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 18, 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

A Novel Architecture for Multiple 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. Our research focuses on designing the baseband processing of a MSCR system. This work explores the advantages of coupling PR and parameterised functional units in one architecture to offer flexibility for OFDM-based baseband processing while minimising reconfiguration time. To the best of our knowledge, there is no published research on dynamical reconfiguration for OFDM-based baseband processing of multiple standards cognitive radio on FPGA.

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," to be prepared.

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 based system are quite long for CRs may not be suitable for adapting fast condition changing requirements.

In an application area with fast moving standards and requiring support for multiple standards, custom ASIC implementation is both 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 lead to increased power consumption and reduced throughput. There are few studies on FPGA based platforms for radio implementation. 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 only limited for NC-OFDM signal 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 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. Using PR 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.
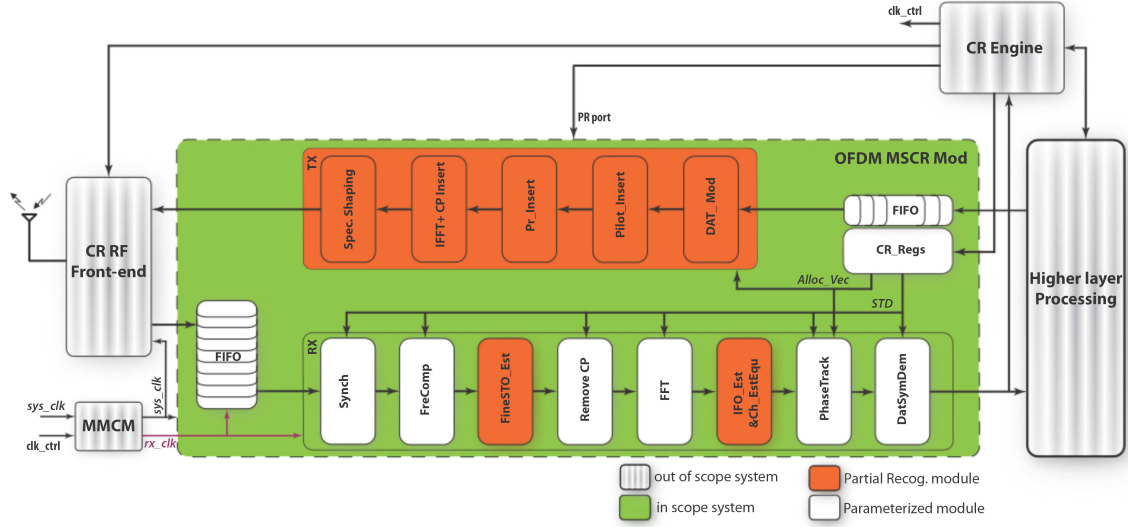
Figure 7.1: The structure of a generic MSCR system

7.3 Proposed OFDM-based baseband modulation 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. 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 modulation that supports transceiving 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 7.3.1.

Basically, The CR implementation is divided into a control plane and data plane. The data plane performs data processing for the radio system in data stream. It tranceives data streams to/from the RF front end through two AXI (Advanced eXtensible Interface) stream interfaces. For transmission, sending data from higher level layers is processed and modulated by the data

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

plane. Then modulated sample streams are transferred to RF front end to convert to analogue signals and subsequently be up-converted to RF band before transmission on the RF channel. For the receiver side, the received signals are down-converted to the baseband followed by analogue to digital convertion form a sample stream. The sample stream is demodulated and processed at the data plane before transferring to a higher level layer. To support multiple standards, the data plane has to provide the flexibility of switching functions accordingly to the parameters specified in different standards. The AXI stream interfaces are used for the overall data plane system as well as each sub-module. This unified interface is required to make partial reconfiguration possible, since modules that are reconfigured should share the same connections. This interface also allows the data from higher layers to be processed in a streaming format. This 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 the adaptive algorithm based on sensing data from the receiver to decide the allocation vector and the required standard. It monitors events in the data plane via an AXI bus interface connected set of registers. The control plane needs to issue commands to the data plane through the same bus register interface. In addition, the CR engine consists of the PR controller that is responsible for download the pre-complied bitstreams stored in DRAM into the corresponding PR modules through the ICAP (Internal Configuration Access Port) interface [11] when the CR engine decides to change the standard. The control plane can be implemented in one of 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 a high throughput. Normally, the data plane processes data streams based on a specified standard to transceive data symbols according to the allocation vectors. The CR engine must be able to performs sensing and adaptive tasks to update suitable allocation vectors based on current channel status. When the frequency band of the current operating standard is mostly occupied by PUs and IUs, the CR engine performs switching to an other 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 registiters in CR_Regs such as allocation vectors (Alloc_vec), symbol modulation type (MOD), and standard (STD).

The scope of this chapter focuses on the baseband modulation in order to deal with the challenges of long configuration. The system takes the advantages of coupling PR and parameterised functional units to offer flexibility while minimising reconfiguration time. The CR system is assumed capable of performing in burst mode in which the system only transmit the packages as soon as data is available [12]. The transmitter subsystem just performs the transmission initiated by the higher layer. If the system needs to transmit data during the periods of reconfiguration time, the data package will be stored in FIFO. The stored data packet will be flushed out of FIFO for transmission after the reconfiguration of the transmitter has been done. Moreover, the hardware usage for the transmitter side is so small compared to the receiver side that it has a much faster reconfiguration time. The hardware usage and reconfiguration time of the transmitter is presented latter in Section 7.4. The reconfiguration time does not cause a critical disruption of the processing chain and loss of transmitted packets. Therefore, the monolithic PR module for the transmitter side is implemented because of simplicity and flexibility. In contrast, the receiver subsystem is required to continuously the receive signal and detect the data frame. Furthermore, the hardware usage for the receiver subsystem is very large resulting in a long reconfiguration time. To avoid losing data frames, a huge FIFO is required to store received samples from the RF front end during reconfiguration and this is not

practical. The critical issue is to minimise the reconfiguration time in the receiver subsystem in an attempt to achieve these aims. The coupling of PR and parameterised functional units is applied to implement the receiver subsystem. The design of each module in the receiver subsystem to obtain the flexibility, while minimising reconfiguration time, will be investigated in the next subsection. In addition, after reconfiguring the receiver subsystem, the FIFO needs to be flushed for the next reconfiguration. A Mixed-Mode Clock Manager (MMCM) is employed to increase the received processing rate (rx_clk). After stored samples for reconfiguration are read out, the received processing rate is reduced 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 samppling rates specified in different standards, shown in Table 7.3.1.

