eXtended eXternal Benchmarking eXtension (XXBX)

Jens-Peter Kaps

Cryptographic Engineering Research Group (CERG)
http://cryptography.gmu.edu
Department of ECE, Volgenau School of Engineering,
George Mason University, Fairfax, VA, USA

SPEED-B 2016
Outline

1. Introduction & Motivation
2. XXBX Hardware
3. XXBX Software
4. Conclusions and Future Work
Introduction & Motivation
Introduction

XXBX is a tool for benchmarking algorithms on microcontrollers that cannot efficiently run their own operating system and compilers.

It uses the following Metrics:
- Throughput - cycles per byte
- ROM usage - bytes
- RAM usage - bytes
- Power - milliwatts
IoT promises a dramatic increase in devices, many will be microcontrollers or SOCs.

32-bit microcontrollers are projected to take lead over 8/16-bit by 2018.

51% of all 32-bit microcontrollers were ARM based in 2012.
SUPERCOP

- Benchmarks many implementations of many primitives across multiple operations on multiple hardware platforms.
- Supports environments capable of running Linux and hosting a compiler.
- Series of shell scripts and C test harnesses, and comprehensive collection of algorithm primitive implementations.
- Verifies correct execution of implementations and times cycles required per byte processed.
- Does not measure ROM and RAM usage or power consumption.

http://bench.cr.yp.to/supercop.html
eXternal Benchmarking eXtension -extends SUPERCOP
Automated testing on real microcontrollers
Compatibility with SUPERCOP algorithm collection ("algopacks") and output format
Low cost hardware and software
Our contribution to original XBX was to port it to the MSP430 platform and provide results for SHA-3 finalists.
Measures ROM and RAM usage. Does not measure power consumption.
XBX Components

Figure: Block Diagram of XBX components
XBX Limitations

- Only supports hash functions
- No power measurements
- Does not use cycle counters
- Benchmarking takes a long time because embedded platforms are slow.
  - Simulation can run faster

Figure: AVR-NET-IO ATmega32 board with MSP430
FELICS

- Fair Evaluation of Lightweight Cryptographic System
- Targeted for lightweight block ciphers
- Uses simulation when available else real hardware
- Supports Atmel AVR, MSP 430, ARM Cortex-M3
- Measures RAM, ROM, execution time.

https://www.cryptolux.org/index.php/FELICS
Expand XBX through
- adding AEAD support,
- adding power measurement,
- replace XBH in order to facilitate power measurement,
- adding resuming partial runs, and
- avoiding breaking when Link-Time Optimization is enabled

⇒ eXtended eXternal Benchmarking eXtension (XXBX)
XXBX Hardware
XBX Harness (XBH)

**Requirements**

- Ethernet to connect to XBS
- \( I^2C \) to connect to XBD
- General purpose I/O to get computation start/stop from XBD and to reset XBD
- Capability to measure execution time on XBD
- Capability to facilitate power measurements.

Hardware under initial consideration

- Raspberry Pi
  - very powerful and inexpensive, however, needs external ADC
- Beaglebone
  - even more powerful but costs more
Linux-based boards very fast, but do not easily meet real-time requirements

- Realtime extension PREEMPT_RT broke MMC driver for SD card with OS.
- Jitter for timing measurements will be in the tens of microseconds.
- Xenomai required reimplementing drivers
New XBH: EK-TM4C129XL

- Tiva Connected Launchpad chosen when it became available
  - ARM Cortex-M4F, 120 MHz with ethernet connectivity.
  - 256 kB of SRAM and 1 MB of ROM
  - Dual 12-bit ADCs capable of 2 MSPS
  - Easily worked on bare metal without an OS
  - Realtime OS (FreeRTOS) available including drivers
  - Inexpensive
  - Boosterpack headers

<table>
<thead>
<tr>
<th></th>
<th>XBH</th>
<th>new XBH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>ATmega32</td>
<td>ARM Cortex-M4F</td>
</tr>
<tr>
<td>Clock</td>
<td>16 MHz</td>
<td>120 MHz</td>
</tr>
<tr>
<td>RAM</td>
<td>2 kB</td>
<td>256 kB</td>
</tr>
<tr>
<td>ROM</td>
<td>32 kB</td>
<td>1 MB</td>
</tr>
<tr>
<td>Price</td>
<td>20 EUR</td>
<td>20 USD</td>
</tr>
</tbody>
</table>
Tiva C Connected Launchpad
XBX Devices under test (XBD)

- **MSP-EXP430F5529LP**
  - 16-bit MSP430,
  - clockable to 25 MHz,
  - 10 kB SRAM and 128 kB flash

