High Frequency Trade Book Builder using FPGA CSEE 4840 Embedded System

Presenters: Choka Thenappan (ct3185) Shivam Shekhar (ss6960) Ameya Keshava Mallya (am6024)



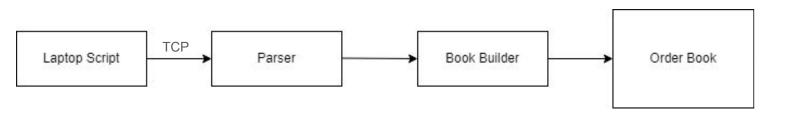
May 12, 2024

- High Frequency Trade Book building using FPGA and simulated NASDAQ ITCH message protocol.
- Main Components -
 - Simulated Market
 - Hardware component
 - HW/SW interface

Background

- Average amount of data coming from the market is around 32 Gb/s. Which means if the data transfer is down even for a millisecond we lose 10000 packets of data.
- Hence we cannot use software methods to parse the data due to latency issues.
- This is why we decided to use FPGA to parse market data to cut latency issues and take advantages of parallel processing speeds.
- Order types :
 - Ask: The term "ask" refers to the lowest price at which a seller will sell the stock.
 - Bid : The term "bid" refers to the highest price a buyer will pay to buy a specified number of shares of a stock at any given time.
 - Sell : Sell refers to the stock being purchased between a buyer and seller.





- We get market data from the simulated market script because of current FPGA limitations.
- The received market data is then processed by a hardware software integrator which sends the data received to the hardware through registers for parsing and book building.
- The parser parses the data and builds and keeps track of the max price.



Incoming Message

Field name	Offset	Length	Format	Binary	Decode	d Notes
Message Type	0	1	Alphanumeric	0x41	`A`	Add Order message
Timestamp	1	8	Numeric	0×0000000000000000	0	Nanoseconds
Order Reference Number	9	4	Numeric	0×000003EA	1002	
Transaction ID	13	4	Numeric	0×0000000	0	
Order book ID	17	4	Numeric	0x000500CD	327885	
Side	21	1	Alphanumeric	0x42	В	"B" = Buy order "S" = Sell order
Quantity	22	4	Numeric	0×0000001	1	
Price	26	8	Price	0×00000000000000000	1	
Yield	34	4	Price	0×0000000	0	

5 | High Frequency Trade Book Builder using FPGA



FPGA_IP = "128.59.19.114" FPGA_PORT = 42000

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

sock.connect((FPGA_IP, FPGA_PORT))

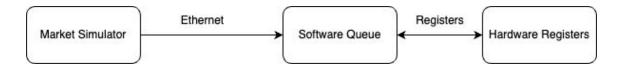


Simulated Market

message = struct.pack("!BIIIIBIQQ8s", msg_type, time_stamp, order_reference_number, transaction_id, order_book_id, side, quantity, price, yield_value, stock return message



- Data coming in from Market sim gets into software queue.
- Software Queue communicates with the Hardware with the defined registers.





HW/SW Interface - Ethernet data

• Reading the network data from ethernet and storing that in a queue.

```
void *network thread f(void *arg) {
    int sockfd = *(int *)arg;
    char recvBuf[BUFFER SIZE];
   while (1) {
        int n = read(sockfd, recvBuf, BUFFER_SIZE);
        if (n < 0) {
            perror("Error reading from socket");
            exit(1);
        } else if (n == 0) {
            // printf("Connection closed by client\n");
            close(sockfd);
            break:
        QueueData newData;
        strcpy(newData.data, data);
        enqueue(&dataQueue, newData);
    return NULL;
```



HW/SW Interface - Writing in registers

Defining registers for all the data in the message and the required flag.

Using the ioctl calls to write/read data.

