Maximize energy efficiency in a normally-off system using NVRAM

- Stéphane Gros
- Yeter Akgul
Value proposition

- Subsystem Processor IP (non volatile)
- Software IP (drivers, apps)
- Memory IP (non volatile compilers)
- eVaderis-powered Customer chip

- Power (active & standby)
- Flexibility (hardware & software)
- Costs (density & process)
Positioning

Autonomy

10 yrs

1 wk

Performance Intelligence

Simple Meter

Secure NFC

Geo

Industrial Meter

Pacemaker

Health Monitoring

Video Surveillance

Ear for industry

Wearable (consumer)

Lifetime Power limitation

Miniaturization Low cost

Computing Amount of Data

May 31, 2017
Summary

THE COMPANY

THE CONTEXT

THE TECHNOLOGY

THE SYSTEM

THE CO-DEVELOPMENT

CONCLUSION
Performances (computing, amount of data)

Autonomy (battery life)

Cell phone dilemma
Increasingly powerful but … … less and less portable!
MCU level

Costly And Complex

Memories, up to 96% of area, 50% of power consumption for advanced designs
Board level

Connected Object
State of the art

Connected Object
Powered by eVaderis

Energy

wireless transmission

Off-Chip Processing-Storage

standby

On-Chip Processing-Storage

10-100X less data send to the cloud

eVaderis-powered Customer Chip

Extended lifetime (10X)

Savings ~10X

Performances

Autonomy

10%
Summary

THE COMPANY

THE CONTEXT

THE TECHNOLOGY

THE SYSTEM

THE CO-DEVELOPMENT

CONCLUSION
## Memory comparison 1/3

<table>
<thead>
<tr>
<th></th>
<th>eFlash/eEE</th>
<th>eSRAM</th>
<th>eSTTRAM</th>
<th>eRRAM (Ox)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Non-volatile</strong></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Compatible with front end logic</strong></td>
<td>No (FinFET, FDSOI)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Scalable</strong></td>
<td>Sub-28nm ?</td>
<td>Yes (size?)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Cell size (density F2)</strong></td>
<td>10-30 (&gt;30 for eEE)</td>
<td>60-80 (HD/LP)</td>
<td>10-30</td>
<td>10-20</td>
</tr>
<tr>
<td><strong>Access time</strong></td>
<td>10-100ns (not destructive)</td>
<td>&lt;1ns (not destructive)</td>
<td>2-10ns (not destructive)</td>
<td>10ns (not destructive)</td>
</tr>
<tr>
<td><strong>Write/erase process</strong></td>
<td>byte/block/page level</td>
<td>bit level</td>
<td>bit level</td>
<td>bit level</td>
</tr>
<tr>
<td><strong>Write/erase time</strong></td>
<td>1us/10ms (erase) (page, byte level)</td>
<td>&lt;1ns</td>
<td>2-10ns (bit level)</td>
<td>10-50ns (bit level)</td>
</tr>
<tr>
<td><strong>Endurance</strong></td>
<td>$10^5$-$10^6$</td>
<td>&gt;$10^{16}$</td>
<td>$10^{10}$-$10^{15}$</td>
<td>$10^7$-$10^9$</td>
</tr>
<tr>
<td><strong>Array standby current</strong></td>
<td>0</td>
<td>1-10µA/Mb(25C)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Memory comparison 2/3

- Storage energy @ constant data (1Mb/1s)

- MRAM vs. SRAM:
  - 100x more energy for writing in MRAM
  - 0 leakage at standby mode (MRAM off)
  - ~4x less area MRAM
Memory comparison 3/3

### eNOR FLASH/EE
- Endurance
- Density
- Performance
- Power efficiency
- Process cost efficiency

### OTP/MTP
- Endurance
- Density
- Performance
- Power efficiency
- Process cost efficiency

### eSRAM/CMOS
- Endurance
- Density
- Performance
- Power efficiency
- Process cost efficiency

---

Versatile eSTTRAM
- Endurance
- Density
- Performance
- Power efficiency
- Process cost efficiency
Many process and architectures tradeoff are possible
Summary

THE COMPANY

THE CONTEXT

THE TECHNOLOGY

THE SYSTEM

THE CO-DEVELOPMENT

CONCLUSION
Non-Volatile System

CPU
- Fetch
- Decode
- Execute
- Write-Back

Code (eFlash)

Data (SRAM)

Storage (external Flash)

PMU
- Timers

UART
- GPIO
- I2C
- SPI

Debug
- DMA

System bus (AHB)

Peripheral bus (APB)

Bridge

May 31, 2017
Non-Volatile System

Flexible data/code partitioning

CPU
- Fetch
- Decode
- Execute
- Write-Back

HP controller
- Code
- Data

System bus (AHB)

Bridge

Peripheral bus (APB)

UART
- GPIO
- I2C
- SPI

IOs

Storage
- (external Flash)

Debug
- DMA

PMU
- Timers
Non-Volatile System

Flexible data/code partitioning

CPU
- Fetch
- Decode
- Execute
- Write-Back

HP controller
- Code & Data

System bus (AHB)

