IET Computers & Digital Techniques

Research Article

IET Journals
The Institution of Engineering and Technology

Low complexity and area efficient reconfigurable multimode interleaver address generator for multistandard radios

ISSN 1751-8601
Received on 1st May 2015
Revised on 11th July 2015
Accepted on 31st July 2015
doi: 10.1049/iet-cdt.2015.0070
www.ietdl.org

Nithish Kumar Venkatachalam[1] ✉, *Lakshminarayanan Gopalakrishnan*[1], *Mathini Sellathurai*[2]

[1]*Department of Electronics and Communication Engineering, National Institute of Technology Tiruchirappalli, Tamilnadu, India*
[2]*School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh, United Kingdom*
✉ *E-mail: nithishkumar_vlsi@yahoo.co.in*

Abstract: Developing a reconfigurable transceiver to support multiple protocols seamlessly and efficiently is an extremely tough task. Wireless standards such as wireless local area network (IEEE 802.11a/g) and WiMAX (IEEE 802.16e) incorporate block interleaving technique to overcome the occurrence of burst errors during transmission. Field Programmable Gate Array (FPGA) implementation of floor and modulus (MOD) functions to perform the two step permutation for attaining the new index is quite complex. In this study, the authors propose a low complexity and area efficient reconfigurable architecture for multimode interleaver address generator to support multiple wireless standards. In addition, a novel MOD_row and MOD_column circuit are proposed to compute MOD function for row and column counter values, respectively. The proposed address generation circuitry supports BPSK, QPSK, 16-QAM and 64-QAM modulation schemes under all possible code rates. The reconfigurable address generator for various block size and modulation scheme are implemented on Xilinx Spartan XC3S400 FPGA and the functionalities are verified through simulation. The synthesis results of the proposed design shows a reduction of 60% in resource utilisation and an improvement of 46% in operating frequency over the existing approaches.

1 Introduction

Multistandard radios utilise distinct transceiver block for each wireless standard to process the corresponding communication signals. Hence, they result in increase of silicon cost and power consumption of the wireless devices which operates on battery mode. To overcome this problem, the concept of the software defined radio (SDR) has been introduced [1]. Main aim of SDR is to provide a single platform consisting of a hardware layer and number of software layers which incorporates set of radios from different communication standards. The reconfigurable nature of SDR enhances an improvement in the physical layer (PHY) of the existing and upcoming modern wireless system. Mitola [2] has introduced the concept of cognitive radio (CR) in his dissertation. SDR based CRs are fully programmable wireless devices that can sense their environment and dynamically adapt their transmission waveform, channels and spectrum reuse for dynamic spectrum access [1]. The wideband input signal of a typical CR exhibits the coexistence of radio channels based on different wireless communication standards simultaneously. Due to abundant growth in broadband wireless access wireless based electronic devices operating at high data rate becomes a more challenging task compared with wired last mile access technologies [3]. IEEE has developed standards (IEEE 802.11a/g) which are popularly known as wireless local area network (WLAN) [4, 5] and (IEEE 802.16e) known as mobile WiMAX [6]. Among the PHY layer subsystems, the forward error correction (FEC) block is the one, which consumes more silicon due to addressing/permutation tables used in the conventional approaches for interleaver and deinterleaver. To support the rapid growth in wireless technology, the interleaver address generator hardware needs to adapt to the different interleaving standards. Hence, there is a scope for researchers to develop a reconfigurable architecture for address generator supporting multiple radio standards which reduces the silicon cost.

The channel interleaver is a mandatory block in the PHY of 802.11a/g and 802.16e based transmitter. A channel interleaver for multistandard SDR has to support multiple interleaving functions. Interleaving plays a vital role in improving the performance of FEC codes in terms of bit error rate. The basic function of interleaver block is to spread the encoded data into a random one. In the conventional approach, the received data is stored row-wise in the memory and read column-wise after applying certain permutations. To achieve more flexibility in the multimode interleaver, an address generator has to be developed using specialised blocks with reconfigurable capabilities.

Upadhyaya *et al.* [7], have implemented the conventional interleaver on field programmable gate array (FPGA) with reduction in area. In [8], an address generating circuit, for 802.11 interleaver based on the conventional look up table (LUT) method is presented. In [9], a simple technique is developed to implement the one dimensional interleaver equation in matrix form (2D). Khater *et al.* [10] carried out the hardware implementation of the address generator for 1/2 code rate WiMAX channel interleaver. Chang [11] implemented an efficient dual mode deinterleaver for both IEEE 802.16e (block deinterleaver) and outer deinterleaver for digital video broadcasting standards. Yu *et al.* [12] presented a high speed block interleaver/deinterleaver to provide arbitrary column wise permutation by adopting a low-power first in first out bank structure and a simple finite state machine (FSM). In [13], the authors have derived 2D translation of the functions for WiMAX channel deinterleaver which is more complex for 64-QAM. Upadhyaya *et al.* have tested an address generating circuit on FPGA for WiMAX multimode interleaver/deinterleaver in [14, 15] and WLAN multimode interleaver in [16] for all permissible code rates and modulation schemes based on FSM. Asghar, *et al.* [17], proposed a twofold interleaver architecture for different spatial stream application in 802.11n. Zhang *et al.* [18] had presented, a low complexity architecture for interleaver/deinterleaver suitable for MIMO application in 802.11a/g/n wireless LAN. Upadhyaya *et al.* [19] have designed and implemented separate hardware for QPSK, 16-QAM, and 64-QAM address generator on FPGA for WiMAX channel deinterleaver.

In this paper, we propose a reconfigurable architecture of multimode interleaver address generator suitable for multistandard SDR by exploring the redundant hardware between different modulation schemes. In addition, we proposed a novel MOD_row and a MOD_column circuits to compute MOD function for row and column counter values. Our work differs from the approach [19] in the style of implementation, which eliminates the redundant hardware resources and the ROM required to store the modulus (MOD) values as well. Moreover we have derived the mathematical expression of address generator for different modulation schemes and the Boolean expression for the MOD_column and MOD_row circuit. A detailed analysis of the proposed architecture is performed and the results are compared with prevailing techniques which show an improvement in terms of logic utilisation and operating frequency.

The rest of the paper is organised as follows: Section 2 discusses the interleaving techniques in WLAN/WiMAX wireless standards. In Section 3, a reconfigurable algorithm is proposed for multimode interleaver address generations. Section 4 explains the implementation of reconfigurable architecture for the multimode interleaver. The performance analysis of the proposed architecture is presented in Section 5. Finally, the paper is concluded in Section 6.

2 Interleaving in WLAN/WiMAX standards

The wireless LAN standards 802.11a [4], 802.11g [5] and WiMAX standard 802.16e [6] supports the conventional block interleaving schemes. The channel interleaving is a process of rearranging code symbols to reduce the effect of burst error. It processes one block of encoded bits at a time with block size equal to one OFDM symbol depending on the modulation scheme for a specific code rate.

The detailed view of the conventional channel interleaver structure is shown in Fig. 1. It consists of an address generator and two RAMs of the same size. Both the memories are controlled by the *sel* signal, generated from the address generator in such a way that when one memory block is being written, the other one is read, and vice versa. When *sel* = 'HIGH', the input data stream written in M1 correspond to the write address as W_E is active. Simultaneously, the memory M2 outputs the interleaved data stream for the read addresses. The read/write operation of memory blocks are carried out according to the interleaver depth. If it reaches the specified depth, then the status of *sel* signal is changed to swap the read/write operation. Table 1 shows the different depths (N_{cbps}) of channel interleaver for IEEE 802.11a/g and 802.16e to incorporate various code rates and modulation schemes [4–6].

