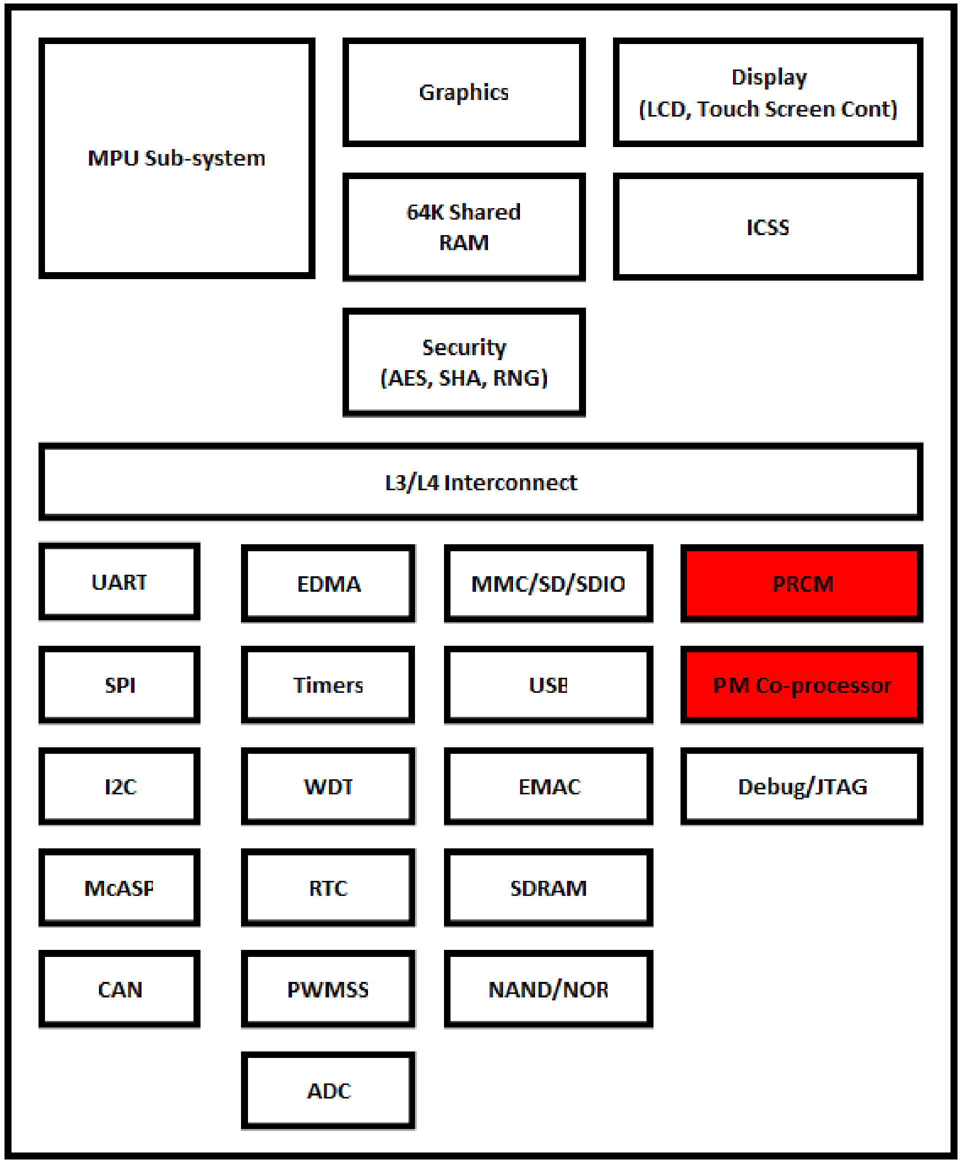
Power Management in Embedded Linux with a Co-Processor

