OpenAirInterface: A Pipeline Structure for 5G

Wang Tsu Han
Eurecom
Biot, France
Email: wangts@eurecom.fr

Raymond Knopp
Eurecom
Biot, France
Email: raymond.knopp@eurecom.fr

***Abstract*—Compare to 4G Long-Term Evolution (LTE), 5G New Radio (NR) has higher timing demand with shorter Transmission Time Interval (TTI). In order to develop OpenAirInterface (OAI) for 5G, adjustments for acceleration are needed. The most timing demanding part of the structure is Physical Layer (PHY). In this paper, the functional acceleration improvement and structural parallelism are used to improve the timing while adding latency, to trade off for more execution time.**

I. Introduction

Since 5G NR is designed to handle multiple transmission scenarios [1], the base station for NR has high complexity. This makes dedicated hardware development difficult and time-consuming. Instead of using hardware to build a base station, software-defined base station provides a higher ability for adjustment and development. This paper is using the OAI [2][3] to construct the software-defined base station. And the focus of this paper is on the critical path of PHY for TTI. The processing of the base station is timing demanding. The real-time nature of the tasks cause low tolerance for timing delay. The timing issue is even more critical in 5G due to the shorter TTI [4] caused by various numerologies.

II. System Description

The structure of wireless communication of 3GPP, specifically LTE, connection is as shown in Fig. 1. Having a User Equipment (UE) connected to base station over the air then accessing internet through the core network. The presented work is based on base station constructed with OAI. For all the procedures happening in base station, PHY is going to be the focus.

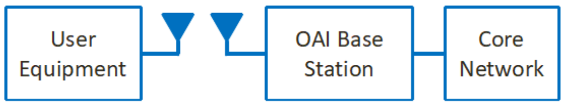

Fig. 1: Overall structure of wireless communication

Dividing the PHY procedure of the OAI into smaller components, there are mainly four parts, Front End Process (FEP) for the Receiver (RX) which is from precoding to SC-FDMA signal in standard, procedures for the Uplink (UL) channel, procedures for the Downlink (DL), and FEP for the Transmitter (TX) which is from precoding to OFDM signal in standard. RX process and UL channel operation work on subframe n and DL channel operation and TX process are for subframe n+4 as shown in Fig. 2. In OAI, all processes are consecutive as a sequence. This makes the next incoming data in subframe n+1 for RX only processable after the previous sequence finishes. Which means that the subframe n+4 for TX has to be written back to the Radio Unit (RU) before next data arrives.

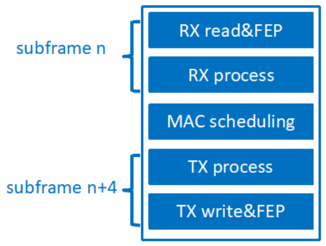

Fig. 2: Physical layer procedure

III. Problem Statement

According to the 5G standard [4], the subframe is still 1 ms, but the number of slots in one subframe can vary with different numerology. Which means that the execution time for the same amount of data is equal or less than that in 4G LTE. There should be an adjustment in order to get the higher throughput that 5G NR needs.

There are mainly two aspects that should be taken into consideration to be able to meet the requirement of 5G. One is to accelerate the function itself and the other is to do structural parallelism. For the first part, execution acceleration is going to be the focus. How long is the time occupied by the function within the whole procedure and how it could be accelerated. For the second part, parallelism is going to be the issue. Throughout the procedure, do all processes have to be consecutive and can the processes be overlapped.

For the functional acceleration, the first task is to determine which process occupies more time than the others, by using timing measurements over each function. Then finding execution dependency within the function. Timing measurements show that most of the effort is spent on the data transport channels DownLink Share Channel (DLSCH) and UpLink Share Channel (ULSCH). Taking DLSCH as an example, before having any optimization, turbo-encoding occupied over half of the execution time and the encoding execution is looping over segmentations which makes turbo-encoding a candidate for acceleration. The same thing also applies to turbo-decoding. FEP is looping over the slots independently and executed right after getting data from RU before sending data to each channel.

