Real Time Packet Processing with FPGAs
A network security toolbox with encryption features designed for FPGA logic-fabrics

Hack In The Box Dubai – 2018

Matteo Collura
About me

- M.Sc in Micro and Nanotechnologies for Integrated systems
- B.Sc in Electronics Engineering
About me

- Amateur and passionate about IT Security
- Speaker at International security conferences since 2013
About me

- Worked for 6-month Master’s Thesis Project at Knowledge Resources GmbH

www.knowres.ch
Outline

Main goals and motivation:
- Framing the problems
- Why are FPGAs useful to this project

Finding the best configuration:
- Processor-centric approach
- Logic-centric approach

The toolbox explained:
- Firewall-like features
- Encryption scheme
Outline

Results and Demo

Follow me @ Armory to get more insights and see the code
How to deliver confidential information securely?
VPN

Virtual Private Network

Tunnel

CONFIDENTIAL

CONFIDENTIAL

6/84
VPN

Virtual Private Network

Tunnel

Real Time?
Ethernet connections run at 1Gbps commonly

IEEE 802.3 standard
Ethernet connections run at 1Gbps commonly

1 Byte every 8 ns
Von Neumann architecture

CPU speed $\approx$ GHz

Diagram:
- CPU
- Memory
- Network Buffer
- Keyboard
- Hacking Station
Von Neumann Bottleneck

Diagram showing a conveyor belt and a computer with a warning symbol. Diagram also shows a computer system with connections to Memory, Network Buffer, CPU, Keyboard, and Hacking Station.
FPGAs: a Non-Von Neumann class of devices
FPGAs: a Non-Von Neumann class of devices
FPGAs: what’s inside

- CLB (2xSlice)
- 36K BRAM cells (2x18K)
- DSP slice (2xDSP48)
FPGAs: Xilinx fabric families

Performances

Virtex

Kintex

Artix

Spartan
Summarizing

Dual Core Cortex A9 processor

Logic-fabric

Real-Time monitoring of network traffic
Carrier Evaluation Kit
ZYNQ architecture

Programmable System

Programmable Logic
Routing Network Connections

Gigabit Ethernet MAC

Multiplexed I/O
Useful libraries to implement network applications

LWIP (LightWeight IP)
- Good at Network+ (3+) OSI Level
- Dynamic allocation of memory

Too high level of abstraction

The processor is always busy
Useful libraries to implement network applications

- LWIP (Light Weight IP):
  - Good at Network Layer at Level
  - Dynamic allocation of mem

- XEmacPS:
  - Low level library, Data Link (MAC) OSI Level
  - Interacts directly with GEM and DMA

Captures ALL the packets

A bit complicated
Routing Network Connections

Gigabit Ethernet MAC

Multiplexed I/O
Gigabit Ethernet MAC

RX/TX Buffer
- Fills up quickly
- Packet size is overestimated but well organized structure

DMA Engine
- 4-state Finite State Machine
- Driven by ARM processor

BRAM
- “Slow” (Hundreds of MHz)
- Too many blocks will mess up the timing constraints.
Towards a fully hardware implementation

- CLK signals are the only reference
- Every thing is executed in parallel
- NO libraries: just electric signals and logic gates

Timing Constraints
Timing Constraints

Do use memory waypoints wisely!
Timing Constraints

Registers / Flip Flops
Timing Constraints – Dependencies

- "Wire" length
- Fabric
- Clock speed

Timing Constraints
Primary goal: the Ethernet extension cord
Primary goal: the Ethernet extension cord

Ensure continuity of data

Be conservative with time of flight

Workstation #1

Workstation #2
FPGA expansion modules
Ethernet at “ground” level

- Single Data Rate (SDR)
  - One signal per clock cycle

- Double Data Rate (DDR)
  - Two signals per clock cycle
Ethernet at “ground” level

- Double Data Rate (DDR)
- RGMII to GMII
- SDR, 8 bit (1 byte) per cycle
- 1 cycle = 8ns (125 MHz) for 1Gbps link

4 bits per semi-cycle
Conveyor belt model

1 byte every 8 ns  →  125 MHz clock
Conveyor belt model

- PHY → RGMII to GMII
- RGMII to GMII → PHY

24 Registers

1 byte every 8 ns → 125 MHz clock

24 Registers x 8 ns = 192 ns
1 Gbps avg latency ≈ 50 μs

Negligible time of flight!
So far so good! What’s next?