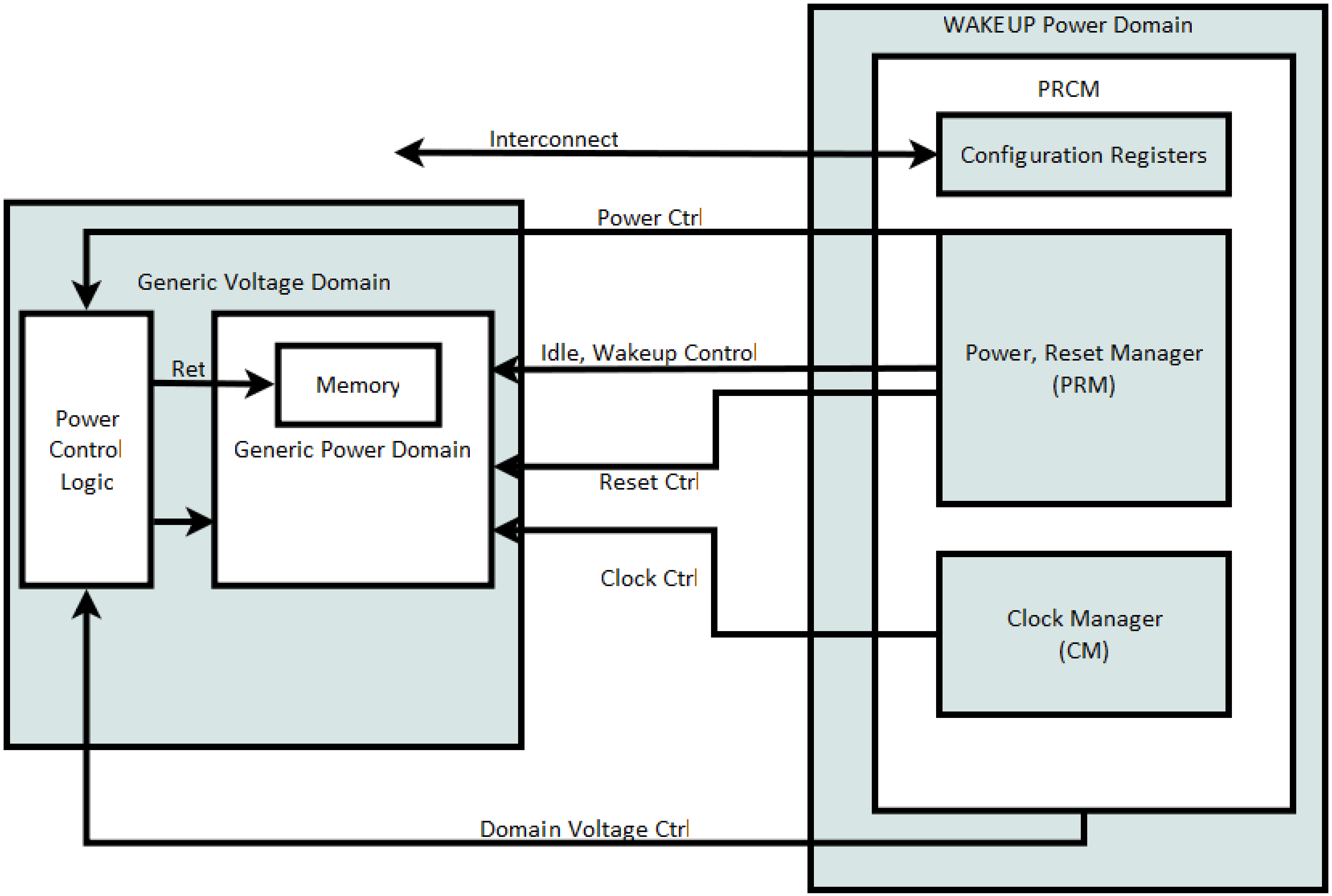
Vaibhav Bedia (vaibhav.bedia@ti.com)