As for the structural parallelism, as mentioned before the overall structure is a consecutive process which each of the following steps requires information provided by former step. This makes the overall structure hard to be parallel within one sequence, from subframe n for RX to subframe n+4 for TX, though there is no dependency between sequences. According to 4G LTE [5], the response time will be 4 TTI which here is the same time as a subframe. In the original structure, even though data only need to be ready at subframe n+4 it still has to finish before the next subframe comes due to consecutive execution. This feature gives a hint to design a system that could tolerate higher latency but still maintain the throughput required.

IV. Proposed Structure

As mentioned before, turbo encoding and turbo decoding are one of the heaviest parts of execution and they have the feature of repetition and independence over segmentation which makes them candidates for function acceleration. Since the number of segments within one data is depending on the amount of data within one subframe for a UE, the number of segmentation need to be calculated before spreading into worker threads. As shown in Fig. 3, after the segmentation master thread generates a signal to worker threads to execute corresponded segments simultaneously. Master thread will wait until all of the workers is finished then it ends the function. The maximum number of worker threads is three but the number of worker threads thats in use depends on the number of segments since segments are independent between each other. It works the same way for decoding and also for the FEP of both TX and RX process. For the FEP process, instead of having the parallelism over segments, it has parallelism over slots.

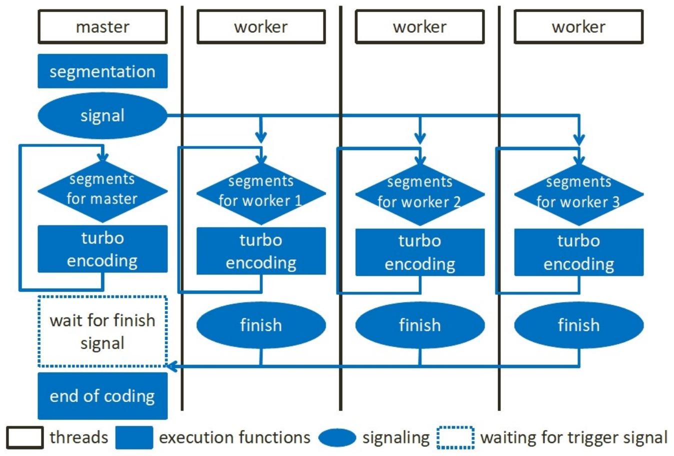

Fig. 3: Coding accelaration with worker threads

The information decoded from subframe n RX is sent to Medium Access Control (MAC) and then being used to schedule for subframe n+4 TX. Since the information decoded from RX subframe n is needed for TX subframe n+4 to be scheduled, it is hard to do much parallelizing within one subframe. Instead of having total independence between each sequence creates an opportunity to make an improvement on the current structure which has to finish the whole sequence before the next one comes.

A pipeline structure provides longer execution timing tolerance for one sequence. By theory, the latency for one sequence become the original timing times number of stages but more stage splits for the structure more latency it will cost for having handshaking between stages. Because of the limitation for response time the proposed structure is set to have four stage shown in Fig. 4. The proposed structure separates the original structure into RX Front End, RX process, TX process, and TX Front End. These four stages correspond with a thread. Processes about Front End including the read-write function for RU are in the RX and TX Front End thread, Uplink channel processes and MAC scheduling are in the RX process thread, as for Downlink channel processes are in the TX process thread. It gains more time for a sequence to finish processing with the proposed structure shown in Fig. 4. Data passing between threads and threading synchronization are an important issue with pipeline structure like this.

