Scalable 10 G TCP/IP Stack Architecture for Reconfigurable Hardware

David Sidler, Gustavo Alonso · Dept. of Computer Science, ETH Zürich
Michaela Blott, Kimon Karras, Kees Vissers · Xilinx Research
Raymond Carley · Carnegie Mellon University
Motivation

- Data center applications require a TCP/IP stack supporting thousands of connections
- Most implementations on FPGAs are optimized for low-latency and support only a few connections
- Allows straightforward integration of specialized hardware into existing infrastructure
Goal

- 10 Gbps throughput
- Support thousands of concurrent connections
- Scalable and flexible architecture
- Use high-level synthesis (C/C++) to shorten development time
Challenges

- Connection-oriented & stream-based protocol
  - Keep state for each connection
  - Data streams need to be segmented and assembled

- Acknowledged data transfer
  - Keep track of each segment
  - Data buffering is required for each transfer

- Various timers
  - Events/packets might be generated at any time

- Control flow
  - Slow-start, Congestion Avoidance, Delayed Acknowledgment
**TCP Module Architecture**

- Data-flow architecture
- Separation between data paths and state-keeping data structures
- Concurrent access to data structures in BRAM
- External buffers in main memory
- Scalable data structure
TCP Module - SYN Processing

1. SYN packet arrives

- Check if port is open
- Insert/lookup session ID
- Check and update state: CLOSED → SYN-RCVD
- Initialize RX SAR Table
- Event is triggered
- Initialize TX SAR Table
- Set Retransmit-Timer
- Reverse session lookup
1. SYN packet arrives
2. Check if port is open
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
6. Event is triggered
1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
6. Event is triggered
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
6. Event is triggered
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
6. Event is triggered
7. Initialize TX SAR Table
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
6. Event is triggered
7. Initialize TX SAR Table
8. Set Retransmit-Timer
TCP Module - SYN Processing

1. SYN packet arrives
2. Check if port is open
3. Insert/lookup session ID
4. Check and update state: CLOSED → SYN-RCVD
5. Initialize RX SAR Table
6. Event is triggered
7. Initialize TX SAR Table
8. Set Retransmit-Timer
9. Reverse session lookup

Diagram:
- RX App If
- TX App If
- RX Buffer
- TX Buffer
- RX SAR Table
- TX SAR Table
- Event Engine
- Timers
- State Table
- Port Table
- Session Lookup
Data Structures Access Requirements

Minimum packet size is 84 bytes
- For linkrate (8 B/c) processing, access time has to be within 11 cycles
- Data structures are shared between modules
  → The sum of all accesses (RX & TX path) can not exceed 11 cycles

State Table

<table>
<thead>
<tr>
<th>RX Eng [rmw]</th>
<th>TX App [r]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

RX SAR

<table>
<thead>
<tr>
<th>TX Eng [w]</th>
<th>RX App [w]</th>
<th>TX Eng [w]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

TX SAR
Data Structures Access Requirements

Minimum packet size is 84 bytes

- For linkrate (8 B/c) processing, access time has to be within 11 cycles
- Data structures are shared between modules
  → The sum of all accesses (RX & TX path) can not exceed 11 cycles
Data Structures Access Requirements

Minimum packet size is 84 bytes

- For linkrate (8 B/c) processing, access time has to be within 11 cycles
- Data structures are shared between modules
  → The sum of all accesses (RX & TX path) can not exceed 11 cycles
Data Structures Access Requirements

- Minimum packet size is 84 bytes
- For linkrate (8 B/c) processing, access time has to be within 11 cycles
- Data structures are shared between modules
  → The sum of all accesses (RX & TX path) cannot exceed 11 cycles

**State Table**

<table>
<thead>
<tr>
<th>RX Eng [rmw]</th>
<th>TX App [r]</th>
<th>idle</th>
<th>RX Eng [rmw]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

**RX SAR**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>

**TX SAR**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>
# Data Structures Access Requirements

- Minimum packet size is 84 bytes
- For linkrate (8 B/c) processing, access time has to be within 11 cycles
- Data structures are shared between modules
  → The sum of all accesses (RX & TX path) can not exceed 11 cycles

## State Table

<table>
<thead>
<tr>
<th>State Table</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15 cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX Eng [rmw]</td>
<td>TX App [r]</td>
<td>idle</td>
<td>RX Eng [rmw]</td>
<td></td>
</tr>
</tbody>
</table>

## RX SAR

<table>
<thead>
<tr>
<th>RX SAR</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15 cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX Eng [r]</td>
<td>RX App [r]</td>
<td>TX Eng [r]</td>
<td>RX Eng [r]</td>
<td></td>
</tr>
</tbody>
</table>

## TX SAR

<table>
<thead>
<tr>
<th>TX SAR</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15 cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX Eng [r]</td>
<td>TX Eng [r]</td>
<td>TX Eng [w]</td>
<td>TX Eng [w]</td>
<td></td>
</tr>
</tbody>
</table>
Data Structures Access Requirements

Minimum packet size is 84 bytes

- For linkrate (8 B/c) processing, access time has to be within 11 cycles
- Data structures are shared between modules
  → The sum of all accesses (RX & TX path) can not exceed 11 cycles
Evaluation - Setup

- TCP/IP stack running on VC709 evaluation board, Virtex7 XC7VX690T, 2x 4 GB DDR3, 10 G network interface
- 10 servers, 8-Core Intel Xeon E5-2609, 64 GB main memory, Intel 82599 10 G NIC, linux kernel 3.12
- Connected via a Cisco Nexus 5596UP switch
Maximum Segment Size (MSS) is 536 bytes. This leads to a theoretical maximum TCP throughput of 8.76 Gbps.
<table>
<thead>
<tr>
<th>Type</th>
<th>Path</th>
<th>Cycle [6.4 ns]</th>
<th>Time[μs]</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN</td>
<td>SYN-ACK</td>
<td>176</td>
<td>1.1</td>
</tr>
<tr>
<td>Payload [1 B]</td>
<td>RX</td>
<td>170</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>TX</td>
<td>131</td>
<td>0.8</td>
</tr>
<tr>
<td>Payload [536 B]</td>
<td>RX</td>
<td>375</td>
<td>2.4</td>
</tr>
<tr>
<td></td>
<td>TX</td>
<td>402</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Excluding PHY, MAC and application latency
## Evaluation - Resources

<table>
<thead>
<tr>
<th>Resources</th>
<th>FF</th>
<th>LUT</th>
<th>BRAM</th>
<th>Total</th>
<th>% of XC7VX690T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Interface</td>
<td>5,581</td>
<td>5,321</td>
<td>8</td>
<td>83,829</td>
<td>9.6%</td>
</tr>
<tr>
<td>Memory Interface</td>
<td>57,637</td>
<td>43,591</td>
<td>36</td>
<td>67,938</td>
<td>15.6%</td>
</tr>
<tr>
<td>TCP/IP Stack</td>
<td>20,611</td>
<td>19,026</td>
<td>279</td>
<td>323</td>
<td>21.9%</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td><strong>83,829</strong></td>
<td></td>
</tr>
</tbody>
</table>

| % of XC7VX690T Resources | 9.6% | 15.6% | 21.9% |
Conclusion

- Novel architecture for a TCP/IP stack
- Resource requirements scale linearly with number of concurrent connections
- Support for 10,000 concurrent connections
- Control flow features and out-of-order segment processing
- Reduced development time and increased design flexibility due to high-level synthesis
Future Work

- FPGA-based network interface could accelerate other functions such as compression, encryption
- Pushing data analytics and processing closer or into the network
- FPGAs as a microserver platform
Demo Tonight

- Key-value store on the FPGA using TCP/IP stack
- Serving thousands of clients concurrently
- Seamless integration with webserver running Apache and PHP