- **EK-TM4C123GXL**
  - 32-bit ARM Cortex M4F,
  - clockable to 80 MHz,
  - 32 kB SRAM and 128 kB flash
Future XBDs (soon)

MSP-EXP430FR5994
- 16-bit MSP430
- clockable to 16 MHz
- 8 kB SRAM and 256 kB FRAM
- AES accelerator

EK-TM4C129EXL
- 32-bit ARMv7E-M, Cortex M4F
- clockable to 120 MHz
- 256 kB SRAM and 1 MB flash
- AES accelerator
Future XBDs (a little bit later)

- STM Nucleo-F091RC
  - 32-bit ARMv6-M, Cortex M0
  - clockable to 48 MHz,
  - 32 kB SRAM and 256 kB flash

- STM Nucleo-F103RB
  - 32-bit ARMv7-M, Cortex M3
  - clockable to 72 MHz,
  - 20 kB SRAM and 128 kB flash
Future XBDs (even later)

Homemade
- ATMEGA1284-PU, 8-bit AVR,
- clockable to 20 MHz,
- 16 kB SRAM and 128 kB flash

chipKIT uC32
- 32-bit PIC32M3xx, MIPS 32,
- clockable to 80 MHz,
- 32 kB SRAM and 512 kB flash
# XBX-XBD and XXBX-XBD Comparison

## XBX Supports

<table>
<thead>
<tr>
<th>Device</th>
<th>Manuf.</th>
<th>Chip</th>
<th>Processor</th>
<th>CPU</th>
<th>Bus</th>
<th>f</th>
<th>OS</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp.Board</td>
<td>Atmel</td>
<td>ATmega1284P</td>
<td>ATmega1284P</td>
<td>AVR</td>
<td>8-bit</td>
<td>20 MHz</td>
<td>bare</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TI</td>
<td>MSP430FG4618</td>
<td>MSP430FG</td>
<td>MSP430X</td>
<td>16-bit</td>
<td>8 MHz</td>
<td>bare</td>
<td>$117</td>
</tr>
<tr>
<td>Artila M501</td>
<td>Atmel</td>
<td>AT91RM9200</td>
<td>ARM920T</td>
<td>ARMv4T</td>
<td>32-bit</td>
<td>180 MHz</td>
<td>Linux</td>
<td>$116</td>
</tr>
<tr>
<td>NSLU2</td>
<td>Intel</td>
<td>IXP420</td>
<td>XScale</td>
<td>ARMv5TE</td>
<td>32-bit</td>
<td>266 MHz</td>
<td>Linux</td>
<td>$90</td>
</tr>
<tr>
<td></td>
<td>TI</td>
<td>LPC1114</td>
<td>ARM Cortex-M0</td>
<td>ARMv6-M</td>
<td>32-bit</td>
<td>50 MHz</td>
<td>bare</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TI</td>
<td>LM3S811</td>
<td>ARM Cortex-M3</td>
<td>ARMv7-M</td>
<td>32-bit</td>
<td>120 MHz</td>
<td>bare</td>
<td></td>
</tr>
<tr>
<td>BeagleBoard</td>
<td>TI</td>
<td>DM3730</td>
<td>ARM Cortex-A8</td>
<td>ARMv7-A</td>
<td>32-bit</td>
<td>1 GHz</td>
<td>Linux</td>
<td>$89</td>
</tr>
<tr>
<td>FritzBox</td>
<td>TI</td>
<td>AR7</td>
<td>MIPS32</td>
<td>4KEc</td>
<td>32-bit</td>
<td></td>
<td>Linux</td>
<td>$300</td>
</tr>
</tbody>
</table>