/* Device Registers */ #define MESSAGE TYPE(x) (x) #define TIMESTAMP(x) ((x)+1)#define ORDER_REF_NUMBER(x) ((x)+2) #define TRANS ID(x) ((x)+3) #define ORDER BOOK ID(x) ((x)+4) #define SIDE(x) ((x)+5) #define OTY(x) ((x)+6) #define PRICE(x) ((x)+7) #define YIELD(x) ((x)+8)#define BUFFER_NOT_EMPTY(x) ((x)+9) #define READPORTT(x) ((x)+10)



static void write_data(vga_ball_color_t *message)

```
iowrite8(message->msg type, MESSAGE TYPE(dev.virtbase) );
iowrite8(message->timestamp, TIMESTAMP(dev.virtbase) );
iowrite8(message->order_ref_number, ORDER_REF_NUMBER(dev.virtbase) );
iowrite8(message->trans_id, TRANS_ID(dev.virtbase) );
iowrite8(message->order_book_id, ORDER_BOOK_ID(dev.virtbase) );
iowrite8(message->side, SIDE(dev.virtbase) );
iowrite8(message->qty, QTY(dev.virtbase) );
iowrite8(message->price, PRICE(dev.virtbase) );
iowrite8(message->yield, YIELD(dev.virtbase) );
iowrite8(message->buffer_not_empty, BUFFER_NOT_EMPTY(dev.virtbase));
dev.message = *message;
```

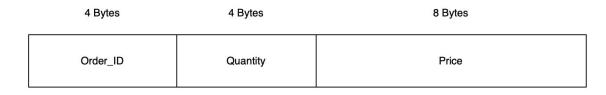


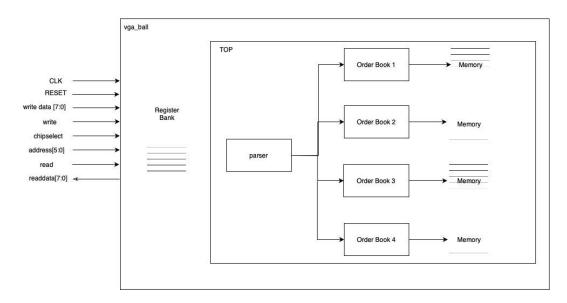
Sensing the hardware to send next data.

```
void write_message(const vga_ball_color_t *c) {
   vga ball arg t vla;
   vla.message = *c;
   unsigned char bufferNotEmpty = ioctl(vga_ball_fd, VGA_BALL_READ_DATA, &vla);
   unsigned char readPort = ioctl(vga_ball_fd, VGA_BALL_READ_DATA, &vla);
   if (bufferNotEmpty && readPort) {
   if (ioctl(vga_ball_fd, VGA_BALL_WRITE_DATA, &vla)) {
       perror("ioctl(VGA BALL WRITE DATA) failed");
       return;
   printf("Msg Type: %02x, Timestamp: %016llx, Order Ref Number: %08x, Order Book ID: %08x, Qty: %08x, Price: %016llx \n",
          vla.message.msg_type, vla.message.timestamp, vla.message.order_ref_number,
          vla.message.order_book_id, vla.message.qty, vla.message.price);
   printf("Done!\n");
       printf("Waiting for ready...\n");
```



HW Design - Implementation



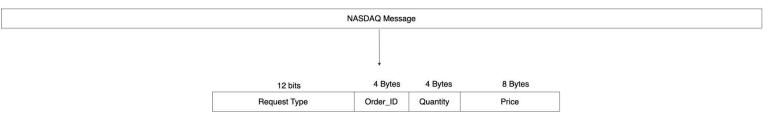


HW Design - Parser

```
module parser(
input [319:0] ff_buffer,
output reg [31:0] out_order_id,
output reg [31:0] out_quantity,
output reg [63:0] out_price,
output reg [11:0] stock_activate
);
```

`define ADD1 12'b10000000000 `define DELETE1 12'b01000000000 `define DECREASE1 12'b00100000000 `define ADD2 12'b00010000000 `define DELETE2 12'b00001000000 `define DECREASE2 12'b000000100000 `define DELETE3 12'b000000010000 `define DELETE3 12'b0000000010000 `define DECREASE3 12'b000000001000 `define DELETE4 12'b00000000010 `define DELETE4 12'b000000000010





