Architectural Exploration of AES-128 in NetFPGA

Presented by:
Tariq Bashir Ahmad (UMASS Amherst)
Fabricio Goncalves (Telefonica Brazil)

Stanford University
August 2, 2013

http://NetFPGA.org
• Introduction and scope of the project
• Architecture
• Simulation
• Build flow
• Results
• Conclusion and Future Work
• Acknowledgements
We’ll focus here
Limitations of XOR / One Time Pad

- If key is short and repeatable, it is weak encryption
- If key is as long as message, then key management becomes cumbersome
Replacing XORING by AES-128

- AES-128 is NIST standard for symmetric cryptography
- Is used heavily in software and hardware cryptography
- Has various modes of operation for processing data at rest and data in flight
Replacing XORING by AES-128

Plain text
Key
Start
Reset

AES-128 Encryption
clk

Cipher text
Done

Output Port Lookup
AXI Lite Bus
Slave
AES
Master
AXI Lite Bus
Endianness
Flow inside NetFPGA

- CPU 3 RxQ
- MAC 3 RxQ
- CPU 2 RxQ
- MAC 2 RxQ
- CPU 1 RxQ
- MAC 1 RxQ
- CPU 0 RxQ
- MAC 0 RxQ
- MAC 3 TxQ
- MAC 2 TxQ
- MAC 1 TxQ
- MAC 0 TxQ
- CPU TxQ

08/02/2013

Tariq Bashir Ahmad
Fabricio Goncalves
Implementation

AXI Slave

FIFO

Get packet payload

Repeat loop

AES-FIFO

AES-Encrypt

AXI Master
Implementation choices for AES-128

Iterative

Fast Iterative

Add round key

Subbytes

Shift rows

Mix columns

Add round key

Reg

Visit http://opencores.org/project,aes-encryption to see my implementations of AES-128 Encryption cores
Unrolled

- add round key
- subbytes
- Shift rows
- Mix columns
- add round key

Pipelined

- add round key
- Reg
- subbytes
- Shift rows
- Mix columns
- add round key
- Reg
- subbytes
- Shift rows
- Add round key
- Reg
- add round key
- Reg
Live Simulation Demo
Xilinx Build Flow

1. RTL Verilog
2. Synthesis
3. Translation
4. Mapping
5. P&R
Implementation Statistics (after P&R)

<table>
<thead>
<tr>
<th></th>
<th>Iterative</th>
<th>Fast Iterative</th>
<th>Unrolled</th>
<th>Pipelined</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUTS</td>
<td>40169</td>
<td>41004</td>
<td>45881</td>
<td>45762</td>
</tr>
<tr>
<td>Registers</td>
<td>57401</td>
<td>58150</td>
<td>63769</td>
<td>63159</td>
</tr>
<tr>
<td>Frequency</td>
<td>117 MHz</td>
<td>125 MHz</td>
<td>28.2 MHz</td>
<td>160 MHz</td>
</tr>
</tbody>
</table>

Timing Not MET

Timing MET For Line Rate

08/02/2013
Can we do Better?

- Yes
- Use Multiple Encryption Engines in Parallel
- We parallelize Iterative and Pipelined Implementations by adding another AES-128 Engine
Implementation Statistics

<table>
<thead>
<tr>
<th></th>
<th>Iterative Parallel</th>
<th>Pipelined Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUTS</td>
<td>40476</td>
<td>51890</td>
</tr>
<tr>
<td>Registers</td>
<td>58300</td>
<td>69202</td>
</tr>
<tr>
<td>Frequency</td>
<td>160 MHz</td>
<td>160 MHz</td>
</tr>
</tbody>
</table>

Timing MET For Line Rate

AXI Slave

FIFO

AES-FIFO

AES-Encrypt

AES-Encrypt

Repeat loop

AXI Master

08/02/2013
Conclusion and Future Work

- This is just the beginning!
- Add Decryption
- Replacing AES-ECB by GCM-AES
Acknowledgements

• Adam Covington, Gianni Antichi and M. Shahbaz
• Wajahat Qadeer

• Guy Hutchison, Awais Nemat and Amer Haider
• Professors Maciej Ciesielski & C.M. Krishna
References

- http://opencores.org/project,aes_core
- http://opencores.org/project,aes-encryption
- http://opencores.org/project,srdydrdy_lib