HLS Based Ultra-low Latency FAST Protocol Decoder

Hongwei Kan
Guangdong Inspur Intelligent Computing Technology Co. Ltd, and Digital Guangdong network construction co. Ltd, Guangdong China
kanhongwei@inspur.com

Rui Hao*
Guangdong Inspur Intelligent Computing Technology Co. Ltd, Guangdong China
haorui@inspur.com

Jiangwei Wang
Guangdong Inspur Intelligent Computing Technology Co. Ltd, Guangdong China
wangjw01@inspur.com

Guoqiang Mei
Inspur Electronic Information Industry Co. Ltd, Jinan Shandong China
meiguoqiang@inspur.com

Dongdong Su
Inspur Electronic Information Industry Co. Ltd, Jinan Shandong China
sudongdong@inspur.com

Songqing Deng
Digital Guangdong network construction co. Ltd, Guangdong China
sqdeng@digitalgd.com.cn

ABSTRACT

Nowadays the advantages of heterogeneous acceleration technology are becoming more and more obvious. Edge computing center began to accelerate its distributed trading business based on FPGA technology, especially in the field of securities. Shanghai stock market uses efficient FIX Adapted for Streaming (FAST) protocol to transmit the market data. FAST protocol has high compression ratio and complex decoding process. Aiming at efficient FAST protocol decoding, we propose a new general FPGA logic design framework, which implements a low complexity parallel FAST field matching state machine and a low latency stock code mapping mechanism. In this paper, Vivado HLS is used to design and generate the relevant codes. Test results show that the clock frequency after place and routing can reach 250MHz under Xilinx VU37P FPGA. For the 10-level and 50-order message of FAST decoding, the processing delay of core parsing module is only 36ns and 56ns, FPGA subsystem delay is about 120ns. The solution in the paper is one order of magnitude more than traditional CPU software solutions, and is superior to other existing designs base on FPGA, so it is expected to be widely used in the financial market.

CCS CONCEPTS

• **Computing methodologies**; • **Concurrent computing methodologies**; • **Concurrent programming languages**;

KEYWORDS

Heterogeneous acceleration, Stock market decoding, FAST protocol, FPGA, HLS, Ultra-low latency

*Corresponding author.

ACM Reference Format:
Hongwei Kan, Rui Hao, Jiangwei Wang, Guoqiang Mei, Dongdong Su, and Songqing Deng. 2021. HLS Based Ultra-low Latency FAST Protocol Decoder. In *The 5th International Conference on Computer Science and Application Engineering (CSAE 2021), October 19–21, 2021, Sanya, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3487075.3487150

With the widely use of the application of distributed artificial intelligence in the field of Fintech(financial technology), FPGA-based acceleration solutions in quantitative trading, high-frequency trading and other businesses have begun to emerge. As input information is received from the distributed network, the transaction order information is transferred from the network interface to the external interface. The processing delay of Ethernet directly determines the profit and loss of the product [1, 2]. Due to its low-latency and high-concurrency processing characteristics [3], FPGA and FPGA Cluster [4, 5] have been widely used in many financial fields such as market analysis, high-frequency trading, and risk control [6, 7]. Shanghai and Shenzhen LEVEL-2 quotations are transmitted using the stream-oriented financial information exchange protocol (FAST) protocol. FAST is a message data stream-oriented coding method with high compression rate and processing efficiency [8, 9], FAST encoding includes the use of XML information template encoding and byte compression [10]. Due to the complexity of the information template, the use of FPGA to achieve low-latency parallel decoding of the FAST protocol has certain challenges.

