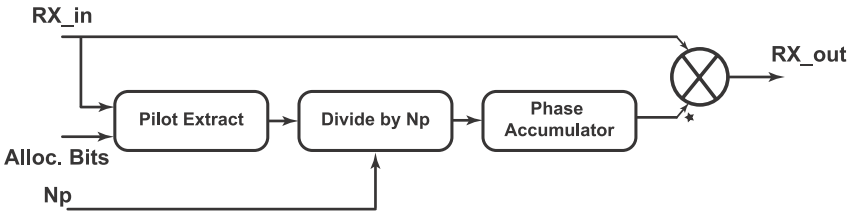

Figure 7.7: Block diagram of phase tracking module.

where $P_{k,l}$ denotes the phase of 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 as below:

$$\begin{aligned} cos\theta_l &= \frac{1}{N_P}\sum_{k\in S_P}\Re\{P_{k,l}\}, \\ sin\theta_l &= \frac{1}{N_P}\sum_{k\in S_P}\Im\{P_{k,l}\}, \end{aligned} \tag{7.8}$$

where N_P denotes the number of received pilots employed for estimation, S_P is a set of used pilot frequency indices. The Fig. 7.7 shows the block diagram of PhaseTrack. *Pilot Extract* finds the employed pilots for phase tracking in the OFDM symbol based on allocation vector ($Alloc.\ Bits$). *Phase Accumulator* computes the residual common phase error according to (7.8). The phase error is compensated for by multiplying the data carriers by the complex conjugate of the estimated common phase error. To support multiple standards, N_P is parameterised and S_P can be determined through the allocation vector with the value shown in Table 7.3

Table 7.3: Allocation vector coding.

Subcarrier Type	Allocation Bits
Null	00
Data	10
Positive pilot	01
Negative pilot	11

Data symbol demodulation (*DatSymDem*)

In the final step, the received bits are extracted from the data symbol by a data symbol 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. All data symbols go through this module, and 2 bits are assigned to the output according to the sign bits of the real and imaginary parts of data symbol.

7.4 Performance Analysis and Discussion

7.4.1 Latency and Stalling for PR-Based Baseband

One crucial challenge when implementing CR systems on reconfigurable hardware is long reconfiguration time when modifying the baseband. PR can take a significant amount of time, especially for a large monolithic module. In the case of the CR receiver, the system would have to stall during reconfiguration, potentially causing the loss of data packets and possibly even the loss of synchronisation. A large FIFO would be required to store a stream of received samples long enough to overcome the reconfiguration latency when a large PR module is reconfigured. A longer reconfiguration time would demand a larger FIFO, resulting in significantly increased hardware resource and power consumption.

We evaluate baseband latency for a monolithic PR module, as well as for a system employing a finer granularity with multiple PR modules, and a mix of PR and parameterised modules for the case when the system switches to a new baseband. Fig. 7.8 illustrates this latency. We consider a system consisting of N modules. T_c refers to the reconfiguration time. We assume that a module can not process data during its reconfiguration time. L_i is the computation latency of the i^{th} module. Received data can, of course, be processed during the computation latency. In the case of a large monolithic PR module employed for the system, the system reconfiguration latency, L_{sys}, and halting time, T_{hlt} that would require a FIFO to buffer received data which would otherwise be lost, is calculated as follow:

$$\begin{aligned} L_{sys} &= T_c + \sum_{i=1}^{N}(L_i) \\ T_{hlt} &= T_c, \end{aligned} \tag{7.9}$$