3 Proposed reconfigurable algorithm for multimode interlever address generator

This section describes the mathematical background of the algorithm for the proposed multimode interleaver address generator. The channel interleaving in WiMAX/WLAN is expressed by a set of equations which perform two step permutation as discussed in [9]. (1) ensures the randomisation of encoded bits and (2) validates that the adjacent coded bits are mapped alternately onto less or more significant bits of the signal constellation.

$$m_n = \left(\frac{N_{\text{cbps}}}{d}\right) * (n \% d) + \left\lfloor \frac{n}{d} \right\rfloor \tag{1}$$

$$k_n = s * \left\lfloor \frac{m_n}{s} \right\rfloor + \left(m_n + N_{\text{cbps}} - \left\lfloor \frac{d * m_n}{N_{\text{cbps}}} \right\rfloor \right) \% s \tag{2}$$

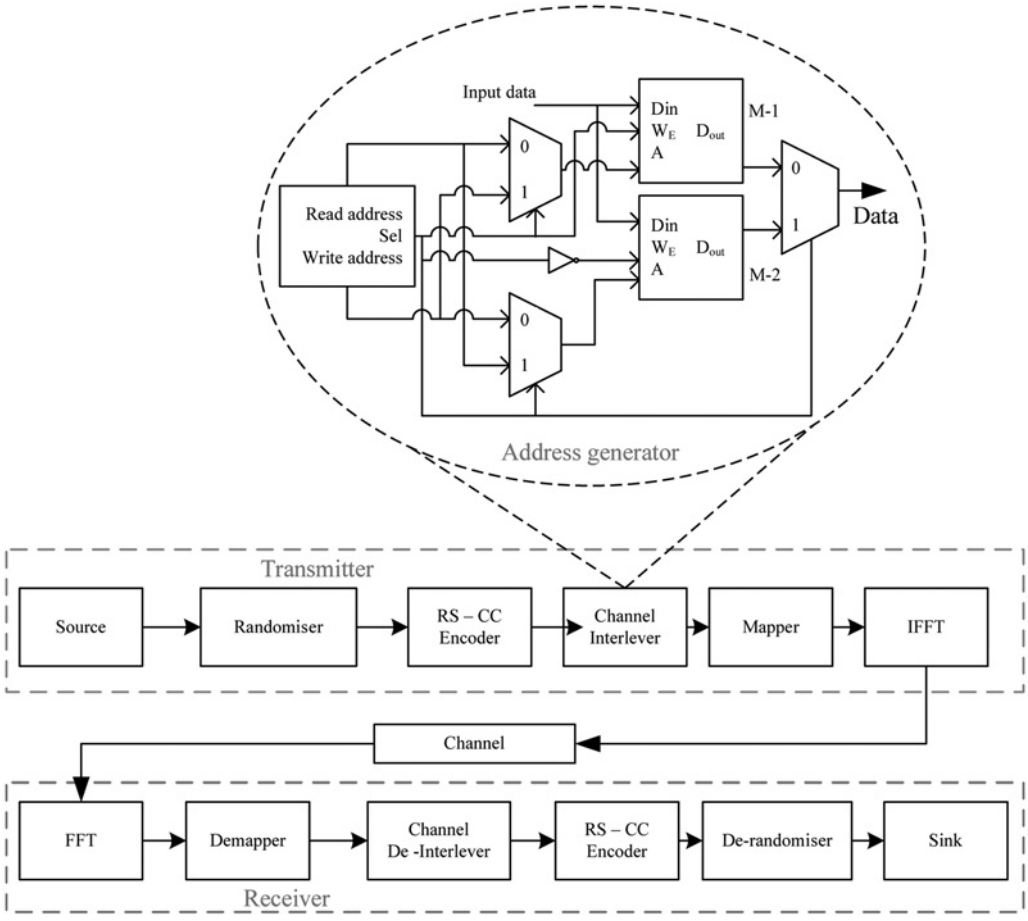

Fig. 1 *General block diagram of Transceiver for Multistandard Radios*

IET Comput. Digit. Tech., 2016, Vol. 10, Iss. 2, pp. 59–68

where, m_n and k_n are the first and second level permutation outputs, respectively; n represents the index of the coded bit in the un-permutated source block which varies from 0 *to* $(N_{cbps}-1)$; s is the parameter defined as max $(N_{bpsc}/2,1)$, where N_{bpsc} represents the number of coded bits per sub-carrier, that is, 1, 2, 4, 6 for BPSK, QPSK, 16-QAM and 64-QAM, respectively [13]. d represents the number of columns and N_{cbps} is the block size corresponding to the number of coded bits per allocated sub-channels. Table 2 shows the theoretical values of N_{bpsc},s, N_{cbps} and number of rows in interleaver memory for different signal constellations [16]. The modulo and floor functions are denoted as % and $\lfloor\rfloor$, respectively.

To generate various permutations values for all modulation schemes and code rates, a MATLAB program is developed using (1) and (2). As d is chosen as 16, the number of columns are fixed $(=d)$ and the number of rows are given by N_{cbps}/d for all N_{cbps}. The general expression for four modulation schemes chosen for discussion is derived by examining the correlation between the addresses as shown in Table 3 [16].

3.1 BPSK/QPSK

For both BPSK/QPSK the value of parameter 's' is 1. Therefore, (2) reduces to $k_n = m_n + 0$

$$k_n = \frac{N_{cbps}}{d} * (n\%d) + \left\lfloor \frac{n}{d} \right\rfloor \tag{3}$$

Introducing the 2D array in which i increments when j expires, the range of i and j as [13]. Now, the interleaver can be realised as a 2D row-column matrix with size $j \times i$. Therefore, (2) can be rewritten as

$$k_{n,\mathrm{BPSK/QPSK}} = \frac{N_{cbps}}{d} * i + j \text{ for } \forall i \text{ and } \forall j \tag{4}$$

where, k_n, $j = 0, 1, \ldots (N_{cbps}/d) - 1$ and $i = 0, 1, \ldots (d-1)$ represents the interleaver address, row and column values, respectively.

3.2 16-QAM

The parameter 's' is 2 for 16-QAM as the number of coded bits per subcarrier is 4. Therefore, (1) and (2) can be rewritten as

$$m_n = \left(\frac{N_{cbps}}{d}\right) * (j\%d) + \left\lfloor \frac{j}{d} \right\rfloor \tag{5}$$

$$k_n = 2 * \left\lfloor \frac{m_n}{2} \right\rfloor + \left(m_n + N_{cbps} - \left\lfloor \frac{d * m_n}{N_{cbps}} \right\rfloor \right)\%2 \tag{6}$$

Considering the 2D i and j, the 2D transformation of interleaver for 16-QAM can be described as:

$$k_{n,16-\mathrm{QAM}} = \begin{cases} \frac{N_{cbps}}{d} * i + j & \text{for } i\%2 = 0 \ \forall j \\ \frac{N_{cbps}}{d} * i + (j+1) & \text{for } i\%2 = 1 \ \& \quad \text{for } j\%2 = 0 \\ \frac{N_{cbps}}{d} * i + (j-1) & \text{for } i\%2 = 1 \ \& \quad \text{for } j\%2 = 1 \end{cases} \tag{7}$$

