MDP 3.0 TICKERPLANT

Daron Lin, Jonathan Liu, Giovanni Ortuno, Mirza Ali

Introduction

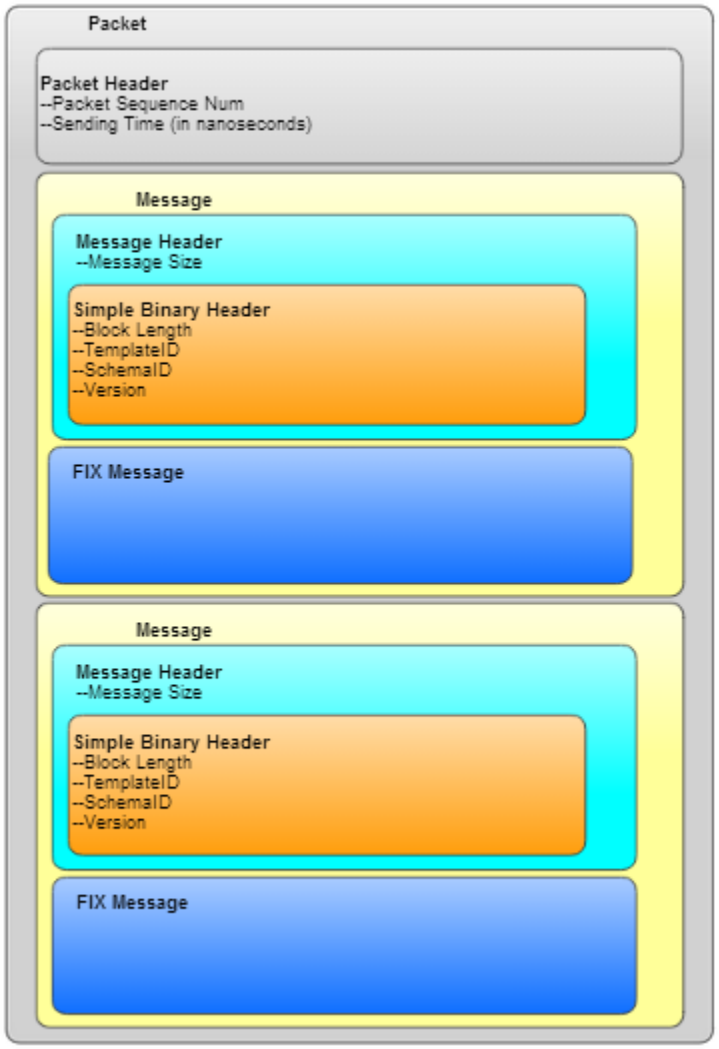
What is MDP3.0

MDP3.0 is a completely new data feed implementation by the CME Group.

Sends Incremental Market Updates among a variety of other information.

Designed to be super quick and efficient

MDP3.0 Protocol



- The encoded FIX transmission is sent in a packet structured as follows:
- Packet header - contains packet sequence number, sending time.
- Message Size - field indicating size of message.
- Message header - contains block length, TemplateID, SchemaID, and Version.
- FIX header - indicates FIX message type (example: 35=X)
- FIX message body - event driven business data such as book updates and trade summary.

Sample Message - Market Data Incremental Refresh (35=X)

```
<!-- MarketDataIncrementalRefresh (35=X) message -->

<sbe:message name="MarketDataIncrementalRefreshTrades" id="02" fixMsgType="X" description="Trade">

<field name="TransactTime" id="60" fixUsage="UTCTimestamp" type="UTCTimestamp" timeUnit="nanossecond" />

<field name="EventTimeDelta" id="37704" fixUsage="String" type="uint16" />

<field name="MatchEventIndicator" id="5799" fixUsage="char" type="MatchEventIndicator" />

<field name="NoMDEntries" id="268" fixUsage="NumInGroup" type="NumInGroup" groupName="MDIncGrp" />

<group name="MDIncGrp">

                <field name="MDUpdateAction" id="279" fixUsage="char" type="MDUpdateAction" />

                <field name="MDEntryType" id="269" fixUsage="char" type="MDEntryType" constant="2" />

                <field name="SecurityID" id="48" fixUsage="String" type="UniqueID" />

                <field name="RptSeq" id="83" fixUsage="int" type="SeqNum" />

                <field name="MDEntryPx" id="270" fixUsage="Price" type="Price" />

                <field name="MDEntrySize" id="271" fixUsage="Qty" type="Qty" />

                <field name="NumberOfOrders" id="346" fixUsage="int" type="uint16" />

                <field name="AgressorSide" id="5797" fixUsage="int" type="AgressorSide" />

</group>

</sbe:message>
```