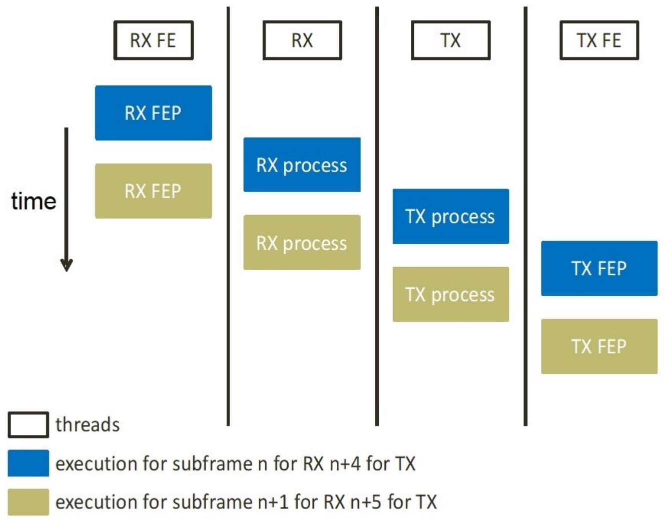

Fig. 4: Physical layer with pipeline structure

Data passing through stages is time and resource consuming so using a global buffer with a structure includes the index for the butter is more efficient. The proposed structure uses frame and subframe number to calculate which part of the buffer this stage should be using. Data is passed down from stage to stage by copying with locks to prevent conflict.

The important part for synchronization is having threads to execute in the correct order with the right parameter. In order to maintain the execution ordering and timing correctness within one sequence, one stage can wake up by its former stage in the proposed structure with threading lock, signal, and wait. Locking the thread with POSIX Thread (PThread) function mutex lock gives the privilege to the thread that no other threads could be using parameters that are being locked. PThread condition wait can put the thread into sleep mode and can be woken up by other thread. A condition variable is used to prevent any unwanted wakeup.

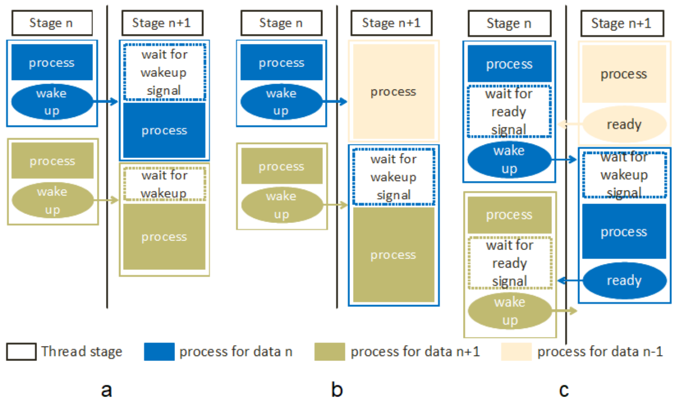

Fig. 5: Threading synchronization issue

Fig. 5a is what the process is expected to do, current stage waking up next stage and passing timestemp frame and subframe parameter to it then start the process for the next incoming data. While only with one side signaling, the thread might still miss some wakeup due to the wakeup timing and wait timing mismatch as shown in Fig. 5b. The error happens when the process for less data n-1 is not yet finished and not yet into the waiting mode when execution for subframe n is heavy loaded but not subframe n+1. This makes two wake-up signals coming too close to each other which cause stage n+1 missing one of the wake-up signals and getting the wake up for data n+1 which process for data n is missing after finishing the n-1 process. In order to solve this, we create a feedback system which will send a ready signal back to the former stage to indicate that this stage is ready to process for the next incoming data as shown in Fig. 5c. This prevents the former stage to send wake up signal before the current stage is ready.

V. Performance

The test is done on three different machines shown as Table I represent different scenarios. Carabe represents the general usage machine, Mozart represents the powerful machine, and Caracal represents the powerful server machine.