At present, there are some domestic and foreign solutions for FPGA decoding for FAST protocol [6, 11–18]. For the international standard FAST decoding, literature [6] implements a microcode engine on FPGA, and the overall delay of FPGA is 2.6us, which is greatly improved compared to software solutions; Literature [12] uses a state machine to parse FAST templates, but only supports the analysis of a single field one by one; Literature [13] implements the entire order system through the HLS method, and the FAST decoding part is not optimized enough. For the domestic Shanghai Stock Exchange market FAST decoding, literature [14] proposed a parallel FAST decoding structure, reaching the performance of 380.4ns decoding 64 field messages; literature [15] uses HLS to automatically generate codes, and the performance is slightly lower than hardware implementation; literature [16] implements parallel

FAST decoding architecture, and the performance of 173ns decoding 64 field messages is achieved, but the method of field dynamic matching is not considered. Literature [17] implement an architecture that uses Cuckoo hashing and use external SRAM modules to store the book with a latency of 253ns. Literature [18] does not give a specific implementation, but gives a detailed test result of the snapshot data. The FPGA penetration delay is 800ns to 1.2us, but the document does not describe the performance of market processing in detail. In the field of securities trading where processing delay is extremely emphasized, it is still necessary to further optimize the decoding design. This paper proposes a new parallel fast decoding method, which compresses the processing delay of the FPGA to less than 200ns, and improves the FPGA processing throughput as much as possible to approach the 10G line speed to reduce the delay jitter.

The main content of this article includes:

- A new type of FAST protocol decoding framework: for FAST protocol

The complex receiving state machine of the proposed decoding proposes a dynamic parallel field matching scheme, which reduces the complexity of the decoding state machine, thereby achieving a higher clock frequency and fewer decoding clock cycles, and reducing the FAST decoding delay.

- Sparse storage and parallel streaming of full information for securities varieties

Waterline market decoding: Starting from the classification characteristics of securities codes, an automatic identification and mapping mechanism is proposed to solve the problems of low-latency securities information reading and irregular updates of securities codes.

- The high-performance decoder of pipeline design is realized by using HLS:

Optimize the data flow for multiple key modules, and increase the parallelism through HLS compilation instructions, so as to achieve a high-performance pipeline processing with 4 bytes of input in 1 clock cycle.

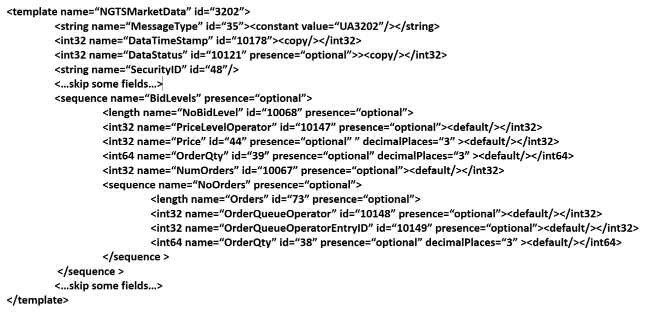
- The relevant performance is verified through experimental simulation: high performance

The overall delay of the FAST decoder is as low as 9 clock cycles, which is significantly lower than similar designs.

Section 2 of this article briefly introduces the background knowledge of FAST decoding; Section 3 introduces the overall structure, main components of the FAST decoding processor; Section 4 introduces some special optimization methods for high-performance FAST decoding processors using HLS; Section 5 introduces FAST decoding The performance of the processor, including clock frequency, area, delay, processing throughput, etc.; Section 6 introduces the conclusion of the article.

1 FAST PROTOCOL DESCRIPTION

The data transmitted by the Shanghai Stock Exchange through the FAST protocol [6] includes information types such as transaction by transaction, index quotes and market quotes. Each information type specifies the data format sent and received through a predefined XML template. Take the most complicated market information as an example, its simplified XML template is as follows:



```
<template name="NGTSMarketData" id="3202">
        <string name="MessageType" id="35"><constant value="UA3202"/></string>
        <int32 name="DataTimeStamp" id="10178"><copy/></int32>
        <int32 name="DataStatus" id="10121" presence="optional">><copy/></int32>
        <string name="SecurityID" id="48"/>
        <...skip some fields...>
        <sequence name="BidLevels" presence="optional">
                <length name="NoBidLevel" id="10068" presence="optional">
                <int32 name="PriceLevelOperator" id="10147" presence="optional"><default/></int32>
                <int32 name="Price" id="44" presence="optional" " decimalPlaces="3" ><default/></int32>
                <int64 name="OrderQty" id="39" presence="optional" decimalPlaces="3" ><default/></int64>
                <int32 name="NumOrders" id="10067" presence="optional"><default/></int32>
                <sequence name="NoOrders" presence="optional">
                        <length name="Orders" id="73" presence="optional">
                        <int32 name="OrderQueueOperator" id="10148" presence="optional"><default/></int32>
                        <int32 name="OrderQueueOperatorEntryID" id="10149" presence="optional"><default/></int32>
                        <int64 name="OrderQty" id="38" presence="optional" decimalPlaces="3" ><default/></int64>
                </sequence >
         </sequence >
        <...skip some fields...>
</template>
```

