FPGA-Based Turbo Decoder Hardware Accelerator in Cloud Radio Access Network (C-RAN)

Shaoxian Tang(✉), Zhifeng Zhang, Jun Wu(✉), and Hui Zhu

College of Electronics and Information Engineering, Tongji University, Shanghai 201804, People's Republic of China
{1433273,zhangzf,wujun,1433250}@tongji.edu.cn

Abstract. In the Cloud Radio Access Network (C-RAN), the Software Defined Radio (SDR) is combined with multi-mode base stations (BSs) together. A lot of BSs are centralized in a Cloud center, the centralized BSs need high bandwidth and cost-effective resource allocation. Since BSs may also run on the virtualized machines, the hardware accelerator can provide faster signal processing speed. This paper uses the Xen virtualization to set up a C-RAN platform, where the SDR and the FPGA hardware connected with PCIe interface to server as the signal processing hardware accelerator. Experimental results demonstrate the turbo decoder accelerator based on the FPGA and Xen platform has good performance to support the SDR signal processing with high bandwidth. The turbo decoder hardware accelerator solved the timing constraints in C-RAN.

Keywords: C-RAN · SDR · Xen · FPGA · Hardware accelerator · Turbo decoder

1 Introduction

Radio Access Network (RAN) is wireless communication infrastructure. However, with the costs of distributed base stations deployment increasing, the mobile operators have to develop a new evolved network architecture. Centralized base station pool with the cloud computing-based architecture supports 2G, 3G, 4G and future wireless communication standards. C-RAN is an eco-friendly and energy efficient infrastructure. The base stations can be reduced with centralized processing of this architecture [5]. The C-RAN emerged to offer a low cost, high reliability, low latency and high bandwidth green network architecture [8].

In order to support multi-network, the operators usually should establish multi-mode base stations (such as GSM, UMTS and LTE), which increase costs. The SDR meets the multi-standards for low cost operation by software reconfiguration. The SDR platform can be implemented when the large scale Baseband Unit (BBU) pool has high-speed and low-latency interconnection. The BBU

Q. Chen et al. (Eds.): ChinaCom 2016, Part I, LNICST 209, pp. 211–220, 2018.
DOI: 10.1007/978-3-319-66625-9_21

with virtual BS pool is shown in Fig. 1. Therefore, the SDR with open platform will become one of the mainstream products.

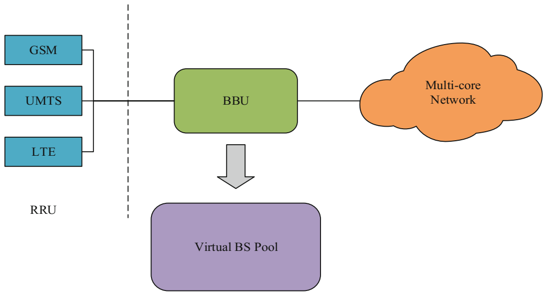

Fig. 1. Virtual base station pool

In the SDR base stations, the baseband process includes some computation intensive tasks. Therefore, using the hardware accelerator instead of software can reduce the processing time of these tasks such as turbo decoder, MIMO decoder and FFT etc. Since the speed of the hardware operation has greatly improved, taking advantage of hardware resources is a good option. Virtualization is the technology to create a virtual version of something, such as computer hardware platforms, operating system, storage devices and network resources. Through the virtualization, we can replace many small physical servers by several larger physical servers to increase the utilization of hardware resources such as CPU and FPGA. Therefore, using the virtual base stations can reduce the cost and fully utilize the hardware resources. Xen is a hypervisor that allows multiple computer operating systems to execute on the same computer hardware concurrently. Xen has para-virtualization, which uses special API instead of simulate hardware to modify the guest OS [1]. So we can use the guest OS to access hardware through the hypervisor with the special API. Using the para-virtualization can minimize the performance loss.

The BS based on the SDR platform has more choices to implement the signal processing, not only uses CPU in the server. The multi-core GPP-SDR platform was proposed in Sora, a programmable software radio platform on PC architectures [6]. However, it only has single terminal to use the platform. Digital Signal Processor (DSP) have high capabilities of floating point, which combined with the GPP has improved the BBU processing density and reduced the power consumption effectively. There are some SDR platforms based on FPGA [4] and DSP [3].