TABLE I: Machine used for testing

Machine	Core Spec	CPUs	OS	Kernel
carabe	Core i7-5775R @3.3GHz	4	Ubuntu	3.19 Lowlatency
mozart	Xeon E5-2687 v3 @3.1GHz	10	Ubuntu	3.19 Lowlatency
caracal	Xeon Gold 6154 @3GHz	36	CentOs	3.10 RealTime

All tests for the functional timing are done under the configuration for 20MHz bandwidth in LTE. The measurement of execution time is with having a timestamp at the beginning and the end of the function. The deduction of two timestamps represents the time used for the function. FEP process is the same in LTE when the bandwidth is fixed. Figs. 6 and 7 verified that there is not much variance throughout different Modulation and Coding Scheme (MCS) for both TX and RX FEP and the functional acceleration reduce about half of the time since they parallel through two slots. For the turbo encoding, the original structure has timing increasing along with the MCS. The proposed structure has the same execution time with the original structure in the lower MCS. Since there are too few segments there is no worker thread being wake up. As the MCS increase, worker threads are woken up one by one that causes the sudden decrease on the timing for the proposed structure in Fig. 8. The critical path in the TX process is coding and FEP in RU process. The acceleration for those function makes this structure possible to execute under the constraint with numerology 1 of 5G in every stage.

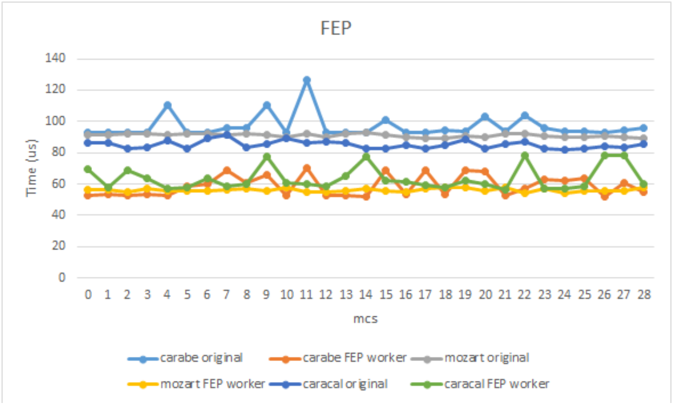

Fig. 6: Timing for FEP process

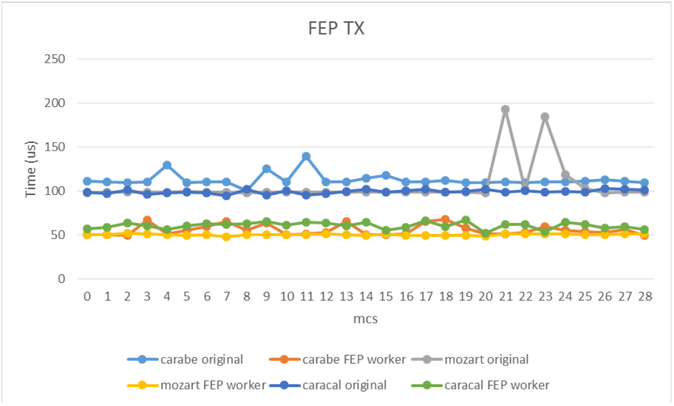

Fig. 7: Timing for FEP process

Fig. 9 shows the timing relationship of all process under pipeline structure with fully loaded resource block and highest MCS for 100 RBs configuration on Mozart. In this case, the gap between finishing time of the last function for subframe n and the beginning time of the first function for subframe n+1 in numerology 0 is narrow. If using the original structure, this makes the program unable to handle any more process or to support more algorithm. While using the pipeline structure,

