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

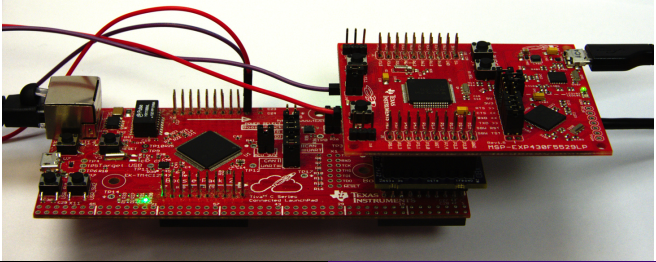
SPEED-B 2016

Outline

Introduction & Motivation

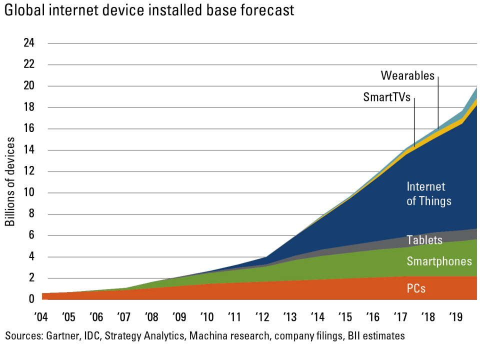
Introduction

- 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



Motivation

- 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

- System for Unified Performance Evaluation Related to Cryptographic Operations and Primitives.
- 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`

XBX

- 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

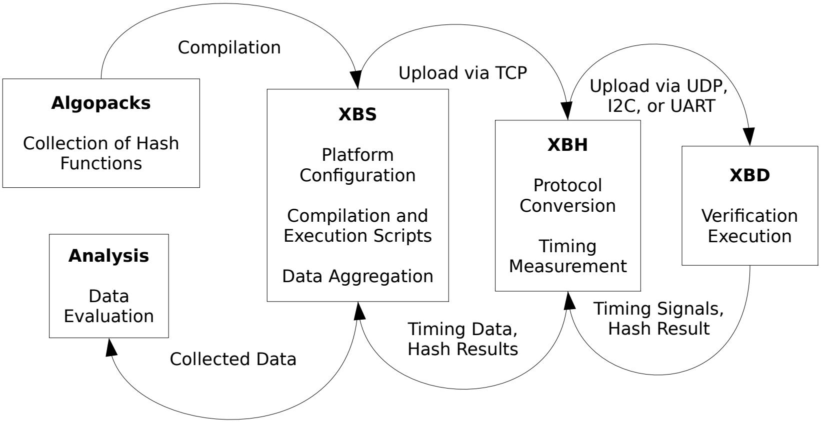


Figure: Block Diagram of XBX components

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

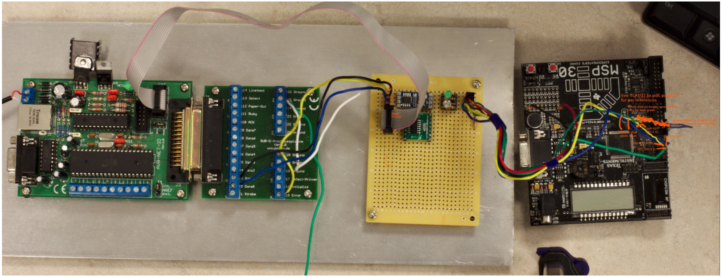


Figure: AVR-NET-IO ATmega32 board with MSP430

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

Design Goals

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

$\Rightarrow$ eXtended eXternal Benchmarking eXtension (XXBX)

XXBX Hardware

XBX Harness (XBH)

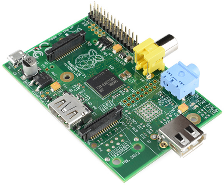
Requirements

- 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.

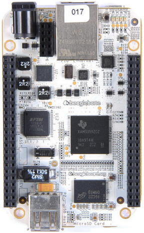
Hardware under initial consideration

- Raspberry Pi
  - very powerful and inexpensive, however, needs external ADC
- Beaglebone
  - even more powerful but costs more