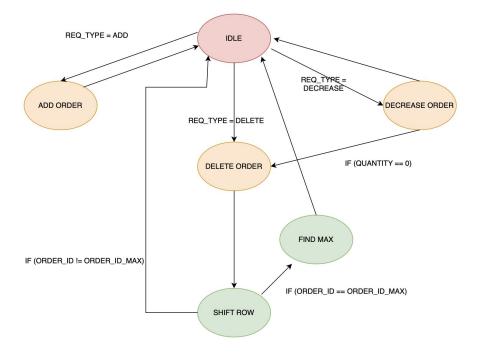


HW Design - Parser Wave Window

2 .	Msgs																			
🗇 /testbench/clk	1'h1																			
🤣 /testbench/resetn	1'h0																			
/testbench/buffer_not_empty	1'h0					1														
	320'h00000	537856	341278563	412000104	2 5378	56341278	56341200	01042e080	0 5378	356341278	3563412000	1042f0800	0 5378	56341278	56341200	10430080	0 15378	356341278	563412000	104
/testbench/slave_ready	1'h1														ſ					
/testbench/system_free	1'hx																		1	
/testbench/master_valid	1'hx	_																		
# /testbench/out_order_id	32'dx	0	66605		ΪO		66606		χo		66607		ΪO		66608		χo		66609	
# /testbench/out_quantity	32'dx	0	100		ΪO		100		10		100		ΪO		100		10		100	
# /testbench/out_price	32'dx	0	85		ΪO		:/testbend	h/out_qua	ntity @ 13	66524 ps	85		ΪO		85		χo		85	
/testbench/stock_activate_out	12'bxxxxxxxx	0000	1000000	00000	10000	00000	_		_	0	1000000	00000	10000	000000	1000000	00000	000	000000	10000000	00000



HW Design - Order Book

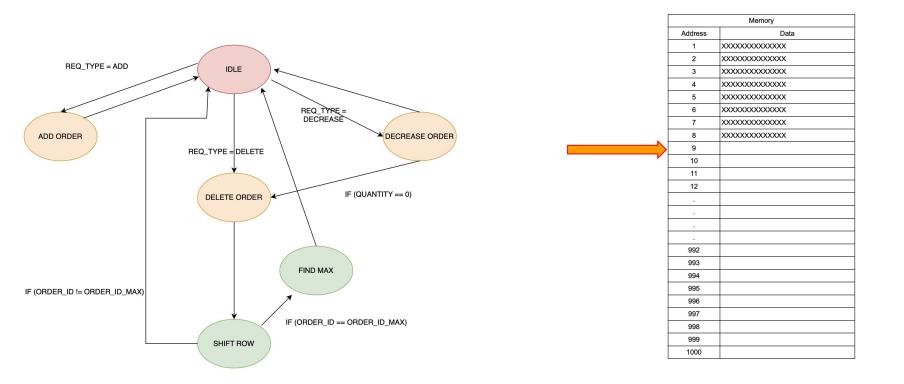


module order_book (
input clk,
input resetn,
input valid,
input [31:0] order_id,
input [31:0] quantity,
input [63:0] price,
input [2:0] req_type,
output reg [31:0] max_order_id,
output reg [31:0] max_quantity,
output reg [63:0] max_price,
output reg ready
);