3.3 64-QAM

For 64-QAM transmission, the number of coded bits per sub-carrier is 6. Thus, using $s = 3$ in (1) and (2), it can be rewritten as

$$m_n = \left(\frac{N_{cbps}}{d}\right) * (j\%d) + \left\lfloor \frac{j}{d} \right\rfloor \tag{8}$$

$$k_n = 3 * \left\lfloor \frac{m_n}{3} \right\rfloor + \left(m_n + N_{cbps} - \left\lfloor \frac{d * m_n}{N_{cbps}} \right\rfloor \right)\%3 \tag{9}$$

The 2D transformation of the interleaver for 64-QAM can be

Table 2 Theoretical values of N_{bpsc}, s, N_{cbps} for different signal constellations

Constellation	N_{bpsc}	s	N_{cbps}	No. of rows
BPSK	1	1	48	3
QPSK	2	1	96	6
16-QAM	4	2	288	18
64-QAM	6	3	384	24

Table 3 First 32 permutation sample address for four code rates and modulation schemes

Parameters	Interleaver addresses							
N_{cbps} = 48-bits, 1/2 code rate, BPSK	0	3	6	9	12	15	18	21
	24	27	30	33	36	39	42	45
	1	4	7	10	13	16	19	22
	25	28	31	34	37	40	43	46
N_{cbps} = 96-bits, 1/2 code rate, QPSK	0	6	12	18	24	30	36	42
	48	54	60	66	72	78	84	90
	1	7	13	19	25	31	37	43
	49	55	61	67	73	79	85	91
N_{cbps} = 192-bits, 1/2 code rate, 16-QAM	0	13	24	37	48	61	72	85
	96	109	120	133	144	157	168	181
	1	12	25	36	49	60	73	84
	97	108	121	132	145	156	196	180
N_{cbps} = 384-bits, 2/3 code rate, 64-QAM	0	26	49	72	98	121	144	170
	193	216	242	265	288	314	337	360
	1	24	50	73	96	122	145	168
	194	217	240	266	289	312	338	361.

Table 1 IEEE 802.11a/g and 802.16e interleaver depths for different code rates and modulation schemes

Modulation Scheme	BPSK		QPSK		16-QAM		64-QAM		
code rate	1/2	3/4	1/2	3/4	1/2	3/4	1/2	2/3	3/4
Interleaver depth, N_{cbps}	48	48	96	96	192	192	288	288	288
	–	–	192	144	384	288	576	384	432
	–	–	288	288	576	576	–	–	–
	–	–	384	432	–	–	–	–	–
	–	–	480	576	–	–	–	–	–
	–	–	576	–	–	–	–	–	–

Algorithm 1

initialise N_{cbps}, *mod_type*, code rate and d

for $j = 0 : 1 : (N_{cbps}/d) - 1$ **do**

for $i = 0 : 1 : d - 1$ **do**

if (EX-OR (*mod_type*) **or** i MOD 2 **or** i MOD 3) = 0

$$k_n = \frac{N_{cbps}}{d} * i + j$$

elseif (i MOD 2 or i MOD 3) = 1

if OR (j MOD 2 or j MOD 3) = 0

if ($mod\_type[0]$) = 0

$$k_n = \frac{N_{cbps}}{d} * i + (j + 2)$$

else

$$k_n = \frac{N_{cbps}}{d} * i + (j + 1)$$

endif

else

$$k_n = \frac{N_{cbps}}{d} * i + (j - 1)$$

endif

elseif (i MOD 2 or i MOD 3) = 2

if (MOD j [1]) = 0

$$k_n = \frac{N_{cbps}}{d} * i + (j + 1)$$

else

$$k_n = \frac{N_{cbps}}{d} * i + (j - 2)$$

endif

endif

endif

endif

end for

end for

Fig. 2 *Proposed reconfigurable algorithm for multimode interleaver address generator*

described as

$$k_{n,64-\mathrm{QAM}} = \begin{cases} \frac{N_{\mathrm{cbps}}}{d} * i + j & \text{for } i\%3 = 0 \ \forall j \\ \frac{N_{\mathrm{cbps}}}{d} * i + (j-2) & \text{for } i\%3 = 2 \ \& \quad \text{for } j\%3 = 2 \\ \frac{N_{\mathrm{cbps}}}{d} * i + (j+2) & \text{for } i\%3 = 1 \ \& \quad \text{for } j\%3 = 0 \\ \frac{N_{\mathrm{cbps}}}{d} * i + (j+1) & \text{for } i\%3 = 2 \ \& \quad \text{for } j\%3 \neq 2 \\ \frac{N_{\mathrm{cbps}}}{d} * i + (j-1) & \text{for } i\%3 = 1 \ \& \quad \text{for } j\%3 \neq 0 \end{cases} \tag{10}$$

Using (6), (9) and (12) an reconfigurable algorithm is developed to eliminate the requirement of modulo and floor function as described in Algorithm 1 (see Fig. 2).

4 Proposed reconfigurable address generator architecture

In this section, the architectural details of the proposed reconfigurable multimode interleaver address generator is presented. The architecture of the proposed reconfigurable multimode interleaver address generator for all modulation schemes is shown in Fig. 3. It comprises of a row counter, column counter, selection unit, multiplier and an adder. The output of column counter is compared with d to generate a fixed count value between 0 to $d-1$. Whereas, the row counter is a variable one, which counts different levels between 0 to $(N_{\mathrm{cbps}}/d) - 1$ based on specific block size selected by multiplexers (MUXs) M6 and M7. From (6), the hardware required to implement BPSK and QPSK is quite similar and also the addresses are equally spaced as shown in Table 4. Whereas, the addresses of 16-QAM and 64-QAM are not equally spaced. Therefore, the address generator has to produce two progressive patterns for 16-QAM and three for the 64-QAM modulation scheme. Thus, the design procedure adapted for QPSK can be extended to 16-QAM and 64-QAM with additional components such as incrementer, decrementer and a separate MOD circuit for row and column counter respectively. The structure shown inside the dashed line in Fig. 3 represents the selection unit for different modulation schemes such as BPSK, QPSK, 16-QAM and 64-QAM block. Each modulation scheme and their corresponding block size are encoded with specific binary values as tabulated in Table 4.

To ensure flexibility in generating address for different modulation schemes, MUX based logic is realised to select incremented/decremented values of row counter based on specific *mod_type*, MOD_row and MOD_column values. The requirement of MOD functions for column and row count values in the address generation of 16-QAM and 64-QAM are satisfied by the proposed MOD_column and MOD_row circuits as shown in Figs. 4 and 5, respectively. From Table 4, the modulation schemes can be categorised into two groups (BPSK, QPSK and 16-QAM, 64-QAM) based on the address spacing in the increment values. In hardware perspective, a simple EX-OR gate is enough to classify the two sets of modulation schemes. The MUX M1 selects the row counter output directly if the EX-OR of *mod_type* is '0' else it selects the incremented/decremented value of the row counter. Likewise, if the MOD_column value is '0', the row counter output is directly selected in M5 for all modulation schemes as given in (6), (9) and (12). If MOD_column value is '1', the MUX M3 should select decrement by 1 logic else the output of MUX M2. Whereas, M2 generates two different outputs, that is, increments by 1 for 16-QAM and increments by 2 for 64-QAM based on *mod_type [0]*. While performing 64-QAM, the MOD_column value reaches a value of 2. In this case, the MUX M4 selects the row counter output incremented by 1 based on MOD_row[1] = '0' else decrement by 2. In our design, we have reduced the size of the MUXs compared with the logic in [19] by having special kind of select signal. Moreover, the control signals are not generated externally, but generated internally from encoded binary values.