Texas Instruments

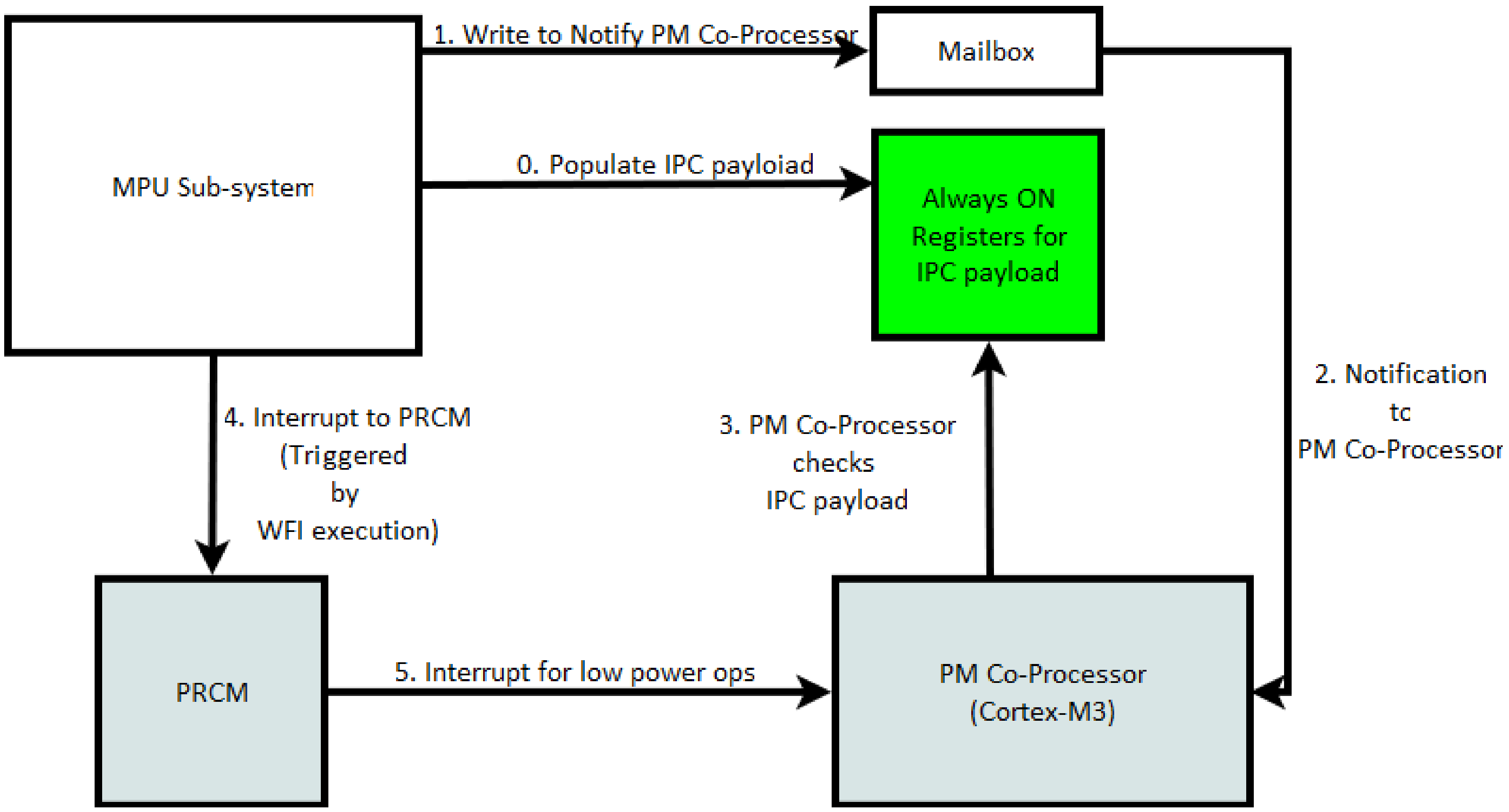
AM335x System Block Diagram



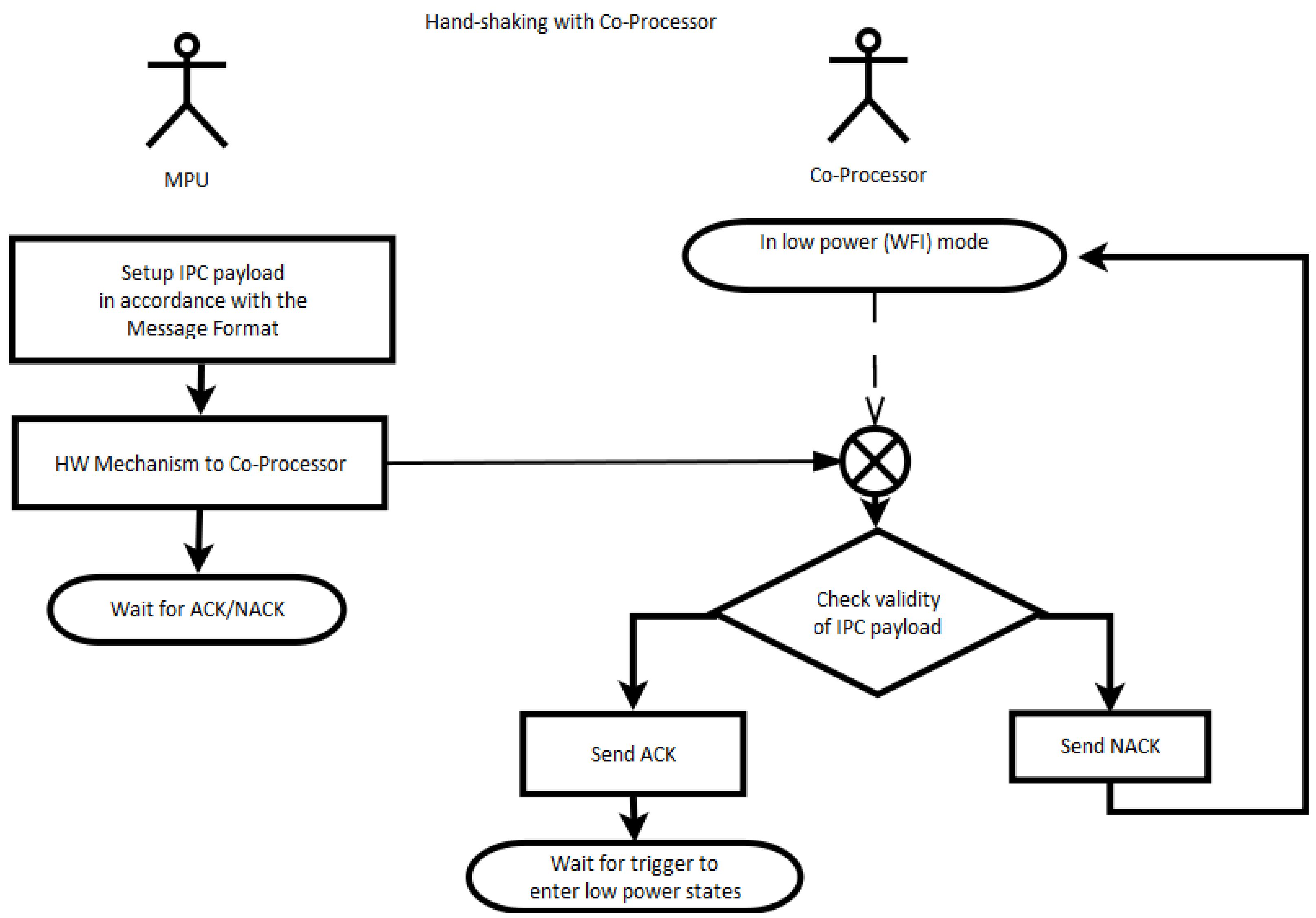
PRCM can use some help



IPC Mechanism


MPU Sub-system
1. Write to Notify PM Co-Processor
Mailbox
0. Populate IPC payloiad
Always ON Registers for IPC payload
2. Notification tc PM Co-Processor
3. PM Co-Processor checks IPC payload
4. Interrupt to PRCM (Triggered by WFI execution)
PRCM
5. Interrupt for low power ops
PM Co-Processor (Cortex-M3)

Overall flow


Hand-shaking with Co-Processor
MPU
Co-Processor
Setup IPC payload in accordance with the Message Format
HW Mechanism to Co-Processor
Wait for ACK/NACK
In low power (WFI) mode
Check validity of IPC payload
Send ACK
Send NACK
Wait for trigger to enter low power states

PM Feature Split

PM Feature	Traditional Systems	AM335x
Reset Control	✓	✓
Wakeup from sleep state	✓	✓
System Clock Disable	✓	✓
SRAM State Management	✓	✓
Power Domain State Management	✓	✓
Clock Management	✓	✓*
PLL Management	✓	✓*
PMIC Control	✓	✓*
Driver Context save and restore	✓	✓
IO Pad Optimization for suspend state	✓	✓*

Key

✓ - HW

✓- MPU

✓- PM Co-Processor

* – Flexibility to do in PM Co-Processor

The right way forward

- Reduces HW complexity

- Flexibility
  - Overall functionality – design and SW stack
  - Develop custom algorithm to optimize power consumption

- Helps workaround some HW bugs

Challenges

- Idle state transition assisted by Co-Processor
  - Want same power savings as suspend state
  - Co-Processor for C-state entry and exit
    - Why?
  - Co-Processor should be ready to take the command at all times
- Idle tied to MPU

Challenges