7.3.2 Module Description

In this subsection, each functional module, shown in Fig. 7.1, in the receiver processing chain is investigated and commonalities across standards are analysed. In term of design options, there is a tradeoff between the simplicity, flexibility, and long reconfiguration latency of PR module and the increased hardware overhead, complexity, but faster configuration of parameterised modules. The comparison between the hardware usage of parameterised modules for multiple standards and that of individual specified standard modules is evaluated. If the hardware overhead required to parameterise the module is less than given percentage above the hardware size of the standard-alone module, then it is judged better to be parameterised. For those modules that require significant changes (i.e. the parameterised overhead is greater than this threshold) or are required to support unspecified parameter for future standard (such as preamble), PR can be used on a per-module level. The bistream for each standard is implemented and precompiled separately. As a result, when switching from one standard to another, only part of the FPGA needs to be reconfigured.

(i) FIFO buffer (FIFO): There are two FIFOs at the transmitter side and the receiver side to buffer the data sent from the higher layer and the RF front end, respectively. The FIFO buffers are implemented based on the 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, meantime, another one is read out from the
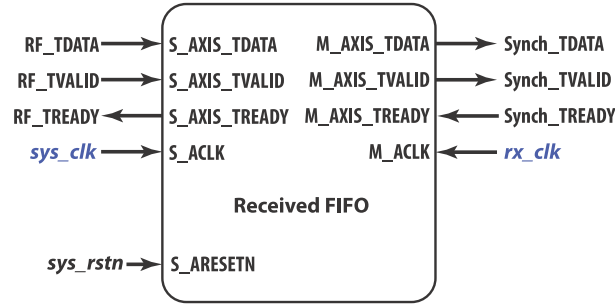
13

Figure 7.2: The received FIFO module

buffer by the transmitter side/ the receiver side at every single system clock. Therefore, the FIFO buffers normally operate in almost empty state. When the reconfiguration is required for switching underlying standard, the transmitted processing and received processing are suspended processing data streams. The FIFO buffers are stored coming data streams to avoid losing frames. Because the system performs in burst mode, the trasmitted streams are not continuously, therefore, the transmitted FIFO can be flushed the stored data during the gap of the burst streams. The data buffer is not a critical issue for the transmitter side. However, the receiver side needs to continuous processing data streams from the RF front end to detect incoming frames. The receiver side does not have spare time to flush the stored data in FIFO bufferred during reconfiguration time. Therefore, the received FIFO is configured with independent clocks as shown in the Fig. 7.2. In order to flushing stored data in the received FIFO after switching standard reconfiguration, the MMCM increases the received processing rate (rx_clk) be multiple time greater than the sampling rate of the RF front end to make the FIFO be almost empty for the next reconfiguration. When the received FIFO is almost empty the received processing rate is reduced back to equals the sampling rate to minimise power consumption.

(ii) Synchronisation (Synch): The CR system performs in burst mode. *Synch* is to detect the presence of a frame and estimate the frequency offset required, based upon the preamble of the received frame. This module performs estimation based on the timing parameters, defined by Schmidl and Cox [13], expressed as below;

$$
\begin{aligned}
P[d] &= \sum_{m=0}^{L-1} (r^*[d+m]r[d+m+L]) \\
R[d] &= \sum_{m=0}^{L-1} |r[d+m+L]|^2 \\
M[d] &= \frac{|P[d]|^2}{(R[d])^2},
\end{aligned}
\tag{7.1}
$$

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

As discussed in Chapter 4, the frame detection is performed by finding the plateau of M when a frame presents at receiver. P is also used to estimate fractional CFO in the subsequent module. The Fig. 7.3 shows block diagram of synchronisation implementation. The timing metrics are calculated by using auto-correlation on received samples. *Coarse Time* detects the new frame and estimate roughly the start of frame based on the comparison the metric P and the value $\frac{R}{2}$ that is equivalents to detect M greater than a threshold of 0.25. The method is blind estimation that provides a generality to be applied for multiple standards. By parameterising the length of L, the synchronisation module can effectively perform for three current supported standards as well as be extensible for future standards. Therefore, this module is implemented as a parameterised version with parameter L whose values are defined together with the length of FFT ($NFFT$), shown in Table 7.2. The parameterised value combinations of L, and $NFFT$ allow for the support of 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.