Figure 1: An Example of a Simplified Market Data Template.

As shown in Figure 1, each line of the template describes a field, including the field type, name, ID number, operator and other key elements. The entire market information contains three major parts, the basic information part before the sequence and the optional beginning of the two sequences Nested loop part; the sequence loop in the 7th line is used to send 10 files of market data, and the loop starting in the 13th line is used to send 50 orders of data.

The XML template indicates the addition, deletion, and modification of the current sending information and the historical sending information. The FAST protocol compresses the sent data to a certain extent; in addition, by encoding the field with variable length, the amount of sent data is further compressed.

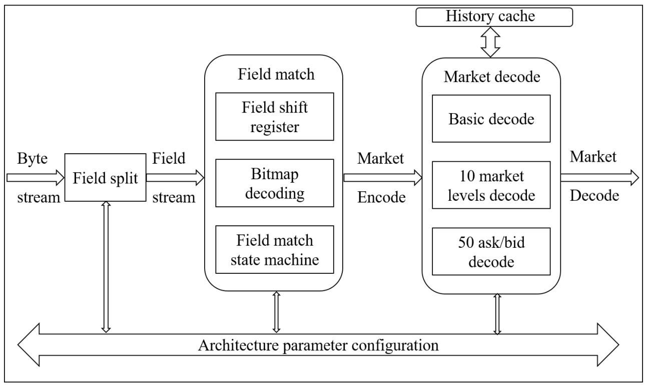


Figure 2: The Overall Architecture of the FAST Decoder.

2 THE OVERALL ARCHITECTURE OF THE FAST DECODING PROCESSOR

Corresponding to the FAST sender described in Section 2, the receiver includes the following processes: field segmentation, field matching, acquisition of the existence bitmap, and final acquisition of field information. The overall architecture of the FAST decoder designed in this paper is shown in Figure 2

As shown in Figure 2, corresponding to a 10Gbits/s network interface, the input is a byte data stream of 4 bytes per clock cycle. According to the characteristics of the stop bit, the field segmentation module processes the byte data stream into a field data stream.

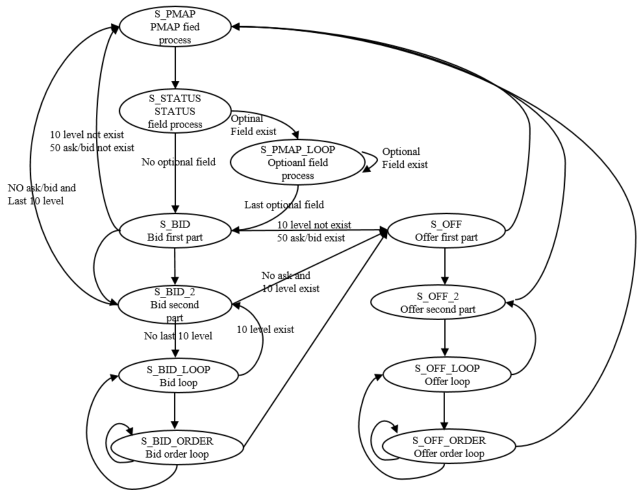


Figure 3: FAST Decoding Parallel Field Matching State Machine.

The field matching module is based on the internal field matching state machine reads the variable fields from the field variable rate shift register, corresponds the input field stream to the actual field, and generates coded market information. The market decoding module combines the current coded market information with the stored historical market information produces the final decoded market information.

