Journal of Circuits, Systems, and Computers
Vol. 29, No. 7 (2020) 2050111 (16 pages)

DOI: 10.1142/S021812662050111X

Design of the Baseband Physical Layer of NarrowBand IoT LTE Uplink Digital Transmitter*

Basma H. Mohamed†, Ahmed Taha†, Ahmed Shawky†, Essraa Ahmed†, Ali Mohamed†, Manar Mohsen†, Rodina Samy‡, Amr ELHosiny‡, Ahmed Ibrahim‡ and Hassan Mostafa†,§,¶

†*Electronics and Communications Engineering Dept., Cairo University (CU), Egypt*

‡*Si-Vision LLC Egypt*

§*University of Science and Technology, Nanotechnology and Nanoelectronics Program, Zewail City of Science and Technology, October Gardens, 6th of October, Giza 12578, Egypt*

¶*hmostafa@zewailcity.edu.e*

Received 6 April 2019
Accepted 7 August 2019
Published 25 September 2019

With the new age of technology and the release of the Internet of Things (IoT) revolution, there is a need to connect a wide range of devices with varying throughput and performance requirements. In this paper, a digital transmitter of NarrowBand Internet of Things (NB-IoT) is proposed targeting very low power and delay-insensitive IoT applications with low throughput requirements. NB-IoT is a new cellular technology introduced by 3GPP in release 13 to provide wide-area coverage for the IoT. The low-cost receivers for such devices should have very low complexity, consume low power and hence run for several years. In this paper, the implementation of the data path chain of digital uplink transmitter is presented. The standard specifications are studied carefully to determine the required design parameters for each block. And the design is synthesized in UMC 130-nm technology.

Keywords: NarrowBand IoT; LTE; physical layer; digital design; transceiver; low power.

1. Introduction

Over the past decade, there has been a tremendous growth in the number of wireless devices. As wireless technology matures, this number is expected to grow at an even higher rate with the goal being able to connect every physical device to the Internet. The idea behind IoT is having a network of computing devices, vehicles and embedded systems devices. The IoT generates, exchanges and processes data

*This paper was recommended by Regional Editor Piero Malcovati.
¶Corresponding author.

intelligently and continuously. About 50 billion devices are estimated to be connected with IoT by 2020, up from about 200 million in the year 2000 and 10 billion in 2013.[1] With such a massive number of devices connected to each other, Machine to Machine (M2M) communication is needed where devices would communicate with each other without any human interaction.

Long Term Evolution (LTE) is one of the latest and widely accepted high-speed wireless communication standard by 3rd Generation Partnership Project (3GPP). However, LTE is designed for high-speed communication and is not optimized for applications that need to support a potentially large number of low-rate, low power and delay tolerant devices. These low cost devices, typically used in applications such as sensors, remote maintenance and tracking, health-care, etc., are expected to have low complexity, low mobility and low power consumption with a long battery life.[2] Hence, the objective is to develop LTE-based wireless systems suitable for low data-rate and low power IoT applications.

This work introduces a low power design for the physical layer of uplink digital transmitter targeting IoT applications, with low throughput and relaxed transmission delay requirements. The overall system, NB-IoT, is based on LTE standard which is discussed in detail in the next section. The rest of the paper is organized as follows: Section 2 introduces NB-IoT LTE standard specifications based on release 14. Section 3 introduces the uplink transmitter block diagram and the design implementation of each individual block, Sec. 4 shows the full chain simulation on ModelSim software in addition to synthesis power and area results which are tabulated and justified. Finally, a conclusion section to summarize what is introduced.

2. NB-IoT LTE Specification

NB-IoT LTE is a low bandwidth (180 kHz per device), low-power and low complexity with extended coverage using repetitions. Besides being variable Transport Block Size (TBS) whose maximum 2536 bits as stated in release 14, and flexible deployment which enables spectrum reforming.

According to 3GPP Release 13 (Ref. 3) type 1 which supports frequency division duplexing (FDD) half-duplex type-B is chosen as the duplex mode whereas legacy LTE also supports full-duplex mode.[4] This frame structure consists of 10 subframes and each subframe consists of 2 time slots. As stated in Ref. 5, NB-IoT supports 15 kHz and 3.75 kHz subcarrier spacing. In this design, 15 kHz subcarrier spacing is chosen, to decrease Inverse Fast Fourier Transform (IFFT) and Resource Element Mapper (REM) size. This subcarrier spacing divides the bandwidth into 12 subcarriers. Further, since the target data-rate and power requirements are low, only lower order modulation schemes are used, Binary Phase-Shift Keying (BPSK) and Quadrature Phase-Shift Keying (QPSK).[5,6]

3. NB-IoT LTE Transmitter Chain Implementation

NB-IoT LTE Physical Uplink Shared Channel (PUSCH) Transmitter chain as shown in Fig. 1 consists of three main parts:

(1) The channel coding which includes: Cyclic Redundancy Check (CRC), turbo encoding, rate matching, data multiplexing and channel interleaving blocks.
(2) Physical channel which includes: scrambler and modulation mapper blocks.
(3) Blocks to generate the Single Carrier Frequency Division Multiple Access (SC-FDMA) symbol which are Fast Fourier Transform (FFT), REM and IFFT.

