Multi-Clock Pipelined Design of an IEEE 802.11a Physical Layer Transmitter

Maryam Mizani and Daler Rakhmatov

University of Victoria
Department of Electrical and Computer Engineering
Victoria, BC, V8P5C2 Canada
{mmizani, daler}@ece.uvic.ca

Abstract

Among different wireless LAN technologies 802.11a has recently become popular due to its high throughput, large system capacity, and relatively long range. In this paper, we propose a reconfigurable architecture for the 802.11a physical layer transmitter, which has low latency and low power consumption due to its pipelined structure. Data from the MAC layer can continuously flow through the pipeline without excessive buffering and handshaking within the physical layer. Dynamically reconfiguring this architecture to work at any data rate supported by 802.11a (eight different modes) can be performed within a few cycles, simply by adjusting the period of two clock signals and changing the value of a 3-bit control signal. Our architecture, prototyped on a Xilinx Virtex-II Pro FPGA, occupies the area of 2059 slices and is estimated to consume 500 mW. These figures can be improved substantially in custom ASIC implementations.

1. Introduction

The rapid advances in CMOS technology have led to remarkable proliferation of mobile devices and wireless networks in the recent years. The current IEEE standards for wireless LAN include 802.11a (5 GHz, 54 Mbps), 802.11b (2.4 GHz, 11 Mbps), and 802.11g (2.4 GHz, 54 Mbps) variants. The 802.11a standard supports nearly five times the data rate and as much as ten times the overall system capacity of 802.11b LANs [1, 2, 3]. In comparison with 802.11g operating at 2.4 GHz, 802.11a may be more popular due to its 5 GHz frequency. The 5 GHz band offers the advantages of higher data rates, more available spectrum, less sharing with other uses (e.g., cordless phones and Bluetooth radios), and a better environment with less noise and interference from other electronic devices [4].

The physical layer of a 802.11a transmitter consists of two protocol functions: the physical layer convergence procedure (PLCP) that maps data units into a framing format suitable for transmitting, and the physical medium dependent (PMD) layer. This paper is focused on the implementation of the PLCP part that is power-efficient and easily reconfigurable to work at different rates during transmission.[1]

The key design idea is to use a pipeline strategy that supports all eight different modes defined in the 802.11a standard. Each stage of the pipeline can work on data of different size. To maintain a continuous flow of data through the pipeline, each stage is driven by a different clock and a different controller. The whole architecture needs four clock signals, two of which are not variable due to the fixed output rate of 20 Msample/s. The other two clocks, however, vary to support different transmission rates. Reconfiguration of this system for working at different modes is very efficient, as switching between different data rates requires changing only the frequency of two clocks and the value of a 3-bit mode signal. Since data is continuously flowing through the pipeline, we can satisfy the throughput constraints at slower clocks and, consequently, at lower supply voltages.

The rest of the paper is organized as follows. In section 2 we present an overview of the physical layer of the 802.11a wireless LAN standard [1], related work, and our contributions. In section 3 we provide the details of our architectural design, and in section 4 we present implementation results (delay, area and power)

[1]We are currently working on the extensions of the proposed architecture to handle data reception as well.

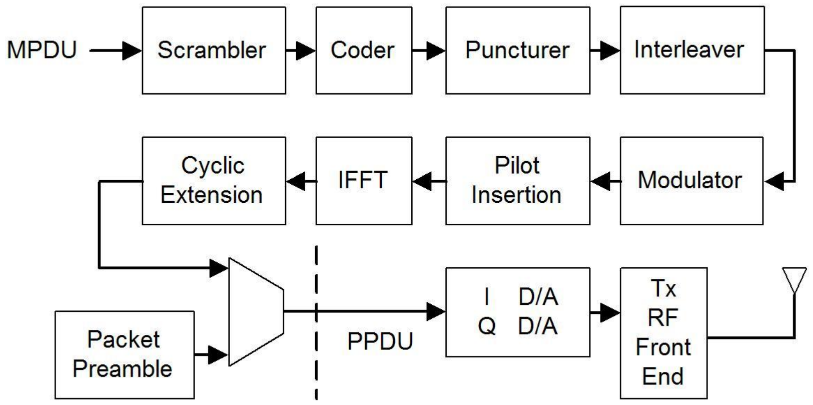

Figure 1. 802.11a PHY transmitter.

for the proposed architecture prototyped on a Xilinx Virtex-II Pro FPGA. Finally, section 5 concludes the paper.