(iii) Frequency Compensation: *FreComp* module performs fractional CFO estimation based on the value of P passed from the previous module *Synch*. Fractional CFO estimation and compensation are mathematically expressed as below;

$$
\begin{aligned}
\widehat{\Delta f} &= \frac{\angle P}{2\pi L} \\
\widehat{r[d]} &= r[d]e^{-j2\pi \widehat{\Delta f} d}
\end{aligned}
\tag{7.2}
$$

15

Table 7.2: Parameterised values according to supported standards

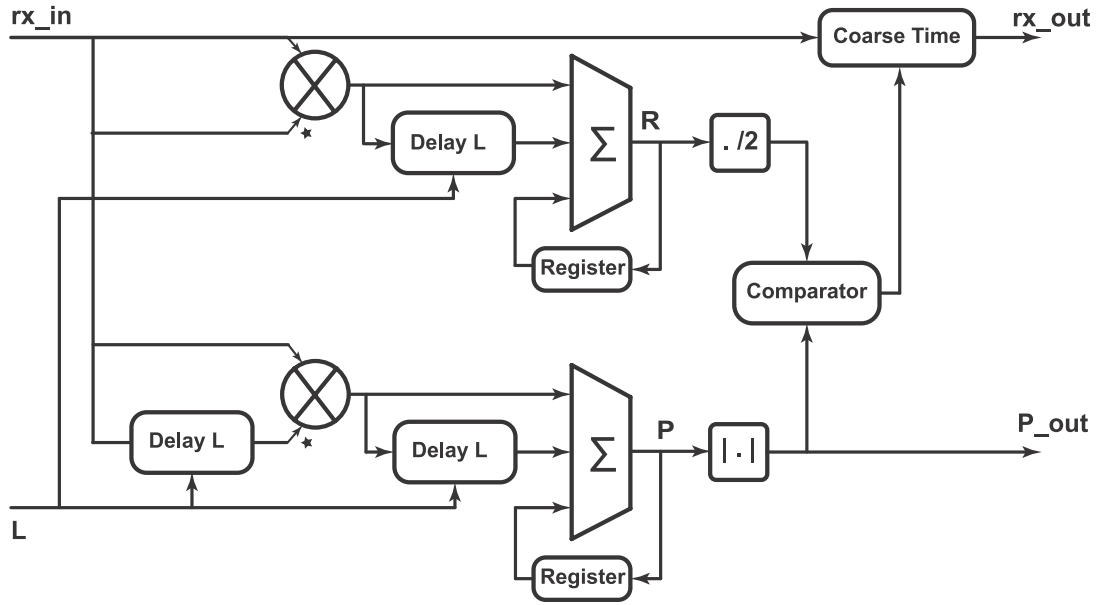| L NFFT	16	32	64	128	256	512
64	802.11					
128						
256			802.16			
512						
1024						
2048						802.22

Figure 7.3: The block diagram of Synchronisation module

where $\widehat{\Delta f}$ is estimated fractional CFO, $\angle P$ denotes the angle of P and N is number of FFT points.

The Fig. 7.4 illustrates the block diagram of frequency compensation. A phase rotation sub-module is used to compensate fractional CFO by rotating the received sample phase with a proper angle. The proper angles are calculated and accumulated based on estimated fractional CFO.

$$
\begin{aligned}
\phi[d] &= \phi[d-1] + \frac{\angle P}{L} \\
\widehat{r[d]} &= r[d]e^{-j\phi[d]}
\end{aligned}
\tag{7.3}
$$

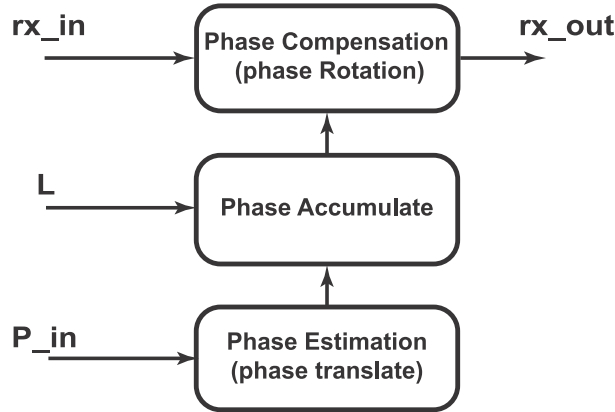According to (7.3), the computation of *FreComp* depends on the periodic length of short

16

Figure 7.4: The block diagram of frequency compensation module

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 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.

(iv) Fine STO Estimation: *FineSTO_Est* is to estimate the starting sample of each OFDM symbol. The RF front-end for MSCR need to access a wide range of frequencies, shown in Table 7.3.1. Depending on the standard in operation, the CFO may be large, resulting in the present of IFO. Therefore, the implementation of *FineSTO_Est* is based on the algorithm presented in [14] that is also robust to IFO. The metric of fine STO estimation is expressed as below;

$$S[d] = \sum_{m=0}^{L-1} |r[d + m + L]|^2 |a[m]|^2, \qquad (7.4)$$

when $|a[m]|$ denotes the normalised amplitude of the preamble at the transmitter. The Fig. 7.5 shows the block diagram of fine STO estimation. The metric is calculated based on the multiplierless correlation between received samples and transmitted preamble. The cross-correlation provides high precision for STO estimation. However, this operation requires a large number of DSP blocks, resulting in increased hardware resource and power consumption of the system. Reducing the hardware resource and power consumption requires using multiplierless correlation [15]. *Peak Detect* finds the maximum value of correlation that is employed to accurately estimate the STO and *Fine Time* determines
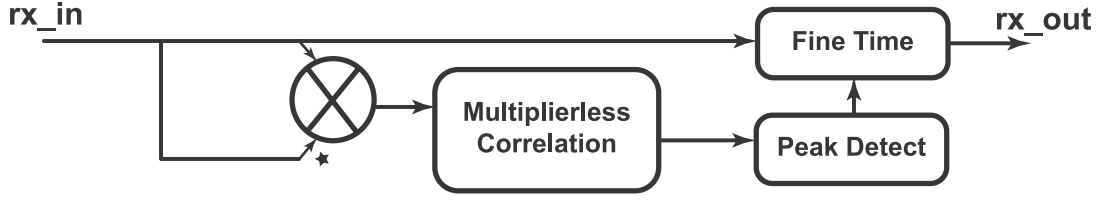
17

Figure 7.5: The block diagram of fine STO estimation module

the exact first sample of the next OFDM symbol (long preamble symbol). However, the multiplierless correlation is not flexible and depends on preambles that are different for each standard. Therefore, we employ the PR module for the FineSTO_Est module to obtain flexibility. Each supported standard is implemented separately and precompiled to a stored bitstream. The suitable precompiled bitstream will then be downloaded to the PR module by the *CR engine* when the underlying standard is switched. The PR module must evidently to be large enough to contain the largest bitstream among the supported standards.