Our Project

We decode market data incremental refresh messages sent from the CME Group

Using this data we generate our own version of order-books for specific securities.

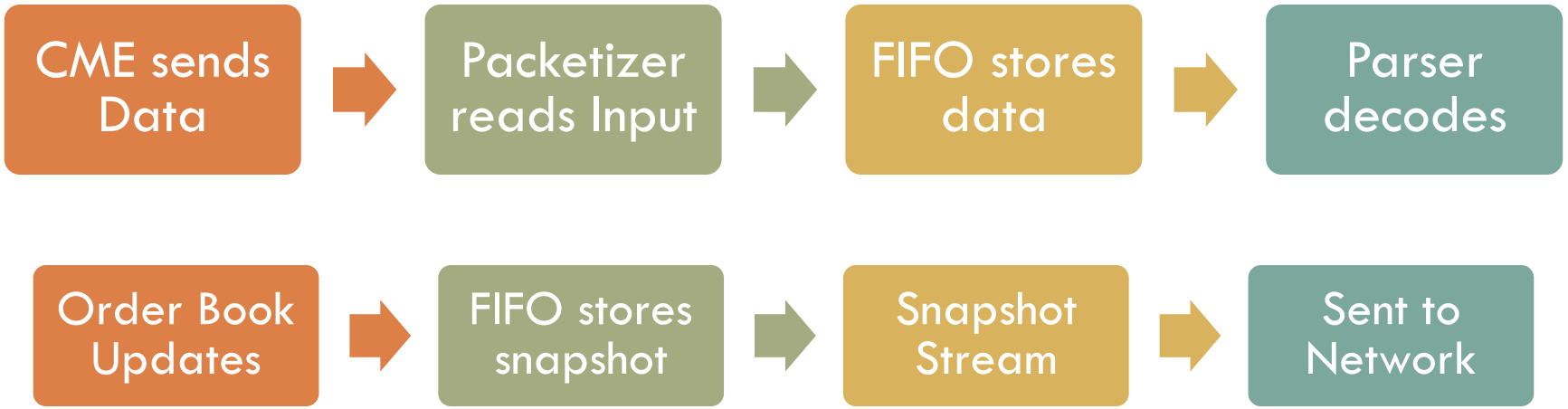
We then send out snapshots of these order books at regular intervals