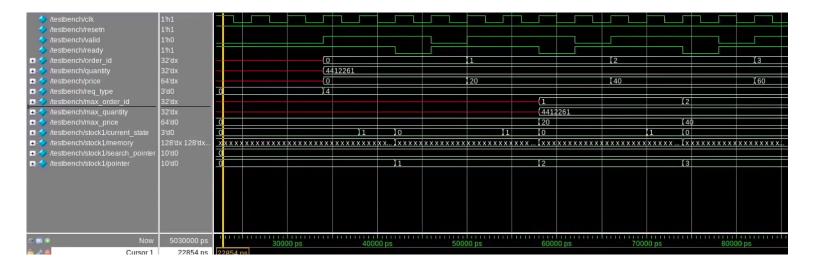


HW Design - Order Book - Add Order





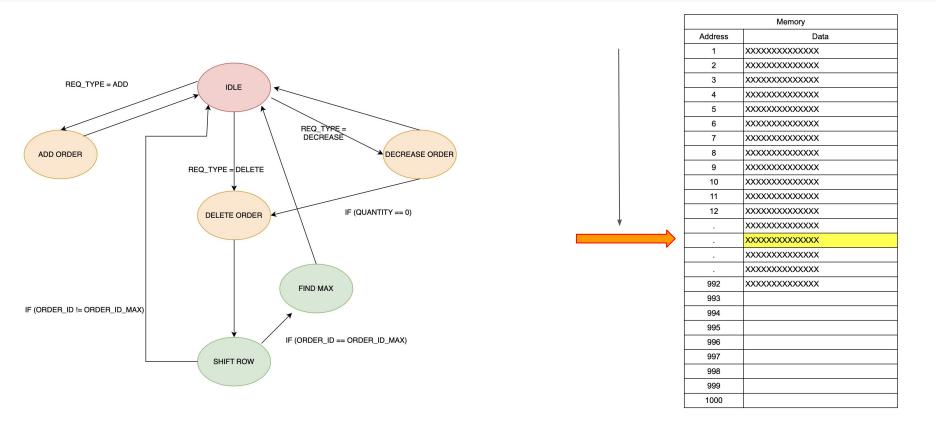
HW Design - Order Book - Add Order - Wave







HW Design - Order Book - Decrease Order - Wave





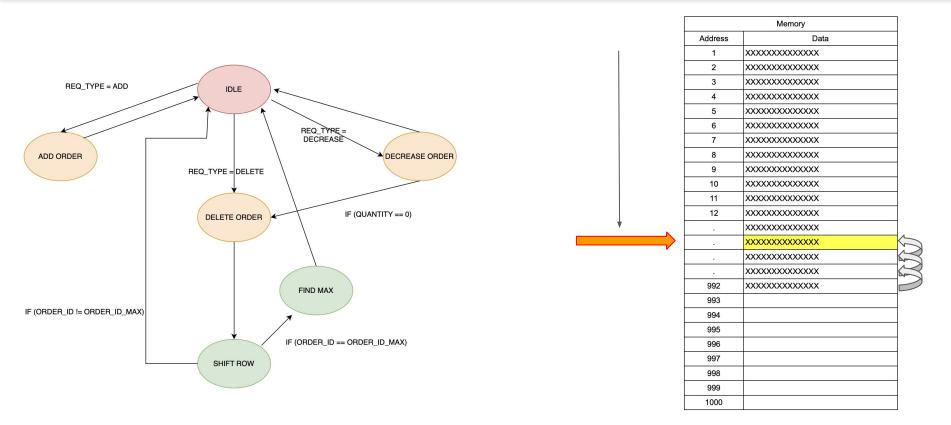
HW Design - Order Book - Decrease Order - Wave

2	Msgs	
/testbench/clk	1'h1	
🗇 /testbench/resetn	1'h1	
Itestbench/valid	1'h1	
/testbench/ready	1'h0	
🖅 🔶 /testbench/order_id	32'h00000064	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
🖅 🤣 /testbench/quantity	32'h0000ffff	00435365 X 0000000 X 000000 X 000000 X 000000 X 000000
🖅 🤣 /testbench/price	64'h000000	
🖃 🔶 /testbench/req_type	3'h2	
💽 🔶 /testbench/max_order_id	32'h00000064	0000064
🖅 🔶 /testbench/max_quantity	32'h00435365	00435365 sim:/testbench/req_type @ 3404075 ps
😟 🥠 /testbench/max_price	64'h000000	0000000003981
🖅 🔶 /testbench/stock1/current_state	3'h4	
/testbench/stock1/memory	128'hxxxxxxx	
🛓 🥠 [1023]	128'hxxxxxxx	
👳 🥠 [1022]	128'hxxxxxxx	