## XXBX Supports (soon)

<table>
<thead>
<tr>
<th>Board</th>
<th>Manuf.</th>
<th>CPU</th>
<th>ISA</th>
<th>Bus</th>
<th>f</th>
<th>HW</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homemade</td>
<td>Atmel</td>
<td>ATmega1284P</td>
<td>AVR</td>
<td>8-bit</td>
<td>20 MHz</td>
<td>$10.00</td>
<td></td>
</tr>
<tr>
<td>MSP-EXP430F5529</td>
<td>TI</td>
<td>MSP430F</td>
<td>MSP430X</td>
<td>16-bit</td>
<td>25 MHz</td>
<td>$12.99</td>
<td></td>
</tr>
<tr>
<td>MSP-EXP430FR5994</td>
<td>TI</td>
<td>MSP430FR</td>
<td>MSP430X</td>
<td>16-bit</td>
<td>16 MHz AES</td>
<td>$15.99</td>
<td></td>
</tr>
<tr>
<td>EK-TM4C123GXL</td>
<td>TI</td>
<td>ARM Cortex M4F</td>
<td>ARMv7E-M</td>
<td>32-bit</td>
<td>80 MHz</td>
<td>$12.99</td>
<td></td>
</tr>
<tr>
<td>EK-TM4C129EXL</td>
<td>TI</td>
<td>ARM Cortex M4F</td>
<td>ARMv7E-M</td>
<td>32-bit</td>
<td>120 MHz AES</td>
<td>$24.99</td>
<td></td>
</tr>
<tr>
<td>NUCLEO-F091RC</td>
<td>STM</td>
<td>ARM Cortex M0</td>
<td>ARMv6-M</td>
<td>32-bit</td>
<td>48 MHz</td>
<td>$10.33</td>
<td></td>
</tr>
<tr>
<td>NUCLEO-F103RB</td>
<td>STM</td>
<td>ARM Cortex M3</td>
<td>ARMv7-M</td>
<td>32-bit</td>
<td>72 MHz</td>
<td>$10.33</td>
<td></td>
</tr>
<tr>
<td>chipKIT uC32</td>
<td>Microchip</td>
<td>PIC32MX3xx</td>
<td>MIPS32 M4K</td>
<td>32-bit</td>
<td>80 MHz</td>
<td>$29.95</td>
<td></td>
</tr>
</tbody>
</table>
Current Sensing: Low Side

- Measured by sensing voltage drop across a small shunt resistor
- Can be single-ended
- Does not have to deal with common mode voltage
- I/O pins could provide alternate ground paths causing measurement errors.

\[ I = I_S = \frac{V_S}{R_S} \]

if \( V_S \ll V_D \) then \( P_D \approx V_{CC} \cdot I \)
Current Sensing: High Side

- Directly measures current delivered by voltages source
- Multiple ground paths do not need to be accounted for
- No issues with ground loops
- Must handle common-mode voltage

\[ V_S = V_{CC} - V_D \quad I = I_S = \frac{V_S}{R_S} \quad P_D = V_D \cdot I \]

If \( V_S \ll V_D \) then \( P_D \approx V_{CC} \cdot I \)
Current Measurement

- Utilize ADCs on Launchpad
  - Input range: 0 – 3.3 V
  - These ADCs have input low-impedance, must be buffered
  - Need amplification, as shunt drop is low
Current Measurement

Utilize ADCs on Launchpad

- Input range: 0 – 3.3 V
- These ADCs have input low-impedance, must be buffered
- Need amplification, as shunt drop is low

Shunt Resistor $R_S = 1\Omega$

Assume: $I_S = I_D = 290\mu A$
$V_S = 290\mu A \cdot 1\Omega = 290\mu V$
ADC resolution = $\frac{3.3V}{2^{12}} = 0.8\text{mV}$
ADC Result: 0
Current Measurement

- Utilize ADCs on Launchpad
  - Input range: 0 – 3.3 V
  - These ADCs have input low-impedance, must be buffered
  - Need amplification, as shunt drop is low

<table>
<thead>
<tr>
<th>Shunt Resistor $R_S = 1\Omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assume: $I_S = I_D = 290\mu A$</td>
</tr>
<tr>
<td>$V_S = 290\mu A \cdot 1\Omega = 290\mu V$</td>
</tr>
<tr>
<td>ADC resolution = $\frac{3.3 V}{2^{12}} = 0.8mV$</td>
</tr>
<tr>
<td>ADC Result: 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shunt Resistor $R_S = 1k\Omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_S = 290 \cdot 10^{-6} A \cdot 1 \cdot 10^3 \Omega = 290mV$</td>
</tr>
<tr>
<td>ADC Result: 360</td>
</tr>
<tr>
<td>But now $V_D = V_{CC} - V_S = 3.01V$ and not 3.3V!</td>
</tr>
</tbody>
</table>
Current Measurement

- Utilize ADCs on Launchpad
  - Input range: 0 – 3.3 V
  - These ADCs have input low-impedance, must be buffered
  - Need amplification, as shunt drop is low
- Considered putting op-amp in front of ADCs
  - Requires precision resistor network
  - More parts to deal with

**Shunt Resistor $R_S = 1\Omega$**

- Assume: $I_S = I_D = 290\mu A$
- $V_S = 290\mu A \cdot 1\Omega = 290\mu V$
- ADC resolution = $\frac{3.3V}{2^{12}} = 0.8mV$
- ADC Result: 0

**Shunt Resistor $R_S = 1k\Omega$**

- $V_S = 290 \cdot 10^{-6} A \cdot 1 \cdot 10^3 \Omega = 290mV$
- ADC Result: 360
- But now $V_D = V_{CC} - V_S = 3.01V$ and not 3.3V!
Current Sensor

Use current sense amplifier in front of ADC - specifically INA225

- Allows high side measurement
- Selectable gain to adjust for different target devices in different ranges (25-200)
- Buffered output to deal with low ADC input impedance
- 250 kHz bandwidth
XBX Power Measurement (XBP)

- Fits between XBH and XBD
- Contains I²C pull-ups
- Space for power regulator
- Eagle files in git
XXBX Software
Original XBX ran bare metal and used TCP/IP stack from Ulrich Radig’s webserver-uvm. Use FreeRTOS with lightweight IP (lwIP) instead of bare-metal.