2. IEEE 802.11a PLCP layer design

The 802.11a PHY transmitter receives MAC Protocol Data Units (MPDU) broken into octets. These octets are encapsulated into a frame called PLCP Protocol Data Unit (PPDU) at the PHY PLCP layer. The PLCP processes MPDU according to the requirements specified by the MAC layer. The appropriate header and the scrambled data are encoded, interleaved, modulated (BPSK, QPSK, 16-QAM, or 64-QAM), mapped onto OFDM carriers, and converted to time domain using IFFT. The output of the IFFT block is cyclicly extended, windowed, and prepended with the signal field and preamble. The resulting PPDU is interpolated and upconverted in the PHY PMD layer and passed to the RF circuit in I/Q format. A block diagram of the physical layer transmitter is shown in Figure 1. For more details the reader is referred to the IEEE 802.11a Standard document [1].

Recent literature reports both software-based and hardware-based implementations of the 802.11a physical layer. Constrained by the sequential execution model, software implementations on high-performance DSPs require large number of instructions per cycle to avoid excessively high clock frequencies in order to achieve the required data rate [5, 6]. Custom hardware, on the other hand, can fully exploit application-specific parallelism to achieve high throughput at low frequencies. However, in the context of 802.11a, hardware-based implementations may suffer from inefficient programmability to handle different data rates. One approach is to create hardware blocks with configuration memories, whose contents can be changed at runtime to tune a block to a different data rate. Changing the state of these configuration memories (e.g., loading control registers) may take tens of cycles, as tens of configuration bytes may need to be loaded [7]. Even if hardware programmability is not an issue, there is still another problem that needs to be addressed: different blocks of the PHY transmitter may have different processing rates. Consequently, the architecture should incorporate certain buffering and handshaking between blocks whose processing rates do not match [8] (unless such blocks are clocked by different signals and carefully pipelined). For example, as MPDU octets enter the transmitter chain, blocks may need to wait for the acknowledgement of the current octet (usually waiting for a few clock cycles) before processing the next octet. As a result, the time to prepare a frame may be quite long, and the system clock frequency may exceed 100 MHz in order to meet the 802.11a throughput constraints [9]. In multi-clock pipelined architectures, where data can be processed non-stop (without waiting for the acknowledge signals), the same throughput can be achieved at lower clock frequencies. Reducing the clock rate enables lower supply voltages, which significantly reduces the power consumption. Our architecture is an example of such an approach. Furthermore, its reconfiguration to work at different modes is very efficient, only requiring the corresponding frequency adjustments of two clock signals and changing the value of the mode signal.

