UTILIZING A BIG.LITTLE™ SOLUTION IN AUTOMOTIVE

JUN. 20, 2018

YOSHIYUKI ITO

AUTOMOTIVE INFORMATION SOLUTION BUSINESS DIVISION
RENESAS ELECTRONICS CORPORATION

Today's Topics & Goal

- Requirement for big.LITTLE™ feature on Automotive Market
  - ➢ Solutions with the Upstream Linux® Kernel
  - ➢ More Performance than battery life
- Parts of EAS contribution from Arm are got accepted by Linux Kernel Community
- Tradeoffs between power consumption and performance
- Solution with the existing upstream features for big.LITTLE
  - ➢ How to put tasks to "big" side as possible
- Now it is contributing big.LITTLE feature to AGL UCB™

Who am I ?

- Name : Yoshiyuki Ito
  I'm working in Renesas Electronics Corporation., Tokyo, Japan.

◆ Career:

- ✓ 1993 – 2003 : Working on multi-core processor development project in NEC.
- ✓ 2003 – 2013 : In charge of Linux® for mobile phone in the NEC Electronics and Renesas Electronics.
- ✓ 2013 – : Joined Linux® for automotive development project.

E-mail : yoshiyuki.ito.ub@renesas.com

Today's Topics & Goal

- Requirement for big.LITTLE™ feature on Automotive Market
  - Solutions with the Upstream Linux® Kernel
  - More Performance than battery life
- Parts of EAS contribution from Arm are got accepted by Linux Kernel Community
- Tradeoffs between power consumption and performance
- Solution with the existing upstream features for big.LITTLE
  - How to put tasks to "big" side as possible
- Now it is contributing big.LITTLE feature to AGL UCB™

Requirement: Solutions with LTS/Upstream Kernel

- Linux kernels of In-vehicle infotainment system (IVI)
  - –Android Automotive: Android Common Kernel
  - –AGL and any other IVI than Android: LTS Kernel

→Some part of EAS still integrated only for Android Common Kernel

Hove to be solved how to provide big.LITTLE feature for non-Android IVI

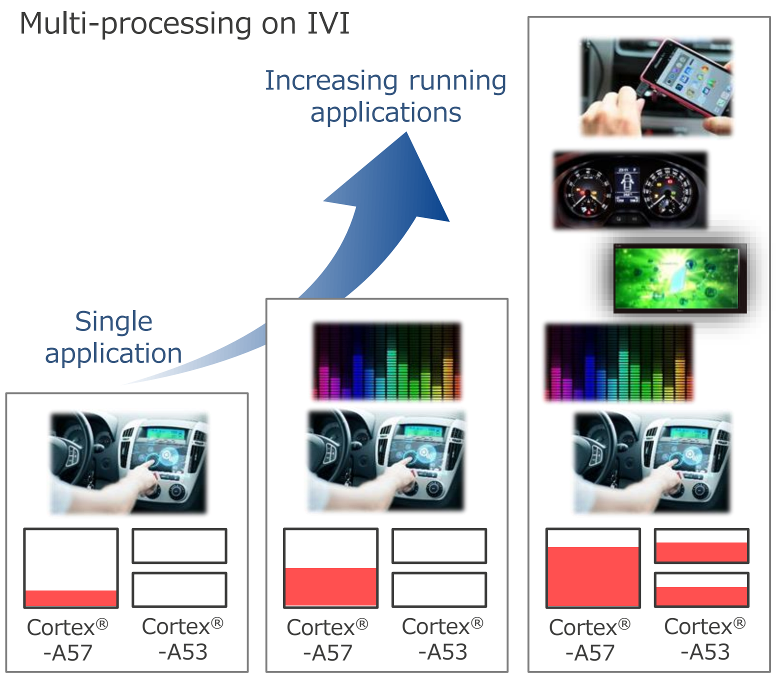
Requirement: Performance than Battery Life

- Target of original EAS
  - Enhancing BATTERY LIFE ← Reducing Power Consumption
    - And then preparing for heavy tasks ← To provide better user experience

- **Requirement of Automotive Market**
  - Keep the PERFORMANCE even in heavy use case