3.1. *Cyclic redundancy check*

It is an error detection technique which enables receiving the transmitted message through an error-introducing channel and determine whether the message has been corrupted or not. This is accomplished by generating a value (called a checksum) in the transmitter and appending it to the message. In the other side, the receiver uses the same function to calculate the checksum of the received message and compare it with the appended one to see if the message is correctly received or not.

In NB-IoT-LTE, the size of the — CRC bits (i.e., the checksum required to be appended) is 24 bit — that are generated by 24 shift registers and XORs according to the following polynomial:

$$g_{CRC24A}(D) = [D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1]. \quad (1)$$

CRC operation starts by entering the input concatenated by 24 zeros then getting the output through the last register. When the 24 CRC bits are ready in the shift registers, they are not taken through the last register; as they are XOR-ed together depending on the given polynomial. The introduced solution is shifting these bits only without "XOR" through multiplexers as shown in Fig. 2.

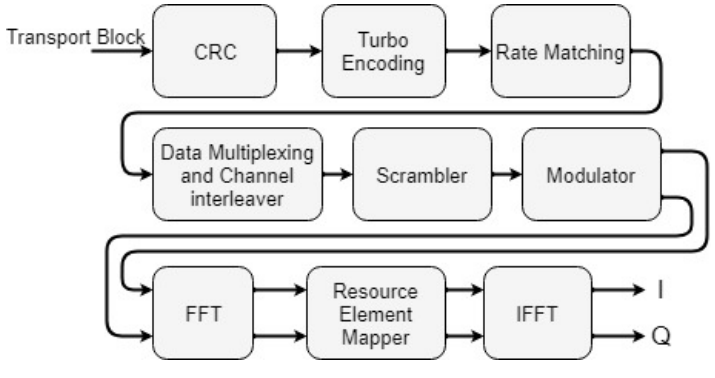

Fig. 1. Transmitter chain block diagram.

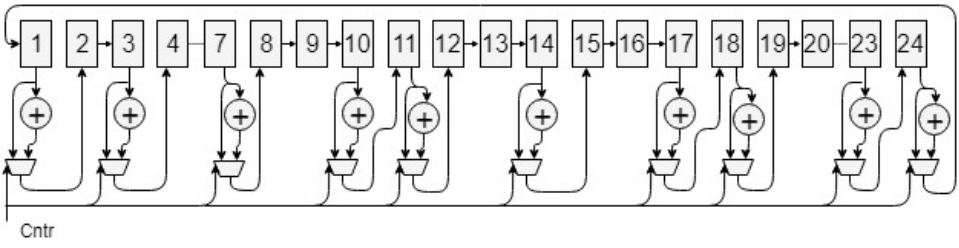

Fig. 2. CRC block architecture.

3.2. *Turbo encoder*

Turbo codes is the coding technique used in many communication and storage systems because its performance is close to Shannon limit. Its function is helping the receiver to correct the received data.

According to Ref. 7, it consists of two Recursive Systematic Convolutional (RSC) encoders with three shift registers XOR-ed according to the following polynomials and one internal interleaver where the input to the encoder is interleaved before entering the 2nd RSC encoder. However, the first one takes the input directly as it is as shown in Fig. 3. The initial value of these shift registers of the eight-state constituent encoders are all zeros.

$$g_0(D) = 1 + D^2 + D^3\,, \tag{2}$$

$$g_1(D) = 1 + D + D^3\,. \tag{3}$$

The relationship between the input $C_0, C_1, C_2, \ldots, C_{k-1}$ and output bits C'_0, C'_1, $C'_2, \ldots C'_{k-1}$ of the internal interleaver is as follows:

$$C'_i = C_{\pi(i)}, \quad i = 0, 1, 2, \ldots (k-1)\,. \tag{4}$$

The relationship between the output index i (ordered bits index) and the input index $\pi(i)$- interleaved index satisfies the following quadratic form:

$$\pi(i) = (f_1 * i + f_2 * i^2) \bmod k\,. \tag{5}$$

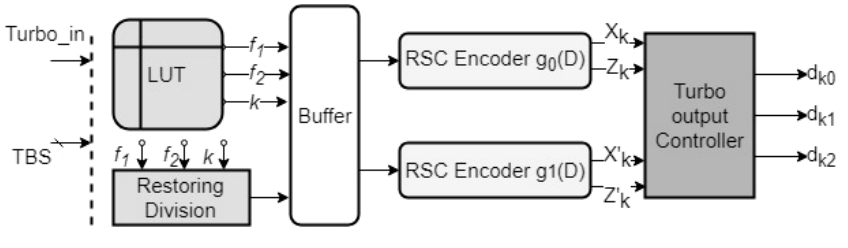

Fig. 3. Turbo encoder block architecture.