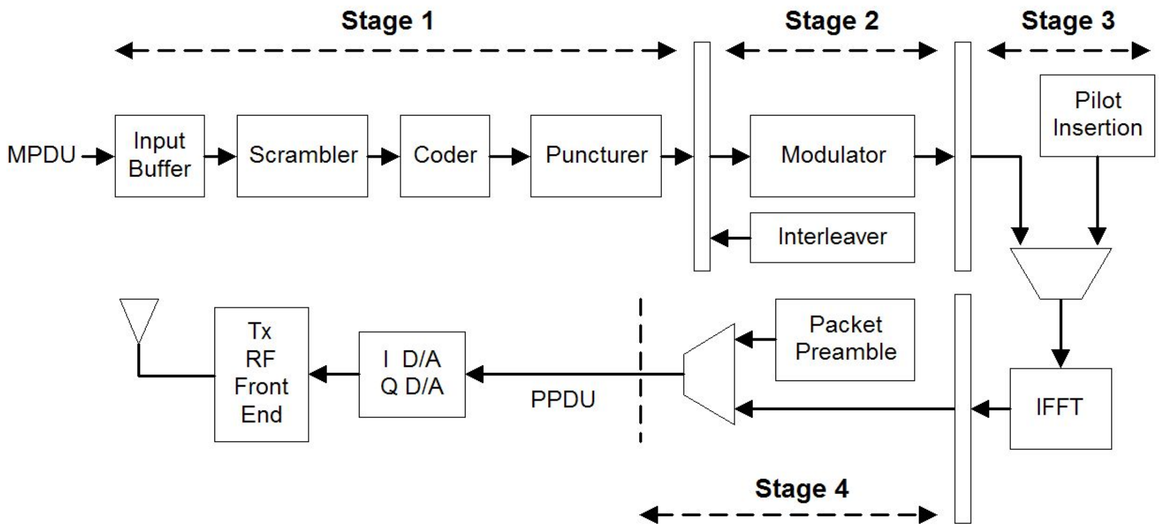

Figure 2. Pipelined architecture of 802.11a PHY transmitter.

3. Proposed pipelined architecture

The proposed pipelined architecture of the 802.11a PLCP layer consists of four stages separated by three inter-stage buffers, as shown in Figure 2. Each of these buffers has two banks – when the previous stage writes one bank, the next stage reads the other bank. For eight different modes, corresponding to eight different data rates (6, 9, 12, 18, 24, 36, 48, and 54 Mbps), Table 1 shows the modulation type, coding rate, and the data rate of four pipeline stages. Stage-by-stage operation of the pipeline is as follows.

Stage 1: Data octets continuously enter the input buffer after receiving START-REQUEST signal from the MAC layer, until the END-REQUEST signal is received. At the same time, the input buffer serially sends out data to the scrambler. After scrambling, data is coded and punctured according to the working mode, and finally saved in the first inter-stage buffer of size 128x16 bits.

Stage 2: The interleaver generates addresses to read data in the interleaved order from the first inter-stage buffer. The modulator receives the interleaved data bit stream and maps it into a mode-specific M-QAM constellation. After modulation, regardless of the mode, there will be 48 complex samples saved in the second inter-stage buffer of size 128x40 bits.[2]

Stage 3: The samples from the second inter-stage buffer and pilot subcarriers enter the IFFT block in the proper order. IFFT has a pipeline structure as well, so that it does not stop the flow of data (see section 3.3 for details). The output of IFFT is stored in the third inter-stage buffer of size 128x40 bits.

Stage 4: When the first OFDM symbol of the data is at the first stage of the IFFT pipeline, the RF front end has already started reading the preamble. Once the preamble is completely transmitted, the RF front end starts transmitting the header (previously saved in bank 1 of the third inter-stage buffer). Meanwhile, the first OFDM symbol of the data is written into bank 2 of the third inter-stage buffer, ready to be transmitted after the header. It should be noted that the header data rate is always fixed at 6Mbps using BPSK modulation. Therefore, to generate the OFDM header symbol, we first enter the header bits into the pipeline and save the generated OFDM symbol in the first bank of the last inter-stage buffer. Then, we adjust the clock rate according to the desired mode and start generating OFDM symbols of the data.

3.1. Fast reconfigurability

For throughput-constrained applications such as wireless networking, the reconfiguration time may be of a special concern since it has a direct influence on the overall system performance. In such systems, the key functionalities that are required to be flexible should be made fast-reconfigurable.

Using our proposed pipelined architecture, the data rate at the third and forth stage of the pipeline is fixed,

[2]We have chosen 40 bits for calculations involving complex numbers.

Mode	Modulation	Coding rate	Stage1 Data rate (Mbps)	Stage2 Data rate (Mbps)	Stage3 Data rate (Msample/s)	Stage4 Data rate (Msample/s)
0	BPSK	1/2	6	12	18	20
1	BPSK	3/4	9	12	18	20
2	QPSK	1/2	12	24	18	20
3	QPSK	3/4	18	24	18	20
4	16-QAM	1/2	24	48	18	20
5	16-QAM	3/4	36	48	18	20
6	64-QAM	2/3	48	72	18	20
7	64-QAM	3/4	54	72	18	20

Table 1. Modulation types, coding rates and data rates for different modes.

as it is shown in Table 1. As a result, the hardware blocks of these two pipeline stages are not reconfigurable. At the first and second stages of the pipeline, only the puncturer, interleaver, and modulator are reconfigurable.