2.1 Field Match

As mentioned in section 2, the market message may contain the cycle data of the two directions of buying and selling and the cycle data of commission. If the decoding is performed according to the number of input fields, the jump relationship is too complicated. In order to reduce the complexity of field matching, the input fields are first cached in the field shift register. The number of input fields in the shift register is the maximum number of fields in each clock cycle. In the case of 10Gbits/s, the input is 4 bytes, corresponding to a maximum of 4 fields; the output of the shift register is the number of reads of the field matching state machine. The throughput and design clock frequency are taken into consideration here, and the maximum reading of 4 fields is supported. Literature [16] uses four parallel FAST decoders, but the control logic of the four decoders working in parallel is extremely complex, which is not explained in detail in this Literature. We design a set of control logic based on state machine jump.

The field matching state machine simplifies the matching by limiting the jump of the FAST decoding state. The number of state machine states and increase the clock frequency, at the same time, by obtaining the field of the number of parallels, the overall throughput capacity is improved.

The specific matching state machine jump design is as follows:

As shown in Figure 3, the matching state machine is changed from the traditional field-by-field jump to the matching jump of multiple parallel fields, which improves the ability of a single clock to process multiple fields; at the same time, the jump between state machines is optimized. Simply jump according to the number of input fields, but by limiting the jump possibilities of the state machine in advance, according to the known information, dynamically read multiple parallel fields, simplifying under the premise of reading as many fields as possible. The complexity of the state machine. Through above two methods, the number of cycles to be processed by the FAST message is reduced and the main frequency of the matching state machine is increased, thereby reducing the overall processing delay.

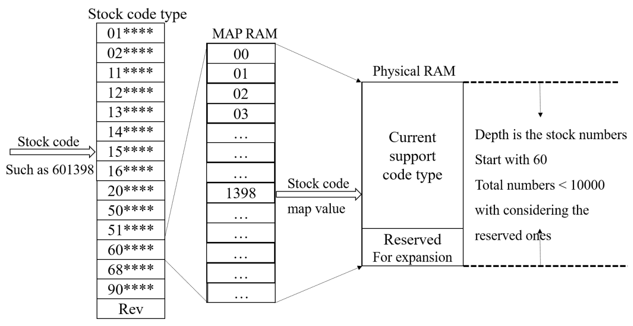


Figure 4: Mapping Method from Security Code to Actual RAM Address.

2.2 Market Decoding Design

The market decoding module mainly consists of two parts: the storage and reading of the full historical market data, the 10 levels and the 50 orders pipeline processing solutions.

Literature [17] implement an architecture that uses Cuckoo hashing and use external SRAM modules to store the book with a latency of 253ns. Instead of hashing, we adopt an efficient RAM address mapping mechanism with a latency of 56ns.

For the storage and reading of all historical market data, the current Shanghai market security code is six digits. Using the security code as the RAM address directly requires too much address space. By analyzing the classification characteristics of the security code, it is found that there are a total of 14 types for two higher digits, such as 01/02/11/12/13/14/15/16/20/50/51/60/68/90, each with a maximum of 10,000 securities codes. Thus 10000*14*16bit can be used as the mapping RAM, and a space is reserved as a new type. A total of 10000*15*16bit RAM is used as the mapping RAM. The specific method of mapping is shown in Figure 4

As shown in Figure 4, the received security code is found as the corresponding mapping RAM according to its high two bits, and then the content of the mapping RAM is read from the low four bits as the final mapping value, which is used to save securities information in the internal RAM of the FPGA real read and write address. If the mapping RAM is all 0s, it means that the security code has not been coded. Write the counter to the mapping RAM and output the calculator. This counter is the address of the FPGA internal RAM corresponding to this security code. At the same time, the counter increments by 1. If the mapping RAM is non-zero, it means that the security code has been coded, and the content in the RAM is output as the address of the FPGA internal RAM corresponding to this security code. It can be seen that through the mapping mechanism, only the actual number of securities for each type of securities plus a certain amount of reserved areas to be expanded are needed, thus realizing low-latency reading plus support for dynamic scalability.