Software Implementation

- Python Code
  - Book Builder

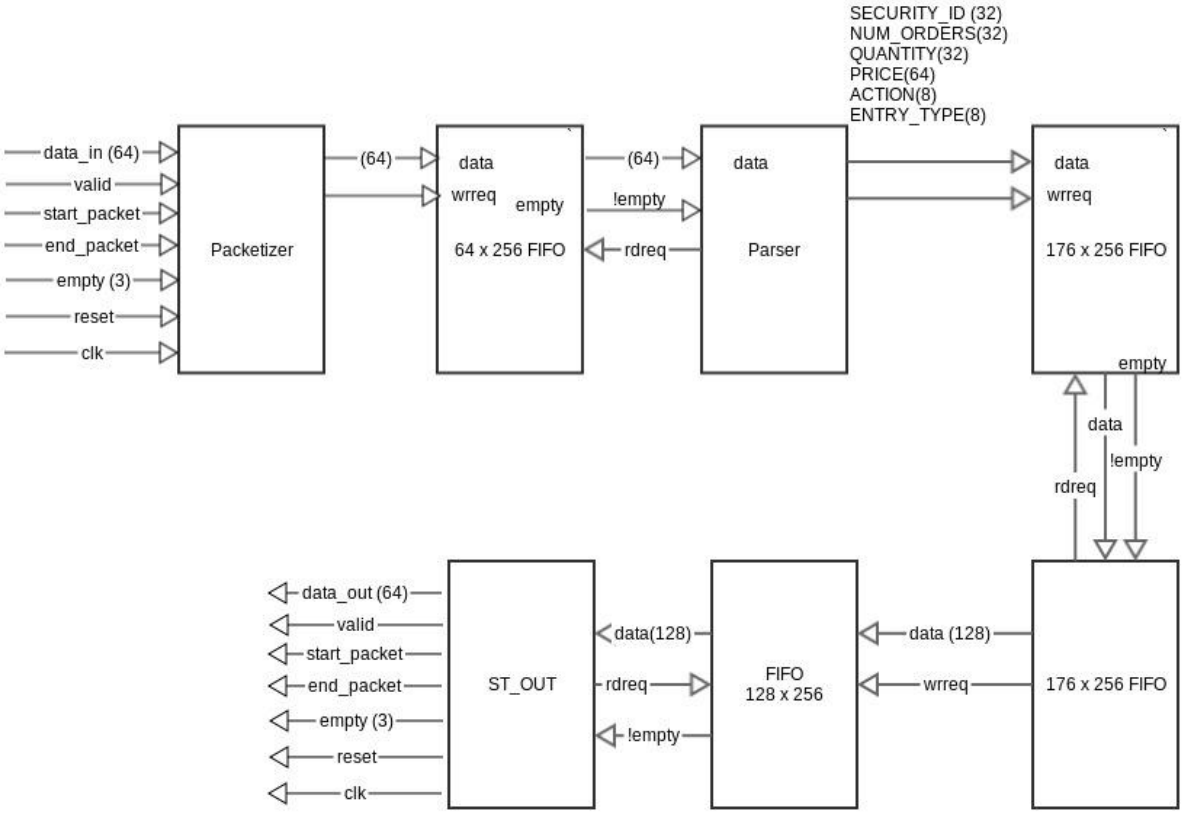
```
2
3  def shift(l, n, rl):
4      if rl=="r":
5          return l[n:] + l[:n]
6      return l[:n] + l[n:]
7
8  class Order():
9      def __init__(self, price, quantity):
10         self.price = price
11         self.quantity = quantity
12
13     def __str__(self):
14         return `self.quantity` + ": $" + `self.price`
15
16 class FullOrderBook:
17     def __init__(self):
18         self.askBook = Book()
19         self.bidBook = Book()
20
21     def __str__(self):
22         askString = 'ASK\n---------\n'
23         bidString = 'BID\n---------\n'
24
25         for bookOrder in reversed(self.askBook.book):
26             askString += `bookOrder.quantity` + ": $"+`bookOrder.price` + '\n'
27
28         for bookOrder in reversed(self.bidBook.book):
29             bidString  += `bookOrder.quantity` + ": $"+`bookOrder.price` + '\n'
30
```