Bridge

Periperal bus (APB)

IOs

UART
- GPIO
- I2C
- SPI

HD controller
- Storage

Debug
- DMA

PMU
- Timers
Non-Volatile System

- Instant ON/OFF with context saving
- Instant context switching
- Flexible data/code partitioning
- Code & Data
- Over the air/on site low power code update and calibration
- UART
- GPIO
- I2C
- SPI
- IOs
- Debug
- DMA
- HD controller
- Storage
- PMU
- Timers
- Distributed NV registers
- Zero leakage counter and timestamp

May 31, 2017
NVP & NVS advantages

- Increase battery life
  - Instant ON/OFF $\rightarrow$ minimize SoC boot energy loss
  - Normally-OFF $\rightarrow$ no idle power consumption (power down)
- Intermittent power supply support
  - Harvesting
  - Avoid rollback
- Simplify sleep modes $\rightarrow$ simplify code dev.
- Simplify code maintenance
  - Flexible memory partitioning
  - Update over-the-air
- Multi-application support
  - Instant context switching
  - Reduced context saving overhead

May 31, 2017
NV CPU and System

Power ON / Reset

Check BOOT_ST

INST_ON

From NV (memory, register)

INST_ON_IRQ

All or partial processing states?
Include configuration states

CURRENT

BOOT_ST=NULL

INST_OFF_IRQ

BOOT_ST=INST_ON

INST_ON_IRQ

BOOT_ST=INST_ON

Architecture states (I/D mem, PC, RF).
Microarchitecture states (pipe, RoB, queue, map table…)

INST_OFF_IRQ

BOOT_ST=INST_ON

Save Context

To NV (memory, register)
Selective/Compression

Power OFF

Power OFF

May 31, 2017
SW retention state

Update boot status

CPU

Boot

Boot status
Reset
Program
Exec stack
State backup
QMEM
Stack pointer

Update boot status

Write context into the stack (GPRs, SPR etc…)

+ Minimal HW impact
- Requires an API \(\rightarrow\) more validation
- Requires memory protection
- Energy cost state dependent
- Duration state dependent \(\rightarrow\) Not suitable for RT applications
- Complications with caches
HW Retention state

• Existing approaches:
  – Retention FF: most efficient but largest
  – Scan-chain based approach:
    • uses existing scan-chain hardware
    • minimal area cost but slow and more power consuming
    • Complex (flow/dft)
  – Drowsy state retention:
    • freeze and reduce voltage
    • lowest area cost but least efficient
    • Complex power management (analog)

• Limitations:
  – Leakage overhead
  – Not power failure tolerant
  – Area overhead
  – HW impact + controller \(\rightarrow\) extra verification
**Our approach:** NV-FF (extend Retention FF)

**Advantages over existing approach:**
- Limited area overhead
- No extra leakage
- Power failure tolerant

**Partial vs Full replacement**
- Reduces area and energy overhead
- Requires detailed knowledge of the design
  → Find best trade-off

---

**HW Retention state**
Summary

THE COMPANY

THE CONTEXT

THE TECHNOLOGY

THE SYSTEM

THE CO-DEVELOPMENT

CONCLUSION
«But how to efficiently exchange data between the two technologies with near instant-on/off capabilities?»
NVSRAM

CPU

SRAM

STT-MRAM

Standby

Dynamic

Static

May 31, 2017
NVS RAM level

Hybridization at bit cell level
- Massives parallels (save/restore) transfers (fast)
- Energy-efficient transfers
- The largest area / SRAM bit cell/array adaptation
- Impact on SRAM read/write power and latency

Hybridization at IP level
- Significant parallels (save/restore) transfers (fast)
- Energy-efficient transfers
- The smallest area
- No impact on SRAM

Hybridization at system level
- Limited parallels (save/restore) transfers (slow)
- Not energy-efficient transfers / complex routing
- Acceptable area
- No impact on SRAM
NVSRAM architecture

System Bus: access to SRAM only (hidden STTRAM), CPU/logic frequency

Internal Wide Bus:
- Transfers between SRAM and STTRAM
- Asynchronous
- Low power
- High bandwidth
NVSRAM vs SRAM vs MRAM vs MRAM+SRAM (sys) energy

- IoT
- Smartphone

Energy (a.u.)
- Memory size: 2048kB
- Cache size: 128kB
- SRAM: 1pA/cell
- MRAM: 100µA@100ns
- MRAM+SRAM (sys): <0.5%

Active time (%)
- 100x to 1000x higher endurance than MRAM

May 31, 2017
Summary

THE COMPANY

THE CONTEXT

THE TECHNOLOGY

THE SYSTEM

THE CO-DEVELOPMENT

CONCLUSION
Conclusion

THE TECHNOLOGY
• Non-Volatile
• Power efficient
• Flexible

THE SYSTEM
• Normally-OFF
• Instant ON/OFF

THE CO-DEVELOPMENT
• Hybrid memory
• Increase efficiency

Find the best trade-off for the application
Thank you.