3 HLS IMPLEMENTATION AND OPTIMIZATION OF FAST DECODER

This section will focus on the FAST decoder and explain how to optimize the relevant HLS code and compilation instructions to complete the high-performance low-latency pipeline FAST decoder design. The overall optimization process includes pipeline processing between functions and within functions, the realization of information decoding shift register processing, the read and write pipeline processing of the large memory and the related processing of timing violations, etc.

By using the stream interface, the function is set to DATAFLOW mode, which can complete the pipeline processing between functions; in addition, the corresponding data structure variable can be set to DATA_PACK, thereby expanding the data structure into a completely decomposed wide variable. In the function, the same pipeline processing is required. At the same time, the logic size of the module can be reduced by setting the specific value of the port. In FAST decoding, the field module is called multiple times, we can use parallel pipeline processing by setting the loop to expand UNROLL and setting the related array variable dim to 1. In addition, data dependency often destroys the pipeline inside the function, and you need to carefully modify the code to deal with it.

For information decoding shift register processing, it needs to be set as a static variable, and the pragma compilation option needs to be set to expand the array to a register, and the function of register shift can be realized through circular assignment.

The FAST decoder needs to store the data before each security code. Due to the large number of security codes, it is necessary to read and write and streamline the super-large MEMORY. Using HLS codes to integrate related memories can make the implementation simple. By defining a static structure Array, and set to dual-port URAM by compiling options, you can generate FPGA on-chip URAM. Since the FAST decoder requires a lot of memory, a single FPGA Die storage may be not enough. You can set the delay RAM delay so that the memory can be placed in other Die. In addition, setting the memory cycle dependency instruction inter RAM false makes the read memory operation can be pipelined.

For more complex logic, timing violations often occur. It is necessary to design with the idea of hardware. Take the processing of the bitmap PMAP as an example, each clock cycle needs to find the position of the first 4 bits in the 43bit vector, so that a full adder is required. The delay of the full adder can be controlled by setting related commands.

Combining the above HLS optimization methods, high-performance FAST decoder design can be completed.

4 IMPLEMENTATION AND VALIDATION RESULTS

The FAST decoder designed in this paper is applied to the FPGA board of Shanghai Stock Exchange in-depth market processing. The board is a FPGA heterogeneous accelerator card F37X, including Xilinx's VU37P FPGA chip. The original data processed by the board is 10 Gigabit fiber input, sent through a network card on the server, as shown in Figure 5

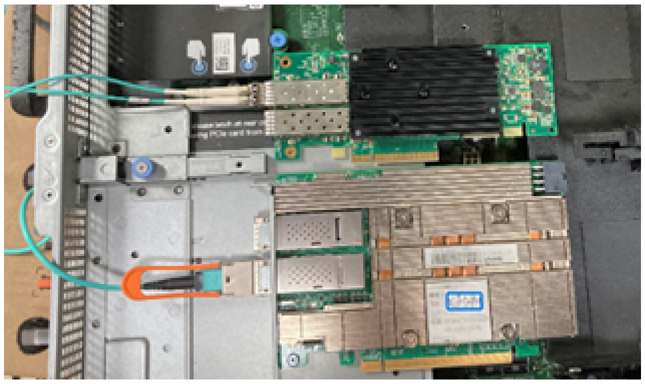


Figure 5: FAST Decoder FPGA Prototype System.

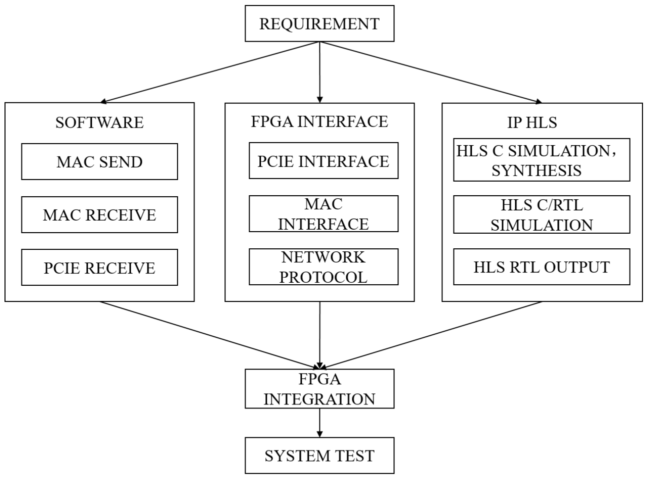


Figure 6: Test Procedures for FPGA Prototype System.