General Architecture

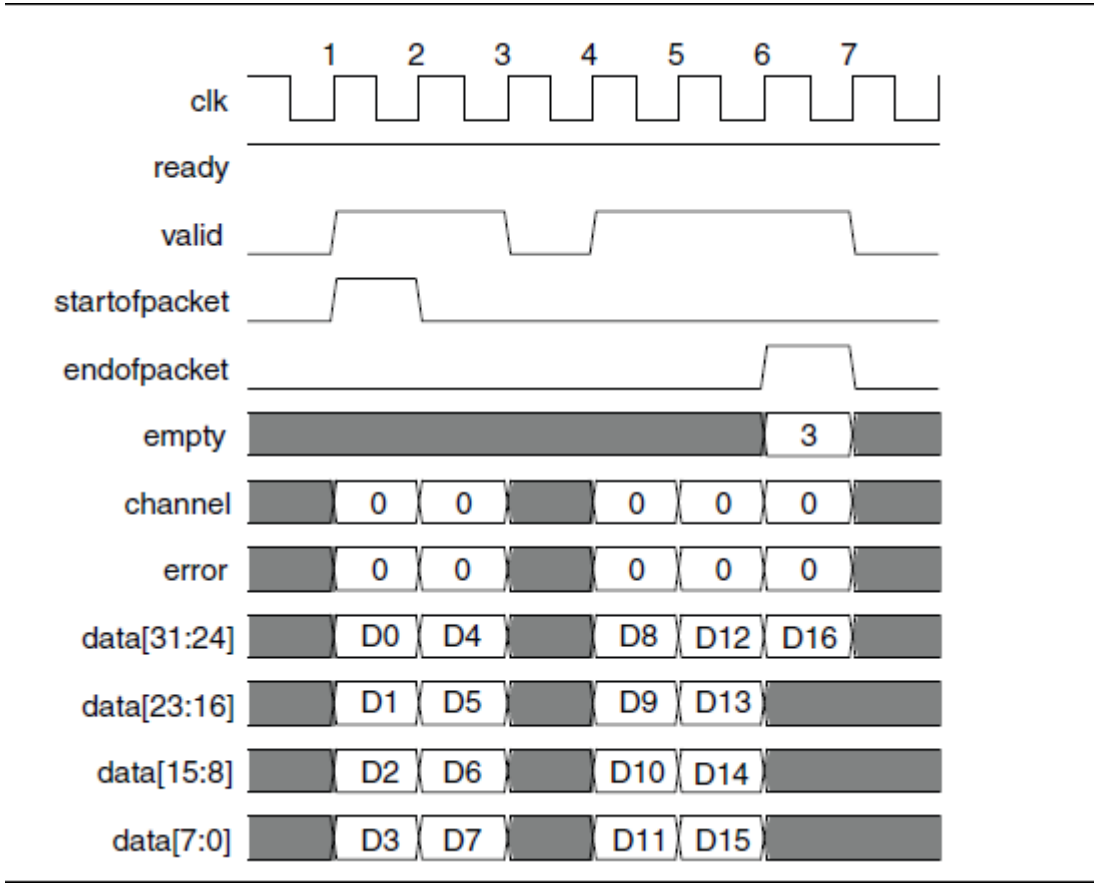


Development Architecture

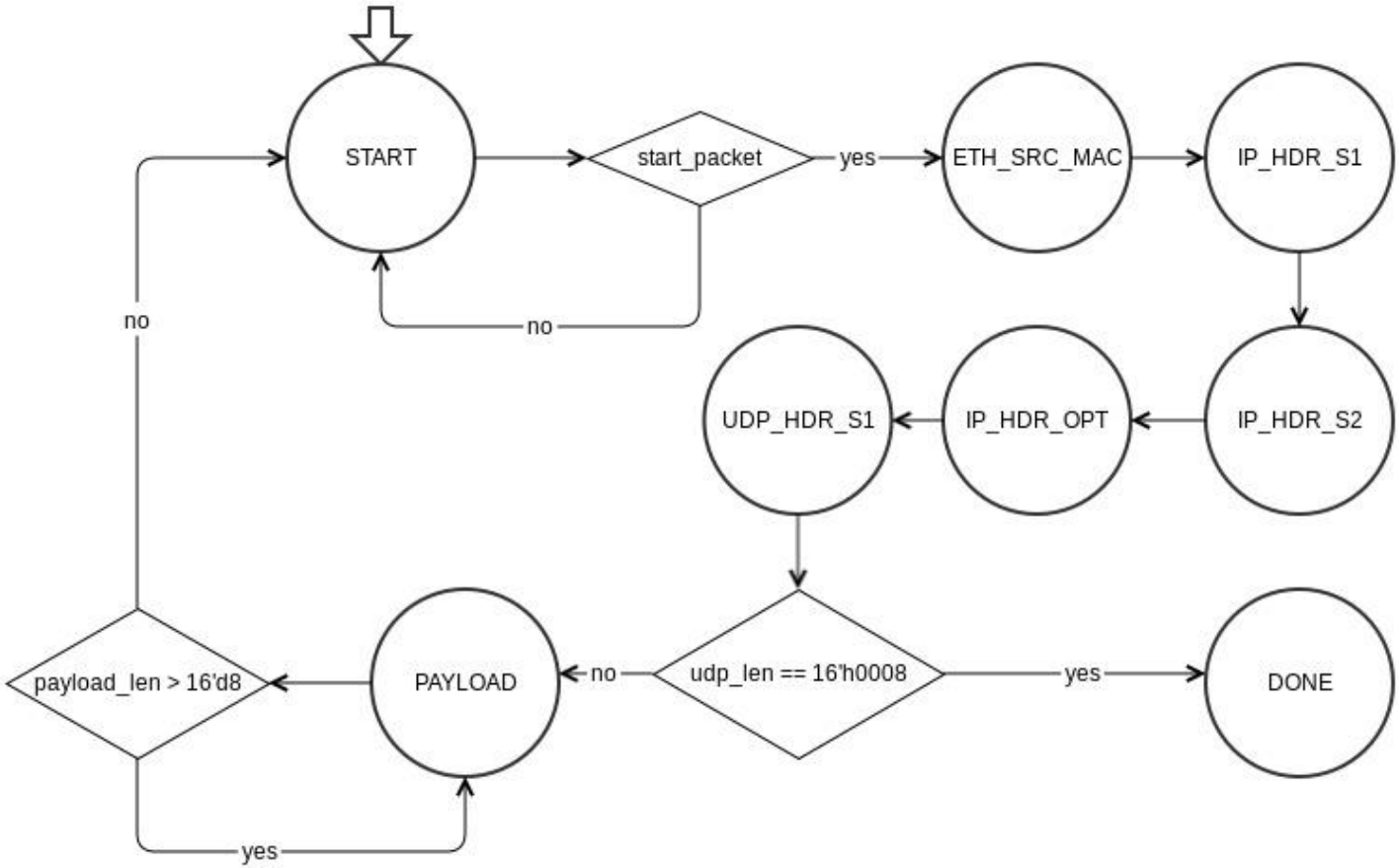
Hardware Implementation



Avalon ST

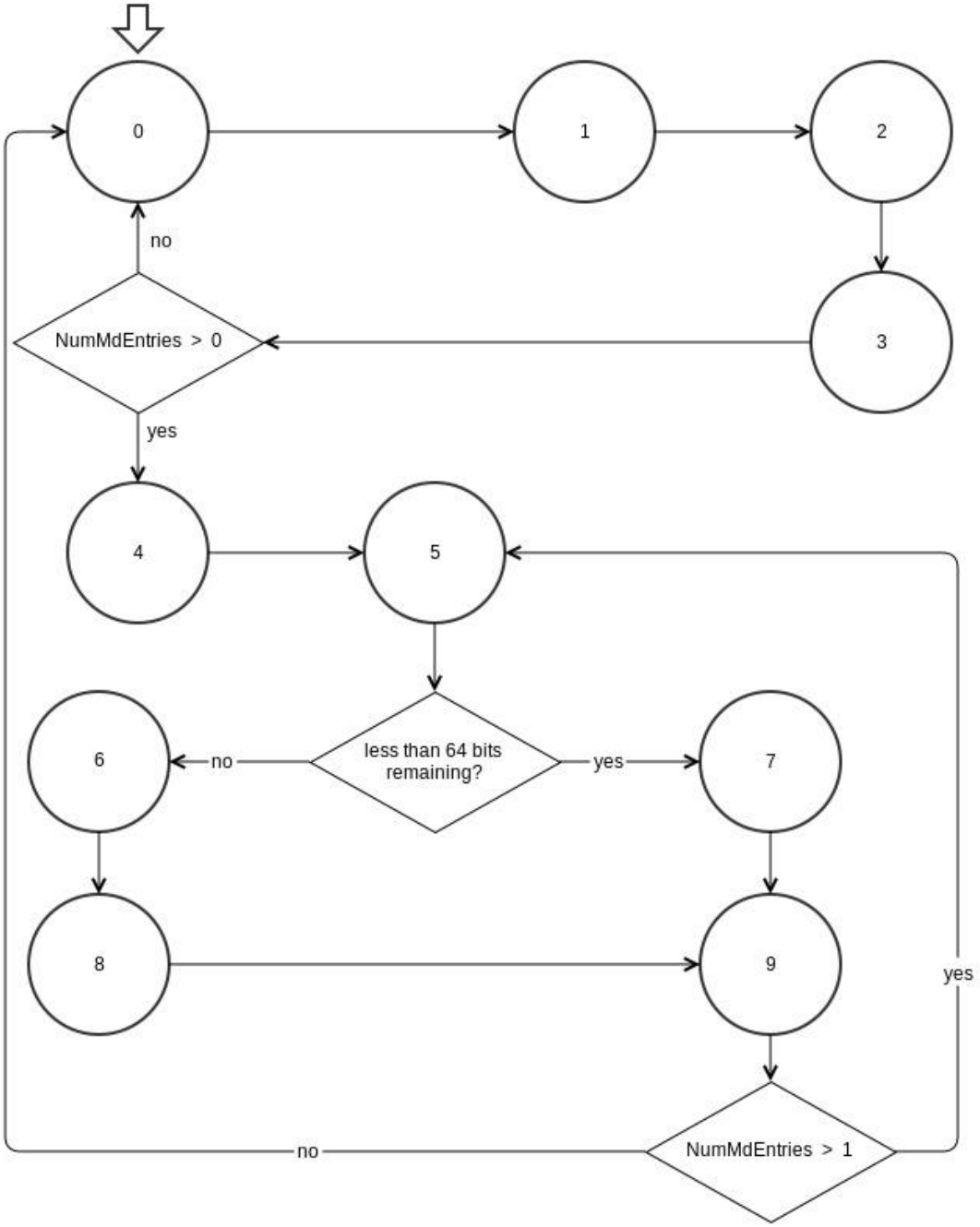

1
2
3
4
5
6
7
clk
ready
valid
startofpacket
endofpacket
empty
3
channel
0
0
0
0
0
error
0
0
0
0
0
data[31:24]
D0
D4
D8
D12
D16
data[23:16]
D1
D5
D9
D13
data[15:8]
D2
D6
D10
D14
data[7:0]
D3
D7
D11
D15

Packetizer


START
start_packet
yes
ETH_SRC_MAC
IP_HDR_S1
IP_HDR_S2
IP_HDR_OPT
UDP_HDR_S1
udp_len == 16'h0008
yes
DONE
no
PAYLOAD
payload_len > 16'd8
yes
no
no

Parser

- Our parser reads in data from the FIFO
- Message headers are always multiples of 64 bits
- But each message can contain multiple entries.
  - Each entry is typically 214 bits (which is not a multiple of 64)
  - This requires us to keep track of the entry offset
- Simple Equation :
  - Offset = (Offset + 40) % 64


0
1
2
3
no
NumMdEntries > 0
yes
4
5
6
no
less than 64 bits remaining?
yes
7
8
9
yes
NumMdEntries > 1
no