Linux Based Boards



Raspberry Pi



BeagleBone

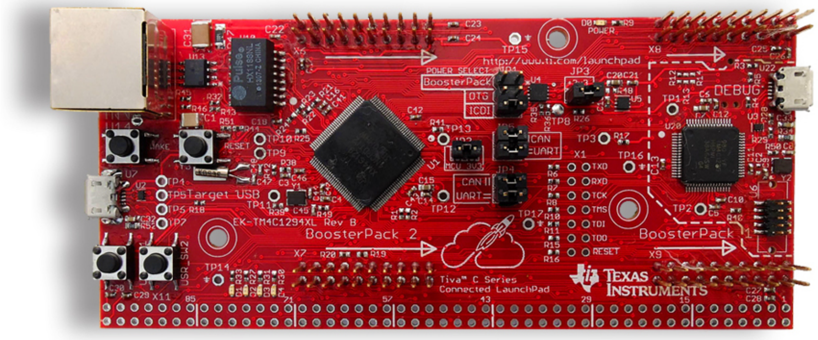
Linux-based boards very fast, but do not easily meet real-time requirements

- 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

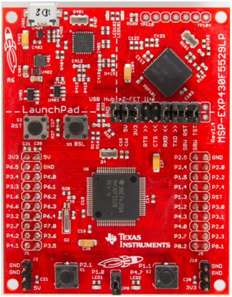
- Tiva Connected Launchpad chosen when it became available
  - ARM Cortex-M4F, 120 MHz with ethernet connectivity.
  - 256 kB of SRAM and 1 MB of ROM
  - Dual 12-bit ADCs capable of 2 MSPS
  - Easily worked on bare metal without an OS
  - Realtime OS (FreeRTOS) available including drivers
  - Inexpensive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

	XBH	new XBH
Architecture	ATmega32	ARM Cortex-M4F
Clock	16 MHz	120 MHz
RAM	2 kB	256 kB
ROM	32 kB	1 MB
Price	20 EUR	20 USD



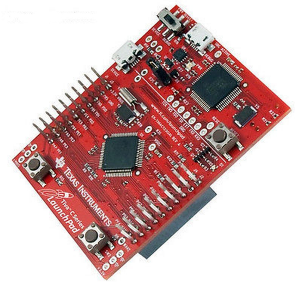
Tiva C Connected Launchpad

XBX Devices under test (XBD)



MSP-EXP430F5529LP

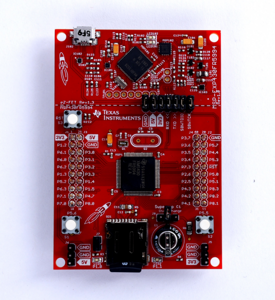
- 16-bit MSP430,
- clockable to 25 MHz,
- 10 kB SRAM and 128 kB flash



EK-TM4C123GXL

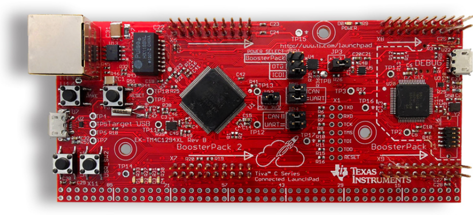
- 32-bit ARM Cortex M4F,
- clockable to 80 MHz,
- 32 kB SRAM and 128 kB flash

Future XBDs (soon)



MSP-EXP430FR5994

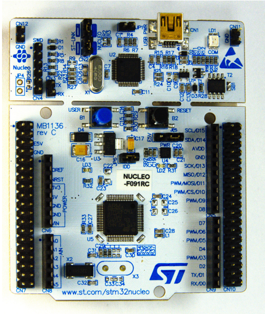
- 16-bit MSP430
- clockable to 16 MHz
- 8 kB SRAM and 256 kB FRAM
- AES accelerator



EK-TM4C129EXL

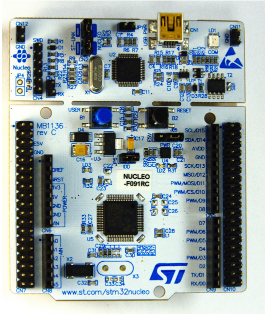
- 32-bit ARMv7E-M, Cortex M4F
- clockable to 120 MHz
- 256 kB SRAM and 1 MB flash
- AES accelerator