4.1 Working of MOD_column

As, the number of column (d) is chosen as 16, the column counter values goes between 0 to $(d-1)$, that is, maximum value of 15. Hence, a 4-bit MOD_column circuit is proposed to compute the MOD value of column counter as shown in Fig. 4. with 4-bit input as C_3, C_2, C_1 and C_0 from MSB to LSB. The final MOD2 or MOD3 value for the column counter is obtained by performing the successive steps as shown in Table 5.

4.1.1 Logical expression for MOD_column: In general, for a 2-bit ripple carry adder with two 2-bit input numbers (B_1, B_0, A_1, B_0), that is, totally 4-bit and the outputs are 2-bit sum (Sum_1, Sum_0) and a carry bit (C_{out}). The general Boolean expression for sum and carry of a 2-bit ripple carry adder is represented as

$$\begin{aligned} \mathrm{Sum}_1 = {} & \overline{B_1} \cdot \overline{B_0} \cdot A_1 + \overline{B_1} \cdot A_1 \cdot \overline{A_0} + \overline{B_1} \cdot B_0 \cdot \overline{A_1} \cdot A_0 \\ & + B_1 \cdot \overline{B_0} \cdot \overline{A_1} + B_1 \cdot \overline{A_1} \cdot \overline{A_0} + B_1 \cdot B_0 \cdot A_1 \cdot A_0 \end{aligned} \tag{11}$$

$$\mathrm{Sum}_0 = \overline{B_0} \cdot A_0 + B_0 \cdot \overline{A_0} \tag{12}$$

$$C_{\mathrm{out}} = B_0 \cdot A_1 \cdot A_0 + B_1 \cdot A_1 + B_1 \cdot B_0 \cdot A_0 \tag{13}$$

The four bit inputs (C_3, C_2, C_1, C_0) are grouped into two 2-bit numbers. To obtain the Boolean expression of the sum (s_1, s_0) and

IET Comput. Digit. Tech., 2016, Vol. 10, Iss. 2, pp. 59–68

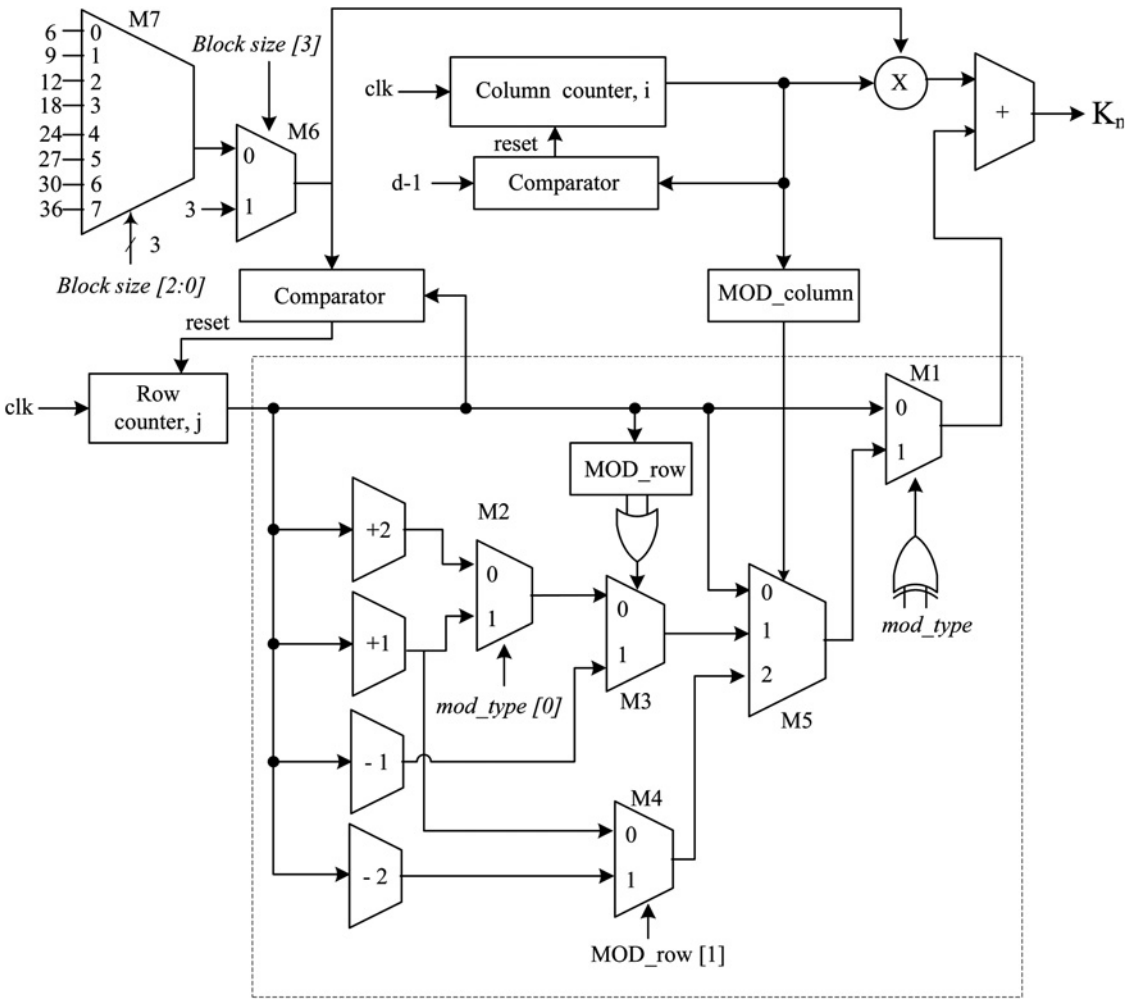

Fig. 3 *Proposed reconfigurable architecture of multimode Interleaver address generator supports IEEE 802.11a/g and 802.16e wireless standards*

carry (c_0) of the first 2-bit adder in Fig. 4, the variables (B_1, B_0, A_1, A_0) in (11), (12) and (13) are replaced with (C_3, C_2, C_1, C_0).

$$s_1 = \overline{C_3}\cdot\overline{C_2}\cdot C_1 + \overline{C_3}\cdot C_1\cdot\overline{C_0} + \overline{C_3}\cdot C_2\cdot\overline{C_1}\cdot C_0 + C_3\cdot\overline{C_2}\cdot\overline{C_1} + C_3\cdot\overline{C_1}\cdot\overline{C_0} + C_3\cdot C_2\cdot C_1\cdot C_0 \quad (14)$$

Table 4 Encoded binary values for various interleaver depths and modulation schemes of IEEE 802.11a/g and IEEE 802.16e

Modulation	*mod_type [1:0]*	Interleaver depths (N_{cbps})	*Block size [3:0]*	Increment values
BPSK	0 0	48	1 0 0 0	3
		96	0 0 0 0	6
		192	0 0 1 0	12
		288	0 0 1 1	18
QPSK	1 1	96	0 0 0 0	6
		144	0 0 0 1	9
		192	0 0 1 0	12
		288	0 0 1 1	18
		384	0 1 0 0	24
		432	0 1 0 1	27
		480	0 1 1 0	30
		576	0 1 1 1	36
16-QAM	0 1	192	0 0 1 0	13,11
		288	0 0 1 1	19, 17
		384	0 1 0 0	25, 23
		576	0 1 1 1	37, 35
64-QAM	1 0	288	0 0 1 1	20,17,17
		384	0 1 0 0	26, 23, 23
		432	0 1 0 1	29, 26, 26
		576	0 1 1 1	38, 35, 35.