(v) 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* 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.

(vi) FFT: *FFT* is based on the Xilinx FFT/IFFT IP core, but now supports run time reconfiguration of the FFT length according to the underlying selected standard. When the standard is changed, the length of the FFT of this core is reconfigured using the configuration input. The FFT/IFFT IP core reconfiguration is very fast, being able to accomplish a switch within a few clock cycles.

(vii) IFO Estimation and Channel Equalisation: *IFO_Est&Ch_EstEqu* corrects the IFO and performs channel equalisation. IFO results in a cyclic shift in the frequency domain. Using differential demodulation of the FFT output, the IFO can be conventionally determined with robustness to frequency selective channels using the correlation function [16] on the second preamble (long preamble) symbol. The function is expressed as below;

$$\hat{\epsilon} = \underset{\tilde{\epsilon}}{\mathrm{argmax}} \left| \sum_{k=0}^{N_{FFT}-1} Y^*[k-1]Y[k]X^*[k-\tilde{\epsilon}]X[k-1-\tilde{\epsilon}] \right| \qquad (7.5)$$

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 kth 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. The Fig. 7.6 illustrates the block diagram of IFO estimation and channel equalisation. The estimated IFO can achieve high precision using cross-correlation in the frequency domain. A low power and low cost IFO estimation architecture presented in Chapter 5 is applied for this module. *IFO Correction* is performed effectively by cyclic 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 subcarriers. By employ the information of the second preamble symbol, the effect of the channel can be estimated. The estimation and compensation of channel and residual effect can be expressed as below;

$$
\begin{aligned}
Y[k] &= X[k] * H[k] + N[k] \\
H[k] &= \frac{Y[k] - N[k]}{X[k]} \\
\hat{H}[k] &= \frac{Y[k]}{X[k]} \qquad (7.6) \\
\hat{R}[k] &= \frac{R[k]}{\hat{H}[k]} \qquad (7.7)
\end{aligned}
$$

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 (7.6), and the compensation for received data carriers is given in (7.7) in which $R[k]$, $\hat{R}[k]$ denote received and compensated data carriers, respectively. Due to symbols are modulated by QPSK, the symbol amplitude is not concerned. So, the complex division of channel estimation and compensation can be equivalently performed by multipling to the conjugation of $X[k]$ and $\hat{H}[k]$, respectively. The operation of this module depends on the second preamble that is specified differently in each standard. Therefore, PR is used for the *IFO_Est&Ch_EstEqu* module to obtain the
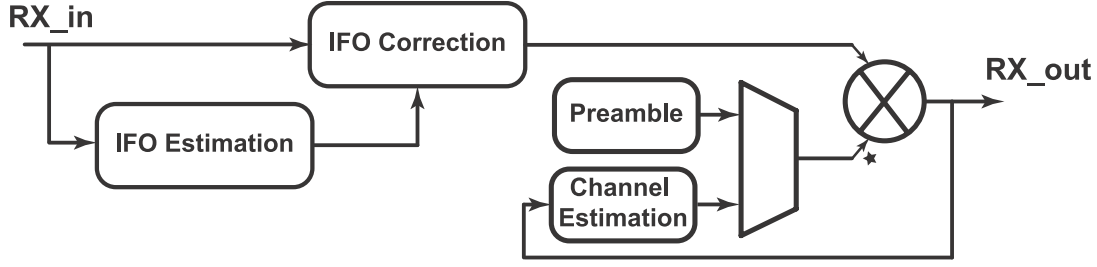
Figure 7.6: The block diagram of IFO estimation and channel equalisation

effective standard-specific implementation. Suitable precompiled bitstreams are down-loaded to the PR module by *CR engine* when the underlying standard is changed. The PR module must evidently be large enough to contain the largest bitstream among the supported standards.

(viii) Phase Tracking: *PhaseTrack* estimates the residual common phase error in each OFDM symbol after channel equalisation. The *PhaseTrack* implementation is based on the algorithm presented by Troya et al. [17]. 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.

$$P_{k,l} = cos\theta_l - \alpha.k.sin\theta_l + j(sin\theta_l - \alpha.k.cos\theta_l), \qquad (7.8)$$

where $P_{k,l}$ denotes the phase of received pilot which has frequency index k in the lth OFDM symbol. $cos\theta_l + jsin\theta_l$ is the residual common phase error of the lth 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;

$$
\begin{aligned}
cos\theta_l &= \frac{1}{N_P} \sum_{k \in S_P} \Re\{P_{k,l}\}, \\
sin\theta_l &= \frac{1}{N_P} \sum_{k \in S_P} \Im\{P_{k,l}\},
\end{aligned}
\qquad (7.9)
$$

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 phase tracking. *Pilot Extract* finds the employed pilots for phase tracking in the OFDM symbol based on
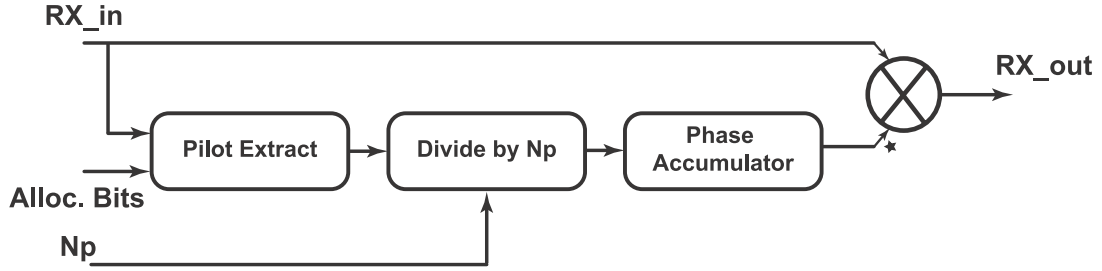
Figure 7.7: The block diagram of phase tracking module

allocation vector ($Alloc.\ Bits$). *Phase Accumulator* and *Phase Accumulator* compute the residual common phase error according to (7.9). The phase error is simply 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