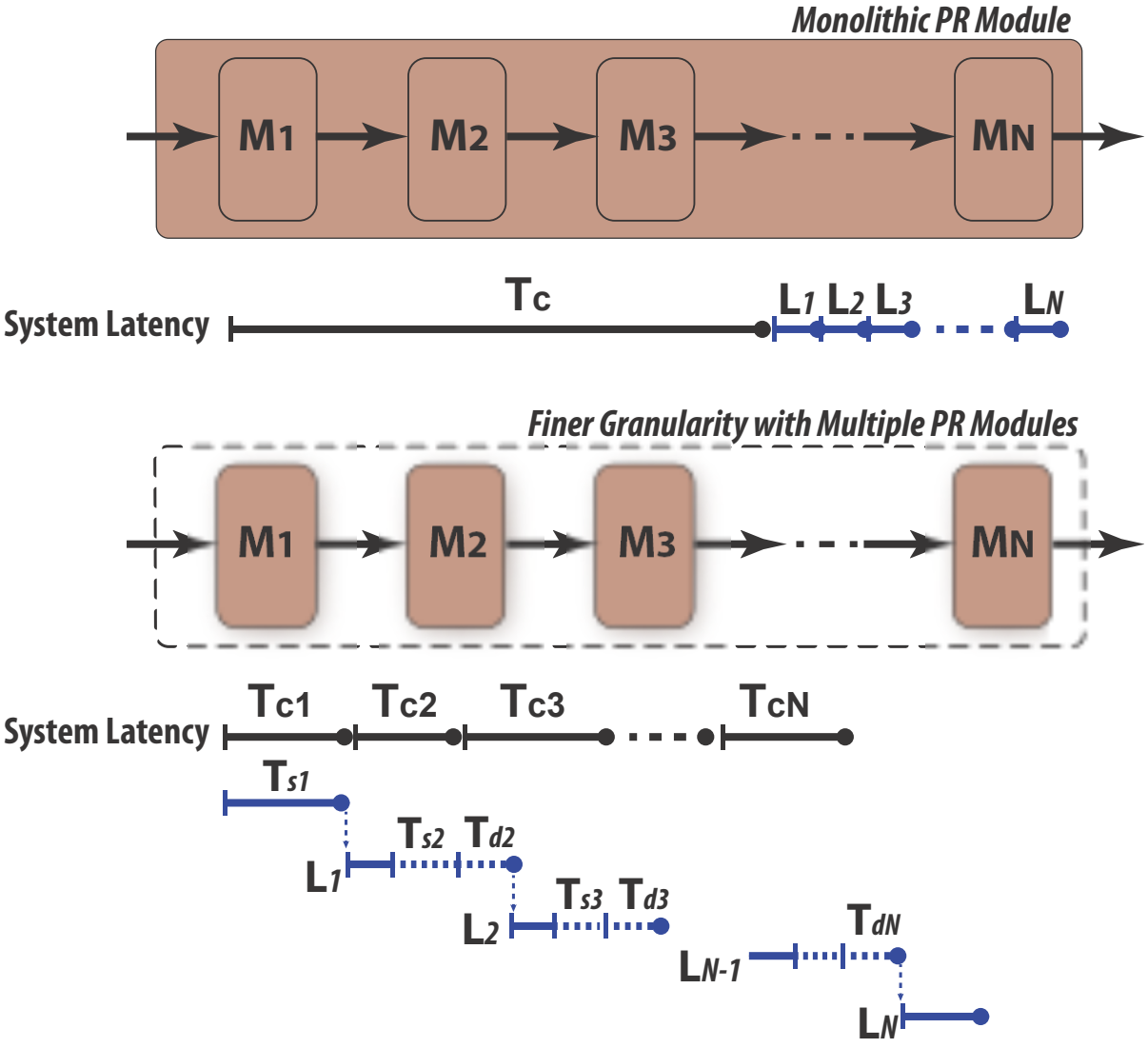

Figure 7.8: Comparison of reconfiguration latency for a single and multiple PR modules.

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 is completely configured, it can process the received data while the following module is configured. Therefore, the system reconfiguration latency and halting time for the case of multiple PR modules can be calculated as follow;

$$\begin{aligned} L_{sys} &= \sum_{i=1}^{N}(T_{ci}) + T_{dN} + L_N \\ T_{hlt} &= \sum_{i=1}^{N}(T_{si}), \end{aligned} \tag{7.10}$$

where T_{di} refers to 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 has to delay operation by a duration T_{di} before it receives input data for processing. Otherwise, the previous module is halted a duration T_{si} until the following module is completely configured. The following module begins processing data just after its configuration is done ($T_{di} = 0$).

$$T_{di} = \begin{cases} Max(T_{ci}, T_{di-1} + L_{i-1}) - T_{ci} & i = 2..N, \\ 0 & i = 1 \end{cases} \tag{7.11}$$

$$T_{si} = \begin{cases} T_{ci} - min(T_{ci}, T_{di-1} + L_{i-1}), & i = 2..N, \\ T_{c1} & i = 1 \end{cases} \tag{7.12}$$

Substituting the above equations into (7.10),

$$\begin{aligned} L_{sys} &= \sum_{i=1}^{N}(T_{ci}) + L_N + \\ &\quad (Max(T_{cN}, (Max(...) - T_{cN-1}) + L_{N-1}) - T_{cN}) \\ T_{hlt} &= \sum_{i=1}^{N}(T_{ci}) - \sum_{i=2}^{N}(min(T_{ci}, T_{di-1} + L_{i-1})), \end{aligned} \tag{7.13}$$