    ←Stable display framerate and/or response time

    are required

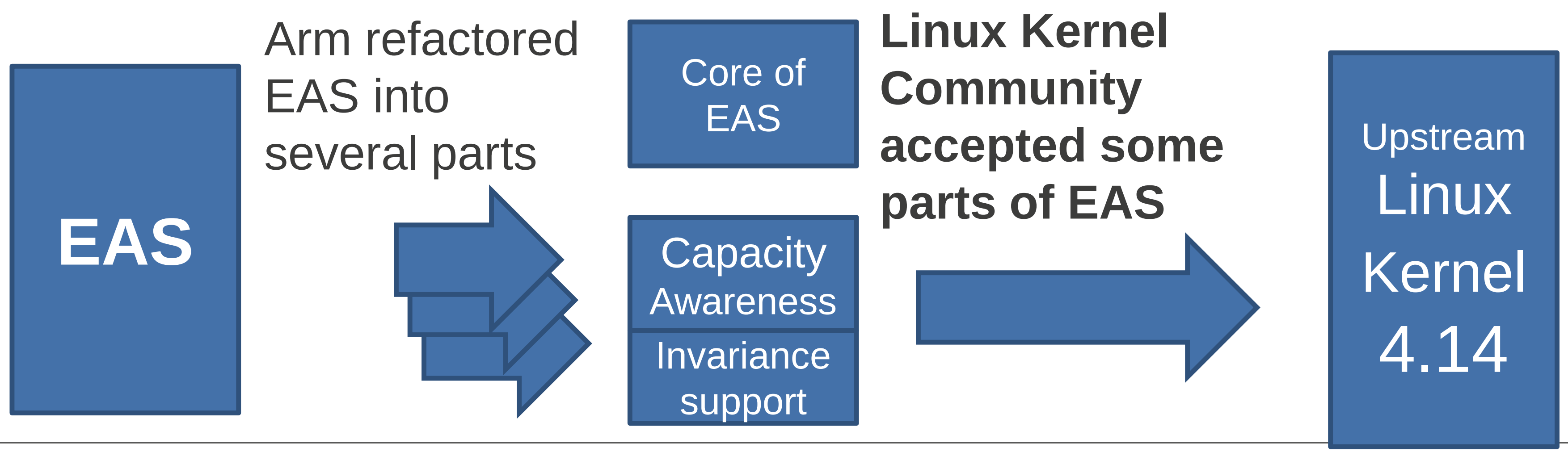


Today's Topics & Goal

- Requirement for big.LITTLE™ feature on Automotive Market
  - Solutions with the Upstream Linux® Kernel
  - More Performance than battery life
- Parts of EAS contribution from Arm are got accepted by Linux Kernel Community
- Tradeoffs between power consumption and performance
- Solution with the existing upstream features for big.LITTLE
  - How to put tasks to "big" side as possible
- Now it is contributing big.LITTLE feature to AGL UCB™