The reconfigurable puncturer applies different puncturing patterns to change the data rate according to the working mode. The coding rate of 1/2 at the output of the coder will be increased to 3/4 for mode 1, 3, 5, and 7 and to 2/3 for mode 6. The reconfigurable interleaver uses different types of permutations according to the working mode, as discussed in section 3.2. The reconfigurable modulator breaks the input bit stream into bit-groups of the required size and maps a complex number to each bit-group. The size of bit-groups, N_{BPSC}, representing the number of coded bits per OFDM subcarrier, varies between 1, 2, 4, and 6 for BPSK, QPSK, 16-QAM, and 64-QAM modulations, respectively.

A major advantage of our proposed architecture is that it allows efficient frame-by-frame reconfigurability since only a few blocks should be adjusted through the reconfiguration. The mode signal which defines the functionality of the reconfigurable hardware blocks is only 3-bit wide. It is an immediate input to the system and its value can be changed very quickly. The mode signal is also fed to the clock circuitry that generates the two adjustable clocks for the first and second stage of the pipeline. The frequencies of these clocks are adjusted according to the selected mode. In other words, changing the reconfiguration of the transmitter to work with a different data rate is as simple as changing the value of the mode signal and waiting for a few clock cycles till the clocks are correctly adjusted. This process is much faster compared to the conventional methods with time-consuming configuration memory downloads.

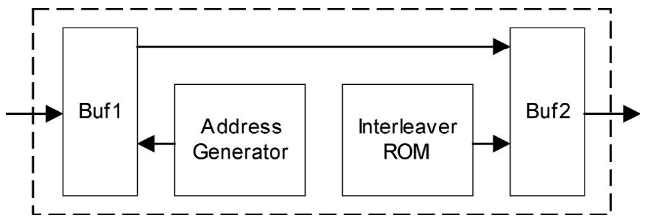

Figure 3. Regular interleaver.

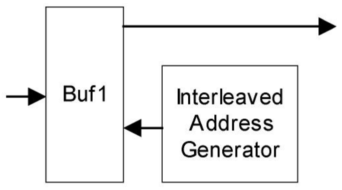

Figure 4. Optimized interleaver.

3.2. Optimized interleaver

Figure 3 illustrates a regular interleaver, where data is read from buffer Buf1 and is written into buffer Buf2 in an interleaved order. Instead of the regular interleaver, we use a simple circuit that provides interleaved addresses to Buf1, so that data is read directly out of Buf1 in the interleaved order. The block diagram is reduced to what is shown in Figure 4. The delay and area of the interleaver ROM and Buf2 are thus eliminated.

The interleaver circuit applies two permutations to generate the interleaved addresses as defined by equations (1) and (2) [1]. In these equations k is the index (address) of coded bits saved in Buf1, i is the generated index after the first permutation, and j is the generated index after the second permutation. In a regular interleaver, j would be the address of interleaved bits that are saved in Buf2. The interleaved address generator shown in Figure 4, directly computes k indices while stepping over the values of j.

$$\begin{gathered} i = (\tfrac{N_{CBPS}}{16})\ (k\ mod\ 16) + \lfloor \tfrac{k}{16} \rfloor, \\ k = 0, 1, ..., N_{CBPS} - 1, \end{gathered} \tag{1}$$

$$\begin{gathered} j = s\lfloor \tfrac{i}{s} \rfloor + (i + N_{CBPS} - \lfloor \tfrac{16i}{N_{CBPS}} \rfloor)\ mod\ s, \\ i = 0, 1, ..., N_{CBPS} - 1, \end{gathered} \tag{2}$$

where $s = max\{\frac{N_{BPSC}}{2}, 1\}$.

The value of the mode signal decides about the N_{CBPS}, and N_{BPSC} in these equations. N_{CBPS} is the block size corresponding to the number of coded bits in a single OFDM symbol, specifically 48, 96, 192, or 288 for data rates 6 and 9, 12 and 18, 24 and 36, or 48 and 54 Mbps, respectively. N_{BPSC} is the number of bits which are considered at the modulator to construct one OFDM subcarrier, specifically 1, 2, 4, and 6 bits for BPSK, QPSK, 16-QAM and 64-QAM, respectively.