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 the approximated equations for (7.13) as below:

$$\begin{aligned} L_{sys} &= \sum_{i=1}^{N}(T_{ci}) + L_N \\ T_{hlt} &= \sum_{i=1}^{N}(T_{ci}) - \sum_{i=1}^{N-1}(L_i), \end{aligned} \tag{7.14}$$

In addition, because of the optimisation in hardware compilation, the overhead of partitioning into multiple PR modules leads to the fact that $\sum_{i=1}^{N}(T_{ci})$ is clearly greater than T_c, in (7.9). Therefore, system reconfiguration latency, L_{sys}, and halting time, T_{hlt} in (7.13) may be greater than that in (7.9). Generally, the finer granularity approach is only efficient in terms of system reconfiguration 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.

Through the above analysis and comparison, we propose a new method employing a mix of PR modules and parameterised modules to obtain a significant reduction in system reconfiguration latency and halting time. For each module in the processing chain, commonalities across different operation modes are analysed. For modules requiring only minor modifications, parameterised versions are created. For the i^{th} module to be parameterised, the configuration time of this module can be eliminated because the parameterised modules can switch operating mode within one clock period. This approximately results in the following simplified equations;

$$\begin{aligned} T_{ci} &\approx 0 \\ T_{si} &\approx 0 \\ T_{di} &\approx T_{di-1} + L_{i-1}, \end{aligned} \tag{7.15}$$

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

7.4.2 Analysing the Proposed OFDM MSCR Approach

We analyse the results of applying this method to the full receiver baseband implemented on a xilinc Virtex 6 FPGA (XC6VLX240T). We compare a large monolithic PR module, finer granularity PR, and the proposed mixture of PR and parameterisation. To compute the configuration

time of a PR module, we generate bitstreams for all modes. The area of a PR region must satisfy the needs of the largest implementation it will house. For the monolithic PR module, it is required that the PR module be able to contain the 802.22 OFDM baseband implementation, which is the largest receiver implementation among the three target implementations.

Similarly for the fine-grained approach, the configurations of the PR modules are computed based on the sub-modules of the 802.22 OFDM-based implementation. Table 7.4 reports the hardware resource usage for each sub-module and transmitter, receiver system 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 then determine the bitsteam size for each functional block according to the number of occupied CLB, DSP, BRAM columns that provide sufficient required resources for the block on FPGA floorplan. Fig. 7.9 illustrates the bitstream sizes of each PR module, which we use to calculate configuration time later. 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. The bitstream of the receiver is nearly triple the size of the transmitter.

Table 7.4: Resources for 802.22 OFDM-based implementation

$modules$	Slices	DSP	BRAM
M_1(*Synch*)	498	5	0
M_2(*FreComp*)	474	4	0
M_3(*FineSTO_Est*)	2414	0	0
M_4(*RemoveCP*)	23	0	0
M_5(*FFT*)	1179	15	11
M_6(*IFO_Est&Ch_EstEqu*)	1249	6	0
M_7(*Phasetrack*)	523	3	0
M_8(*DatSymDem*)	4	0	0
M_R(Receiver)	6363	33	11
M_T(Transmitter)	1668	15	11

Processing latencies for functional modules are shown for the three standards in Fig. 7.10. We can see that 802.11 has the shortest latency because this standard uses the shortest FFT

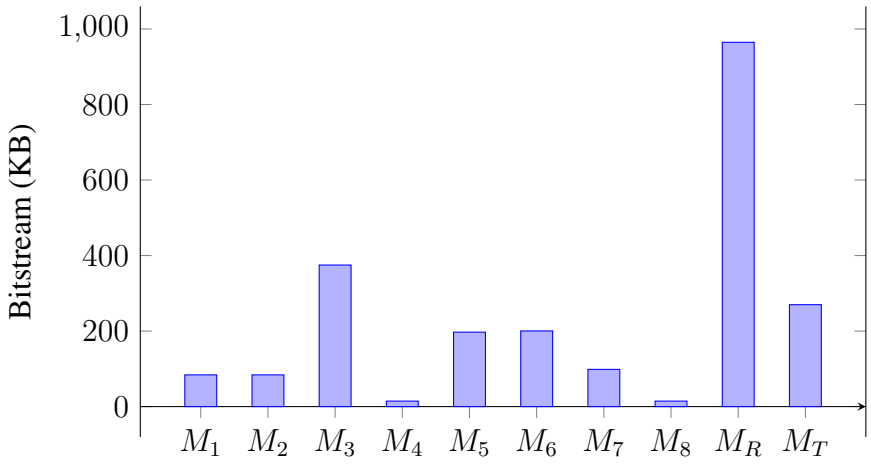

Figure 7.9: Bitstream sizes for PR modules.