The processed results of the decoder FPGA prototype system are transmitted to information users through PCIE or 10 Gigabit fiber. The development, simulation, synthesis, and test procedures of the entire board system are shown in Figure 6

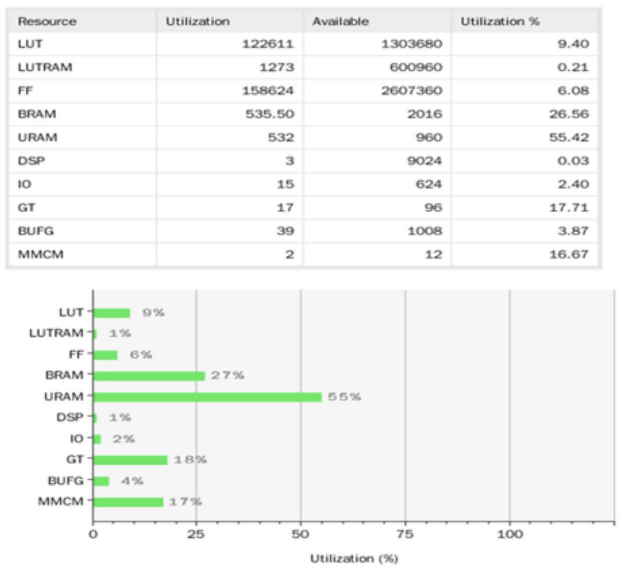
The FPGA system test data adopts the actual collected trading data of the Shanghai Stock Exchange throughout the day, and uses custom software to accelerate the playback within half an hour, and transmits it to the FPGA for processing through the 10G network port. The processed information is transmitted to the host computer through the network port or PCIE interface. The following is the FPGA placement and routing results and the actual test results of the software.

As shown in Table 1, the timing of the FPGA system's overall clock and core state machine fully meets the 4ns constraint, and the operating frequency can reach 250MHz.

As shown in Figure 7, the logic resources and register resources occupied by the decoder are relatively low relative to the entire FPGA, leaving sufficient space for functions such as algorithmic

Table 1: FAST Decoder System and Core Module Clock Frequency

Path type	Worst Negative Slack (ns)	Max clock frequency(MHz)
All path	0.048	253
currState_reg/D	0.31	271



Resource	Utilization	Available	Utilization %
LUT	122611	1303680	9.40
LUTRAM	1273	600960	0.21
FF	158624	2607360	6.08
BRAM	535.50	2016	26.56
URAM	532	960	55.42
DSP	3	9024	0.03
IO	15	624	2.40
GT	17	96	17.71
BUFG	39	1008	3.87
MMCM	2	12	16.67

Figure 7: FAST Decoder System FPGA Logic Resources.

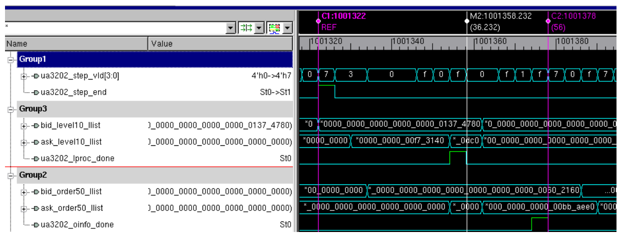


Figure 8: Simulation Results of Prototype System.

trading after market acquisition; URAM and BRAM occupy a large amount, which is due to the types of market data. If there is a greater demand for on-chip SRAM in the future, a certain compression scheme can be considered.