3.3. Pipelined implementation of IFFT

The IFFT operation in the 802.11a physical layer is performed on 64 data points. We implemented our 64-point IFFT block in a pipelined fashion to improve the throughput and power consumption. A 64-point IFFT has six stages, each containing 32 butterfly operations, and one more stage for scrambling the output data. To perform the total of 192 butterfly operations, we use six pipeline stages, as shown in Figure 5. There is one Processing Element (PE) per stage, responsible for the processing of 32 pairs of complex numbers. Representing a pair of complex numbers as $Re(in0) + Im(in0)$ and $Re(in1) + Im(in1)$, the operations performed by each PE can be expressed as follows.

$$\begin{aligned} Re(out0) &= Re(in0) + [Re(in1) \cdot \cos(\omega) - Im(in1) \cdot \sin(\omega)], \\ Im(out0) &= Im(in0) + [Im(in1) \cdot \cos(\omega) + Re(in1) \cdot \sin(\omega)], \end{aligned}$$

$$\begin{aligned} Re(out1) &= Re(in0) - [Re(in1) \cdot \cos(\omega) - Im(in1) \cdot \sin(\omega)], \\ Im(out1) &= Im(in0) - [Im(in1) \cdot \cos(\omega) + Re(in1) \cdot \sin(\omega)], \end{aligned}$$

where $\cos(\omega)$ and $\sin(\omega)$ are coefficients generated by the W-Gen blocks. The size of RAM1 is 32x40 bits, while the other RAMs are 64x40-bit. It takes 32 cycles to complete each stage. When IFFT starts working, 32 complex numbers enter RAM1. In the following 32 cycles, these numbers and the next 32 numbers entering IFFT will enter PE1 in pairs, and the results will be saved in RAM2. The other stages work in a similar manner, as it is shown in Figure 6. In the last stage, data is saved in either RAM70 or RAM71, and it will be read out in a scrambled order using the addresses that are generated by the scrambler block. This IFFT structure has high throughput: when the pipeline is full, a complex sample is produced every clock cycle, i.e., IFFT does not stop the flow of input and output data. In other words, it is ideally suited in the proposed pipelined architecture of the 802.11a PHY transmitter.

4. Implementation results

We have prototyped the proposed architecture on a Xilinx Virtex-II Pro XC2VP50 FPGA [10]. The architecture uses 2059 slices, 27 18Kb RAM blocks, and 72 18x18-bit multipliers.

Table 2 shows the maximum clock frequencies for the four pipeline stages in our implementation, along with the minimum required clock frequencies for the fastest 54 Mbps input data rate. If the clock is slower than the minimum frequency, then the system will not be fast enough to handle the 54-Mbps mode. The time slack corresponding to the difference between the maximum and the minimum clock frequencies can be used to scale voltage down, resulting in significant power savings.

The minimum clock frequencies are computed as follows. The clock frequency of Stage 1 is 54 MHz, which corresponds to the input data rate of 54 Mbps. The second stage of the pipeline must provide 48 complex numbers every 4 μs (Every 48 complex number will be used later to construct an OFDM symbol in 4 μs of the symbol time interval). In order to generate one complex number in the 54-Mbps mode, the modulator takes 6 clock cycles. Therefore, the minimum clock frequency of Stage 2 is 72 MHz. The clock frequency of Stage 3 is fixed for all modes at 18 MHz, which is the result of dividing 72 by 4 μs, where 72 is the number of clock cycles required by the pipelined IFFT block to generate 64 OFDM subcarriers. The clock frequency of Stage 4 is also fixed at 20 MHz, which is obtained by dividing 80 samples per OFDM symbol by the 4 μs of the symbol time interval.

The larger the difference between the maximum frequency allowed by the implementation and the minimum required frequency to support a given data rate, the larger delay slack that can be utilized for voltage scaling. In ASIC implementations, the critical path delay is usually much smaller than in FPGA imple-