The parameters f_1 and f_2 depend on the block size k. Their values are tabulated in Ref. 7. The output from the turbo encoder is as given in (6) appending to them the trellis termination bits which are illustrated in (7).

$$d_k^{(0)} = x_k\,, \tag{6a}$$

$$d_k^{(1)} = z_k\,, \tag{6b}$$

$$d_k^{(2)} = z'_k\,, \tag{6c}$$

$$d_k^{(0)} = x_k, \quad d_{k+1}^{(0)} = z_{k+1}, \quad d_{k+2}^{(0)} = x'_k, \quad d_{k+3}^{(0)} = z'_{k+1}\,, \tag{7a}$$

$$d_k^{(1)} = z_k, \quad d_{k+1}^{(1)} = x_{k+2}, \quad d_{k+2}^{(1)} = z'_k, \quad d_{k+3}^{(1)} = x'_{k+2}\,, \tag{7b}$$

$$d_k^{(2)} = x_{k+1}, \quad d_{k+1}^{(2)} = z_{k+2}, \quad d_{k+2}^{(2)} = x'_{k+1}, \quad d_{k+3}^{(2)} = z'_{k+2}\,, \tag{7c}$$

The proposed design is as shown in Fig. 3 consists of a Lookup Table (LUT) to store the values of f_1 and f_1, a buffer for storing input stream, restoring division unit for index calculation, in addition to the two main blocks. The first block is responsible for shifting and XOR operation for the input stream. It consists of shift registers and multiplexers to generate the trellis termination bits. The second block is the output controller. It is used to control the positions of the 12 trellis terminations bits in the proper locations in the three streams. It contains six multiplexers to help this operation to be done according to (7).[8]

It is considered that the input transport block coming from the upper layer enters the CRC and the encoder block at the same time to save clock cycles. Then the turbo is disabled, waiting for the last 24 CRC bits to be ready to continue operating and start sending the bits to the two convolutional encoders.

Another implementation is proposed in Ref. 9 which provides low power and area solution but in cost of larger latency and complexity as it uses successive subtraction instead of restoring algorithm.

3.3. *Rate matching*

The basic function of this module is matching the number of bits in transport block to the number of bits that is transmitted through the channel. It also controls the rate as the turbo encoder gives 1/3 rate.

According to Ref. 7, it consists of three sub-block interleavers followed by bit collection block then bit selection. The sub-block interleavers are 2D array of 32 columns. The number of rows depends on the input stream's length whose maximum number of bits are 2564 bits. Each input sequence is written row by row to the 2D array. If the array is not filled completely with data bits, the dummy bits are used at the beginning of the array to make it completely filled for proper interleaving process. After that inter-column permutation is performed, data is read column by column. For the first two streams, interleaving is performed according to the permutation

Table 1. Inter-column permutation pattern for 1st and 2nd sub-block interleavers.

Number of columns C^{TC}_{subblock}	Inter-column permutation pattern $\langle P(0), P(1), \ldots, P(C^{TC}_{\text{subblock}} - 1)\rangle$
32	⟨0, 16, 8, 24, 4, 20, 12, 28, 2, 18, 10, 26, 6, 22, 14, 30, 1, 17, 9, 25, 5, 21, 13, 29, 3, 19, 11, 27, 7, 23, 15, 31⟩

Table 2. Inter-column permutation pattern for 3rd sub-block interleaver.

Number of columns C^{TC}_{subblock}	Inter-column permutation pattern $\langle P(0), P(1), \ldots, P(C^{TC}_{\text{subblock}} - 1)\rangle \bmod 32$
32	⟨1, 17, 9, 25, 5, 21, 13, 29, 3, 19, 11, 27, 7, 23, 15, 31, 2, 18, 10, 26, 6, 22, 14, 30, 4, 20, 12, 28, 8, 24, 16, 0⟩

shown in Table 1. While the last stream interleaving is done according to an equation in Ref. 7. This implies adding one to each column index of the same table used for the first two streams as shown in Table 2. The bit collection collects the three interleaved streams then bit selection starts reading the required output sequence from a certain starting point according to (8a) which is simplified for NB-IoT to be as in (8b).

$$k_o = R^{TC}_{\text{subblock}} \left(2 \left\lceil \frac{N_{cb}}{8R^{TC}_{\text{subblock}}} \right\rceil rv_{idx} + 2 \right), \tag{8a}$$

$$k_o = R^{TC}_{\text{subblock}} (24 rv_{idx} + 2). \tag{8b}$$

The operation of the designed block starts by writing input stream which is serially received, so it is converted to parallel to be stored in the 2D array row by row during write state. That is why a 32-bit serial to parallel converter is used. Switching to the read state, which depends on (8b) that determines two fixed starting columns according to redundancy version value (rv_{idx} equals 0 *or* 2). If rv_{idx} equals to 0, reading starts from the 9th column in the first sub-block interleaver while if rv_{idx} equals to 2, reading starts from the 19th column in the second sub-block interleaver. From this point, it is obvious that no bit collection block is needed as data bits is directly read from the sub-block interleavers. In addition, reading directly from the sub-block interleavers makes locating dummy bits easier, as they are located in the first row of each sub-block interleaver so it is easily avoided while reading.[10–12]

3.4. *Data multiplexing and channel interleaver*

The main function of this block in narrowband, taking into consideration that there is no control information, is overcoming burst errors. As for the channels with deep fading characteristics, errors often occur in bursts, these consecutive bits may be of the same symbol. This block overcomes this type of noise by changing this burst error into a group of discrete errors by reordering the bits so that the error happens in one bit only inside the same symbol.