We perform simulations on the FPGA prototype system. As shown in the purple line interval in Figure 8, 10-level message processing delay is 36ns and 50-order message processing delay is 56ns. As shown in Table 2, the processing delay of each module inside the FPGA is given, and the total is 120ns. As shown in Table 3, even at the same clock frequency, it still has a 23% performance improvement compared with the results in Literature [13].

Another key indicator is market processing bandwidth, so we take an all-day market data on the Shanghai Stock Exchange as an example for testing. The host adopts fast playback mode and uses FPGA to receive and decode the data. The data input is 702.96M

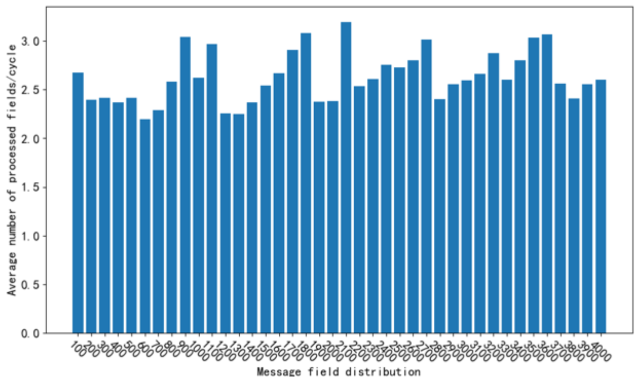


Figure 9: FPGA Results of Prototype System.

bytes, and 434.83M fields are generated after field decomposition, which is processed by 175.74Mcycles. After that, the average processing performance reaches 2.47 fields/cycle, and the statistical results are as follows:

As shown in Figure 9, the number of fields of all input messages is counted, and the average processing capacity of each length interval is counted according to the field length. As a result, the average field processing capacity has little correlation with the message field length. For messages with 64 fields, the average processing clock cycle is 2.47 cycle, and the corresponding average processing time is 158.08ns. The comparison with the existing design is as follows:

It can be seen from Table 4 that the design in this paper is based on the lower measured frequency of the board, and it still has a certain performance improvement compared with the simulation results in [14, 16].

5 CONCLUSION

In summary, this paper proposes a high-performance FAST protocol decoding processor based on HLS, which achieves a 10Gbits/s line-rate decoding process. Through a system scheme of shift register division and field state machine matching, we achieves a minimum of 120ns FPGA subsystem processing delay, and the average processing performance reaches 2.47 fields/cycle. At the same time, this article proposes related optimization schemes for HLS design, which verifies that the prototype algorithm can be quickly and efficiently implemented through HLS, which greatly reduces the workload of development and verification. The follow-up will further study based on software compilation ways to achieve low-latency market decoding of universal templates.

Table 2: Delay of Each Module inside FPGA

Module	Clock frequency	Cycles	Delay
Eth parse	312.5MHz	1	3.125ns
TCP filter	312.5MHz	1	3.125ns
Step_parse	312.5MHz	1	3.125ns
Fast filed split	312.5MHz	1	3.125ns
Fast filed FIFO	250MHz	12	48ns
Fast Decode	250MHz	1	4ns
Level 10 /Order 50 pro	250MHz	9/14	36/56ns
Sum		31	120.5ns

Table 3: Delay Comparison between this Article and Related Literature

Literature	Testmethod	Clockfrequency	Cycle/Delay
[13]	Board Test	156MHz	42Cycles 269ns
The paper	Board Test	250MHz	31Cycles 120.5ns

Table 4: Delay Comparison between this Article and Related Literature

Literature	Testmethod	Clockfrequency	64-field processingtime
[14]	Simulation	156MHz	380.4ns
[16]	Simulation	312.5MHz	173ns
The paper	Board Test	250MHz	158.08ns

ACKNOWLEDGMENTS

This work was supported in part by National Key R&D Program of China (2020YFB1805505), and 2019 Quancheng "5150" Talents Doubles Plan (Innovative Talents).

REFERENCES

[1] Brogaard J (2010). High frequency trading and its impact on market quality [J]. Northwestern University Kellogg School of Management Working Paper.