EAS contribution and Linux Kernel community

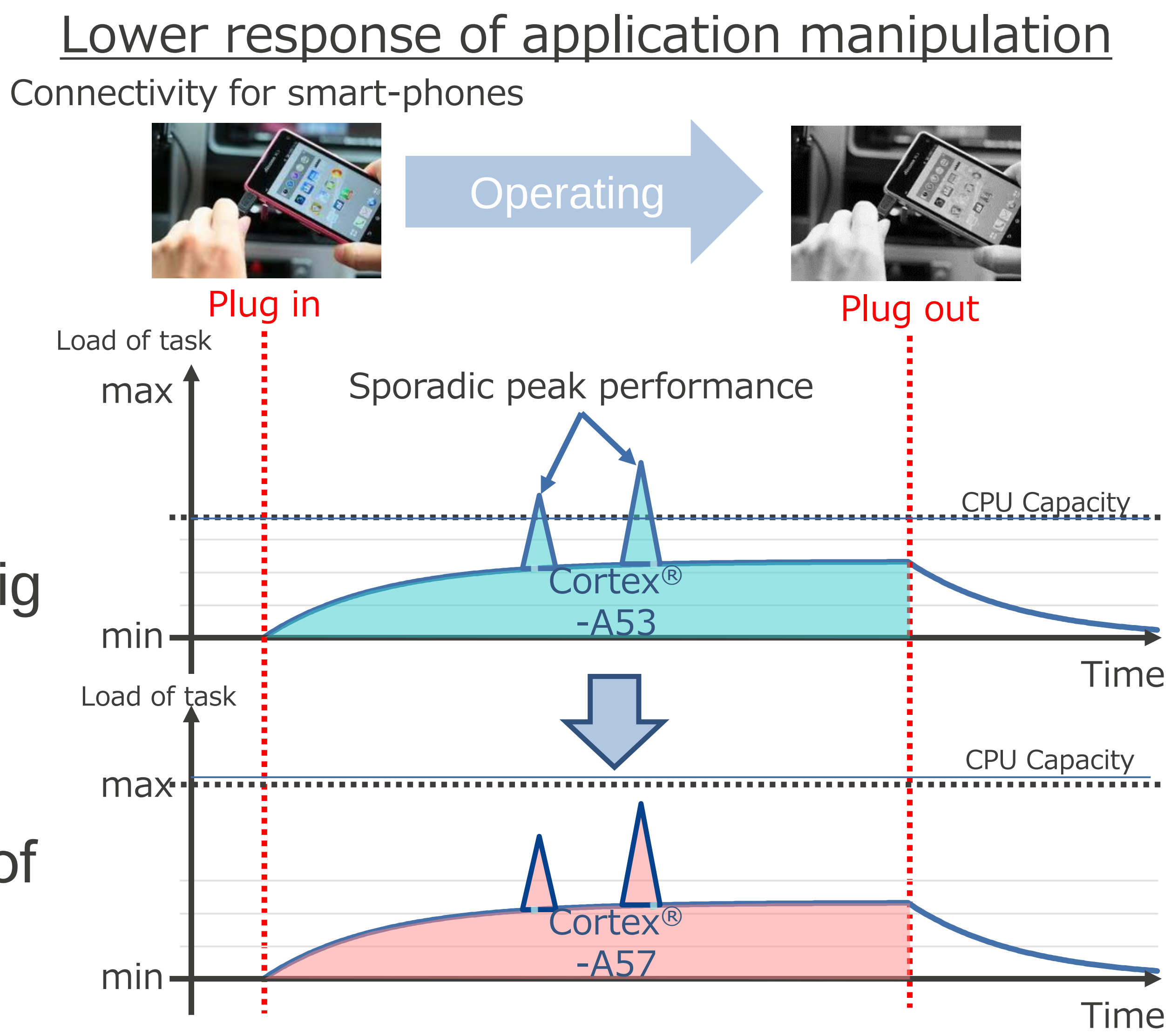
- It got accepted by upstream as some parts of EAS such as capacity awareness, Freq./Arch. invariance support, etc., as a result of Arm contribution effort.
  - Arm refactored EAS into several parts of scheduler functions to contribute for upstream.
  - Core part of EAS is still struggling to contribution (as of 4.14).



Tradeoffs between power consumption and performance

- When put the priority for effectiveness of “BATTELY LIFE”
  - Unexpected task allocation to the LITTLE would be expected
  - Even while it able to allocate big CPU