	100000000					
= 🔶 [45]	Sector States and the sector should be a	3565348704991348601489400529796				
🕂 🎝 [44]	128'd34861	3486120542477084263895856579440				
😐 🔶 [43]	128'd34068	3406892379962819926302312629084				
😐 🔶 [42]	128'd33276	3327664217448555588708768678728				
😐 🥠 [41]	128'd32484	3248436054934291251115224728372				
主 🔶 [40]	128'd31692	3169207892420026913521680778016		31692078921433	25752416037503776	
💼 🥠 [39]	128'd30899	3089979729905762575928136827660				
💼 🥠 [38]	128'd30107	3010751567391498238334592877304			sim:/testbench/stock1/memor 316920789214332575241603750	
🛓 🥠 [37]	128'd29315	2931523404877233900741048926948			510520705214552575241005750	3/10
📻 🔶 [36]	128'd28522	2852295242362969563147504976592				
🚊 🥠 [35]	128'd27730	2773067079848705225553961026236				
😐 🛷 [34]	128'd26938	2693838917334440887960417075880				
n now	5030000 ps	3300000 ps	3350000 ps	34000 <mark>00 ps</mark>	3450000 ps	3500000 ps
🗟 🥕 🥌 Cursor 1	3401417 ps			3401417 ps		



HW Design - Order Book - Delete Order





HW Design - Order Book - Delete Order - Wave

Msgs		
/testbench/clk 11h1	արտուրտուրտուրտուր	ուսվուսակուսպիսուսվուս
/testbench/resetn		
/testbench/valid 1'h0		
/testbench/ready 1'h1		
	199 X40 X15	<u> </u>
	3980 4000	14721
■	1	¥2
A / testbench/max order id 32'dx 100		<u>h</u> 2
+ / testbench/max_guantity 32'dx 4412261		
+ / testbench/max_price 64'd0 14721		
• /vestbench/stock1/current_state 3'd0	χ χο χ 5	X0 X2
🖬 🍫 /testbench/stock1/memory 128'dx 128'dx 🛛 🕻 🗴 🗴 🕮 🕹	*	<u> </u>
/testbench/stock1/search_pointer 10'd0		х→
• // /testbench/stock1/pointer 10'd0 197 198 199	X200	
Now 5030000 ps 3200000 ps	3250000 ps 3300000 ps 335	50000 ps 3400000 ps 345000t
💼 🖋 🗧 Cursor 1 22854 ps		
<u>.</u>	128'h000000 0000006b004353650000000b0000085c	10000006b00435365000000000000000
+ − ↓ [106]	128'h00000 0000006a0043536500000000000848	0000006b0043536500b00000000085c
🖷 🔶 [105]	128h00000 000000690043536500000000000834	X0000006a00435365000000000000848
🕂 🔶 [104]	128h00000 000000680043536500000000000820	00000069004353650000000000834
• • [103]	128'h00000 0000067004353650000000000000000000000	X000000680043536500d00000000820
	128'h00000 00000660043\$3650000000000007f8 128'h00000 00000650043\$3650000000000007e4	X000006700435365000000000000000000 X0000006600435365000000000007f8
	128/100000 000006800433365000000000000784	10000006600433365000000000000000000000000
	128'h00000	A0000000045338300000000007E4
1 [98]	128'h00000 0000062004353650000000000007a8	
• [97]	128'h00000 0000006100435365000000000000794	
A Re No	w 5030000 ps 35000 ps 35000	00 ps 400000 ps 4500000 ps 50



HW Design - Order Book - Overall Window

.	Msgs									
🗇 /testbench/clk	1'h1									
🤙 /testbench/resetn	1'h0									
/testbench/buffer_not_empty	1'h0									
/testbench/system_free	1'h1									
	320'h53785	537856341278563	4120001043e0800000	6000000042000	юоьводоооооорооо	537856341278	5634120001043	000000000000000000000000000000000000000	04200000069000	00000000000
	32'dx	66621				66622				
🗉 🚽 /testbench/max_quantity_1	32'dx	183				184				
	64'd0	251				253				
/testbench/max_order_id_2	32'dx									
/testbench/max_quantity_2	32'dx									
/testbench/max_price_2	64'd0	0								
/testbench/max_order_id_3	32'dx									
🖃 🍫 /testbench/max_quantity_3	32'dx									
🖅 🤣 /testbench/max_price_3	64'd0	0	sim:/testbench/m	ax_quantity_3 @ 8	78147 ps					
🖅 🔶 /testbench/max_order_id_4	32'dx		x							
/testbench/max_quantity_4	32'dx									
/testbench/max_price_4	64'd0	0								
/testbench/bid_side/stock1/current_state	3'd0	0	1			0			1	
	10'd0	0							عد تحديد الد	
# /testbench/bid_side/stock_activate_out	12'd2048	2048								
Itestbench/bid_side/stock1/memory	128'dx 128'dx	<u> </u>	*****	*****	x x x x x x x x x x x x x x x .		*******	*****	*********	(XXXXXXX
Itestbench/bid_side/p_block/stock_id	32'd0	0								
/testbench/bid_side/p_block/current_state	2'dx									
/testbench/bid_side/p_block/ns_command	32'dx 32'dx 6	XXXXXX								
/testbench/bid_side/p_block/cs_command	32'dx 32'dx 6	XXXXXX								