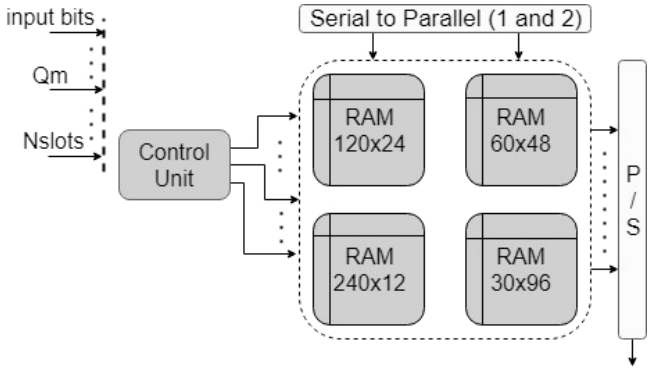

Fig. 4. Data multiplexer and channel interleaver block architecture.

This block is divided into two sub-blocks, the first is the data multiplexing and then the channel interleaver, which are performed in the same memory to reduce power, area and number of cycles as shown in Fig. 4.

The data multiplexing depends on the modulation scheme Q_m which is either BPSK or QPSK. In BPSK case, the input stream is written row by row simply in the memory. However, in QPSK case, first bit is written in the first row and the second bit in the second row and so on till they are filled then start filling the third and the fourth rows in the memory. The uncompleted rows are filled by zeros.

The number of used columns in each case $C_{\max}$ is calculated by (9) to be the number of columns of the matrix.

$$C_{\max} = (N_{\text{symb}}^{UL} - 1) * N_{\text{slot}}^{UL} . \tag{9}$$

N_{symb}^{UL} is the number of SC-FDMA symbols for NPUSCH in an uplink (UL) resource unit. N_{slot}^{UL} is the number of subcarriers in the frequency domain for NB-IoT and N_{sc}^{RU} is the number of consecutive subcarriers in an UL resource unit for NB-IoT.

The proposed design consists of four memories for the stream to be stored row by row and read column by column, 2 serial to parallel blocks for writing, 1 parallel to serial block for reading, multiplexers to select the proper memory and gate the others. Gating the memories depends on an upper layer signal N_{slot}^{UL} which is the number of slots and a control unit to manage the reading and writing operation as shown in Fig. 4. Both serial to parallel blocks are used in QPSK case and only one serial to parallel is used in BPSK case.

3.5. *Scrambler*

A scrambler is an algorithm to randomize the interference randomization between receiving cells. Each receiver is characterized by a number $N_{ID}^{N\text{cell}}$, which is used to

generate a unified scrambling code to scramble the transmitted data and this data is only descrambled by the receiver carrying this number.

Scrambler mainly consists of two Linear Feedback Shift Registers (LFSR) which simply generates a $L = 31$-golden sequence $C(n)$ by 2 paths of flip flops. These 2 LFSR are initialized by two different values. The first LFSR is initialized as in (10a), while the second LFSR is initialized by the binary representation of C_{init} calculated by (10b).

$$x_1(0) = 1, \quad x_1(n) = 0, \quad n = 1, 2, \ldots, 30\,, \tag{10a}$$

$$C_{\text{init}} = n_{RNTI} 2^{14} + n_f \bmod 2.2^{13} + \lfloor n_s/2 \rfloor 2^9 + N_{ID}^{\text{Ncell}}\,. \tag{10b}$$

1600-shift cycles should be performed directly after initialization and before enabling scrambler, to induce extra randomness, then XOR operation is performed between input and the two golden sequences.

3.6. *Modulation mapper*

Modulation mapper is used to map the incoming data stream to the corresponding symbol in the constellation of either BPSK or QPSK. These symbols are defined in Ref. 5.

As stated, this corresponding symbols are located in a LUT in the design. For a BPSK case, the modulation mapper maps the input bit to the corresponding symbol in each clock cycle. While for the QPSK case, each two bits is mapped to the corresponding symbol so it takes two clock cycles to generate one symbol to the next block.

3.7. *Modulation buffer*

As discussed in the modulation mapper, the output of the modulator is produced every clock cycle as in BPSK or every two cycles as in QPSK. Therefore, a buffer is required after the modulation mapper and before the FFT to store the results of the modulator and provide data to the FFT every clock cycle in BPSK and QPSK as well.

The buffer operates as follows: it stores the data from the modulator till it is full, and when it is full it is disabled with all of the previous blocks. When the FFT is ready to receive new input, the buffer is enabled and FFT reads the data from the buffer till its empty again then all of the disabled blocks re-operate till the buffer is full and so on.

