An Instruction-Based NAND Flash Processor Unit (NPU)

Yilei WANG,
Architect, Memblaze
Controller capability review

- **Flash Protocol standard**
  - Unified ONFI 4.0, toggle 3.0
  - Vendor specified protocols
- **Geometry organization**
  - Block/page geometry
- **Timing difference**
- **Complex of NAND management/ NAND validation requirement**
Hardware controller proposal

- **Effective area cost**
  - 2k FPGA LUTs in Memblaze solution

- **Performance optimized**
  - Most effective NAND scheduling

- **Fixed pre-designed functions**
  - Buffer geometry/commands/scheduling

- **Poor capability for variety/future NAND device**
Software controller proposal

• **Heave area cost**
  • 7k FPGA LUTs+ 8KB memory in Memblaze solution
  • Unused CPU instruction/block

• **Flexibility and control optimized**
  • Soft scheduling/NAND sequence
  • Soft sequence/NAND command

• **Strong capability for varity/future NAND device**
  • Undocumented command/operation; variable NAND geometry
Balance point of controller design

• **Feature demands**
  - Configurable timing/sequence/NAND addressing/data frame size/scheduling algorithm
  - HW scheduled and thread switch for LUN management
  - Other accelerate function as buffer/timer/power management

• **Feature undemands**
  - Competing performance/interrupt/stack management
  - Other CPU feature
Defines a NPU- specs

• **Instruction set define**
  • General instruction as jump, bitwise, arithmetic operate, call, return…
  • NAND timing/sequence generation instruction
  • LUN thread/scheduling instruction
  • User command parser/queue and data Buffer management instruction

• **Hardware define**
  • NAND IO generation, atomic operation for NAND command/address/data
  • Buffers/scheduler/user command Queues
  • Instruction ram/general registers/status register/thread stack
Defines a NPU- arch

**Arch define**

- IO logic & PHY, atom operation for NAND command/address/data
- Thread FSM, running the instruction sequence as normal CPU as step of fetch/decode/execute from instruction RAM
- Scheduler, a programmable Thread switcher, and Scheduler.
- User command queue and data buffer, the user interface for the controller
- GR/Field/Stack, the local storage and thread stack
Defines a NPU- thread switches

- **Thread switches**
  - A thread is assigned to the LUN operation
  - Thread could be yielded at any time depending on the instruction programming. E.g., an erase thread could be yielded while status read
  - The yielded thread releases the thread FSM. thus the thread FSM could serve another LUN operations
NPU Programming model
Memblaze NPU design

- Memblaze NPU features
  - Micro-code programmable interface behavior & timing controller
  - Micro-code programmable NAND operation
  - Programmable NAND PHY ADDR define
  - Programmable data frame length
  - Programmable CE configuration
  - FSM-> 32 instructions,
  - FSM-> full Turing machine
  - Multiple command queue for each LUN
  - Scheduler-> 2D programmable, mixed HPQ/WRR
  - Scheduler-> target enable/disable
  - Timer for each LUN
  - Hard Macro for LUN supports
  - Hard Macro for buffer size/length configuration
  - Hard Macro for bus size configuration
  - Hard Macro for instruction RAM size configuration
  - Hard Macro for each register address configuration
  - Hard Macro for each register default value configuration
## Memblaze NPU design

<table>
<thead>
<tr>
<th>Feature</th>
<th>Memblaze Hardware NAND Flash controller</th>
<th>Memblaze software NAND Flash controller</th>
<th>Memblaze NAND flash Processor Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>-</td>
<td>FPGA soft CPU</td>
<td>Thread FSM</td>
</tr>
<tr>
<td>Pipeline</td>
<td>-</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Interrupt</td>
<td>-</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>General register</td>
<td>16*32</td>
<td>1*32</td>
<td></td>
</tr>
<tr>
<td>Instruction</td>
<td>Standard</td>
<td>Full Customized</td>
<td></td>
</tr>
<tr>
<td>Instruction Memory</td>
<td>32KByte</td>
<td>8KByte</td>
<td></td>
</tr>
<tr>
<td>Data memory</td>
<td>64KByte (share for No)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logic</td>
<td>2k LUTs</td>
<td>7k LUTs</td>
<td>4k LUTs</td>
</tr>
<tr>
<td>Coding</td>
<td>fix function</td>
<td>C/C++</td>
<td>Assembly</td>
</tr>
<tr>
<td>Scheduler</td>
<td>fix</td>
<td>pure software</td>
<td>Hardware</td>
</tr>
<tr>
<td>Scheduler Algorithm</td>
<td>WRR</td>
<td>any</td>
<td>2D mixed WRR+HP</td>
</tr>
<tr>
<td>Hardware thread Support</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Number of threads support</td>
<td>16</td>
<td>16</td>
<td>Hard parameter</td>
</tr>
<tr>
<td>Timer</td>
<td>No</td>
<td>1 timer</td>
<td>Yes, per-thread</td>
</tr>
<tr>
<td>Hardware Buffer manage</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Power Management Support</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Hardware Power loss manage</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Soft bit read</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Channel Lock</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Target lock</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PHY control</td>
<td>fixed</td>
<td>Fixed</td>
<td>Microcode</td>
</tr>
<tr>
<td>Test debug</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User command</td>
<td>fixed</td>
<td>full flexibility</td>
<td>full flexibility</td>
</tr>
</tbody>
</table>

Memblaze Pblaze3 with Hardware NAND controller

Memblaze flash validation platform with software NAND controller (only 1 channel valid)

Memblaze flash validation platform with NPU NAND controller (full 4 channel valid)
Summary

- NPU is the just right controller for NAND
- Right for any NAND in feature and any undisclosed NAND Vendor operations
- Right for both performance and FPGA/ASIC area cost to hit the balance point.
- Right for offload the task of main SSD processors
- Right for the precision NAND control, specially, for fine tuning in different strategy of enterprise applications
NAND Flash Processor Unit (NPU)

Thank you!!

yilei.wang@Memblaze.com