- Firewall-like features
- Deep packet inspection
- Encryption/Decryption system
Network Packets

Information travels via network packets according to IEEE 802.3 standard
Assembly chain model

On-the-flight manipulation of packets
Preamble detector

- Constant pattern “01010101011...1101” → 0x55......0xD5
- Helps clock synchronization

Preamble 7 bytes  SFD 1 byte

Data valid from GMII interface

Initialization of CRC engine

when IDLE =>
  IS_CURRENT_STATE_IDLE <= '1';
  if (DV_IN = '1') then
    counter_next <= "001";
    state_next <= PREAMBLE_DETECT;
  crc_rst <= '1';
  else
    state_next <= IDLE;
  end if;

when PREAMBLE_DETECT =>
  if (PIPELINE(SIZE_OF_PIPE-1) = "01010101") then
    if (counter_reg = "01") then
      if (D_IN = "11010101") then
        state_next <= PREAMBLE_OK;
      crypto_start <= '1';
      else
        counter_next <= (OTHERS => '0');
        state_next <= IDLE;
      end if;
    else
      counter_next <= (OTHERS => '0');
      state_next <= IDLE;
    end if;
  else
    counter_next <= (OTHERS => '0');
    state_next <= IDLE;
  end if;
MAC and EtherType filters

Ensure continuity of data

The assembly chain never stops

Preamble 7 bytes
SFD 1 byte
Beginning constant

Destination MAC address 6 bytes
Source MAC address 6 bytes
Ether Type 2 bytes
Payload 46-1500 bytes
Checksum 4 bytes

Receiver
Sender
Type of content
Content

38/84
MAC and EtherType filters

Search algorithm progress: 0%
MAC and EtherType filters

Blacklist

Search algorithm progress: 0%
MAC and EtherType filters

Blacklist

Search algorithm progress: 0%
MAC and EtherType filters

Destination MAC
MAC and EtherType filters

Start search algorithm

Blacklist

Search algorithm progress: 20%
MAC and EtherType filters

Blacklist

Search algorithm progress: 40%
MAC and EtherType filters

Search algorithm progress: 60%

Blacklist
MAC and EtherType filters

Blacklist

Search algorithm progress: 80%
MAC and EtherType filters

Search algorithm progress: 100%

Blacklist

Matched!
MAC and EtherType filters

Blacklist

Matched!

Drop the whole frame
MAC and EtherType filters – Preparation

- Smart Lookup in memory
  - Arrange the Blacklist properly
  - Dicotomic (Tricotomic) search

- Ascending order pre-sort
  - Handled by the CPU

MAC Address Library

1. 00:01:03:02:30:22
2. 02:23:43:C2:B4:FF
3. Co:CA:Co:1A:00:00

Qsort is useful!
MAC and EtherType filters

Fetch and Match
- The incoming MAC is compared with the blacklist entries

DSP makes fast comparisons
- Up to 48 bit entries
MAC and EtherType filters

Smart Lookup in memory
- Arrange the Blacklist properly
- Dicotomic (Tricotomic) search

Ascending order pre-sort
- Handled by the CPU

Fetch and Match
- The incoming MAC is compared with the blacklist entries

DSP makes fast comparisons
- Up to 48 bit entries

Choose wisely the variable size
MAC address

00:10:FA:CC:Co:1A

MAC address – 48 bits

Organizationally Unique Identifier (OUI) – 24 bits

Do save time!

Compare 24-bit operands

DSP will evaluate 2 comparisons per time
Search Algorithm version 1 – tricotomic

x > n₁?  ⇒  x – n₁ > 0
x > n₂?  ⇒  x – n₂ > 0
Search Algorithm version 1 – tricotomic

MAC OUI taken from network

MAC OUIs stored in memory

x > n1? \rightarrow x - n1 > 0
x > n2? \rightarrow x - n2 > 0
MAC filter state machine

Start

DSP_Idle_Dest
  cycle = 1

DSP_Recv
  cycle++

If cycle = 5

DSP_Recv
  cycle++

If matched AND Dest list = 1

Drop

DSP_Recv
  cycle = 1

DSP_Recv
  cycle = 1

DSP_Recv
  cycle = 1

DSP_Recv
  cycle = 1

If matched AND Src list = 1

Done
MAC and EtherType filters configuration

Set up MAC filter for Source and Destination address:

Source : Packets coming from the chosen MAC are flushed
Destination : Packets directed towards the chosen MAC are flushed

Would you like to add all the inserted MACs to:
(1) Source blacklist
(2) Destination blacklist
(3) Both Source and Destination blacklist
(4) I would like to choose one by one

Please type per each MAC:
(1) --> Source blacklist
(2) --> Destination blacklist
(3) --> Both Source and Destination blacklist

0 | 00:0A:35:XX:XX:XX DESTINATION
1 | 0A:0A:0A:XX:XX:XX SOURCE
2 | 12:23:45:XX:XX:XX SOURCE & DESTINATION
3 | 12:31:23:XX:XX:XX