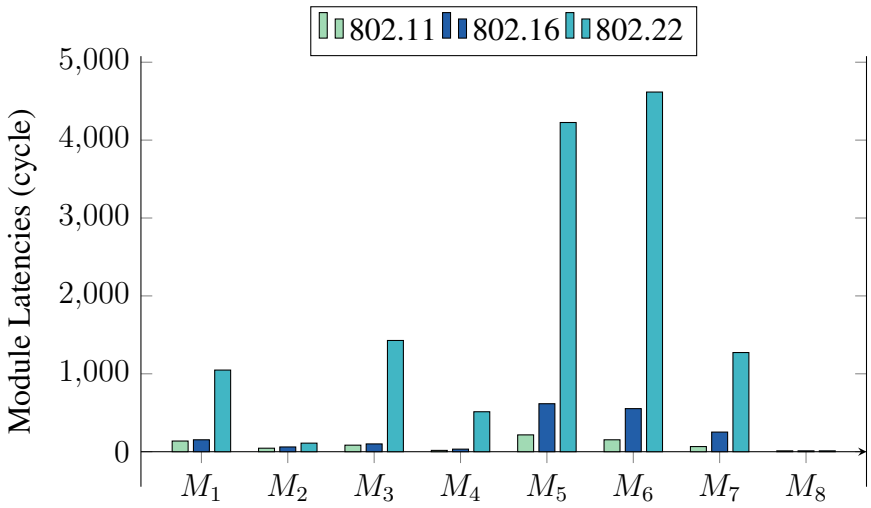

Figure 7.10: The latency of sub-modules for three standards

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 time has ended but the reconfiguration of the following module has not yet been completed. The worst case halting time is a case of the shortest latency and the longest reconfiguration time. Therefore, the latencies for 802.11 are used to calculate system halting time.

Partial reconfiguration is performed with a high throughput ICAP controller that supports a data rate of 380 MBps, closet to the theoretical limit of the FPGA. We use a sampling frequency of 10 MHz (i.e. the clock period = 0.1 us) that is typically defined for the 802.11p standard.

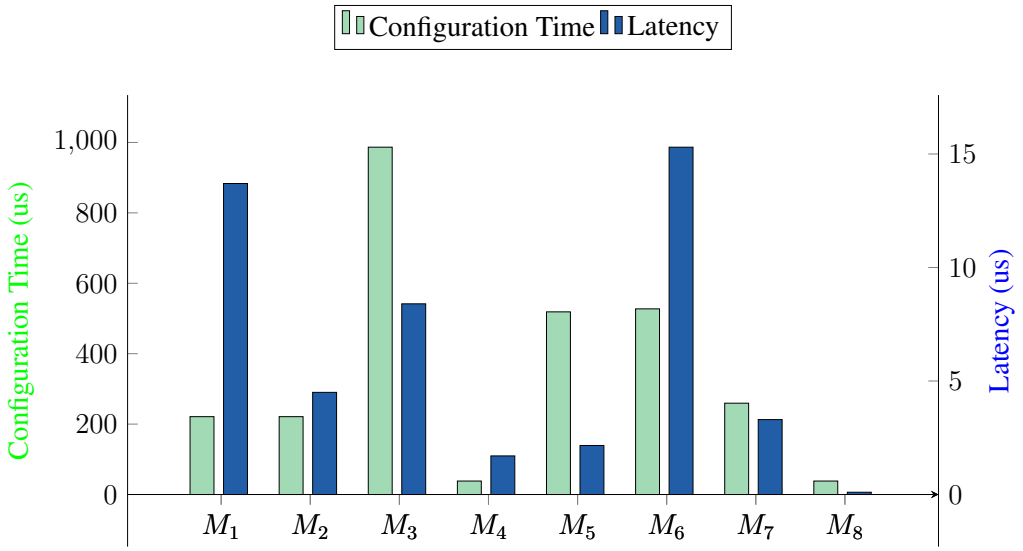

Figure 7.11: The configuration time and latency of sub-modules for OFDM-based MSCR system

Compute latency is calculated for the 802.11 standard, as shown in Fig. 7.11. As can be seen, the latency is very small in comparison to the configuration time.It is clear that overlapping reconfiguration and data processing is not sufficient to hide reconfiguration delay, and so the finer granularity approach may not improve significantly over the monolithic PR module.