3.8. *FFT*

FFT is a block added to generate SC-FDMA symbols instead of Orthogonal Frequency Division Multiplexing (OFDM) symbols. As the latter have high Peak to Average Power Ratio (PAPR) and require highly linear power amplifiers to avoid

excessive intermodulation distortion. However, SC-FDMA solves this problem by distributing given number of modulated symbols over all assigned subcarriers for transmission instead of distributing one symbol over one subcarrier as in OFDM; therefore, the total PAPR decreases due to this fair distribution.

The number of allocated subcarriers in NB-IoT with NPUSCH format 1 and subcarrier spacing of 15 kHz, varies between 1, 3, 6 and 12 subcarriers,[5] which arises the need of implementing an FFT block that supports different numbers of inputs. Moreover, none of the number of subcarriers is power of 2, so the simple radix 2 algorithm does not suit this case. That is why a mixed radix FFT algorithm is proposed and implemented where both radix 2 and radix 3 algorithms are used.[13]

This algorithm is used to implement multi-point FFT block that is either 3-FFT, 6-FFT or 12-FFT. To implement low-power FFT design, the memory-based hardware architecture is used.[14–19] In the proposed FFT processor, only four single-port memories are needed for buffering the computational data. Therefore, the area and test costs are much lower than those of the existing memory-based FFT processors which use two-port memories.[20]

The Signal Flow Graph (SFG) for the 3-FFT, 6-FFT and 12-FFT are built taking into consideration no destructive overwrite takes place in the memories.

Memory-based FFT is the most suitable choice for low-power implementation which is the main goal in case of IoT systems.[21] Memory-based architecture usually performs the FFT in serial, i.e., one butterfly operation at a time instead of more than one in parallel, and this results in a low area cost for implementing the memory-based FFT processor.

As shown in Fig. 5, data path of this design consists of four single port RAMs, five multiplexers, butterfly unit, one ROM and one controller. The RAMs are used for

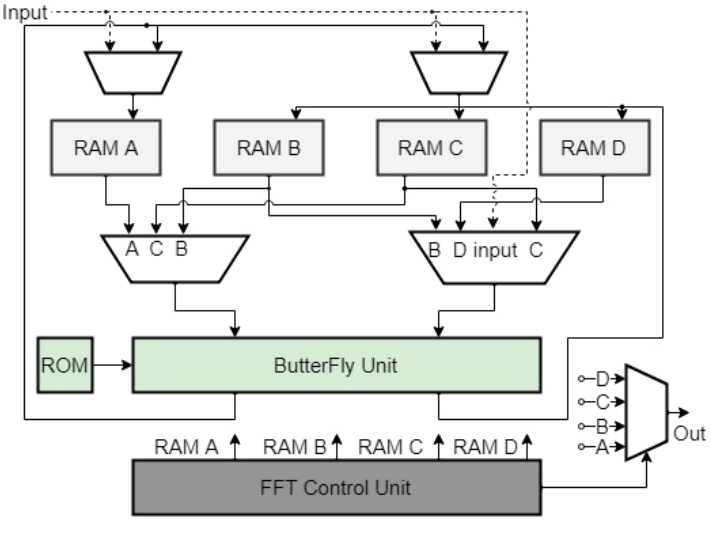

Fig. 5. Implemented memory based FFT architecture.

storing the intermediate results of the 3, 6, 12-point FFT. The multiplexers are responsible for switching the data flow between the storage and arithmetic components. The butterfly unit executes the computation of the two-point FFT. ROM stores the twiddle factors. The butterfly addition and multiplication operations are performed as in Ref. 20. The control unit generates the controlling signals for the multiplexers and the RAMs.

3.9. *Resource element mapper*

The function of the REM is assigning the output symbols from the FFT to the proper subcarriers at the input of the IFFT block and handle the repetition of the symbols to enhance coverage. In 12-point FFT case, Fig. 6 shows that all the symbols are assigned to the 12 subcarriers allocated for the NB-IoT bandwidth and this is provided as an upper layer information. In 6-point FFT case, Fig. 7 the output 6 symbols of the FFT is assigned to a certain 6 subcarriers of the 12 subcarriers and the rest are padded with zeros and the same is done with 3-point FFT. The position of the symbols within the 12 subcarriers in case of 3 and 6 subcarriers is calculated from a table in Ref. 5.

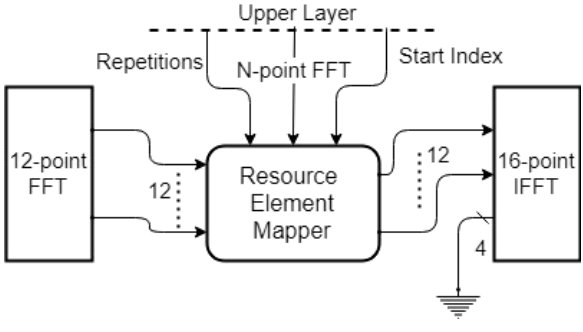

Fig. 6. Resource allocation in 12-point FFT.