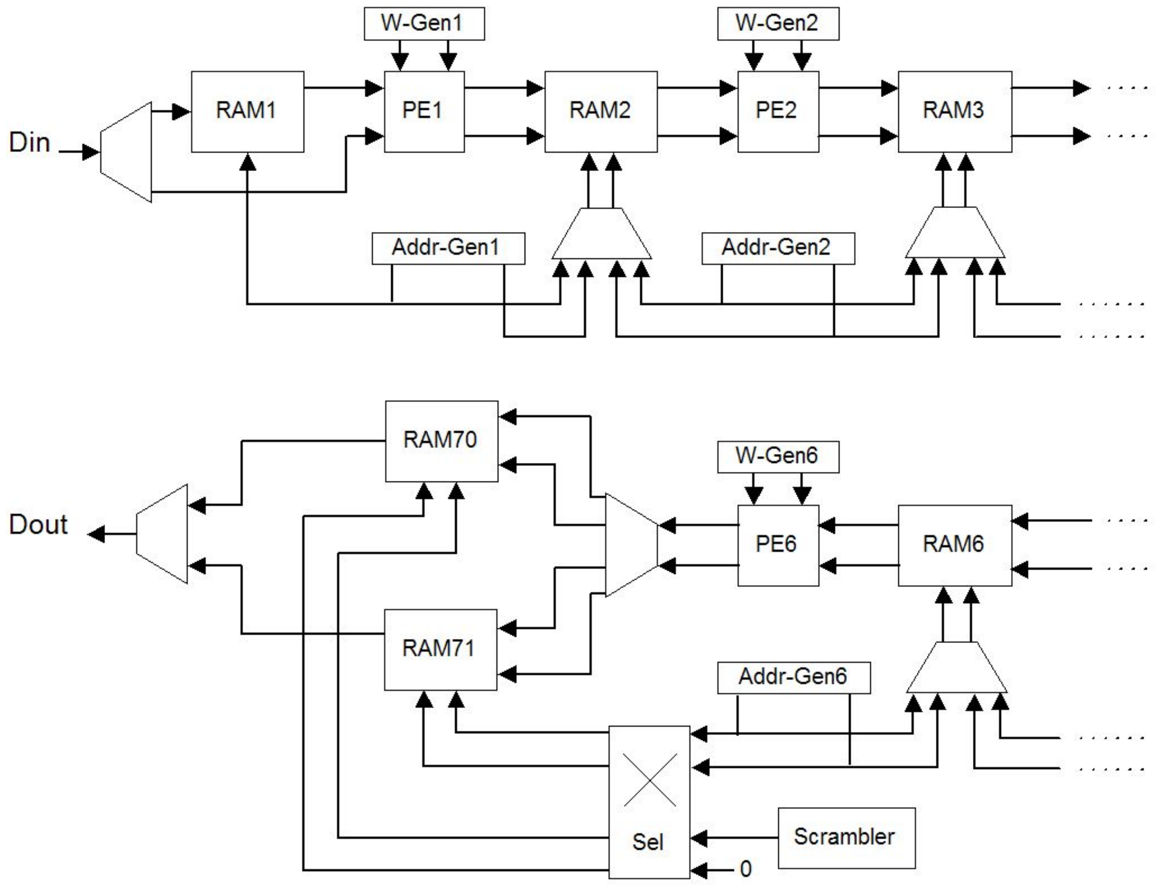

Figure 5. Pipelined architecture of IFFT block.