The system halting time is an accumulated value of the halting time in each module described in (7.12). 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 a scenario of a transmission, shown in Fig. 7.12 when a standard switch is required, Both the transmitter and receiver spend time to reconfigure the system for the new baseband standard. In the proposed receiver, the synchronisation module is a parameterised module, so this module can change its function within a clock period and hence quickly process input samples. However, a new frame cannot be sent so 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 whose hardware characteristics were reported in Table 7.4.

Fig. 7.13 shows the system halting time of the three approaches. Mon, Mul, Pro denote the halting time of the monolithic PR module approach, the multiple PR module approach,

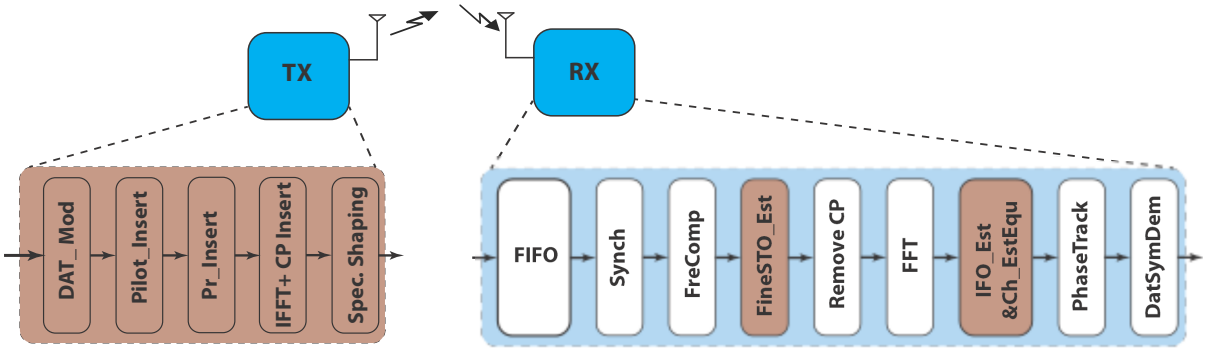

Figure 7.12: A scenario of a transmission

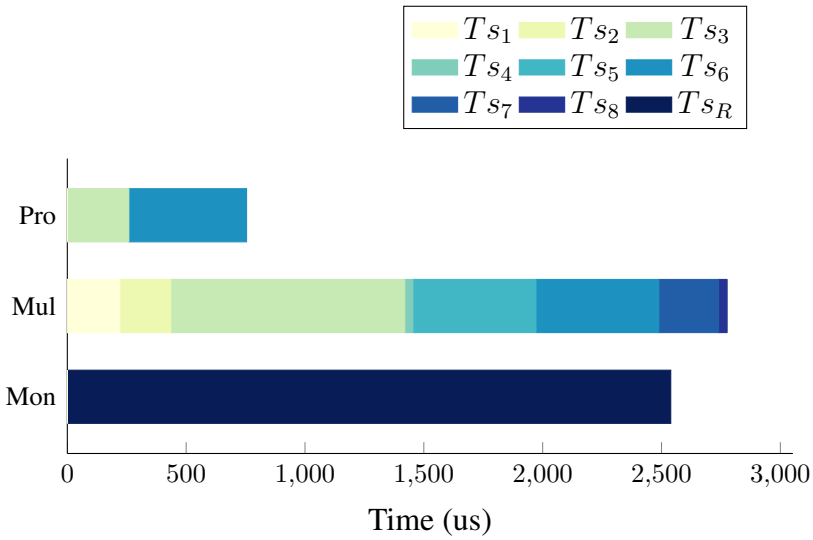

Figure 7.13: The halting time comparison of the system for three different approaches.