• Up to 81 entries → $3^4$
• Source and/or Destination MAC attribute is defined per each entry
Demo: MAC filtering
Demo: MAC filtering
Demo: MAC filtering
MAC filter conclusions

80* ns to browse the whole memory

The process ends always on the same checkpoint

The checkpoint is defined by the state machine

* IMPROVABLE TO 64 ns

80 ns = 10 bytes * 8 ns
EtherType filter

Ensure continuity of data

The assembly chain never stops

Preamble 7 bytes
SFD 1 byte
Destination MAC address 6 bytes
Source MAC address 6 bytes
Ether Type 2 bytes
Payload 46-1500 bytes
FCS 4 bytes

Beginning constant
Receiver
Sender
Type of content
Content
Checksum
EtherType filter configuration

CURRENT LIST:

<table>
<thead>
<tr>
<th>#</th>
<th>EtherType_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x0800</td>
</tr>
</tbody>
</table>

Do you want to add a new EtherType ID to your filter?
You still have 29 entries left
(y)es, please    (n)o, that's enough for today

Ok, fine.

Set up EtherType filter to work as a Whitelist or Blacklist

Whitelist: Only the packets with matching EtherTypes are accepted
Blacklist:  Packets with matching EtherTypes are flushed
Would you like to set up a Blacklist or a Whitelist?
(1) Blacklist
(2) Whitelist

• Up to 30 entries
• Blacklist/Whitelist mode
Search Algorithm version 2

Blacklist entries [0:29]

14 – 15

\[ n_1 \quad n_2 \]

6 – 7

22 – 23

2 – 3

10 – 11

18 – 19

26 – 27

14 – 15

6 – 7

22 – 23

2 – 3

10 – 11

18 – 19

26 – 27

x > n_1 \Rightarrow x - n_1 > 0

x > n_2 \Rightarrow x - n_2 > 0
Encryption Environment

Ensure continuity of data
The assembly chain never stops
Point-to-point Encryption system
PoC level

Preamble 7 bytes
SFD 1 byte
Destination MAC address 6 bytes
Source MAC address 6 bytes
Ether Type 2 bytes
Payload 46-1500 bytes
FCS 4 bytes

Beginning constant Receiver Sender Type of content Content Checksum
Encryption configuration

Do you want to setup the Crypto Engine? Press (y)es or (n)o
Alright!

What is the port exposed the outer world?
Type 1 (parallel wrt board) or 2 (perpendicular)

You chose port #2

It's time to choose a password. Maximum characters allowed: 32
Type your password here below. The ENTER key will validate your password

******
Please, type your password once again to confirm.
******

Congratulations: it's time to generate the seed now
Encryption schema
Encryption schema
Encryption schema
Encryption schema
Encryption architecture

The following slides can seriously hurt Crypto specialists
Encryption architecture

Password input by user

32-bit hash: Fowler Noll Vo

32 8-bit subsets are cycled
Encryption architecture

JohnDo3

Password input by user

0x67D3E282

32-bit hash: Fowler Noll Vo
Encryption architecture

0x67D3E282

8-bit subsets are cycled

ob01100111
ob11001111
ob10011111

ob011001111010011110001010000010
Encryption architecture

PS

Crypto Box

J 3 0
0 J 3
H 0 J
0 H
D n H
n D
3 0 D
Encryption architecture – “Signature”

- Checksum will be recomputed
- Packet size is preserved
- MAC headers are preserved (indeed)

- Preamble
- SFD
- Destination MAC address
- Source MAC address
- Ether Type
- Payload
- Checksum

??
Encryption architecture – “Signature”

- Checksum will be recomputed
- Packet size is preserved
- MAC headers are preserved (indeed)

EtherType
**Encryption architecture – “Signature”**

**Match EtherTypes with another ID**

**Only IPv4 is encoded atm**

<table>
<thead>
<tr>
<th>EtherType</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0800</td>
<td>Internet Protocol version 4 (IPv4)</td>
</tr>
<tr>
<td>0x0806</td>
<td>Address Resolution Protocol (ARP)</td>
</tr>
<tr>
<td>0x0842</td>
<td>Wake-on-LAN[^9]</td>
</tr>
<tr>
<td>0x22F3</td>
<td>IETF TRILL Protocol</td>
</tr>
<tr>
<td>0x22EA</td>
<td>Stream Reservation Protocol</td>
</tr>
<tr>
<td>0x6003</td>
<td>DECnet Phase IV</td>
</tr>
<tr>
<td>0x8035</td>
<td>Reverse Address Resolution Protocol</td>
</tr>
<tr>
<td>0x809B</td>
<td>AppleTalk (EtherTalk)</td>
</tr>
<tr>
<td>0x80F3</td>
<td>AppleTalk Address Resolution Protocol (AARP)</td>
</tr>
<tr>
<td>0x8100</td>
<td>VLAN-tagged frame (IEEE 802.1Q) and Shortest Path Bridging</td>
</tr>
<tr>
<td>0x8137</td>
<td>IPX</td>
</tr>
<tr>
<td>0x8204</td>
<td>QNX Qnet</td>
</tr>
<tr>
<td>0x860D</td>
<td>Internet Protocol Version 6 (IPv6)</td>
</tr>
<tr>
<td>0x8808</td>
<td>Ethernet flow control</td>
</tr>
<tr>
<td>0x8809</td>
<td>Ethernet Slow Protocols[^11] such as the Link Aggregation</td>
</tr>
</tbody>
</table>