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



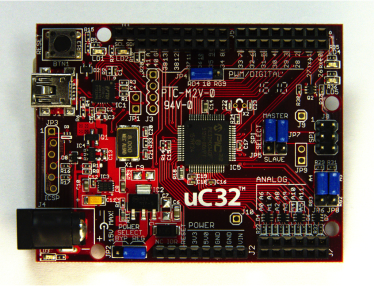
STM Nucleo-F103RB

- 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

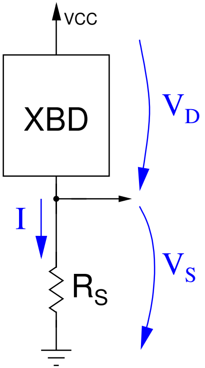
Device	Manuf.	Chip	Processor	CPU	Bus	*f*	OS	Price
	Atmel	ATmega1284P	ATmega1284P	AVR	8-bit	20 MHz	bare	
Exp.Board	TI	MSP430FG4618	MSP430FG	MSP430X	16-bit	8 MHz	bare	$117
Artila M501	Atmel	AT91RM9200	ARM920T	ARMv4T	32-bit	180 MHz	Linux	$116
NSLU2	Intel	IXP420	XScale	ARMv5TE	32-bit	266 MHz	Linux	$90
	IXP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

XXBX Supports (soon)

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

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.



$$I = I_S = \frac{V_S}{R_S}$$

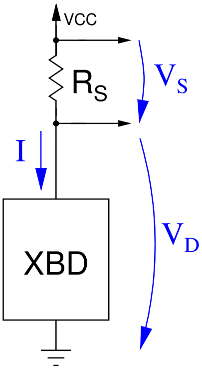
if $V_S << V_D$ then $P_D \approx V_{CC} \cdot I$

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 $V_S << V_D$ 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: $I_S = I_D = 290\mu\text{A}$

$V_S = 290\mu\text{A} \cdot 1\Omega = 290\mu\text{V}$

ADC resolution $= \frac{3.3\text{V}}{2^{12}} = 0.8\text{mV}$

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

Shunt Resistor $R_S = 1\Omega$

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

Shunt Resistor $R_S = 1\text{k}\Omega$

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

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

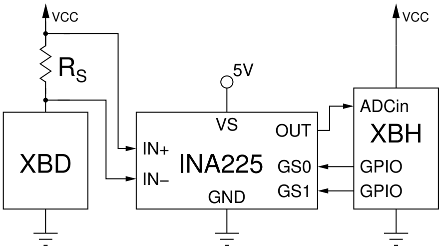
Shunt Resistor $R_S = 1\mathrm{k}\Omega$

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

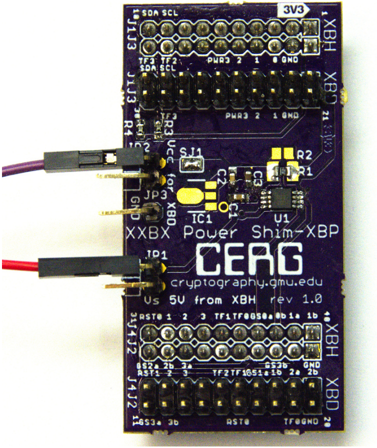
Current Sensor

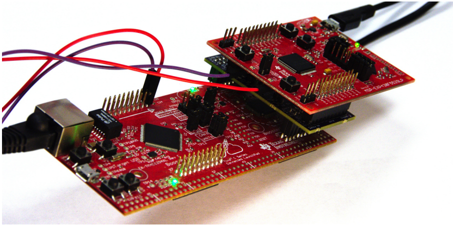
Use current sense amplifier in front of ADC - specifically INA225

- 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 I^2C pull-ups
- Space for power regulator
- Eagle files in git

XXBX Software

XBH 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 $\leftrightarrow$ 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 I^2C for XBH $\leftrightarrow$ XBD
- Uses XBH $\leftrightarrow$ 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
2. XBH command execution and XBD communication (same priority as XBH server)
3. Ethernet Receive/Transmit – sends transmit and receive descriptors to lwIP
4. 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.

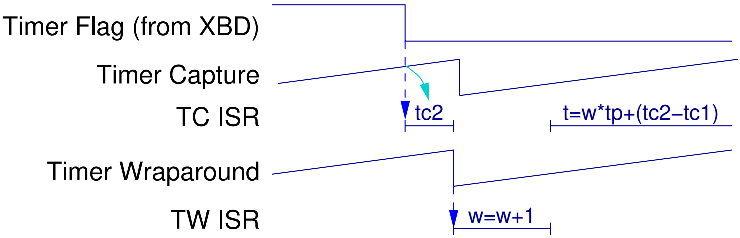
XBH Interrupts

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.



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

	SUPERCOP	XBX	XXBX
Target Platform	Desktop/Server	Embedded	Embedded
Speed Benchmarks	✓	✓	✓
Memory Benchmarks		✓	✓
ROM Benchmarks	N/A	✓	✓
Supports AEAD	✓		✓
Power Benchmarks			✓

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 variablility
- Measure idle power

Thanks for your attention.

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