(ix) Data symbol demodulation (*DatSymDem*): At 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 with 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 the 2 bits are assigned to the output according to the signed bits of the real and imaginary parts of data symbol.
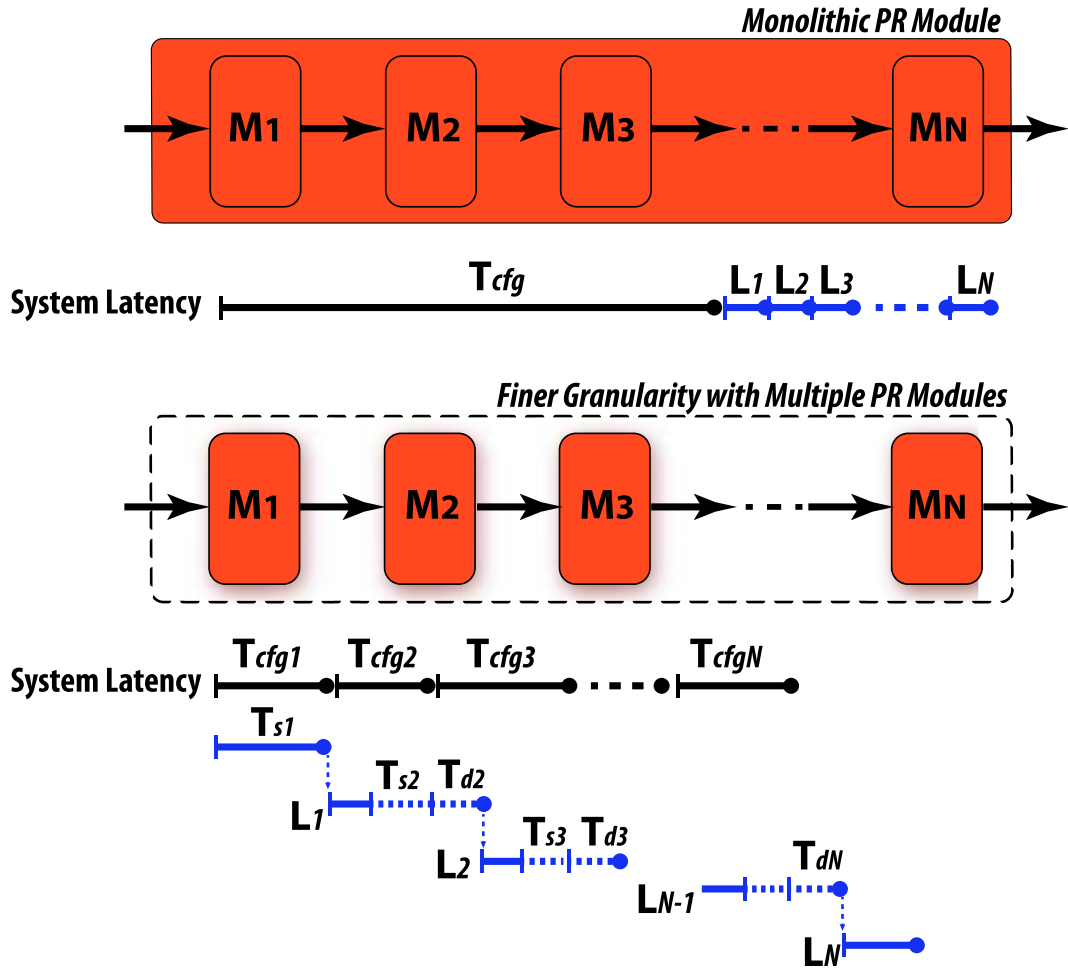
Figure 7.8: Comparison of system latency

7.4 Performance Analysis and Discussion

7.4.1 Analysing the latency and halting time of PR module-based systems

One crucial challenge when implementing CR systems on reconfigurable hardware is the long adaptation time, including compiling time and reconfiguring time, when changing standards. Despite the PR technique can eliminate the compiling time (since PR uses a database of pre-compiled modules), the partial reconfiguring time is still relatively long, particularly for a large monolithic module. Especially in the case of the CR receiver, the system is halted during reconfiguration, potentially causing the loss of data packet and possibly even the loss of synchronisation. A huge FIFO may be required to store a stream of received samples to overcome the reconfiguration latency when a PR module is reconfigured. A longer reconfiguration time demands a larger FIFO, which results in significantly increased hardware resource and power consumption. Mathematical analysis is used to evaluate the system latency for cases of monolithic PR modules, as well as for a system employing a finer granularity with multiple PR modules, and a mix of PR and parameterized modules for the case when the system switches to a new channel condition. Fig. 7.8 illustrates the system latency for monolithic PR modules and for a system with finer granularity with multiple PR modules. We consider a system consisting of N modules. T_c refers to the reconfiguration time. The assumption is that the system or module can not process data during its reconfiguration time. L_i is the computation latency of the nth 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 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 follow;

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

Finer granularity approaches divide the system into multiple sub-modules, each of which employes a PR module. When a module is completely configured, it can process the received data while the following module begins to be configured. Therefore, the system latency and halting time for the case of multiple PR modules can be calculated as follow;

$$L_{sys} = \sum_{i=1}^{N}(T_{ci}) + T_{dN} + L_N$$

$$T_{hlt} = \sum_{i=1}^{N}(T_{si}), \tag{7.11}$$

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 has to delay operation by a duration T_{di} before it receives input data for processing. Otherwise, the previous modules 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 ($T_{di} = 0$).