[^9]: Wake-on-LAN is a protocol that allows a computer to be awakened from a power-saving state by receiving a special packet on the network.

[^10]: The IEEE 802.1aq standard specifies the use of VLANs for network redundancies and fault tolerance.

[^11]: Ethernet Slow Protocols include Link Aggregation Control Protocol (LACP) among others.
Encryption architecture – “Signature”
Encryption architecture – “Signature”
Encryption architecture – “Signature”

Encrypted IPv4 packets have custom EtherType

The board automatically detects and decrypts such packets
Demo: Encryption

Test workstation #1

Test workstation #2

Ethernet

Ethernet 2

UART

JTAG

RJ45

50-pin ZIF

HS3 connector
Demo: Encryption

--------------------------------
-----| Crypto Engine Setup |-----
--------------------------------
Do you want to setup the Crypto Engine? Press (y)es or (n)o
Alright!

What is the port exposed the outer world?
Type 1 (parallel wrt board) or 2 (perpendicular)

You chose port #2

It's time to choose a password. Maximum characters allowed: 32
Type your password here below. The ENTER key will validate your password

********
Please, type your password once again to confirm.
********

Congratulations: it's time to generate the seed now
Demo: Encryption
Demo: Encryption
Demo: Decryption

Test workstation

Ethernet2

UART

JTAG

Plaintext

Ciphertext

“Echoed” Ciphertext

Plaintext

“FPGA”

“FPGA”

“!&$%”

“!&$%”
Demo: Decryption
Checksum generation

- **Preamble**: 7 bytes
- **SFD**: 1 byte
- **Destination MAC address**: 6 bytes
- **Source MAC address**: 6 bytes
- **Ether Type**: 2 bytes
- **Payload**: 46-1500 bytes
- **FCS**: 4 bytes
- **Beginning constant**: Receiver
- **Sender**: Type of content
- **Content**: Checksum
Checksum generation

CRC_MUX_OUT <= CRC_OUT_o when crc_append = '1' else PIPELINE(14);
CRC_MUX_IN <= CRYPTO_OUT_o when is_crypto_detected_reg = '1' else PIPELINE(15);

Flag raised when the last payload’s byte crosses PIPELINE(14)
Flag raised when the custom IPv4 sequence is detected
Checksum generation

CRC logic

CRC_IN_3 <= crc_out(31 downto 24) when crc_append = '0' else (OTHERS => '0');
CRC_IN_2 <= crc_out(23 downto 16) when crc_append = '0' else CRC_OUT_3;
CRC_IN_1 <= crc_out(15 downto 8) when crc_append = '0' else CRC_OUT_2;
CRC_IN_0 <= crc_out(7 downto 0) when crc_append = '0' else CRC_OUT_1;
Checksum generation

CRC logic

CRC_OUT_i: ff_gen
generic map (NBITS => DEPTH)
port map(
  CLK => CLK,
  RST => RST,
  DIN => CRC_IN_i,
  DOUT => CRC_OUT_i
); (OTHERS => '0');

32-bit checksum stored in 4 chunks
Moving towards a higher level...

- Preamble: 7 bytes
- SFD: 1 byte
- Destination MAC address: 6 bytes
- Source MAC address: 6 bytes
- Ether Type: 2 bytes
- Payload: 46-1500 bytes
- FCS: 4 bytes

Beginning constant: Receiver: Sender: Type of content: Content: Checksum
Moving towards a higher level...

Byte 0x18
Moving towards a higher level...

Killing ICMP packets!

when ICMP_KILL =>
  if (is_icmp = '1') then
    state_next <= DROP;
  else
    state_next <= WAIT_TILL_DONE;
  end if;
Achievements

• Transparent modification of data

• Real-time equivalent

• Pipelining allows multiple firewall-feature implementation

• The whole time of flight (192 ns) is below avg latency

• P2P Encryption
Future Upgrades

• Filter upper layer protocols (IP, etc.)

• Deep Packet Inspection (custom word lookup in payload, etc.)
Questions?
Thank you for your attention!

Follow me on LinkedIn: linkedin.com/in/matteocollura