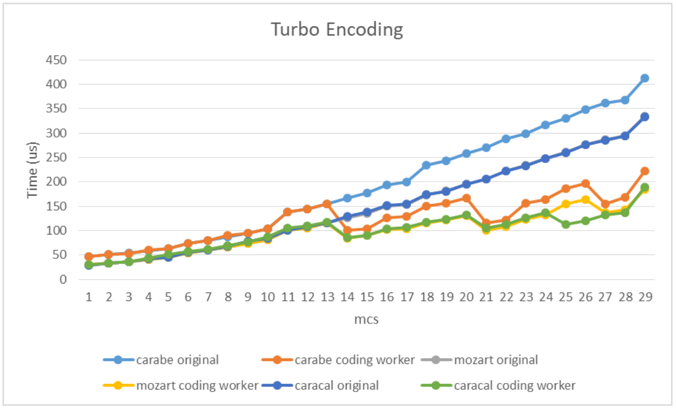

Fig. 8: Timing for encoding process

the program is able to handle sorter time slot by overlapping the process time like shown in Fig. 9 numerolog1. While the TX process for subframe n+4 is still active, RX process for subframe n+1 has already started. The wake-up mechanism cost less than 10us in the proposed structure. And the pipeline structure provides more flexibility to support more complex structure with multi RUs and multi eNBs.

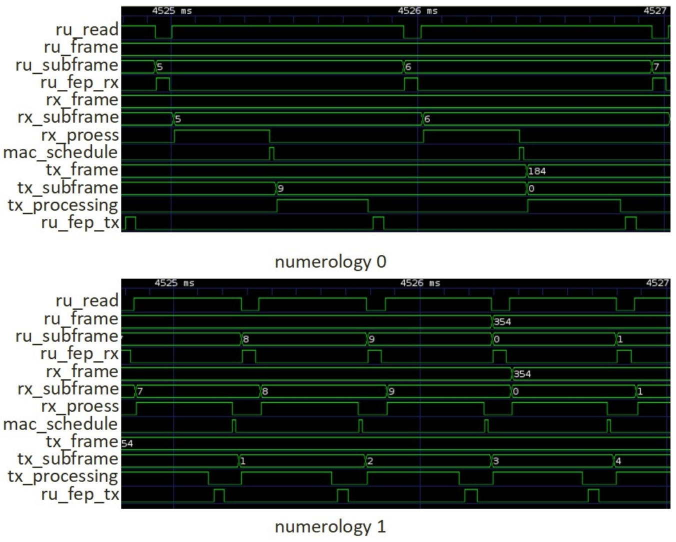

Fig. 9: Overall timing and relationship between stages

All tests for the pipeline structure are done with fully allocated Resource Block (RB) of 20MHz bandwidth configuration in LTE adding numerology concept for NR. The measurement for the capacity of the structure is done with having a thread emulate the Radio Frequency (RF) device to wake up every TTI and also check if the lest process finished before it starts the next round. Table II shows the result of the maximum MCS that could be handled by each structure with helps of functional acceleration. Improvement on the less powerful machine is more than that on the server machine. The total execution time on the general machine is more than 850 us and the server machine is less than 650 us. Timing on the general machine is much tighter than that on the server machine so the improvement on the general machine is more severe.

TABLE II: Maximum capability

Machine	Origianl (mcs UL/DL)		Pipeline (mcs UL/DL)	
	Numerology 0	Numerology 1	Numerology 0	Numerology 1
carabe	0/0	X	28/28	28/28
mozart	28/28	X	28/28	28/28
caracal	28/28	X	28/28	28/28

VI. Conclusion

The proposed structure spreads the workload to different threads which gives the opportunity for the Operating System (OS) to arrange better while having one core to execute all the process for base station makes it vulnerable of any impulse. The proposed structure introduces latency as a tradeoff for longer execution time while still maintain the real-time property. It is less affected by the task created by the OS and other background process and having more flexibility for the proposed structure. In able to meet the requirements of 5G latency, both data parallelism and task parallelisation are used to meet fully loaded 40MHz bandwith with mcs28.

Acknowledgment

This work was supported by colleague from EURECOM

References

[1] ”Study on Scenarios and Requirements for Next Generation Access Technologies”, 3GPP Technical Report 38.913.
[2] Nikaein, Navid, et al. ”OpenAirInterface: A flexible platform for 5G research.” ACM SIGCOMM Computer Communication Review 44.5 (2014): 33-38.
[3] The OpenAirInterface Initiative. http://www.openairinterface.org/
[4] 3GPP TS 38.211: ”NR; Physical channels and modulation”.
[5] 3GPP TS 36.213: ”Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures”.