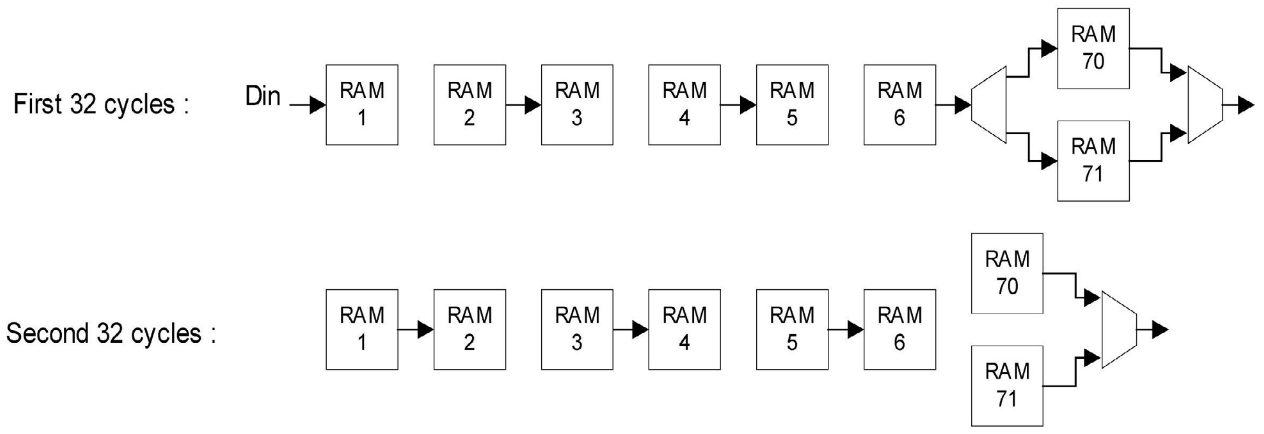

Figure 6. Flow of data in the pipeline at the first and second 32 cycles of 64-IFFT.

	Stage1	Stage2	Stage3	Stage4
Maximum clock frequency (MHz)	126	117	57	226
Minimum clock frequency (MHz)	54	72	18	20

Table 2. Maximum and minimum clock frequencies of the pipeline stages for the 54-Mbps mode.

mentations, leading to the greater delay slack available for more aggressive voltage scaling.

Estimated power consumption of our FPGA prototype is approximately 500 mW, without utilizing available delay slack for voltage scaling. Since the maximum clock frequencies in the 54 Mbps mode are approximately 1.6x greater than the minimum required clock frequencies (see Table 2), the supply voltage can be reduced by the factor of 1.6, which can lead to the 2.56x reduction in power consumption. If we choose not to reduce the voltage and the clock frequency by half, then our implementation can handle 86 Mbps, i.e., 1.6 times the data rate of 54 Mbps. Finally, we note that the estimated latency of the transmitter pipeline is 57 μs, after which complex samples are produced at each clock cycle at 20 Msamples/s.

5. Conclusion

In this paper, we described a multi-clock pipelined architecture for the IEEE 802.11a PHY transmitter. The proposed design supports all eight different modes of the 802.11a transmission. Frame data octets continuously flow through the pipeline, whose four stages are driven by four individual clocks and four individual controllers. Two of the clocks are configurable to support different transmission rates. As data octets are processed non-stop, the throughput constraints can be met at low clock frequencies, leading to lower supply voltages. Our architecture has been prototyped on a Xilinx Virtex-II Pro FPGA, with a custom implementation to follow soon. Our current design efforts are focused on extending this architecture to handle data reception as well as transmission.

References

[1] IEEE Standard 802.11a-1999: Wireless LAN MAC and PHY Specifications - High-Speed Physical Layer in the 5GHz Band, 2000.

[2] W. Eberle et al. Digital 72Mbps 64-QAM OFDM Transceiver for 5GHz Wireless LAN in 0.18 CMOS. In *IEEE Proc. ISSCC-2001*, pages 336–337, Feb. 2001.

[3] P. Ryan et al. A Single Chip PHY COFDM Modem for IEEE 802.11a with Integrated ADCs and DACs. In *IEEE Proc. ISSCC-2001*, pages 338–339, Feb. 2001.

[4] A. Nogee. WLAN Chipset Market - The Incredible Journey Is Just Beginning. *In-Stat rep. no. IN020271WY*, 2002.

[5] M. Meeuwsen et al. A full-rate software implementation of an IEEE 802.11a compliant digital baseband transmitter. In *IEEE Workshop on Signal Processing Systems*, pages 124–129, 2004.

[6] Yiyan Tang, Lie Qian, and Yuke Wang. Optimized Software Implementation of Full-Rate IEEE 802.11a Compliant Digital Baseband Transmitter on Digital Signal Processor. *GLOBECOM´05*, 2005.

[7] M. Krstic et al. Baseband processor for IEEE 802.11a Standard with Embedded BIST. *Facta Universitatis, Series: Electronics and Energetics*, volume 17, no. 2, pages 231–239, Aug. 2004.

[8] P. Coulton and D. Carline. An SDR inspired design for the FPGA implementation of 802.11a baseband system. In *IEEE International Symposium on Consumer Electronics*, pages 470–475, Sept. 2004.

[9] Duolog Technologies. Cb0020: 802.11a Signal Processor Product Brief. In *IEEE International Symposium on Consumer Electronics*, 2002.

[10] Xilinx. Virtex-II Pro Paltform FPGA Handbook, 2002.