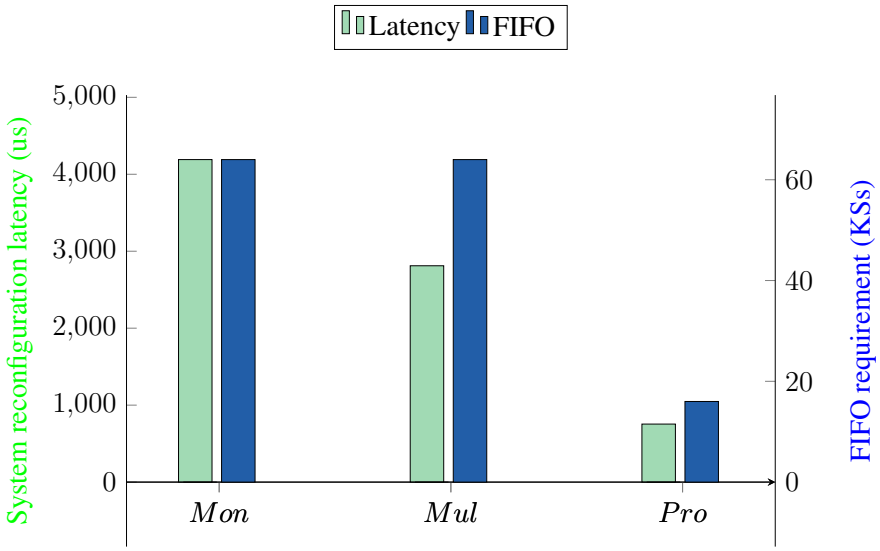

Figure 7.14: A comparison of the three approaches in terms of system reconfiguration latency and FIFO requirements.

and the proposed approach, respectively. 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 that the halting time of the multiple PR module approach is greater than that of the monolithic PR module approach, because the gain achieved by overlapping reconfiguration and data processing is less than the overhead incurred by partitioning into multiple PR modules. The proposed approach can significantly reduce the halting time to less than one-third of that of the monolithic PR module approach. This results in a reduction in FIFO buffering requirements.

Table 7.5: Memory resources for 32 bit AXI4 interface FIFOs implementation

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

Fig. 7.13 compares the three approaches in terms of system reconfiguration latency and FIFO buffering requirements. The system latencies are computed in the worst case which is the latency for the 802.22 standard. The FIFO requirements are calculated based on multiplying

the sampling frequency by the halting time, followed by rounding up to the next power of two. Table 7.5 reports required resources for 32 bit AXI4 interface FIFO implementation with some different availabe size configurations. The system reconfiguration latency of the proposed method is significantly reduced to 18 % and 27 % of the system reconfiguration latency of the monolithic PR module and the multiple PR module approaches, respectively. The FIFO requirement for the proposed approach is only 16 kilo-samples (KSs) while the other approaches require a FIFO to store up to 64 KSs.

7.5 Summary

We have shown that configuring a radio baseband composed of multiple modules can be made more efficient in terms of time, and buffering requirements, by mixing parameterisation and PR. Implement the whole baseband as a single PR module leads to long reconfiguration time, during which no processing can be performed, and hence significant FIFO buffering to prevent loss of samples in the receiver. Breaking the baseband chain into multiple PR modules could be beneficial as it would allow individual modules to start processing as the complete their reconfiguration, overlapping this processing with the reconfiguration of the next module. However, since reconfiguration time is significantly higher than the computational latency, we find that the benefit is outweighed by the overhead of multiple PR modules. Our proposed approach only applies PR to modules where different modes are very different in hardware, with other modules designed to be parameterisable. As a result, overall halting time is reduced by more than half. The FIFO buffering requirement of the proposed method is also decreased to 25 % of that required for the other methods. We are working on finalising a demonstrator for this approach.

Chapter 8

Conclusion and Future Work

References

[1] T. H. Pham, S. A. Fahmy, and I. V. McLoughlin, "Efficient Multi-Standard Cognitive Radios on FPGAs," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL)*, 2014.

[2] Free Software Foundation, Inc., "GNU Radio - The GNU Software Radio," 2009.

[3] 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, pp. 114–122, Sep 2010.

[4] 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 International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference (NEWCAS-TAISA)*, pp. 355–358, June 2008.

[5] G. J. Minden, J. B. Evans, L. Searl, D. DePardo, V. R. Petty, R. Rajbanshi, T. Newman, Q. Chen, F. Weidling, J. Guffey, D. Datla, B. Barker, M. Peck, B. Cordill, A. M. Wyglinsky, and A. Agah, "KUAR: A flexibile software-defined radio development platform," in *Proceedings of IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, pp. 17–22, Apr. 2007.

[6] 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 the International Conference on Field Programmable Logic and Applications (FPL)*, 2007.

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

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

[9] IEEE std.802.22-2011, *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*.

[10] 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.

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

[12] B. Ai, Z.-X. Yang, C.-Y. Pan, J.-H. Ge, Y. Wang, and Z. Lu, "On the synchronization techniques for wireless OFDM systems," *IEEE Transactions on Broadcasting*, vol. 52, pp. 236–244, June 2006.

[13] 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, pp. 1374–1385, April 2007.