→ Performance Instability on the “Sporadic peak performance” type of the task

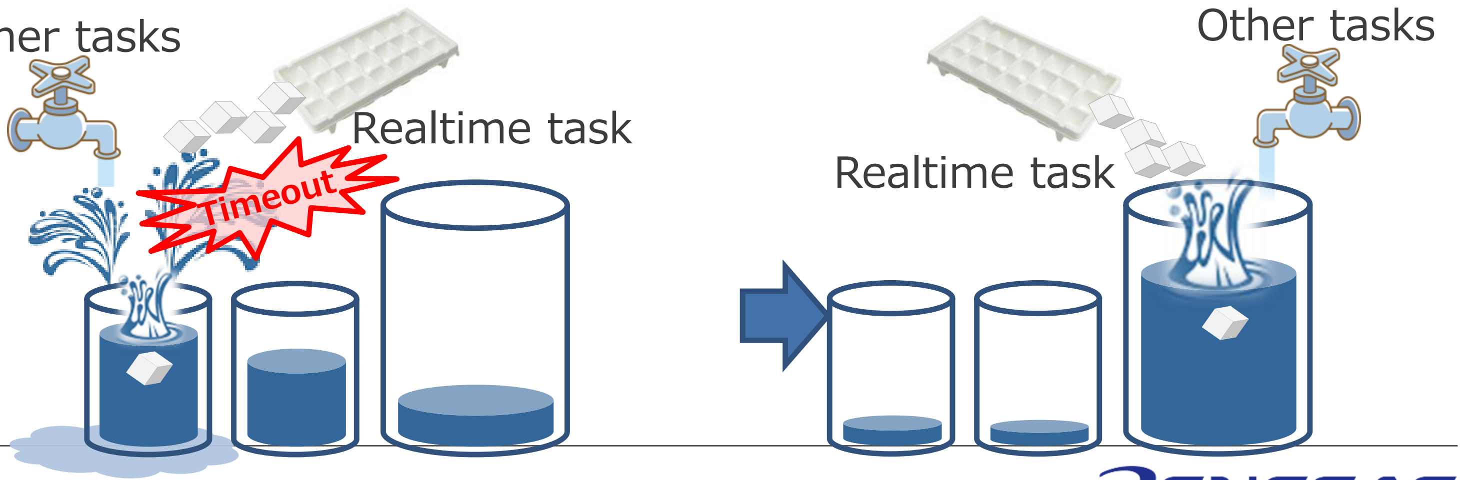


Today's Topics & Goal

- Requirement for big.LITTLE™ feature on Automotive Market
  - Solutions with the Upstream Linux® Kernel
  - More Performance than battery life
- Parts of EAS contribution from Arm are got accepted by Linux Kernel Community
- Tradeoffs between power consumption and performance
- Solution with the existing upstream features for big.LITTLE
  - How to put tasks to "big" side as possible
- Now it is contributing big.LITTLE feature to AGL UCB™

How to solve “unexpected task allocation to the LITTLE”

- In the case of Android Common Kernel:
  - It’s able to control task assignment via **sched tune** facilities.
- In the case of Upstream Kernel:
  - It have to be apply parameter tunings to **utilize “big” side** as maximum.