$$T_{di} = \begin{cases} Max(T_{ci}, T_{di-1} + L_{i-1}) - T_{ci} & i = 2..N, \\ 0 & i = 1 \end{cases} \tag{7.12}$$

$$T_{si} = \begin{cases} T_{ci} - min(T_{ci}, T_{di-1} + L_{i-1}), & i = 2..N, \\ T_{c1} & i = 1 \end{cases} \tag{7.13}$$

Substituting the above equations into (7.11),

$$L_{sys} = \sum_{i=1}^{N}(T_{ci}) + L_N +$$
$$(Max(T_{cN}, (Max(...) - T_{cN-1}) + L_{N-1}) - T_{cN})$$

$$T_{hlt} = \sum_{i=1}^{N}(T_{ci}) - \sum_{i=2}^{N}(min(T_{ci}, T_{di-1} + L_{i-1})), \tag{7.14}$$

As can be seen in the equations, the system latency and halting time in the case of multiple PR modules are theoretically reduced thanks to being able to overlap the reconfiguration and data processing periods. Practically, the reconfiguration times usually are much 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 (7.14) as below;

$$L_{sys} = \sum_{i=1}^{N}(T_{ci}) + L_N$$

$$T_{hlt} = \sum_{i=1}^{N}(T_{ci}) - \sum_{i=1}^{N-1}(L_i), \tag{7.15}$$

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 (7.10). Therefore, the system latency, L_{sys}, and halting time, T_{hlt} in (7.14) may be greater than that in (7.10). Generally, the finer granularity approach can only achieve the 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.

Through the above analyses and comparison of large monolithic PR modules with a finer granularity approach, we proposed a new method employing a mix of PR modules and parameterised modules to obtain a significant reduction in system latency and halting time. Each module in the processing chain of the granularity approach is investigated, and commonalities across different operation modes are analysed. For modules requiring only minor modifications, parameterized versions are created. For the ith module to be parameterised, the configuration time of this module can be eliminated because the parameterised modules can switch operation mode within one clock period. This approximately results in the following simplified equations;

$$
\begin{aligned}
T_{ci} &\approx 0 \\
T_{si} &\approx 0 \\
T_{di} &\approx T_{di-1} + L_{i-1},
\end{aligned} \tag{7.16}
$$

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

7.4.2 Analysing results of proposed OFDM-based MSCR architecture

The system latency and halting time of OFDM-based MSCR systems are investigated based on the results of hardware consumption and system performance when implemented on a Vertex6 FPGA (XC6VLX240t). A comparison of large monolithic PR module, finer granularity and proposed approaches is given to show the advantages of the proposed architecture. To compute a configuration time of a PR module, the bitstreams of all implementations have to be computed. The area of the PR module, i.e., the occupied and reserved hardware resources, must

satisfy the needs of the largest implementation. For the monolithic PR module approach, it is required that the PR module must be able to contain the 802.22 OFDM-based implementation, which is the largest receiver implementation among the three target implementations. Similarly to granularity 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 resources usage of each sub-module and transmitter, receiver system for 802.22 on the Vertex6 FPGA (XC6VLX240t). 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. These results are employed to calculate the bitstream size of PR modules for each functional block. Fig. 7.9 illustrates the bitstream sizes of each PR module, that are employed to calculate the configuration time later. The bitstream sizes of PR module for the functional modules are relatively small compared to the monolithic PR module for the receiver sub-system. The M_3 bitstream is the largest bitstream among the functional granular modules. Interestingly, the bitstream of the receiver sub-system is nearly triple that of the transmitter sub-system.

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

The latencies of functional modules are considered for three standards. Fig. 7.10 shows the latencies in number of clock cycles. As can be seen, the latencies for the 802.11 standard are smallest, because this standard uses the shortest FFT length, i.e., shortest symbol length for OFDM modulation. It should be noted that during the latency time the module still receives
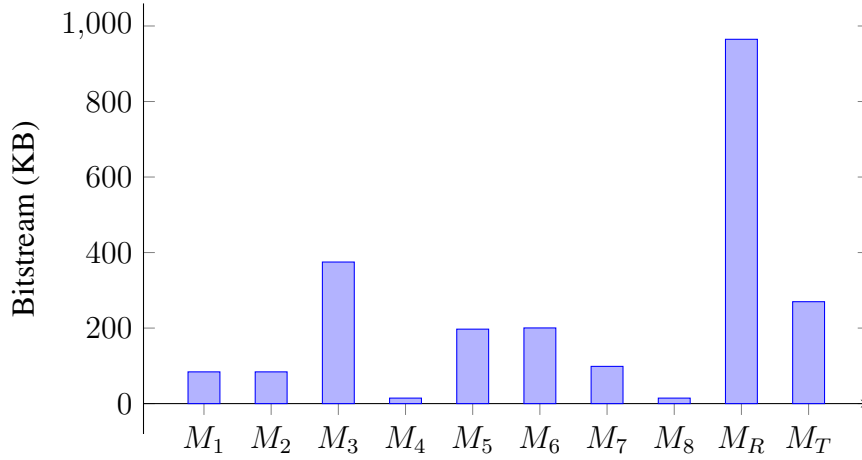
Figure 7.9: The bitstream size of PR modules

input data for processing. The processing chain is halted when the latency time has ended but the reconfiguration of the following module has not been completed yet. Therefore, the worst case halting time is a case of the shortest latency and the longest reconfiguration time. So, the latencies for 802.11 are taken to calculate the system halting time. The latency of the synchronisation module depends on the timing offset that is the duration from the time of receiving input samples to the time when the first sample of the coming frame is received. The synchronisation module does not output data if no frame is detected. Generally, the synchronisation latency is calculated as $L_1 = Offset + processing time$. To solely evaluate the processing latency of the synchronisation module, the timing offset is considered to equal 0. We will consider the presence of timing offset later for calculating the system halting time and FIFO requirement.