$$s_0 = \overline{C_2}\cdot C_0 + C_2\cdot\overline{C_0} \quad (15)$$

$$c_0 = C_2\cdot C_1\cdot C_0 + C_3\cdot C_1 + C_3\cdot C_2\cdot C_0 \quad (16)$$

Similarly, the Boolean expression of the sum (s_3, s_2) of the second 2-bit adder in Fig. 4, can be obtained by substituting the values of

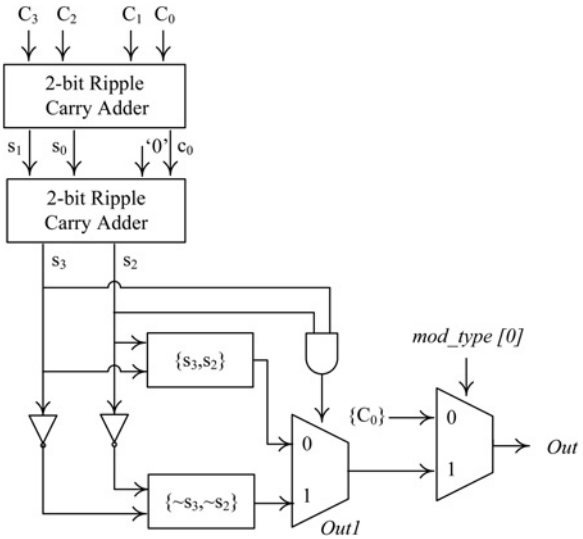

Fig. 4 *Proposed architecture of MOD_column circuit with 4-bit input*

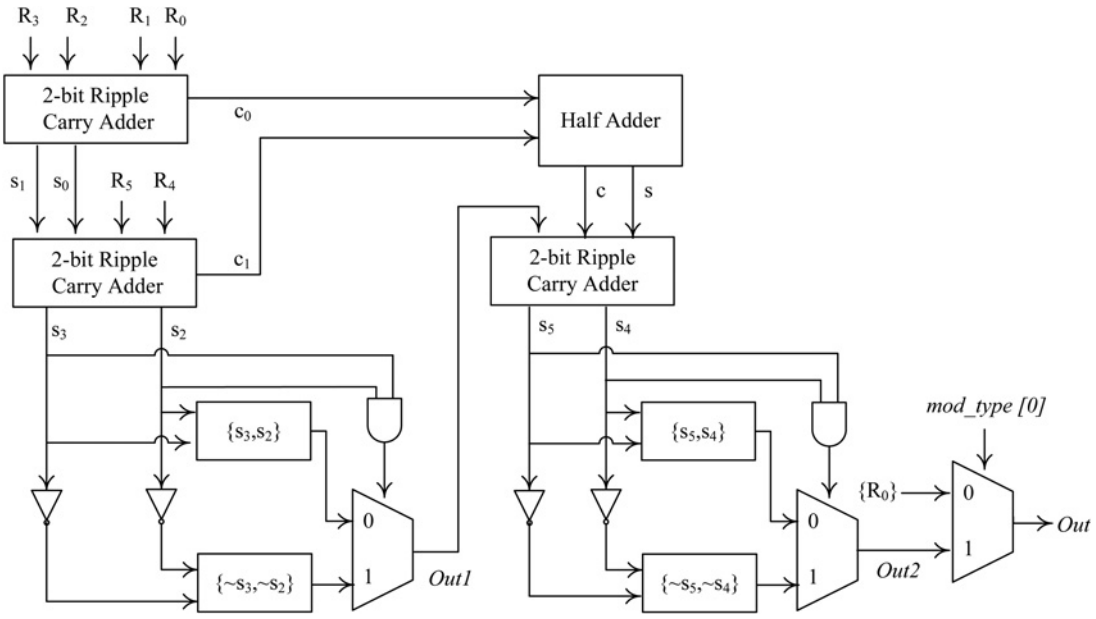

Fig. 5 *Proposed architecture of MOD_row circuit with 6-bit input*

Table 5 Working of proposed MOD_column circuit

STEP	For the count value of 12.	Output	Operation
1	12 = '1100' as 11 00	$C_1C_0 = 00$ $C_3C_2 = 11$	Group in to two bits each
2	$C_1C_0 = 00$ $+ C_3C_2 = 11$ 0) 11	$s_1s_0 = 11$ (sum) $c_0 = 0$ (carry)	2-bit ripple carry addition
3	$s_1s_0 = 11$ $+ 0\ c_0 = 00$ 0) 11	MSB is fixed to '0' $s_3s_2 = 11$ (sum)	2-bit ripple carry addition
4	If $s_3s_2 = 11$, complement each bit; Else, pass the same.	*Out1* = 00, that is, MOD3 value	Complement
5	C_0 represents MOD2 value	$C_0 = 0$, that is, MOD2 value	
6	Final MUX with the selection line as *mod_type*[0][a]	*mod_type*[0] = 0 for 64-QAM	MOD_column Output, that is, *Out* = $C_0 = 0$

[a]From Table 4, the four modulation schemes (mod_type) are assigned with binary values, that is, *mod_type[0]* = 0 for (BPSK, 64-QAM) and *mod_type[0]* = 1 for (QPSK, 16-QAM).

(s_1, s_0, 0, c_0) in the place of (B_1, B_0, A_1, A_0) in (11), (12) and (13).

$$s_3 = \overline{C_3}\cdot C_2\cdot\overline{C_1}\cdot C_0 + \overline{C_3}\cdot\overline{C_2}\cdot C_1 + \cdot C_2\cdot C_1\cdot\overline{C_0} + C_3\cdot\overline{C_2}\cdot\overline{C_1} + C_3\cdot\overline{C_1}\cdot\overline{C_0} + C_1\cdot C_3\cdot C_0 \quad (17)$$

$$s_2 = \overline{C_3}\cdot\overline{C_2}\cdot C_1\cdot\overline{C_0} + \overline{C_3}\cdot\overline{C_2}\cdot C_0 + \overline{C_3}\cdot C_2\cdot\overline{C_0} + C_2\cdot\overline{C_1}\cdot\overline{C_0} + C_2\cdot C_1\cdot C_0 + C_3\cdot\overline{C_1}\cdot C_0 \quad (18)$$

From the Table 6, it is noted that the sum value (s_3, s_2) coincides with the MOD3 (modulo 3) of the 4-bit inputs, except for the case of sum value is 3 ('11'). Where, the required MOD3 value is 0 ('00'), that is, complement.