- Easier multitasking- OS handles task switching instead of doing it explicitly
- TCP/IP runs in background while application executes
- Easier to write network code - lwIP socket API can be used
- lwIP and FreeRTOS port included in examples provided by Texas Instruments
- Upgraded TI’s versions of both to newer versions
- TiwaWare driver library and lwIP freely licensed, not examples

Hardware abstracted away
XBH code differences to older XBH

- Only support TCP/IP for XBS ↔ XBH comms
- Add length prefix to delimit messages
- Power measurements streamed to XBS in realtime
  **Future:** Processing on XBH, so only maximum and average power are sent to XBS.
- Only support I²C for XBH ↔ XBD
- Uses XBH ↔ XBD protocol from original XBH
XBH code tasks

Lowest to highest priority:

1. lwIP TCP/IP
2. XBH Server – handles communication to XBS, cues commands for XBD
3. XBH command execution and XBD communication (same priority as XBH server)
4. Ethernet Receive/Transmit – sends transmit and receive descriptors to lwIP
5. Power Measurement – woken up periodically by timer interrupt to perform measurements and enqueuing them to the XBH server task.

Execution time is measured through interrupts.
Highest to lowest priority:

- 0: Unused
- 1: Timer Wraparound
- 2: Timer Capture
- 3: Max FreeRTOS SysCall Priority
- 3: Power Sample Timer
- 4: Watchdog
- 5: Unused
- 6: Unused
- 7: FreeRTOS kernel
Timing Measurements

- 16-bit timer TC to capture timing flag from XBD.
- Need additional timer TW at same rate to get interrupts when timer wraps around.
- Higher priority TW counts wraps (w).
- TW can interrupt processing of TC ISR!
- Maximum time (t) is 35.8 seconds (64-bit value) at 120 MHz.

Diagram:

```
<table>
<thead>
<tr>
<th>Timer Flag (from XBD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer Capture</td>
</tr>
<tr>
<td>TC ISR</td>
</tr>
<tr>
<td>Timer Wraparound</td>
</tr>
<tr>
<td>TW ISR</td>
</tr>
</tbody>
</table>
```

- Timer Flag (from XBD)
- Timer Capture
- TC ISR
- Timer Wraparound
- TW ISR

Equation: 
\[ t = w \times \text{tp} + (tc2 - tc1) \]

Equation: 
\[ w = w + 1 \]
XBD Software

- Largely the same as original XBX
- Replaced self-test implementation with SUPERCOP’s
- Refactor out hash-specific code to make it easier to add other operations
- Add AEAD payload processing
  - XBH doesn’t know anything about the operation under test, just routes it blindly to XBD from XBS.
  - XBD must know what is being in run order to unpack parameters and messages
XXBX Benchmarking System (XBS) Software

- Completely rewritten in Python 3
- Now supports resuming runs if run fails and XBS crashes due to hung hardware
- Results now stored in a SQLite database
- Dropped unused features such as KAT-file verification and loading XBD in formats other than IHEX
- Builds performed in parallel
Conclusions and Future Work
Conclusions

- XBX extended to include support for AEAD
- Enables benchmarking of power
- Allows resuming partial runs
### SUPERCOP, XBX, XXBX Feature Comparison

<table>
<thead>
<tr>
<th></th>
<th>SUPERCOP</th>
<th>XBX</th>
<th>XXBX</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target Platform</strong></td>
<td>Desktop/Server</td>
<td>Embedded</td>
<td>Embedded</td>
</tr>
<tr>
<td><strong>Speed Benchmarks</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Memory Benchmarks</strong></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>ROM Benchmarks</strong></td>
<td>N/A</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Supports AEAD</strong></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Power Benchmarks</strong></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Remaining work

- Integrate the power measurement hardware
- Perform a full benchmarking run on all AEAD and hash algorithms that have implementations that can run
- Extend platform support to AVR and MIPS
- Documentation
- Use cycle counters when available
- Make sure XBD CPU does not have memory wait states
- Option to run with and without cache on XBD
- Check constant time variability
- Measure idle power
Thanks for your attention.

https://crytography.gmu.edu/xxbx
https://github.com/GMUCERG/xbx