FPGA virtualization, such as [2] brings FPGA as a shareable resources in the cloud, which demonstrate that FPGA can be used in the C-RAN for the signal processing with good performance. pvFPGA [7] also runs in the Xen virtualized environment and uses FPGA to accelerate the process. It uses the accelerator to compute FFT with Xilinx IP core. So using the accelerator for decoding in the SDR base stations can reduce the overall time effectively.

In this paper, we design a turbo decoder hardware accelerator in the SDR base stations based on FPGA and Xen virtualization. The rest of paper is organized as follows. Section 2 is the system design. Section 3 is the detailed design of the turbo decoder in FPGA. Section 4 gives simulation and evaluation of the turbo decoder hardware accelerator. Finally, Sect. 5 concludes the paper.

2 System Design

The system has the structure shown in Fig. 2, which includes the SDR platform based on general purpose computer server, and the custom designed hardware accelerators. Virtualization is utilized to support multiple BSs on one physical server. Figure 2 is a simplified SDR platform with only one single virtual machine. The Guest OS is based on Xen virtualization. Although only one guest OS is showed, the platform supports multiple guest OS running on the host OS, each corresponds to one virtualized BS. The system has hardware accelerators to meet the requirement of high bandwidth baseband signal processing. The accelerator is implemented with FPGA, using Verilog HDL and IP cores provided by Xilinx. PCIe interface is used to support data exchanges between the hardware accelerator and the host OS. The Xen VMM and the PCIe backend driver is located in the host OS, and the front-end driver is in the Guest OS, which is used for transfer the data between the SDR platform and backend driver.

2.1 Virtualized SDR Platform

The general server system runs Xen hypervisor, also called Virtual Machine Monitor (VMM), to build the virtualized SDR platform. The architecture of Xen is shown in Fig. 3. The SDR platform is located in the Guest OS (VM), uses the front-end driver to communicate with Dom0 (a privileged virtual domain). The

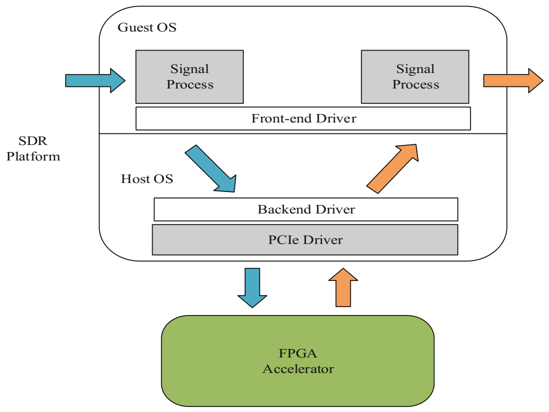

Fig. 2. System design

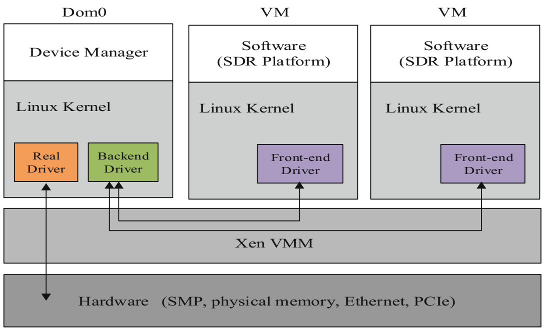

Fig. 3. Xen architecture

data transfer in Xen VMM is made up of three different paths, namely the shared memory, the event channel and grant table. Large data is exchanged through shared memory to achieve high throughput without memory copy, and the message use event channel and grant table to transfer requests and small amount of information between Dom0 and other VMs. Supporting virtually centralized BS, or in other words, supporting multiple BS in one physical server requires a significant higher throughput of encoded and decoded data in compares with the traditional distributed BS structure. To meet such a requirement, PCIe is used as data transfer interface between server and the FPGA-based hardware accelerator. For the virtual platform, we implement the PCIe backend driver in Dom0 and front-end driver in VMs. The SDR platform can use the front-end driver as the real PCIe driver to write encoded data and read decoded data after computing process.

2.2 Cloud Base Stations

The centralized BBUs need to take advantage of the base station resources. In the cloud base stations, the SDR platform can abstract resources as a open interface and share them between multiple users at the same time. In the cloud base station, we care about the LTE physical layer, which includes signal processing. In our system, the SDR platform within the server offers its interface to users to take advantage of virtual resources in the guest OS. The FPGA are connected to the server with PCIe. On top of the native server, the VMM manages all the hardware resources, including the hardware accelerators. Users in the VMs use the accelerator and then complete other processing. The Dom0 is responsible for the accelerators scheduling and uplink and downlink data transmission control.