The key point is that, if the sum value goes to '3' we have to bring into '0'. Hence, we introduced a MUX for selecting either the direct value of sum (s_3, s_2) or its complement value ($\sim s_3$, $\sim s_2$) with the control signal (*sel*) generated by performing AND operation of the

Table 6 Truth table for the MOD_column circuit

C_3	C_2	C_1	C_0	s_1	s_0	'0'	c_0	s_3	s_2	*sel*	MOD3		MOD2
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	1	0	0	1	1
0	0	1	0	1	0	0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0	1	1	1	0	0	1
0	1	0	0	0	1	0	0	0	1	0	0	1	0
0	1	0	1	1	0	0	0	1	0	0	1	0	1
0	1	1	0	1	1	0	0	1	1	1	0	0	0
0	1	1	1	0	0	0	1	0	1	0	0	1	1
1	0	0	0	1	0	0	0	1	0	0	1	0	0
1	0	0	1	1	1	0	0	1	1	1	0	0	1
1	0	1	0	0	0	0	1	0	1	0	0	1	0
1	0	1	1	0	1	0	1	1	0	0	1	0	1
1	1	0	0	1	1	0	0	1	1	1	0	0	0
1	1	0	1	0	0	0	1	0	1	0	0	1	1
1	1	1	0	0	1	0	1	1	0	0	1	0	0
1	1	1	1	1	0	0	1	1	1	1	0	0	1

IET Comput. Digit. Tech., 2016, Vol. 10, Iss. 2, pp. 59–68

Table 7 Working of proposed MOD_row circuit

STEP	For the count value of 35.	Output	Operation
1	35 = '100011' as 10 00 11	$R_1R_0 = 11$ $R_3R_2 = 00$ $R_5R_4 = 10$	Group in to two bits each
2	$R_1R_0 = 11$ + $R_3R_2 = 00$ 0) 11	$s_1s_0 = 11$ (sum) $c_0 = 0$ (carry)	2-bit ripple carry addition
3	$s_1s_0 = 11$ + $R_5R_4 = 10$ 1) 01	$s_3s_2 = 01$ (sum) $c_1 = 1$ (carry)	2-bit ripple carry addition
4	If $s_3s_2 = 11$, complement each bit; Else, pass the same.	*Out1* = 01	No Complement
5	$c_0 = 0$ + $c_1 = 1$ 0) 1	s = 1 (sum) c = 0 (carry)	Half adder
6	$c\ s = 01$ + *Out*1 = 01 0) 10	$s_5s_4 = 10$ (sum)	2-bit ripple carry addition
7	If $s_5s_4 = 11$, complement each bit; Else, pass the same.	*Out2* = 10, that is, MOD3 value	No Complement
8	R_0 represents MOD2 value	$R_0 = 0$, that is, MOD2 value	
9	Final MUX with the selection line as *mod_type*[0][a]	*mod_type*[0] = 1 for QPSK	MOD_column Output, that is, *Out* = '10' = 2.

[a]From Table 4, the four modulation schemes (mod_type) are assigned with binary values, that is, *mod_type[0]* = 0 for (BPSK, 64-QAM) and *mod_type[0]* = 1 for (QPSK, 16-QAM).

sum bit s_3 and s_2 as given in Fig. 4.

$$sel = s_3 \cdot s_2 = (C_3 \oplus C_1)(C_2 \oplus C_0) + C_3 \cdot C_2 \cdot C_1 \cdot C_0$$

The output of the MUX is

$$Out1 = \{\sim s_3,\ \sim s_2\} \cdot sel + \overline{sel} \cdot \{s_3, s_2\}$$

Finally, the output of the MOD_column is obtained by selecting either MOD2 (C_0) or MOD3 (*Out1*) with the selection signal as *mod_type*[0].

4.2 Working of MOD_row

From Table 4, the maximum Block size (N_{cbps}) for IEEE 802.11a/g and IEEE 802.16 interleaver is 576. Therefore, the row counter counts up to maximum value of $(N_{cbps}/d) - 1 = 35$. In binary form, it can be represented with 6-bit as '100011'. Hence, a MOD_row circuit is proposed to compute MOD function for 6-bit input named as R_5, R_4, R_3, R_2, R_1, R_0 from MSB to LSB as shown in Fig. 5. The final MOD2 or MOD3 for the row counter is obtained by performing the successive steps as shown in Table 7.

The Boolean expression for the MOD_row circuit can be derived in the same manner as MOD_column. The MUX selects either the direct value of sum (s_5, s_4) or its complement value ($\sim s_5$, $\sim s_4$) with the control signal (*sel*) generated by performing AND operation of the sum bit s_5 and s_4 as given in Fig. 5. Finally, the output of the MOD_row (*Out*) is obtained by selecting either MOD2 values (R_0) or the MOD3 value (*Out*2) with the selection signal as *mod_type[0]* (Table 7).

Therefore, the proposed MOD circuit for row and column, eliminates the need for two ROMs of dimension 16×3-bit and 64×3-bit in [19] to store the MOD values for 16-QAM and 64-QAM respectively, which yields a substantial amount of reduction in resources and improvement in performance.

5 Performance analysis of the proposed architecture

In this section, the performance analysis of the proposed MOD function circuit and the reconfigurable address generator with existing architectures are discussed in detail.

5.1 Mod circuit analysis and synthesis results

Several implementation techniques for computing modulo function with varying input and modulo bit width on FPGA and ASIC platform are discussed in [20–22]. In [20], Sivakumar *et al.* discusses the implementation of *X mod m* architecture with the MOD values of 3, 5, 6, 7, 9 and 10 in an ASIC chip using 3 µm CMOS technology. Since, our proposed MOD circuit for address generation is implemented on FPGA, it is difficult to make a direct comparison with [20].

In [21], Butler, *et al.* have developed two methodologies for computing *x mod z* with *z* fixed as 3, where *x* is the n-bit input and *z* is the MOD value. First approach is based on combinational circuits and the second design uses cascade of LUTs. To have better comparison with [21], we have also performed the synthesis of the proposed MOD circuit on Altera Stratix IV EP4SE530F43C3NES FPGA using ALTERA Quartus II tool with the input bit width of 8 and 16.

From Table 8, it is inferred that the proposed MOD circuit is much faster and consumes fewer resources than the first approach in [21]. It shows an average reduction of 65% in total registers and 28%

Table 8 Resource utilisation comparison of MOD circuit on Altera Stratix IV EP4SE530F43C3NES FPGA with first approach of [21]

n-bit input	*x mod z* [21]						Proposed MOD					
	freq. (MHz)	# of r-input LUTs				est. # of ALMs	total # of registers	freq. (MHz)	# of r- input LUTs			est. of ALMs	total # of registers
		6	5	4	3				6	5	4	3	
8	573.5	0	14	7	15	29 (0%)	50 (0%)	745.12	0	2	0	7	9 (0%)	20 (0%)
16	498.1	5	12	17	124	134 (0%)	226 (0%)	624.81	0	4	2	14	21 (0%)	43 (0%)

Table 9 Resource utilisation comparison of MOD circuit on Altera Stratix IV EP4SE530F43C3NES FPGA with second approach of [21]

n-bit input	*x mod z* [21]						Proposed MOD					
	freq. (MHz)	# of r-input LUTs				est. # of ALMs	total # of registers	freq. (MHz)	# of r-input LUTs			est. # of ALMs	total # of registers
		6	5	4	3				6	5	4	3	
8	1520.2	0	0	0	12	14 (0%)	27 (0%)	745.12	0	2	0	7	9 (0%)	20 (0%)
16	1520.2	0	0	0	28	60 (0%)	119 (0%)	624.81	0	4	2	14	21 (0%)	43 (0%)

Table 10 Resource utilisation comparison of MOD circuit on Xilinx FPGA with 10-bit input for $P = 7$

Device	*X mod P*[22]		Proposed MOD	
	no. of slice LUTs	time, in ns	no. of slice LUTs	time, in ns
Virtex 7 XC7V285t	49 (0%)	8.28	18 (0%)	3.591
Spartan 3 XC3S1000	92 (0%)	12.75	30 (0%)	13.517

improvement in operating speed with inputs of 8 and 16 bit width. From the results, it is concluded that for higher input bit width, the proposed circuit provides further reduction in resource utilisation and improvement in operating speed.