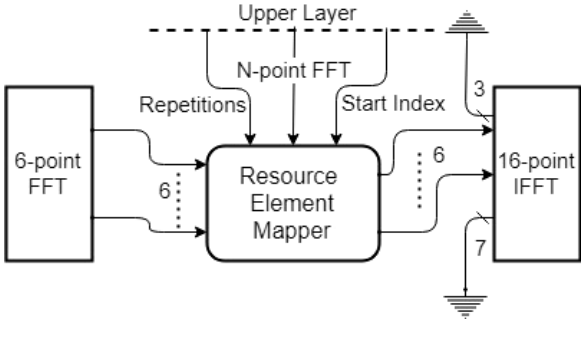

Fig. 7. Resource allocation in six-point FFT.

3.10. *IFFT*

IFFT is used to transform the symbols from frequency domain to time domain samples to be transmitted through the Radio Frequency (RF) chain. It is required to satisfy the output rate mentioned in Ref. 5. Not only produce this symbol in the SC-FDMA symbol duration but also with a constant rate without any bubbles.

The IFFT subcarriers are grouped into sets of 12 subcarriers with spacing of 15 kHz between each adjacent subchannels, each group is called a resource block. NB-IoT has one resource block only per UE transmitter. 16-point IFFT is used as a minimum size, to implement IFFT block which supports 12 subcarriers as the max number of allocated subcarriers. This IFFT size is used to decrease latency and power dissipation of the block and output sample rate is compensated by an upsample filter to upgrade the rate to a convenient rate for the digital to analog converter (DAC) block.

Memory-based IFFT is the most suitable choice for low-power implementation as illustrated in FFT. Almost the same architecture of the FFT is used based on a 16-point IFFT paper using the mentioned SFG in this paper.[20] Noticing that it is a bit simpler than implementing the mixed radix.

Since REM supports 12 symbols as mapped symbols and last 4 symbols of 16 are always zeros, these zeros are encountered instead of output from REM at cycles from 13 to 16.

The problem in the used design as shown in the 16-point IFFT SFG[20] is that it produces output samples in a burst style (i.e., from cycle 33 to 48) only. The rest 32 cycles have no valid output. To achieve the required constant output rate, a buffer with different read and write rates is used. It ensures that read rate from the buffer satisfies the required constant rate, while writing in it is in a burst style as it is. An external control unit is added to control the read and write process and insert the cyclic prefix properly. Figure 8 shows block diagram which consists of IFFT, control unit and a buffer. Figure 9 describes how reading and writing from and into the buffer occurs versus time cycles.

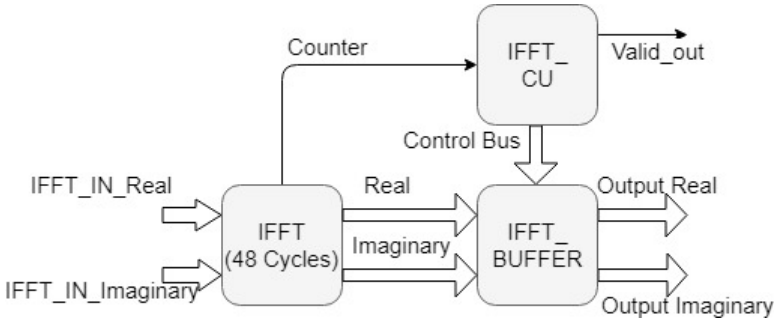

Fig. 8. Constant rate controller, buffer and IFFT.

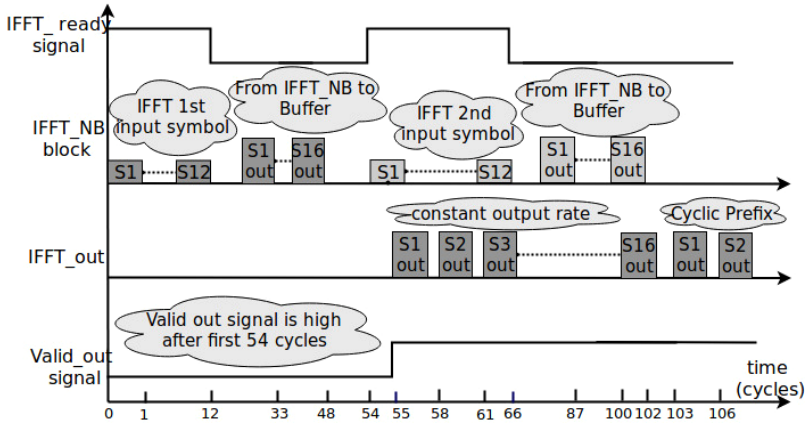

Fig. 9. IFFT timing diagram after controlling the output rate.

As shown in Fig. 9, writing in the buffer from IFFT block occurs at cycles from 33 to 48 in a burst style then these symbols are read from buffer by a constant rate distributed over the next 54 cycles. This is the minimum number of cycles that allows new SC-FDMA to be produced as the 16-pint FFT requires 48 cycles.[20] Furthermore, it is divisible by 18 which is the total number of samples per SC-FDMA symbol duration to make the block produce output samples with a constant rate during these 54 cycles. The control unit handles all the invalid data overwriting operation by sending the addresses and the read/write enable signals based on the counter cycle value.