The partial reconfiguration is performed with the ICAP interface that supports downloading speeds of up to 400 MBps. Practically, because of the overhead of PR controller, a speed of 380 MBps is used for downloading the PR bitstream to FPGA. Therefore, the accuracy of computing the reconfiguration time for PR modules by dividing the bitstream size by the downloading speed should be acceptable. In addition, the sampling frequency for the radio signal is varied according to the currently chosen standard. 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. The latency is calculated from the results of 802.11 standard in Fig. 7.10 multiplied by the clock period of 0.1 us. Fig. 7.11 illustrates the time of configuration and latency for each module.
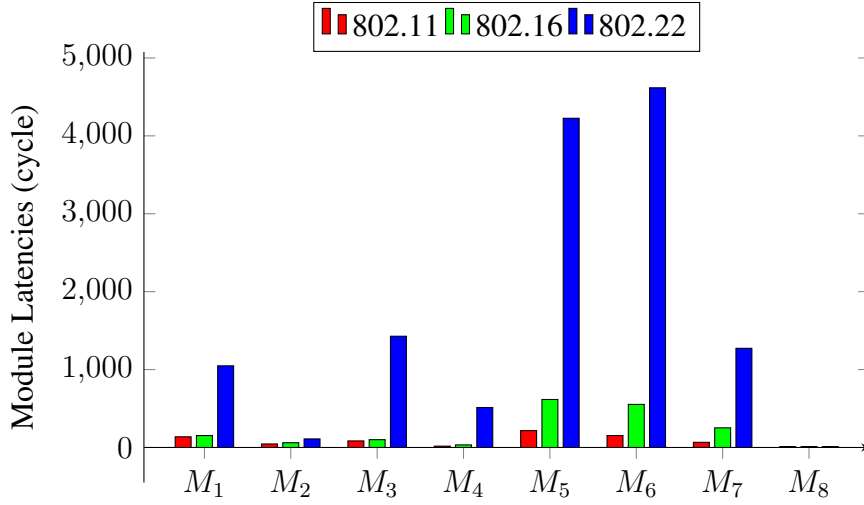
Figure 7.10: The latency of sub-modules for three standards

The values of latency are scaled by a factor of 10 for better observation. As can be seen, the latency is very small in comparison to the configuration time. It is clearly not efficient to perform the overlapping reconfiguration and data processing. Therefore, the finer granularity approach may not obtain a good improvement.

The system halting time is an accumulated value of the halting time in each module described in (7.13). During the halting time, the processing chain is halted and a FIFO is required
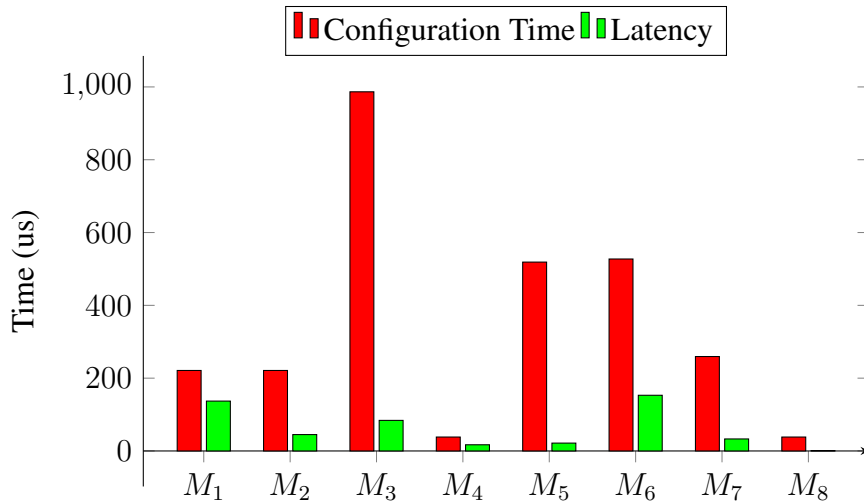

Figure 7.11: The configuration time and latency of sub-modules for OFDM-based MSCR system
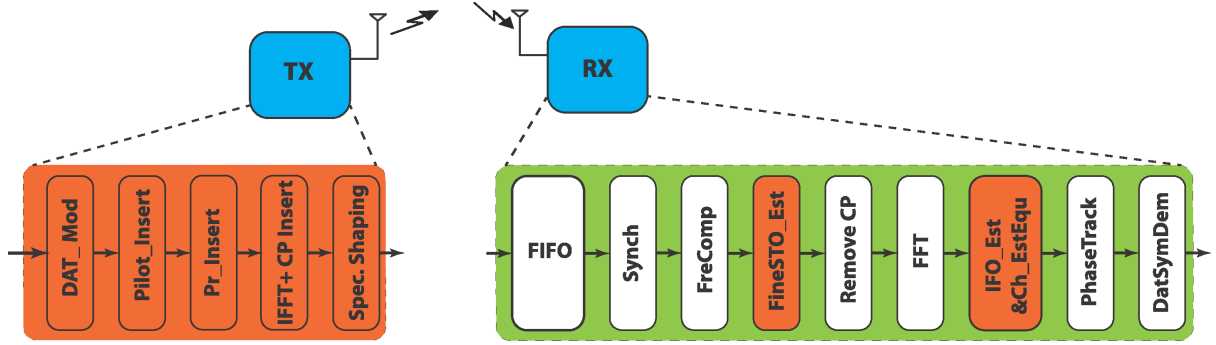
Figure 7.12: A scenario of a transmission

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 operating standard. In the proposed receiver, the synchronisation module is a parameterised module, so this module can change its function very fast - within one clock period and hence quickly process input samples. However, a new frame is not able to 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, 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. As can be seen, the halting time of the multiple PR module approach is greater compared to the monolithic PR module approach, because the gain achieved by overlapping the reconfiguration and data processing is greater compared to the overhead of partitioning into multiple PR modules. The proposed approach can significantly reduce the halting time that only less than one-third of that of the monolithic PR module approach. This results in a reduction in the FIFO requirement.