[2] LOCKWOOD J W, GUPTE A, MEHTA N, *et al.* (2012). A low-latency library in FPGA hardware for high-frequency trading (HFT) [C]//2012 IEEE 20th annual symposium on high-performance interconnects. IEEE, 9–16.

[3] D. Yuan, H. Kan and S. Wang (2020). Ultra Low-latency MAC/PCS IP for High-speed Ethernet," 2020 International Conference on Space-Air-Ground Computing (SAGC), pp. 73-75.

[4] H. Kan, R. Li, D. Su, Y. Wang, Y. Shen and W. Liu (2020). Trusted Edge Cloud Computing Mechanism Based on FPGA Cluster," 2020 IEEE 8th International Conference on Computer Science and Network Technology (ICCSNT), pp. 146-149.

[5] Wang, Y. , Kan, H., D Su, Shen, Y., & Ou, M. (2020). Energy efficient computing offloading mechanism based on FPGA cluster for edge cloud. EITCE 2020: 2020 4th International Conference on Electronic Information Technology and Computer Engineering, pp. 1120-1125.

[6] LEBER C, GEIB B, LITZ H (2011). High frequency trading acceleration using FPGAs[C]//2011 21st International Conference on Field Programmable Logic and Applications. IEEE, 317–322.

[7] M. Guoqiang, H. Rui, W. Jiangwei, K. Hongwei and L. Rengang (2020). A FPGA based intra-parallel architecture for PageRank graph processing," 2020 IEEE International Conference on Edge Computing (EDGE), pp. 31-38.

[8] D. Rosenberg (2006). FAST Specification Version 1.1[EB/OL]. https://www.fixtrading.org/packages/fast-specification-version-1-1/.

[9] FINANCIAL INFORMATION EXCHANGE PROTOCOL (FIX), Version 4.2 with Errata 20010501, FIX Protocol organization, 2001.

[10] F. El-Hassan and D. Ionescu (2009) SCBXP: An Efficient Hardware-Based XML Parsing Technique, Proc. Fifth Southern Conf. Programmable Logic (SPL '09), Apr. 2009.

[11] R. Pottathuparambil, J. Coyne, J. Allred, W. Lynch, and V. Natoli (2011). Low-Latency FPGA Based Financial Data Feed Handler, in 19th Field-Programmable Custom Computing Machines (FCCM2011), May, 2011.

[12] HUA D, REN J, LIU C, *et al.* (2013). Hardware Accelerated Decoding of FIX/FAST and Book Building of Market Data[J]. Spring, 2013.

[13] BOUTROS A, GRADY B, ABBAS M, *et al.* (2017). Build fast, trade fast: FPGA-based high-frequency trading using high-level synthesis[C]//2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig). IEEE, 1–6.

[14] Lockwood J W, Gupte A, Mehta N, *et al.* (2012). A low-latency library in FPGA hardware for high frequency trading (HFT)[C]//High-Performance Interconnects (HOTI), 2012 IEEE 20th Annual Symposium on. IEEE: 9-16.

[15] TANG Q, SU M, JIANG L, *et al.* (2016). A scalable architecture for low-latency market-data processing on FPGA[C]//2016 IEEE Symposium on Computers and Communication (ISCC). IEEE, 597–603.

[16] YU L, FU Y, LIU T (2017). A Hardware Structure for FAST Protocol Decoding Adapting to 40Gbps Bandwidth [J]. DEStech Transactions on Computer Science and Engineering, (csma).

[17] <number> [17]<number> M. Dvorakand, J. Korenek (2014). Low Latency Book Handling in FPGA for High Frequency Trading," in International Symposium on Design and Diagnostics of Electronic Circuits Systems.

[18] Z. Yue Z. Qiang L. Cheng Hao and Z. Jian Yu. Accelerate the realization of market quotations and improve the energy efficiency of financial transactions-FPGA-based ultra-low latency market quotation system," THE FORELAND OF TRADING TECHNOLOGY, ITRDC_TechMag_040_202011, pp. 30-36.