Compared with the second design using LUTs, our proposed design also consumes less logic resources. From Table 9, it is observed that the proposed design achieves an average reduction of 44% in total registers. Since, the LUT approach adapts LUT in each stage rather than combinational logic, the design able to operate at high speed with increase in latency. The trade-off between the complexity and latency were discussed in [21].

In [22], Gorodecky has developed a circuit for the calculation of *X mod P* with the Boolean expressions are represented in Reed–Muller XOR polynomial form. It uses XOR and AND operators for the computation. For comparison of the proposed MOD circuit with [22], we have extended the logic behind our proposed MOD 3 circuit to develop a MOD 7 function with input of 10-bit width and the synthesis are performed on both Virtex 7 XC7V285t and Spartan 3 XC3S1000 FPGA. The results are tabulated in Table 10 to show its better performance in terms of LUTs and propagation time.

5.2 Reconfigurable address generator analysis and results

The proposed reconfigurable address generator architecture is developed using Verilog HDL [23] and the functionalities are verified using Xilinx ISE. The whole design is implemented on Xilinx Spartan XC3S400 FPGA [24] device to have a better comparison of the proposed work with the existing approaches in [15], [19].

5.2.1 FPGA simulation results: The Xilinx ISE simulation results for $N_{cbps} = 48$-bits, 3/4 code rate, BPSK; $N_{cbps} = 96$-bits, 1/2 code rate, QPSK; $N_{cbps} = 192$-bits, 1/2 code rate, 16-QAM and 64-QAM modulation scheme for $N_{cbps} = 384$-bits, 2/3 code rate

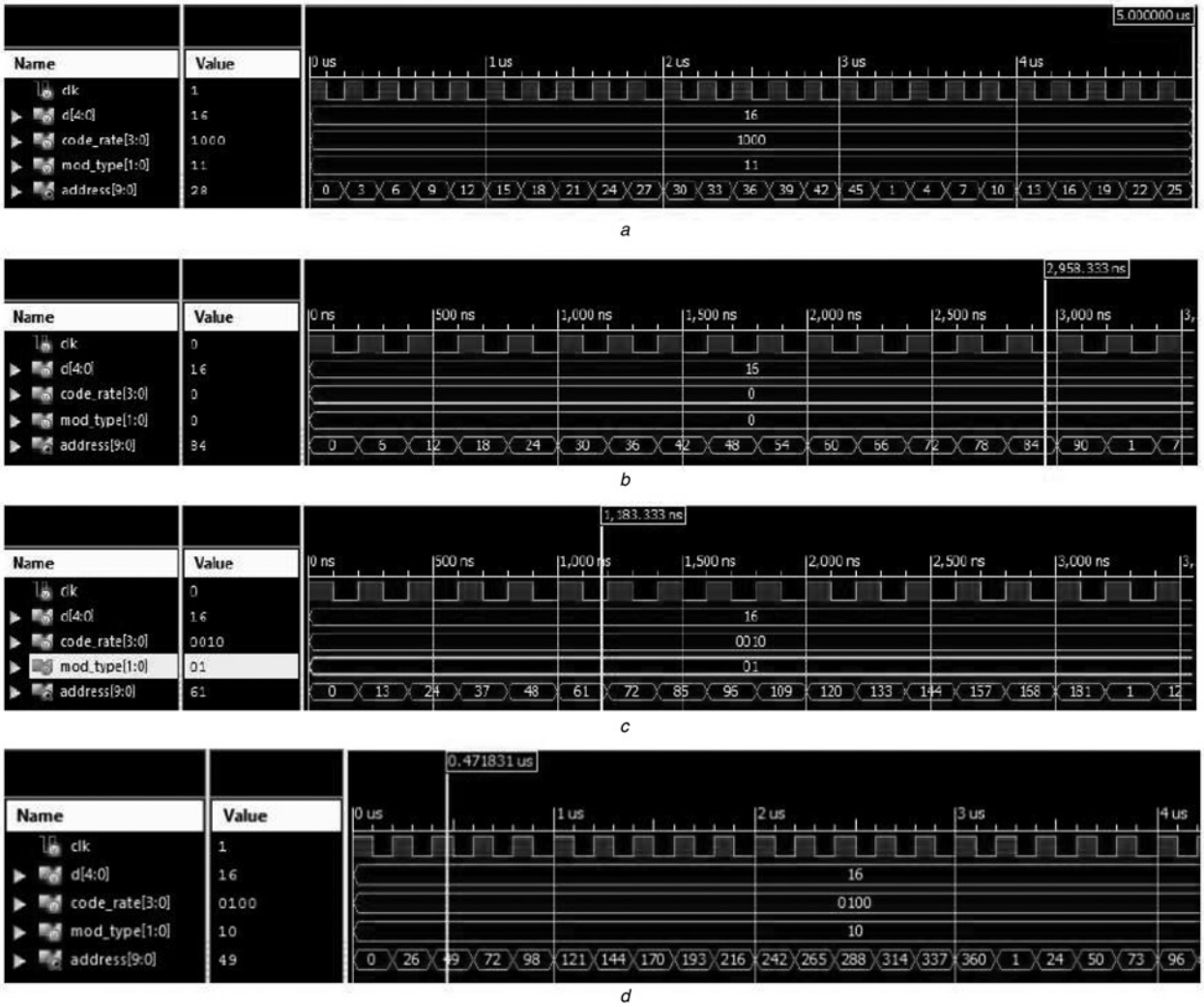

Fig. 6 *Simulation results showing the addresses for*

a $N_{cbps} = 48$-bits, 3/4 code rate, BPSK
b $N_{cbps} = 96$-bits, 1/2 code rate, QPSK
c $N_{cbps} = 192$-bits, 1/2 code rate, 16-QAM
d 64-QAM modulation scheme for $N_{cbps} = 384$-bits, 2/3 code rate

IET Comput. Digit. Tech., 2016, Vol. 10, Iss. 2, pp. 59–68

Table 11 Comparison of logic resources between the proposed and existing approach

Logic resources	FSM based method [15]		LUT based method [19]		Mux based method [19]		proposed method	
	used	utilisation	used	utilisation	used	utilisation	used	utilisation
slices	108	6.75%	633	17.66%	125	3.49%	56	1.56%
flip-flops	37	0.67%	55	0.78%	35	0.5%	10	1.03%
4–input LUT's	398	5.5%	1229	17.15%	240	3.35%	106	1.47%
operating frequency, MHz	191.05		62.51		121.82		178.42	

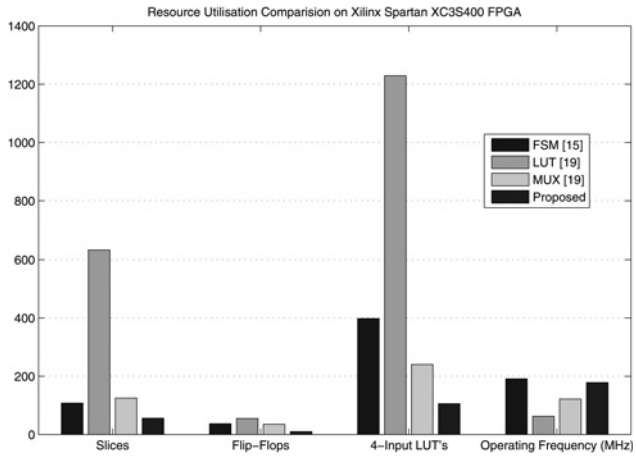

Fig. 7 *Resource utilisation comparison of the proposed and existing approaches*

has been presented in Fig. 6. These FPGA simulation results are verified with the outputs obtained from the MATLAB.