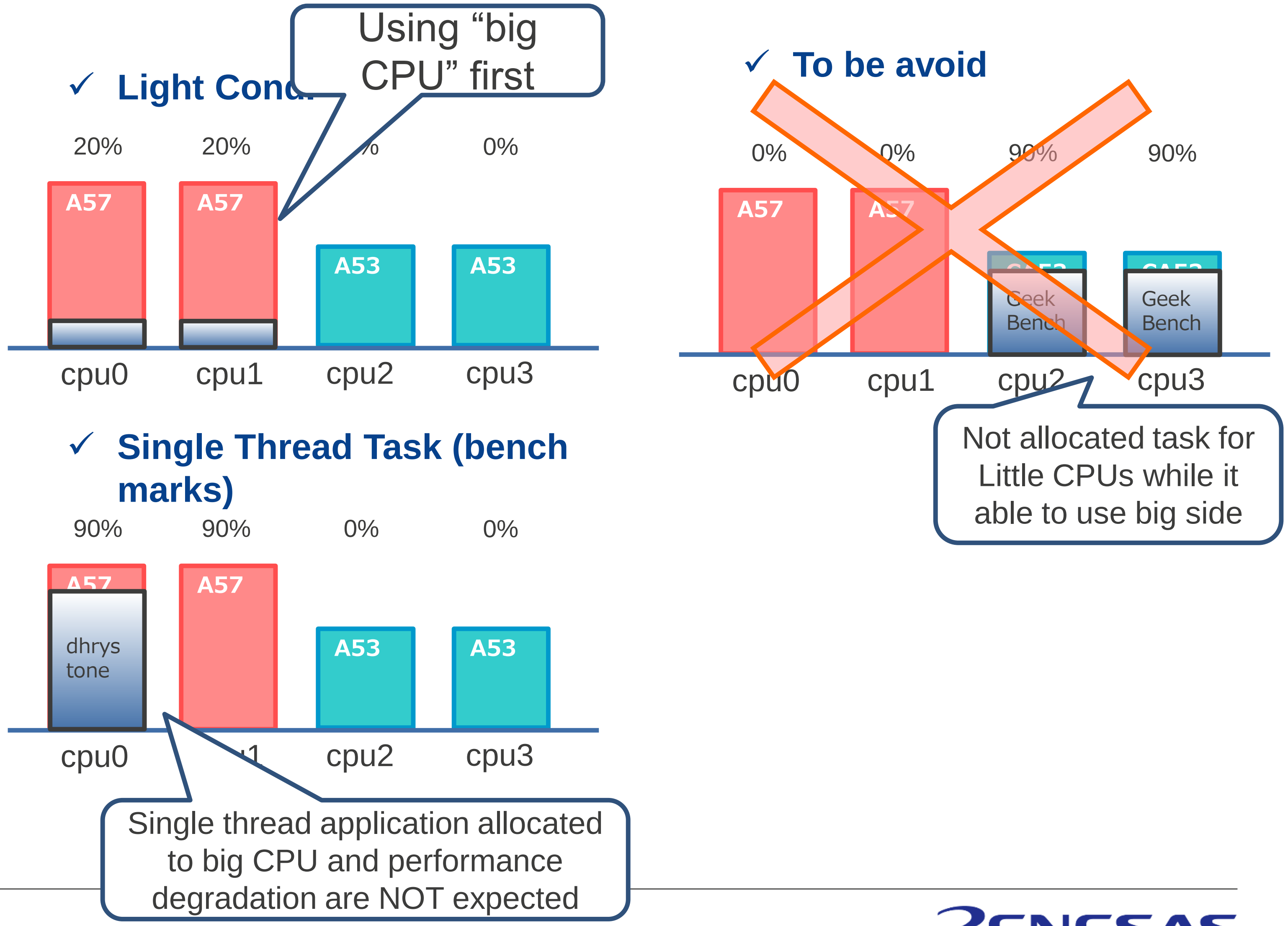
Parameters to achieve task allocation to utilize "big" side (1/2)

- It must allocate tasks for "big" side

←Preventing degradation of the performance of single thread process

  – Untightening request for power consumption, but put on weight for performance

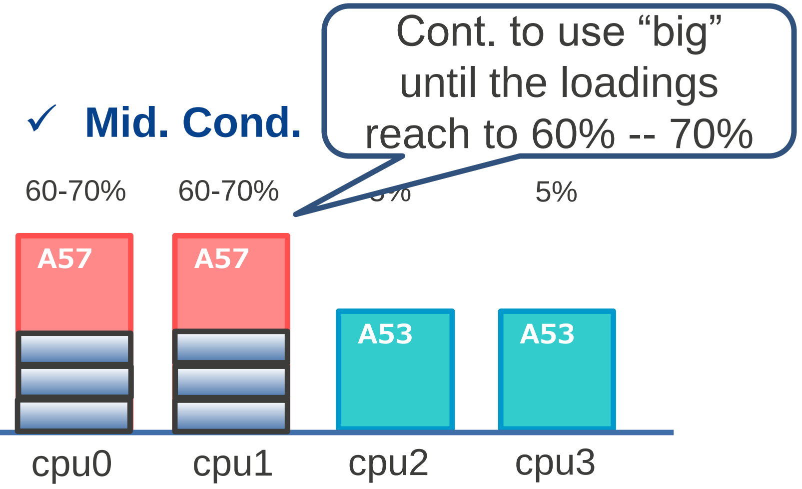
- Renesas got many suggestions from Arm SW Team to implement how it modify to allocation priority