4. Full Transmitter Chain Simulation and Synthesis Results

Validation of the full chain is carried out by building a full MATLAB model of the integrated chain. Comparing the output of different test cases, generated from this model, RTL implementation is carried out using automated test bench. Taking into consideration in this comparison the quantization noise resulting due to truncation and fixed point representation, the number of bits of this representation is chosen to be 10 fraction bits after performing Signal to Quantization Noise Ratio (SQNR) simulation.

Moreover, the output rate is taken into consideration in the test bench where the RTL design gives a new output sample and each 3 clock cycle with a total number of samples per SC-FDMA symbol duration is calculated by (11).

$$\text{Total number of samples} = (16(\text{IFFT size}) + 2(\text{CP}))\,. \tag{11}$$

System throughput has been calculated by dividing Total number of samples (11) by SC-FDMA symbol duration defined in Ref. 5, and is equal to $\sim$ 253 Ksample/s.

Synthesis of this implemented design is done using Synopsys tools and UMC 130-nm technology at a clock frequency of 756 kHz to estimate the utilized area and

power consumption of this implementation. The Switching Activity Interchange Format (SAIF) files are being added to get more realistic values for the dynamic power results. The clock frequency was calculated by considering the SC-FDMA symbol duration given in Ref. 5 and the number of IFFT clock cycles which represents the total number of cycles per SC-FDMA symbol duration and are equal to 54 cycles, as it is the last block in the chain. These 54 cycles result from memory-based IFFT architecture which has high latency but has the least power consumption that meets the standard specification unlike LTE which requires higher data rate. Although this clock frequency is low, it satisfies the design requirements as it deals with IoT devices with low data rates.

Area and power results are shown in Figs. 10 and 11, respectively. CRC and scrambler have the lowest utilization area because they mainly consist of LFSRs. Modulator also has a very low utilization area as it contains mainly one LUT. One significant result as well is for FFT and IFFT, which shows relatively low power and utilization area after using memory-based architecture which is much more optimized from power and area point of view than the pipelined architecture. On the other hand, channel interleaver, REM and encoder has the largest utilization area because they have large storage elements. For the encoder, a 1D buffer is used and it is powered one third of the total time of the encoder activity. The channel interleaver

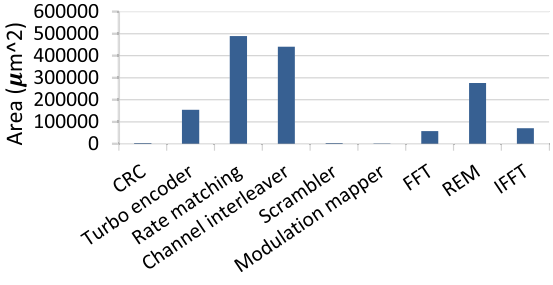

Fig. 10. Area results of all blocks in the chain.

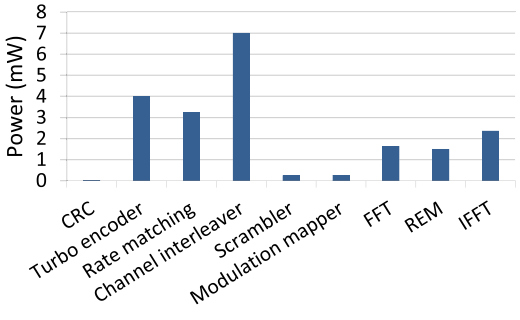

Fig. 11. Power results of all blocks in the chain.

has four memories which are gated but they are still large ones. The same applies for the REM block.

Although the rate matching block has larger area than channel interleaver, the latter has larger significant power. This stems from the fact that channel interleaver is enabled and disabled many times according to the modulation buffer state. As mentioned before, when the buffer is full it disables all the previous blocks till the interleaver. Then it enables them again when it is ready to receive new input symbols so the interleaver is disabled and enabled many times while processing every transport block.

The justification for the same conflict in power and area between the rate matching and encoder blocks is that the latter operates more cycles than the rate matching. This is because of the encoder division unit which takes 33 clock cycles to calculate the new index of each output bit, so the total number of cycles taken by the encoder is given by (12a). However, number of operating cycles for the rate matching is calculated by (12b).

$$\text{number of encoder operating cycles} = K + 33 * K + 4\,, \tag{12a}$$

$$\text{number of rate matching operating cycles} = K + 4 + E\,. \tag{12b}$$

K is the length of the input bits length to the encoder which reaches 2560 bits depending on the transport block size. E is the length of the output bits from the rate matching whose maximum is 2880 bits.

5. Conclusion

In this paper, the NB-IoT physical layer digital transmitter is designed using new creative solutions and designs for each block to fulfill the performance requirements of IoT such as significant coverage extension using repetitions, low device complexity, long battery lifetime and supporting a massive number of IoT devices through using smaller bandwidth for each user 180 kHz. This digital transmitter is implemented to support both QPSK and BPSK modulation schemes with 15 kHz subcarrier spacing. Synthesis results of power and area are tabulated using UMC 130-nm technology and clock frequency of 756 kHz. They are justified as well showing a significant low power and area for some blocks.