Request Type	Number of Cycle
ADD ORDER	1 cycle
DELETE ORDER	N cycles
DELETE ORDER is MAX	N + N - 1
DECREASE ORDER	~Depends on the position

Here N is the number of valid entries in the Book



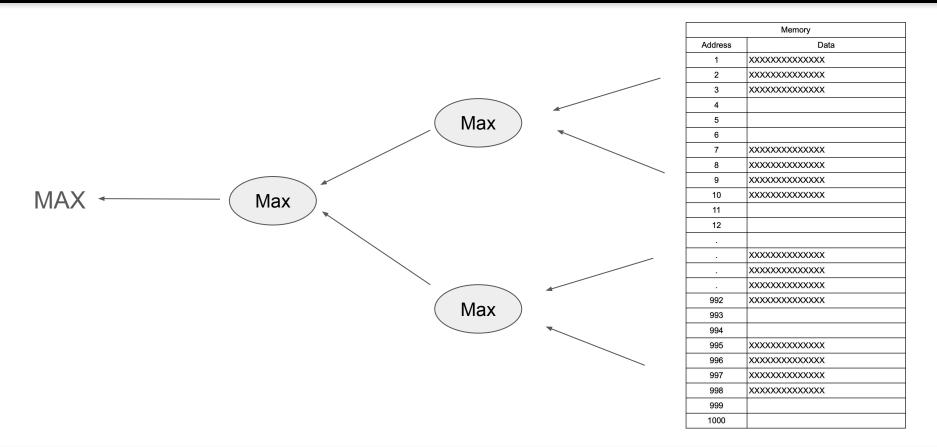
HW Design - Optimizations

	Memory			Memory
Address	Data		Address	Data
1	XXXXXXXXXXXXXX		1	XXXXXXXXXXXXXX
2	XXXXXXXXXXXXXX	4	2	XXXXXXXXXXXXXX
3	XXXXXXXXXXXXXX	1-way memory	3	XXXXXXXXXXXXXX
4	XXXXXXXXXXXXXX		4	
5	XXXXXXXXXXXXXX	2-Bits 62 Bits	5	
6	XXXXXXXXXXXXXX		6	
7	XXXXXXXXXXXXXX		7	xxxxxxxxxxxxxx
8	XXXXXXXXXXXXXX		8	xxxxxxxxxxxxx
9	XXXXXXXXXXXXXX		9	xxxxxxxxxxxx
10	XXXXXXXXXXXXXX		10	xxxxxxxxxxxxx
11	XXXXXXXXXXXXXX		10	
12	XXXXXXXXXXXXXX		12	
	XXXXXXXXXXXXXX			
	XXXXXXXXXXXXXX		•	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
	XXXXXXXXXXXXXX		· ·	
	XXXXXXXXXXXXXX			XXXXXXXXXXXXX
992	XXXXXXXXXXXXXX			XXXXXXXXXXXXXX
993			992	XXXXXXXXXXXXXXX
994			993	
995		4-way memory	994	
996			995	XXXXXXXXXXXXXX
997			996	XXXXXXXXXXXXXX
998			997	XXXXXXXXXXXXXX
999			998	XXXXXXXXXXXXXX
1000			999	
L			1000	





HW Design - Optimizations





HW Design - Performance

Extra Combinational Logic and States - did 10 ways

Request Type	Number of Cycle
ADD ORDER	1 cycle
DELETE ORDER	~N/4 cycles
DELETE ORDER is MAX	N/4 + 3 Cycles
DECREASE ORDER	~Depends on the position (/4 the previous time)

Here N is the number of valid entries in the Book







Thank You!