2.3 FPGA Hardware Accelerator

In this section, the hardware accelerator is described. Xilinx virtex-6 FPGA is used as the hardware resources to implement the turbo decoder and the PCIe interface. The design of hardware accelerator include two sub-modules: PCIe interface and turbo decoder logic.

PCIe Interface. For supporting high channel bandwidth, the LTE needs to finish decoding process as fast as possible. The requirement is to finish it in 1 ms for a sub-frame. We use Xilinx PCIe IPcore v1.7 for the PCIe and DMA design. The PCIe interface has a transfer speed of 400 MB/s. Based on the calculation, a data transfer interface based on PCIe can meet the bandwidth requirement of our experimental system, supporting read and write operation at the same time. The actual data driver transfer data with FPGA module via the PCIe data acquisition card. The PCIe interface use DMA to control the encoded data transform the server data pool to the FIFO in FPGA decoder layer. The block size in data pool is from 4 KB to 1 MB, we can reconfigure it when the driver is loaded.

Turbo Decoder Logic. The detailed design of turbo decoder is shown in next section.

3 Turbo Decoder Design

The decoder is used with a compared encoder to provide an extremely effective way of transmitting data reliably over noisy data channels. The turbo decoder operates very well under low signal-to-noise conditions and provides a performance close to the theoretical optimal performance. The turbo code has good error correction performance, it has been widely used in wireless communication.

Turbo decoder has long interleaving length and need iterate many times, so it has large delay to get the decoding results. For the CPU, it takes very long time to process the decoding operation. We use FPGA instead of CPU for turbo decoding. The FPGA has high clock frequency and parallel processing, all of which can speed up the decoding operation.

3.1 Interface Design

The data needs to use the soft decision input and output, the data received from the SDR platform is 16 bits. Through the quantization, 8 bits (include 5 integer bits and 3 fractional bits) data is input to the decoder accelerator. The PCIe interface has 64 bits width and it can transfer 64 bits data in one clock cycle. So we can use it transfer two groups data at the same clock cycle. The algorithm is known as the Max-scale algorithm, which is a simplified algorithm based on the Log-Map algorithm.

3.2 Design of Turbo Decoder

Figure 4 shows the architecture of turbo decoder. It has three layers.

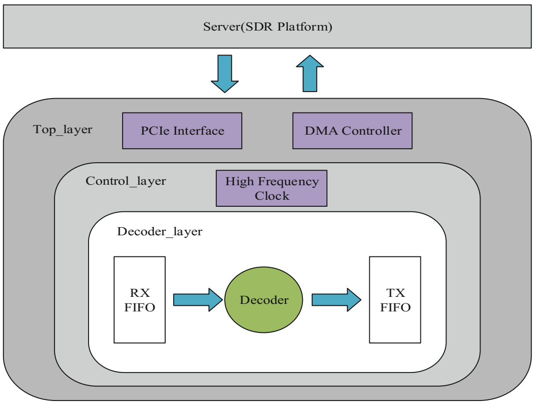

Fig. 4. Turbo decoder design

Top Layer. PCIe hardware interface and the DMA controller is in the top layer. The PCIe driver in the FPGA coordinates with the driver in server to transfer data and control signal.

Control Layer. FPGA reset the decoder core upon the power is on, and it is always waiting for the encoded data and signals input. When the data and parameters are delivered into the FPGA through PCIe, the control layer sends the parameters to the decoder layer first, such as the data block size, reset signal and data count information. Then the encoded data is sent to decoder layer for decoding. This layer also catch the decoded data from decoder layer and send it to the PCIe driver in server with the DMA controller.

Decoder Layer. The decoder layer includes the turbo decoder intellectual property core (IP core) which is the Xilinx IP core for the 3GPP LTE turbo decoder. The main interfaces of the IP core are data input, data output and control signals. We use FIFO to cache data and isolate asynchronous clock between DMA and decoder. The decoder core use 250 MHz clock frequency to achieve higher data throughput while the data transfer clock frequency is 62.5 MHz. When the control layer receives control signal and the encoded data, the information is written into the corresponding RX FIFO in the decoder layer. If the decoder core is free and the data is in the RX FIFO, the data would be transferred into the core.