FIFO

- Buffer between components
- 64-bits wide
- 256 blocks deep

FIFO

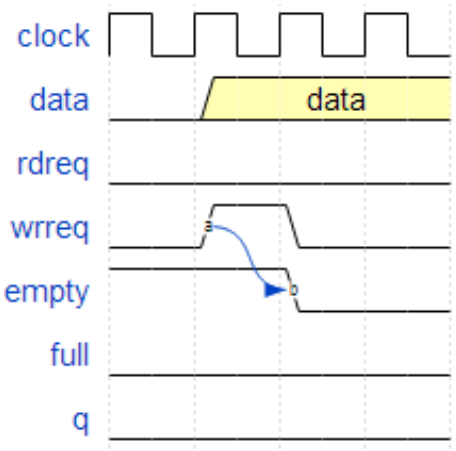
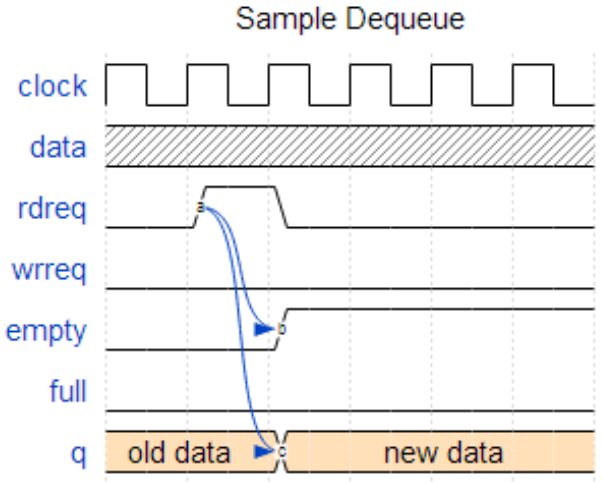
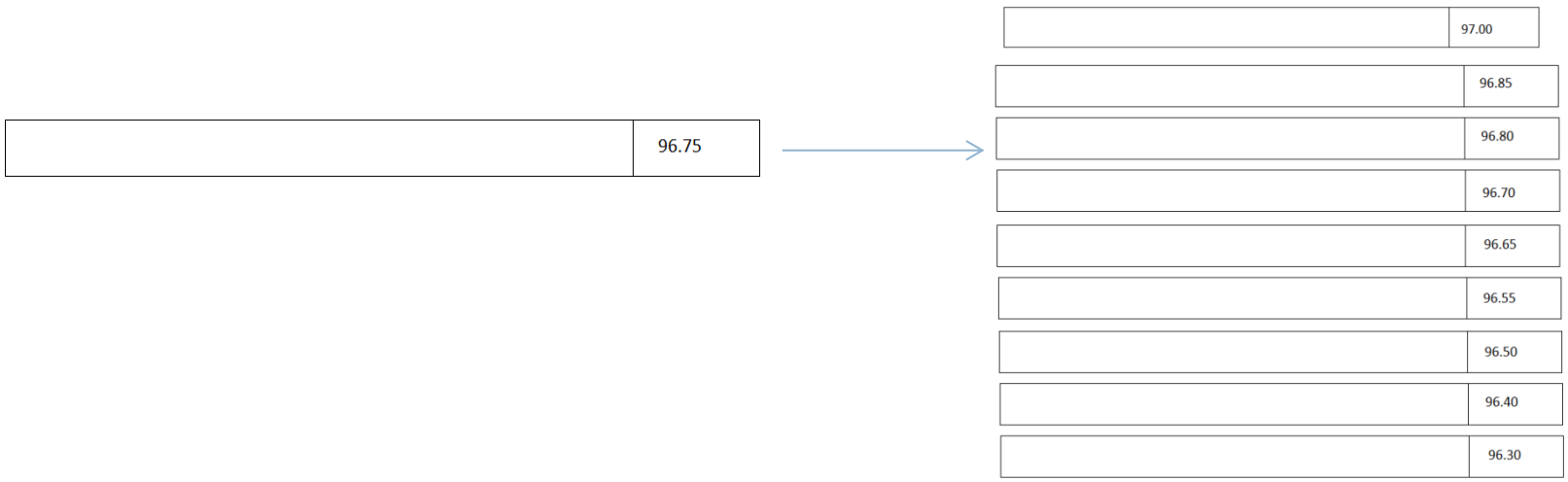


Figure 1



Order Book

- 10 levels of Bid and Ask prices



Challenges

- Oversimplified Initial sample data
- Needed a robust testing suite
- Too much trust in Modelsim
- New data format

Lessons Learned

- More robust Modelsim tests
- The initial design should have been more macro focused
- Clarify confusing financial concepts earlier

Future Work

- Implied Orders
  - Implied “IN”
    - Order In spread from outright
  - Implied “OUT”
    - Order In the outright from spread
  - Our future work on the project aims to be able to read the saved Order Books across different months to create Implied books

Conclusion

- Thanks for all the help!
  - Prof Edwards & Lariviere
  - Qiushi Ding