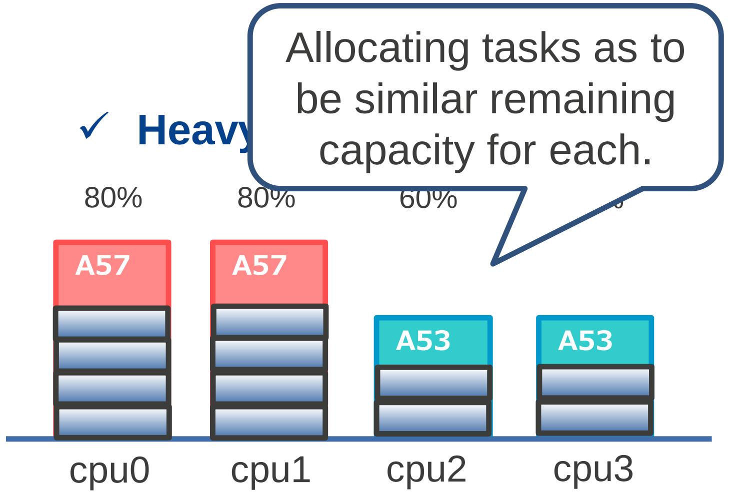
Parameters to achieve task allocation to utilize "big" side (2/2)

- When filled out so far the capacity of big side, it starting allocation to the LITTLE side.
  → It's better to run on LITTLE side instead of the waiting on the run queue.
- It able to achieve that to control task allocation along with making remaining capacity as a similar level on both big side and LITTLE side.
- Renesas consider that it is appropriate to control load level as to corresponds to 60% - 70% on the big side. (Note: It depends on max CPU capacity.)

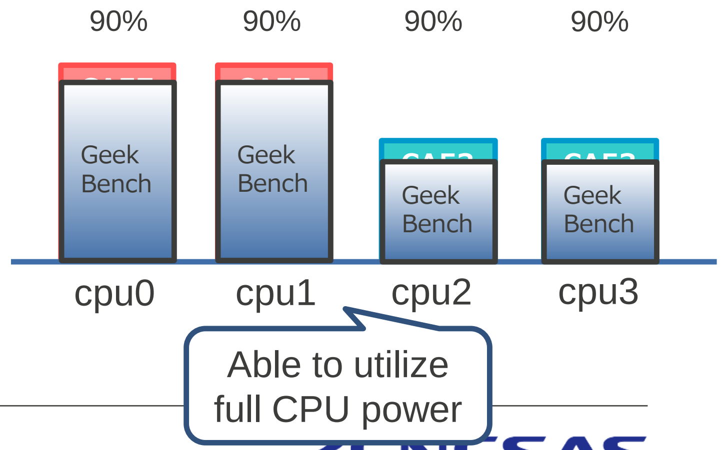
✓ **Mid. Cond.**



✓ **Heavy**



✓ **Multi-threaded Tasks**



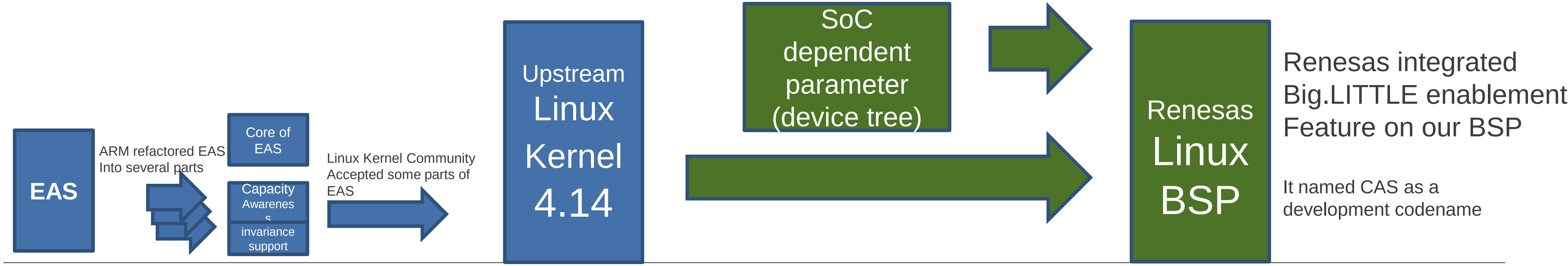
Today's Topics & Goal

- Requirement for big.LITTLE™ feature on Automotive Market
  - Solutions with the Upstream Linux® Kernel
  - More Performance than battery life
- Parts of EAS contribution from Arm are got accepted by Linux Kernel Community
- Tradeoffs between power consumption and performance
- Solution with the existing upstream features for big.LITTLE
  - How to put tasks to "big" side as possible
- Now it is contributing big.LITTLE feature to AGL UCB™

How to utilize big.LITTLE w/ upstream kernel (1/2)

– Already enough when it's not required strict battery life enhancement

- Renesas: **Confirmed those current partial EAS features of the upstream Linux Kernel** (capacity awareness, Freq./Arch. invariance support, etc.) **are enough to utilize big.LITTLE feature for automotive use case.**
  - With some additional scheduler parameter refinement via device tree to make CPU performance as optimal.



How to utilize big.LITTLE w/ upstream kernel (2/2)

– Just 5 patches required

1. Added CPU capacity definitions of Cortex®-A57/Cortex®-A53 into the device tree

   - arm64: dts: renesas: r8a7795: Add cpu capacity-dmips-mhz

   - arm64: dts: renesas: r8a7796: Add cpu capacity-dmips-mhz

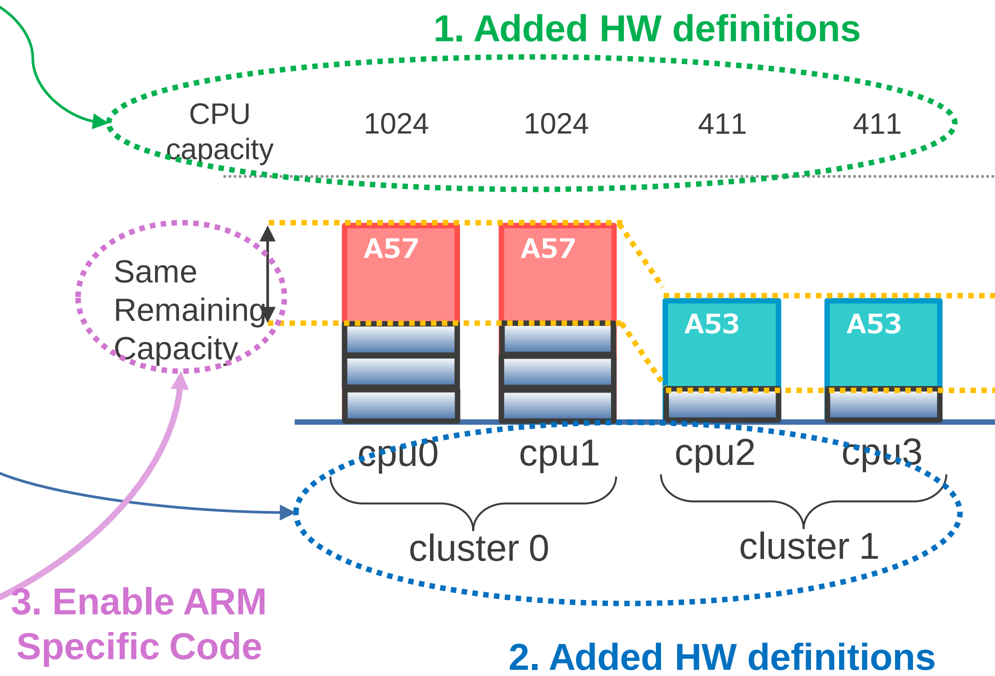
2. Defines cluster structure of A57/A53 into the device tree

   - arm64: dts: renesas: r8a7795: Add multi-cluster definition

   - arm64: dts: renesas: r8a7796: Add multi-cluster definition

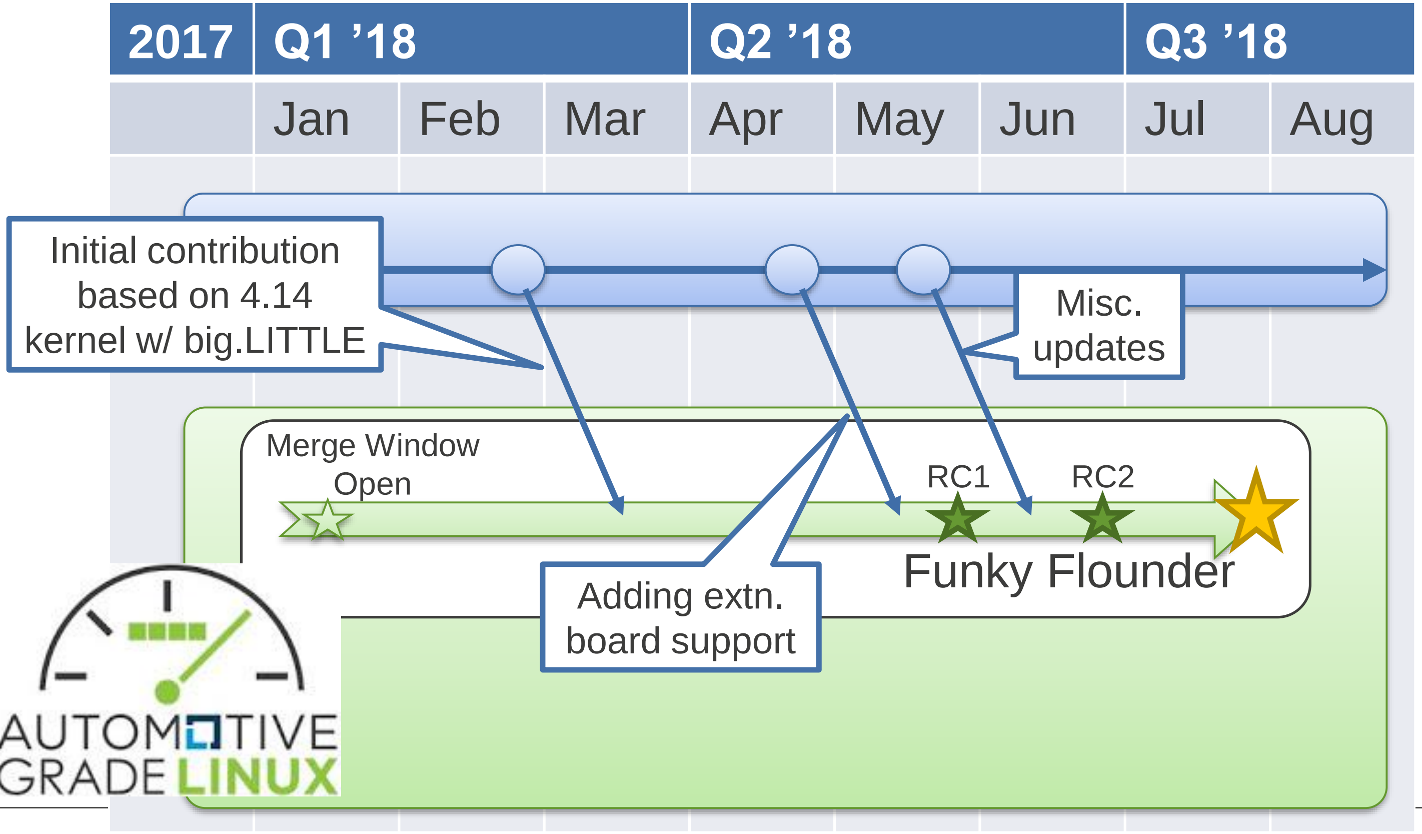
3. Enable the “flag” to capacity awareness of the scheduler

   - soc: renesas: rcar-topology: Add support to be aware cpu capacity



It's already starting contribution to AGL FF

- Renesas is now contributing BSP with big.LITTLE feature into AGL UCB™.
  - Also contributing Renesas's small patches to the upstream.



CONCLUSION

- Renesas is now contributing big.LITTLE feature for AGL FF release
  - Which big.LITTLE feature tweaked by parameter setting:
    - To utilizing “big” core as maximum to fit with Automotive use case
- It implemented with small patches for device tree and upstream scheduler
- As a result, it got explicit performance improvement on multi-threaded benchmarks without any performance degrade of single-thread benchmark compared with SMP of “big” cores

And, I’m very appreciate to the Arm Software Team who put heavy effort to contributing EAS to the upstream and held quite fruitful discussion with us about the these patch.

Renesas.com

Thank you

yoshiyuki.ito.ub@renesas.com

- AGL™ is the registered trademark of The Linux Foundation in the United States and/or other countries.
- Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.
- Arm, Cortex and big.LITTLE are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
- Android is a trademark of Google LLC.
- All names of other products or services mentioned in this press release are trademarks or registered