The decoder layer masks the invalid data into the RX FIFO, which comes from the entire block data transfer through the DMA. After the decoding operation, the decoded data is put into the TX FIFO. At the same time, control layer detect the TX FIFO status and the valid data is read. When the SDR platform receives the decoded data from the PCIe driver, the whole operation is completed.

3.3 Pipeline Process

We write and read data simultaneously with the full-duplex mode by PCIe interface. For the encoded data, the data length ranges from to 24 KB for all 188 different code length in the range of 40 to 6144 bits. The average time for writing data into decoder is about 24 µs and for the reading data from decoder is 10 µs. Taking the decoding stage into consideration, we can make the overall process pipelined. Figure 5 shows the pipelined process for the data write, decoding computation and data read. For the pipelined process, each data block decoding time can be the maximum of reading data, decoding and writing data. In order to use the pipeline operation and adapt to single user, the decoder will compute every data block independently.

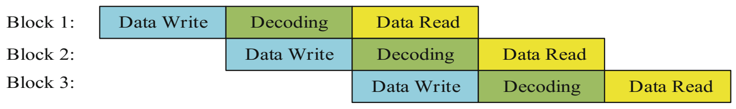

Fig. 5. Pipeline process

3.4 Implementation of Turbo Decoder

The proposed layers of the hardware accelerator is implemented by Verilog HDL at a structural level and then synthesized with Xilinx Synthesis Technology (XST). Then it was placed and routed using ISE Implement Design. After that, the programming file is generated for the FPGA to process the turbo decoder. The implementation results can be illustrated by the Table 1. It shows the utilization of the FPGA resources.

Table 1. Resource utilization

Resource	Used	Available	Utilization
Slice register	45208	301440	14%
Slice LUT	28955	150720	19%
36 Kb block RAM	98	416	23%
18 Kb block RAM	56	832	6%

There are a number of factors which influence the throughput of the FPGA accelerator, such as the processing clock frequency, iterations and algorithm type. Table 2 shows the factors we set to increase the decoder throughput. Within the decoder layer, we use higher clock frequency to decode than the data transfer. The accelerator has the same number of iterations and algorithm type with software implementation, which can compare the performance fairly. We use the

Table 2. Throughput factors

Factors	Settings
Data transfer clock frequency	62.5 MHz
Processing clock frequency	250 MHz
Iterations	6
Algorithm type	Max-scale
Processing units	8
Parallel data input/output width	2 words (64 bits)

maximum processing units and the parallel data input/output width in the IP core to achieve higher throughput. The number of processing units is six, and the parallel data input/output width is 64 bits which matches PCIe interface width.

4 Simulation and Evaluation

In order to minimize the overhead and reduce the time consumption, we can simulate the turbo decoder to evaluate the costs. Xilinx ISE and Modelsim are used to design and simulate the turbo decoder.

Table 3 shows the simulation decoding time for the code length from 40 bits to 6144 bits, including write data time, decoding time and read data time. Because the turbo code rate is 1/3, so the encoded data length is 3 times longer than the decoded data. The input data can be calculated immediately while all the valid data input into the RX FIFO, and the invalid data can be masked by the decoder layer. However, because of the characteristic of the DMA, all the block data should be send to the data pool, and then the driver can read the decoded data from this block. So the read data time is 10 µs for every single code length. In order to reduce this time consumption, we choose the minimize block size: 4 KB for the DMA data transmission.

Table 3. Simulation decoding time

Code length	Data input	Decoding	Data output
40 bits	0.5 µs	25.1 µs	10 µs
512 bits	4.2 µs	30.1 µs	10 µs
1024 bits	8.3 µs	31.8 µs	10 µs
2048 bits	16.5 µs	32.5 µs	10 µs
3008 bits	24.2 µs	38.4 µs	10 µs
4032 bits	32.4 µs	45.2 µs	10 µs
5184 bits	41.6 µs	52.1 µs	10 µs
6016 bits	48.3 µs	59.5 µs	10 µs

Figure 6 shows the experimental results of decoding time for the code length from 40 bits to 6016 bits. The hardware accelerator decoding time for the max code length is almost one tenth of the software decoding. And the average decoding time is one eighth of the software decoding. So we can use the hardware accelerator to support at least 5 M bandwidth of LTE platform. While the software decoding of LTE platform cannot support even the 2.5 M bandwidth because the total time exceeds 2 ms.