5.2.2 FPGA synthesis results: In the case of FSM based method [15], for each block size, a unique MUX is used for all modulation schemes which results in increase of hardware and decrease in critical path delay. From Table 11, it is evident that the proposed technique shows reduction of 48% in FPGA slices, 72% in flip flops, 73% in 4-input LUT's and with 6% lesser operating frequency compared with FSM method [15]. Compared with LUT based approach [19], our work eliminates the conventional usage of block RAMs to house the address for each interleaver depths which results in significant reduction in slices (by 91%), in flip flops (by 81%), in 4-input LUT's (by 91%) and the design operates at 1.85 times faster rate. Our work shows an reduction of 55% in terms of logic slices, 71% in terms of flip flops and 56% in terms of 4-input LUT's and operates at 46% faster in terms of operating speed compared with MUX based approach in [19]. For a better comparison of the implementation results, a bar chart of resource utilisation for various methods against the proposed method is shown in Fig. 7.

6 Conclusion

In this paper, a reconfigurable address generator architecture of multimode interleaver for IEEE 802.11a/g and 802.16e wireless standards supporting all possible code rates and modulation schemes has been presented. The complete hardware was developed using Verilog HDL and implemented on Xilinx Spartan 3 FPGA. In addition, a novel MOD_row and a MOD_column circuits were proposed to compute MOD function for row and column counter values. The synthesis results of the proposed MOD circuit shows significant reduction in resource utilisation and improvement in operating speed compared with existing modulo algorithms. MATLAB simulation results endorse the functionality of the address generator for various interleaver depths. A detailed analysis of the implementation results has been made to show the performance of the proposed method is improved compared with the existing methods. The proposed work shows an average of 60% reduction in resource utilisation and an improvement of 46% in operating frequency compared with the existing approaches. Thus, the proposed reconfigurable address generator architecture can be used to support various modulation schemes in multimode interleaver of multistandard SDR devices.

7 Acknowledgment

The authors thank UK-India Education and Research Initiative (UKIERI) Thematic Partnerships under grant no. UKUTP201100134, India for providing necessary support in this work.

8 References

1 Mitola, J., Maguire, G.Q.: 'Cognitive radio: making software radios more personal', *IEEE Pers. Commun.*, 1999, **33**, (4), pp. 13–18
2 Mitola, J.: 'Cognitive radio – an integrated agent architecture for software defined radio'. PhD thesis, Royal Institute of Technology (KTH), 2000
3 Konhauser, W.: 'Broadband wireless access solutions–Progressive challenges and potential value of next generation', *Wireless Pers. Common.*, 2006, **37**, (4), pp. 243–259
4 IEEE 802.11a: 'Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: high-speed physical layer in the 5 GHz band', 1999
5 IEEE 802.11g: 'Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: further higher data rate extension in the 2.4 GHz band', 2003
6 IEEE 802.16e: 'Local and metropolitan networks–part 16: air interface for fixed broadband wireless access systems', 2005
7 Upadhyaya, B.K., Sanyal, S.K.: 'VHDL modeling of convolutional interleaver–deinterleaver for efficient FPGA implementation', *Int. J. Recent Trends Eng.*, 2009, **2**, (6), pp. 66–68
8 Sghaier, A., Ariebi, S., Dony, B.: 'A pipelined implementation of OFDM transmission on reconfigurable platforms'. Canadian Conf. on Electrical and Computer Engineering, December 2007, pp. 801–804
9 Asghar, R., Liu, D.: 'Low complexity multimode interleaver core for WiMAX with support for convolutional interleaving', *Int. J. Electron. Commun. Comput. Eng.*, 2009, **1**, (1), pp. 20–29
10 Khater, A.A., Khairy, M.M., Habib, S.E.D.: 'Efficient FPGA implementation for the IEEE 802.16e interleaver'. Proc. Int. Conf. Microelectron., Marrakech, Morocco, December 2009, pp. 181–184
11 Chang, Y.N., Ding, Y.C.: 'A low-cost dual mode de-interleaver design'. Proc Int. Conf. Consum. Electron., Las Vegas, USA, January 2007, pp. 1–2
12 Yu, C., Yen, M.H., Hsiung, P.A.: 'Design of a high-speed block interleaving/deinterleaving architecture for wireless communication applications'. Proc. Int. Conf. Consumer Electronics, Las Vegas, USA, January 2009, pp. 1–2
13 Asghar, R., Liu, D.: '2D realization of WiMAX channel interleaver for efficient hardware implementation'. Proc. World Academy of Science and Engineering Technology, Hong Kong, April 2009, pp. 25–29
14 Upadhyaya, B.K., Misra, I.S., Sanyal, S.K: 'Novel design of address generator for WiMAX multimode interleaver using FPGA based finite state machine'. Proc. 13th Int. Conf. Computer and Information Technology, Dhaka, Bangladesh, December 2010, pp. 153–158
15 Upadhyaya, B.K., Sanyal, S.K.: 'Novel design of WiMAX multimode interleaver for efficient FPGA implementation using finite state machine based address generator', *Int. J. Commun.*, 2012, **6**, (2), pp. 27–36
16 Upadhyaya, B.K., Sanyal, S.K.: 'Design of a novel FSM based reconfigurable multimode interleaver for WLAN application'. Proc. Int. Conf. on Devices and Communications, Mesra, India, February 2011, pp. 1–5
17 Asghar, R., Dake, L.: 'Low complexity hardware interleaver for MIMO-OFDM based wireless LAN'. IEEE Int. Symp. on Circuits Syst. Circuits and Systems, Taiwan, May 2009, pp. 1747–1750

18 Zhang, Z., Wu, B., Zhou, Y.M., *et al.*: 'Low-complexity hardware interleaver/deinterleaver for IEEE 802.11a/g/n WLAN', *Hindawi VLSI Design*, **2012**, 2012. doi:10.1155/2012/948957
19 Upadhyaya, B.K., Sanyal, S.K.: 'Efficient FPGA implementation of address generator for WiMAX deinterleaver', *IEEE Trans. Circuits Syst. II*, 2013, **60**, (8), pp. 492–496
20 Sivakumar, R., Dimapolos, N.J.: 'VLSI architecture for computing X mod m', *IEE Proc. Circuit Syst.*, 1995, **142**, (5), pp. 313–320
21 Butler, J.T., Sasao, T.: 'Fast hardware computation of *x mod z*'. Proc. 18th Reconfigurable Architectures Workshop (RAW 2011), Anchorage, Alaska, USA, May 2011, pp. 294–267
22 Gorodecky, D.A.: 'Reed-muller realization of X (mod P)', CoRR, 2015, abs/1504.04773,- arxiv.org
23 Michael, D.C.: 'Advanced digital design with the Verilog HDL' (Prentice-Hall, 2010, 2nd edn.)
24 Xilinx: 'DS099 Spartan-3 FPGA family data sheet', http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf