The *Tick* Programmable Low-Latency SDR System

Haoyang Wu\(^1\), Tao Wang\(^1\), Zengwen Yuan\(^2\), Chunyi Peng\(^3\), Zhiwei Li\(^1\), Zhaowei Tan\(^2\), Boyan Ding\(^1\), Xiaoguang Li\(^1\), Yuanjie Li\(^2\), Jun Liu\(^1\), Songwu Lu\(^2\)
New Requirements on SDR

Drives for Software-Defined Radio

New wireless standard
Trending applications
Protocol evolutions

- Low latency
- Good Programmability
- Good Protocol evolutions
## Current SDR Hard to Achieve Both

<table>
<thead>
<tr>
<th>Platform</th>
<th>Latency over milisecond</th>
<th>Good Programmability</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNU Radio</td>
<td><strong>Real-time 802.11a/g</strong></td>
<td>X</td>
</tr>
<tr>
<td>Sora</td>
<td><strong>Real-time 802.11ac</strong></td>
<td>X</td>
</tr>
<tr>
<td>WARP</td>
<td><strong>Real-time 802.11n</strong></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td><strong>Real-time 802.11ac</strong></td>
<td>X</td>
</tr>
</tbody>
</table>

### Low latency vs. Good Programmability

- **Low latency**
- **Good Programmability**

---

*Hard*
Why?

PHY & MAC differ in their data and control flows

**PHY**
- Computational-intensive data-flow
  - FFT, channel estimation, ...
- Low-latency oriented design
  - ASIC, DSP, FPGA, ...
- In HW, hard to program

**MAC**
- Complex cond./branch control-flow
  - Retransmission, backoff, ...
- Processor for program
  - C programming
- Hard to satisfy MAC timing

Low latency vs. Program mability
Achieving the Best of Both Worlds

It’s not a simple trade-off!

Call for new design
Tick: A New SDR that Achieves Both

Tick uses HW/SW co-design in PHY & MAC

1. How is latency reduced?
   - PHY
     - New parallel pipeline techniques in FPGA (HW)
   - MAC
     - Separate data and control flows (HW)

2. How is programmability retained?
   - PHY & MAC
     - Modular programming (SW)
**Low Latency at PHY: MCDP**

**Problem**

- Different computing workload

**Cause:** single clock limits latency reduction in non-bottleneck modules

<table>
<thead>
<tr>
<th>Clock frequency</th>
<th>Latency cycles</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>45MHz</td>
<td>1</td>
<td>0.02 μs</td>
</tr>
<tr>
<td>45MHz</td>
<td>17</td>
<td>0.37 μs</td>
</tr>
<tr>
<td>430MHz</td>
<td></td>
<td>0.04 μs</td>
</tr>
</tbody>
</table>

**Reduce** 0.33 μs

- Synchronization
- Interleaver
- Bottleneck

 clk 45MHz
Low Latency at PHY: MCDP

Problem

- Different computing workload

MCDP: Multi-Clock-Domain Pipeline

- Each module operates with its own clock

How to match rates at different clock

- Async FIFO

Higher clk frequency $\rightarrow$ Lower latency
Low Latency at PHY: Field-Based Processing

Wireless Frame

Training, Data fields, ...

Whole pipeline latency

Not all fields need to traverse the entire pipeline

Latency: 2.49 μs

Latency: 1.86 μs
Low Latency at PHY: Field-Based Processing

Solution: Field-based processing

- Parallel processing fields in different pipeline stages

Benefit:

- Flexible fields customization with control

802.11a

802.11ac

<table>
<thead>
<tr>
<th>L-header</th>
<th>L-Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L-header</th>
<th>VHT-header</th>
<th>VHT-Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Low Latency at MAC: Accelerator-Rich

Problem

- Data blocks time critical control flow

Accelerator-rich design

- Offload tasks

Separate data & control flows

- Significantly reduce latency
How?

- **Bus-based?**
  - Larger latency per R/W (18%)
  - Repetitive control

Why register-based?

- **Smaller accumulated R/W latency**
  - Reduce 1.5 μs per 50 times R/W
  - Save SIFS time by 10%
- Avoid repetitive control by shared reg.

Solution: Control & status register (CSR)

- CR/SR: avoid rewrite conflict
**Retain Programmability**

**Element-based design: similar to Click**

- Unified interface
- User programmable PE

**Programming abstraction**

- A graph of elements
- XML-based config reduces coding efforts

![Diagram](image)
Implementation

Three **real-time** 802.11 SDR systems with Tick:

- 802.11ac, SISO *(first-ever)*
- 802.11ac, 2x2 MIMO *(first-ever)*
- 802.11a/g, full-duplex

**Public code release soon**
Implementation

COTS Hardware: Xilinx Kintex-7 KC705

Rich Library: supporting 802.11a/g/ac

- Easy-to-use APIs
  - XML/GUI based programming and config

- 28 PHY elements
  - BPSK, QPSK, 16/64/256-QAM
  - Coding rate: 1/2, 2/3, 3/4, 5/6
  - Long GI, short GI ...

- 12 MAC accelerators
  - CRC-32
  - Backoff, Timer
  - MAC header process ...

Tick board (w/o host)
Lessons Learnt from Implementation

Many viable choices need further fine tuning

- **PHY:**
  - Eliminate jitter caused by async FIFO in MCDP

- **MAC:**
  - Ensure critical timing for accelerators via timestamp passing

- **HW/SW co-design:**
  - Choose low-latency interfaces

- ...


Evaluation

Latency Benchmark on 802.11ac SISO, 80MHz

- Reduction compared to a reference design
- **PHY**: sizable reduction (2.2x)
  - MCDP and field-based processing
- **MAC**: over two-order-of-magnitude reduction
  - Decoupling control and data flows

Overall Evaluation

- Latency
- Programming effort
- FPGA resource usage
Evaluation: Latency Benchmark

802.11ac SISO v.s. Ref. design: 80 MHz, 64 QAM, 3/4 rate

PHY latency: **2.2x/1.6x** reduction for RX/TX, MTU frame

- **MCDP**: speed up clock frequency
- **Field-based Proc.**: bypass bottleneck

### SCDP vs. MCDP

<table>
<thead>
<tr>
<th>Latency (μs)</th>
<th>SCDP</th>
<th>MCDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.98</td>
<td>2.01</td>
<td>2.49</td>
</tr>
<tr>
<td>2.3</td>
<td>1.5</td>
<td>1.86</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tick PHY Tx latency reduction **(1.6x)**

MAC latency: **> 497x** reduction (**8938 → 18 μs**), MTU frame

- Reduce data processing bottleneck: CRC + Frame header
**Evaluation: Overall System**

**Overall latency**
- End-to-end latency on host: 334 μs (WARP: 2.5 ms)
- SIFS: 14.7 μs (< 16us, 802.11ac)

**Programmability**
- Reduce 22.8% line-of-codes (LoC)

**Moderate FPGA resource usage**
- 27.9% LUT in Kintex-7 FPGA (65.3% for 2x2 MIMO)
Case Study: 802.11ac 2x2 MIMO

First real-time 802.11ac MIMO prototype on SDR

Performance for 64-QAM

- Max PHY proc.: Tx: 1 Gbps (1113 Mbps); Rx: 632 Mbps
- Satisfy SIFS timing requirements

Coding effort

- 2x2 MIMO
  reuses 14/20
  SISO modules
Case Study: 802.11a/g Full-Duplex

Lightweight coding effort

- Only 8% of the original 802.11a/g codes are modified
- 15/22 elements are unmodified and reused

Throughput

- 245 Mbps
Conclusion

Future SDR calls for lower latency while retaining programmability

Existing designs cannot meet both for new wireless standards

Tick:

- Three-year dev effort at PKU, UCLA
- Explore HW/SW co-design techniques
- First real-time 802.11ac SDR implementation (SISO & MIMO)
- Stimulate community efforts for domain-specific arch design for wireless networking
Thanks

Visit [http://metro.cs.ucla.edu/tick.html](http://metro.cs.ucla.edu/tick.html)

Peking University

- Haoyang Wu, Tao Wang, Zhiwei Li, Boyan Ding, Xiaoguang Li, Jun Liu

UCLA

- Zengwen Yuan, Zhaowei Tan, Yuanjie Li, Songwu Lu

Purdue University

- Chunyi Peng