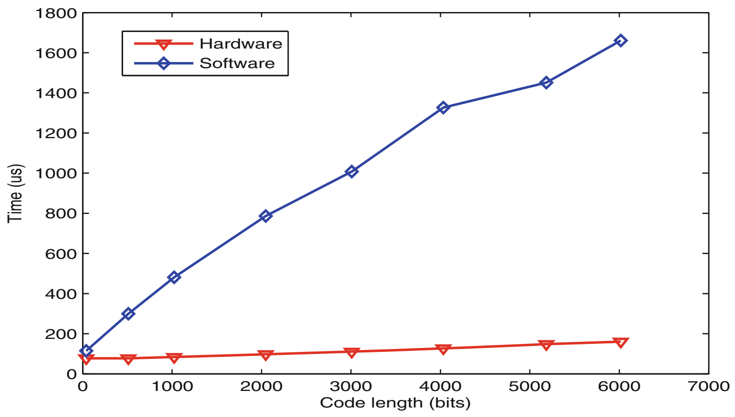

Fig. 6. Decoding time

We run the LTE physical layer in VM and native server respectively to evaluate the overhead of virtualization. Table 4 shows the overhead of Xen virtualization. For the long code length, the overhead can be relatively small (about 10 µs), which can be accepted. For the short length, we can change the algorithm to combine multiple data blocks together to improve the performance.

Table 4. Xen virtualization overhead

Code length	Native	VM	Xen VM overhead
40 bits	68.3 µs	77.6 µs	13.6%
512 bits	69.2 µs	77.2 µs	11.6%
1024 bits	78.9 µs	83.9 µs	6.3%
2048 bits	87.1 µs	95.5 µs	9.6%
3008 bits	103.1 µs	110.1 µs	6.8%
4032 bits	117.4 µs	127.1 µs	8.3%
5184 bits	135 µs	148.2 µs	9.8%
6016 bits	150.8 µs	160.4 µs	6.4%

5 Conclusions and Future Work

This paper designs and implements a FPGA-based turbo decoder hardware accelerator for C-RAN. Xen virtualization is used to support multiple virtual BSs on the C-RAN platform, and the FPGA hardware connected with PCIe interface to server as hardware accelerator. Experimental results demonstrate the turbo decoder accelerator based on the FPGA and Xen platform has much better performance to support the real time signal process with high bandwidth. The turbo decoder hardware accelerator can solve the timing constraints well in the C-RAN.

We believe the accelerators can be used for all VMs in C-RAN. So we will design the scheduling algorithm and make the accelerators support parallel processing of multitasking. And we will also design more accelerators for the C-RAN such as FFT and MIMO decoder to further reduce overall processing latency of C-RAN.

Acknowledgments. This work was supported in part by National Science and Technology Major Project of China under Grant 2014ZX03003003, in part by the National Natural Science Foundation of China under Grant 61390513.

References

1. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. ACM SIGOPS Oper. Syst. Rev. **37**(5), 164–177 (2003)
2. Chen, F., Shan, Y., Zhang, Y., Wang, Y., Franke, H., Chang, X., Wang, K.: Enabling FPGAs in the cloud. In: Proceedings of the 11th ACM Conference on Computing Frontiers, p. 3. ACM (2014)
3. Glossner, J., Hokenek, E., Moudgill, M.: The sandbridge sandblaster communications processor. In: Tuttlebee, W.H.W. (ed.) Software Defined Radio, pp. 129–159. Wiley (2004)
4. Haruyama, S.: FPGA in the software radio. IEEE commun. Mag. **37**, 109 (1999)
5. China Mobile. C-RAN: the road towards green RAN, version 2 . White Paper (2011)
6. Tan, K., Liu, H., Zhang, J., Zhang, Y., Fang, J., Voelker, G.M.: Sora: high-performance software radio using general-purpose multi-core processors. Commun. ACM **54**(1), 99–107 (2011)
7. Wang, W.: Accessing an FPGA-based hardware accelerator in a paravirtualized environment. Ph.D. thesis, Université d'Ottawa/University of Ottawa (2013)
8. Wu, J., Zhang, Z., Hong, Y., Wen, Y.: Cloud radio access network (C-RAN): a primer. IEEE Netw. **29**(1), 35–41 (2015)