- Wakeup capability
  - Not all peripherals have it
  - No way to come back from some C-states

- Calls for an additional constraint
  - Wakeup constraint - Prevent entry to C-states
  - Driver control over constraint?

Challenges

- PM layer init dependent on a binary blob
  - Requires Firmware API
    - Co-Processor code is available online…
    - Could be blocked till user-space comes up
    - Use initramfs?
  - C-states or OPP gets added or removed at runtime
    - Ensure that there's no static dependency

Challenges

- PMIC driver on Co-Processor?
  - Parts of I2C/GPIO driver on Co-Processor
  - Hooking up the regulator f/w with Co-Processor

Future work

- Standardize the message passing scheme
  - Alignment for making things generic
- Passing configuration data to Co-Processor
  - Extend DT to configure Co-Processor based on boards
  - Current use-cases
    - Optimizing IO pad configuration for the board
    - PMIC info
    - …

Advantages

- Interfacing the PMIC with the CM3
  - Most generic solution since not tied to a PMIC
  - Control can be from Co-Processor to keep MPU powered down
- Ability to workaround HW and ROM bugs
  - Bugs around “change of mind” (suspend -> don’t suspend) scenarios
- Test-bed for future PM features
  - Test codes on Co-Processor to experiment
  - Move things to HW in future?

References

- AM335x Technical Information
  - www.ti.com/am335x
- AM335x PM Firmware code
  - http://arago-project.org/git/projects/?p=am33x-cm3.git;a=summary

Requirements

- Co-processor aware idle state entry/exit
- Driver constraints for indicating wakeup capability in deeper idle states

Backup