6. Future Work

Some of the future work that could be carried out as an extension to this paper include:

(1) Supporting NB-IoT second mode that has subcarrier spacing 3.75 kHz as addressed in Ref. 5.

(2) Implementing Physical Random Access Channel (PRACH) chain which generates the random access preamble and transport access requests when the device wants to communicate with the base station as in call origination or paging response.
(3) Physical implementation for the synthesized chain.
(4) Relating the operating clock frequency to market conventional crystal oscillator frequencies.

Acknowledgement

This work was partially funded by ONE Lab at Zewail City of Science and Technology and at Cairo University, and Si-Vision LLC, Egypt.

References

1. J. Bradley, J. Barbier and D. Handler, Embracing the internet of everything to capture your share of $14.4 trillion, White Paper, Cisco (2013).
2. R. Ratasuk, N. Mangalvedhe and A. Ghosh, Overview of lte enhancements for cellular iot, *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015 IEEE 26th Annual International Symp. IEEE* (2015), pp. 2293–2297.
3. 3GPP Technical Specifications 36.201, General description.
4. H. Malik, H. Pervaiz, M. M. Alam, Y. Le Moullec, A. Kuusik and M. A. Imran, Radio resource management scheme in NB-IoT systems, *IEEE Access* **6** (2018) 15051–15064.
5. 3GPP Technical Specifications 36.211, Physical channel and modulation.
6. 3GPP Technical Specifications 36.213, Physical channel and modulation.
7. 3GPP Technical Specifications 36.212, Physical channel and modulation.
8. A. Abdelbaky and H. Mostafa, New low area NB-IoT turbo encoder interleaver by sharing resources, *2017 29th Int. Conf. Microelectronics (ICM)*, Beirut (2017), pp. 1–4.
9. A. Abdelbaky and H. Mostafa, New low area NB-IoT turbo encoder interleaver by sharing resources, *2017 29th International Conference on Microelectronics (ICM)* 10 December 2017, pp. 1–4.
10. J. F. Cheng, A. Nimbalker, Y. Blankenship, B. Classon and T. K. Blankenship, Analysis of circular buffer rate matching for LTE turbo code, *2008 IEEE 68th Vehicular Technology Conf.*, Calgary, BC (2008), pp. 1–5.
11. K. G. Lenzi, J. A. Bianco, F. A. de Figueiredo and F. L. Figueiredo, Optimized rate matching architecture for a LTE-Advanced FPGA-based PHY, *2013 IEEE Int. Conf. Circuits and Systems (ICCAS)*, Kuala Lumpur (2013), pp. 102–107.
12. J. C. Ikuno, S. Schwarz and Michal Simko, LTE rate matching performance with code block balancing, *17th European Wireless 2011 — Sustainable Wireless Technologies*, Vienna, Austria (2011), pp. 1–3.
13. J. Löfgren and P. Nilsson, On hardware implementation of radix 3 and radix 5 FFT kernels for LTE systems, *2011 NORCHIP*, Lund (2011), pp. 1–4.
14. C.-H. Chang, C.-L. Wang and Y.-T. Chang, A novel memory-based FFT processor for DMT/OFDM applications, *IEEE Int. Conference on Acoustics, Speech, and Signal Processing* (1999), pp. 1921–1924.
15. C.-L. Wang and C.-H. Chang, A DHT-based FFT-IFFT processor for VDSL tranceivers, *IEEE Int. Conference on Acoustics, Speech, and Signal Processing* (2001), pp. 1213–1216.

16. C.-K. Chang, C.-P. Huang and S.-G. Chen, An efficient memory-based FFT architecture, *Proc. IEEE Int'l Symp. on Circuits and Systems (ISCAS)* (2003), pp. II–129–II–132.
17. S.-C. Moon and I.-C. Park, Area-efficient memory-based architecture for FFT processing, *Proc. IEEE Int'l Symp. on Circuits and Systems (ISCAS)* (2003), pp. 129–132.
18. S.-Y. Lee, C.-C. Chen, C.-C. Lee and C.-J. Cheng, A low-power VLSI architecture for a shared-memory FFT processor with a mixed-radix algorithm and a simple memory control scheme, *Proc. IEEE Int'l Symp. on Circuits and Systems (ISCAS)* (2006), pp. 157–160.
19. C.-H. Su and J.-M. Wu, Reconfigurable FFT design for low power OFDM communication systems, *IEEE Int'l Symp. on Consumer Electronics* (2006), pp. 1–4.
20. Y.-X. Yang, J.-F. Li, H.-N. Liu and C.-L. Wey, Design of cost-efficient memory-based FFT processors using single-port memories, *2007 IEEE International SOC Conf.*, Hsin Chu, Taiwan (2007), pp. 29–32.
21. W. L. Tsai, S. G. Chen and S. J. Huang, Reconfigurable radix-2 k×3 feedforward FFT architectures, *2019 IEEE International Symp. Circuits and Systems (ISCAS)*, IEEE, 26 May 2019, pp. 1–5.