Fig. 7.13 compares the three approaches in terms of system latency and the FIFO requirement to avoid losing data frame. The system latencies are computed in the worst case which is the latencies for the 802.22 standard (the longest latencies). The FIFO requirement is calculated based on multiplying the sampling frequency by the halting time, followed by rounding
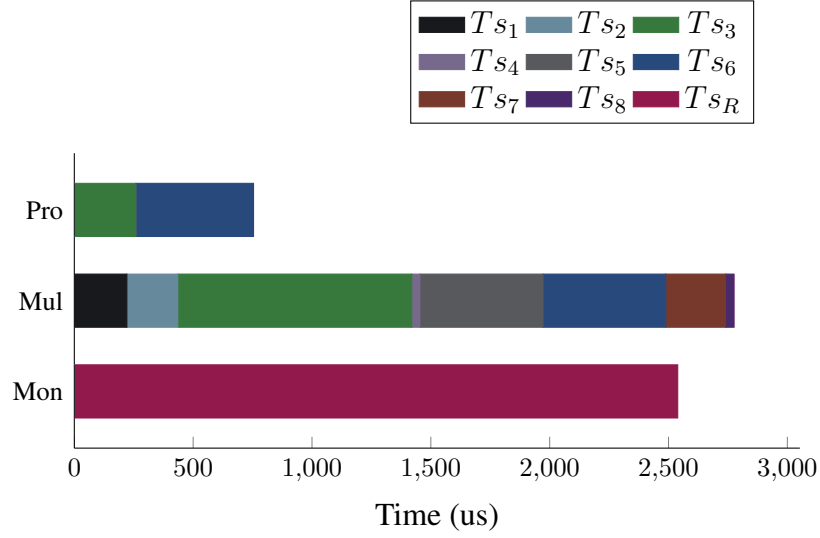
Figure 7.13: The halting time comparison of the system for three different approaches
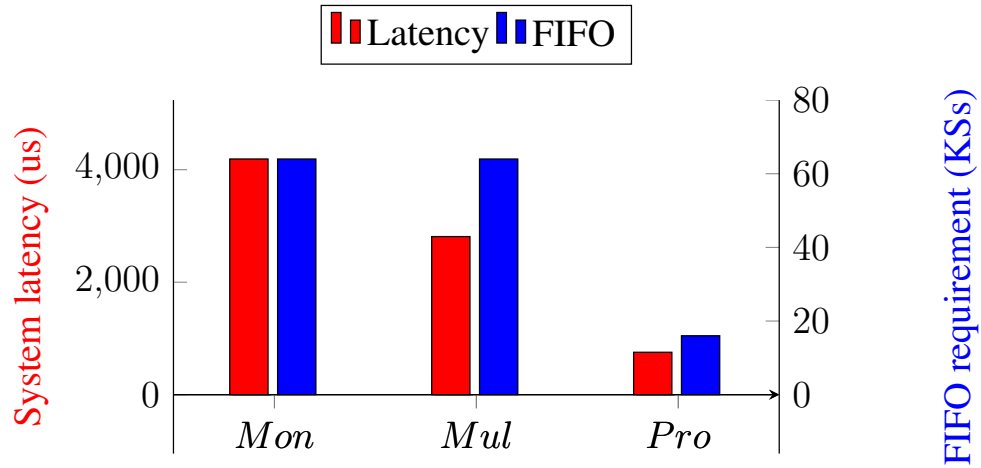

Figure 7.14: A comparison of the three approaches in terms of system latency and FIFO requirement

up to the next power of two value. The system latency of the proposed method is significantly reduced to 18 % and 27 % of the system latency of the monolithic PR module approach and the multiple PR module approach, respectively. The FIFO requirement for the proposed approach is only 16 kilo-samples (KSs) while the two other approaches require a FIFO which must store up to 64 KSs.

7.5   Summary

Our research explores the feasibility of designing efficient multi-standard radios to enhance bandwidth efficiency and avoid spectrum congestion. Our proposed architecture assumes the use of OFDM radio, and implemented on an FPGA, coupling parameterized modules and PR modules to achieve flexibility while minimizing reconfiguration time. A mathematical analyses of the FPGA synthesis results shows that the proposed architecture achieves a significant reduction of 82 % compared to conventional approaches in terms of system latency. The FIFO requirement of the proposed method is also decreased to 25 % of that required for the conventional method.

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 6th 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 *Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, (Dublin, Ireland), 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 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 *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 Trans. on Broadcasting*, vol. 52, pp. 236–244, Jun 2006.

[13] T. Schmidl and D. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Transactions on Communications*, vol. 45, pp. 1613 –1621, Dec. 1997.

[14] T. H. Pham, I. V. McLoughlin, and S. A. Fahmy, "Robust and Efficient OFDM Synchronisation for FPGA-Based Radios," *Circuits, Systems, and Signal Processing*, vol. 33, pp. 2475–2493, Aug. 2014, Springer.

[15] 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, pp. 1549–1553, August 2013.

[16] M. Park, N. Cho, J. Cho, and D. Hong, "Robust integer frequency offset estimator with ambiguity of symbol timing offset for OFDM systems," in *Proc. Vehic. Techn. Conf. (VTC)*, pp. 2116–2120, 2002.

[17] 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.