# PRU Hardware Overview

**Building Blocks for PRU Development: Module 1**

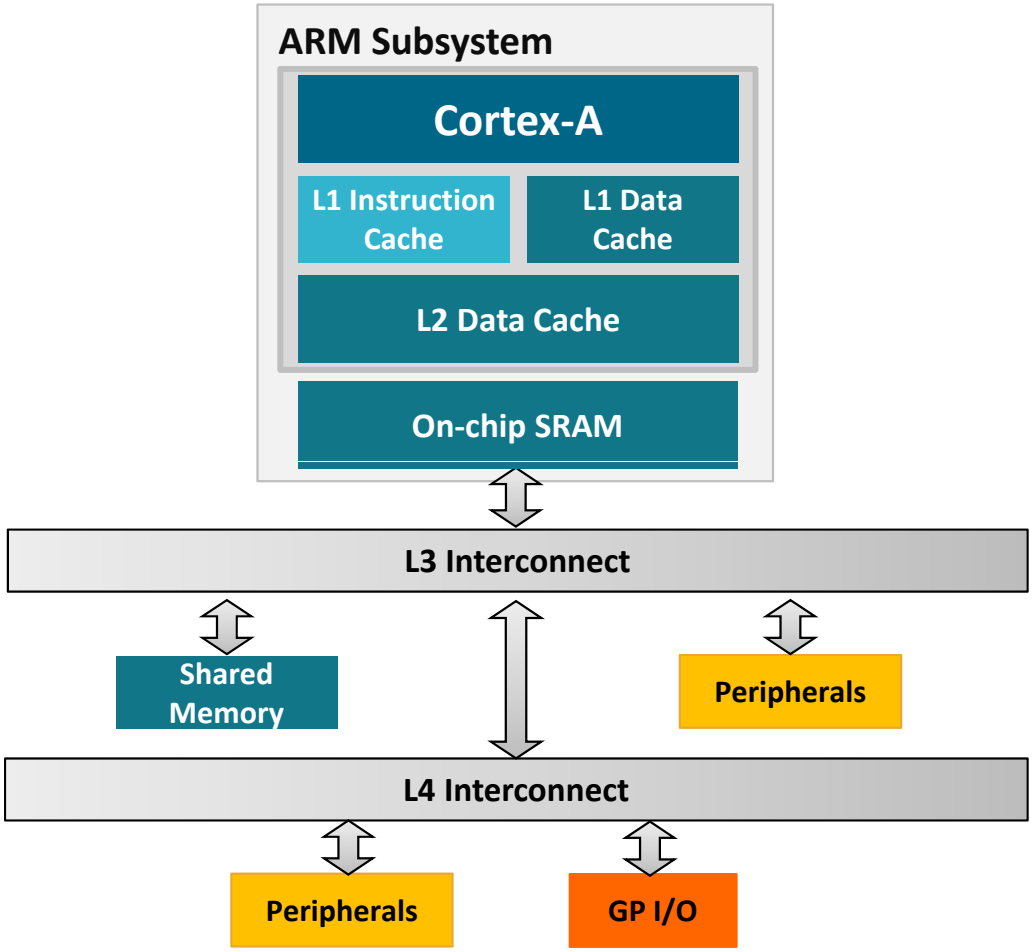
# Agenda

- SoC Architecture
- PRU Submodules
- Example Applications

# SoC Architecture

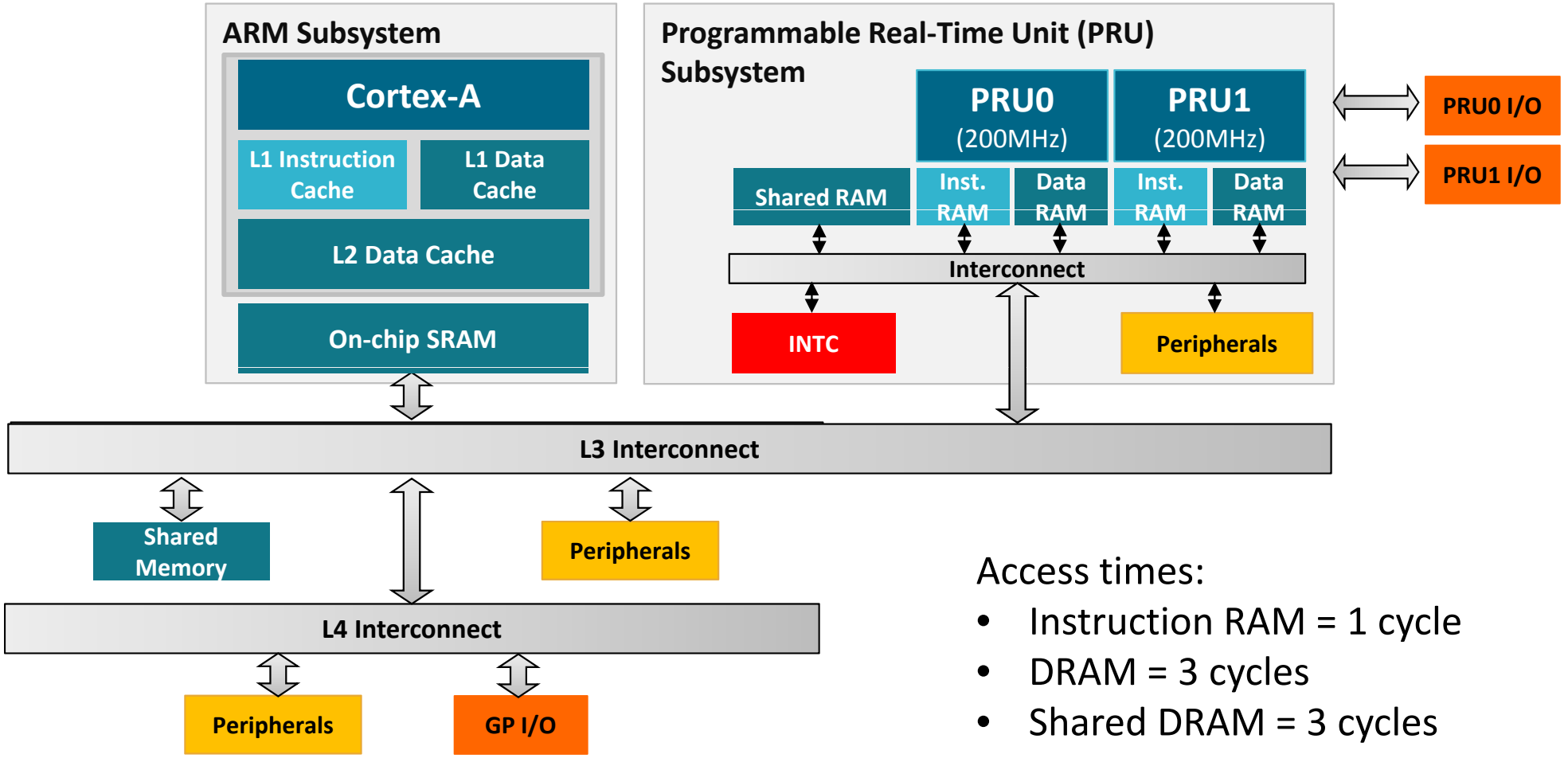
**Building Blocks for PRU Development: PRU Hardware Overview**

# ARM SoC Architecture




- L1 D/I caches: Single-cycle access
- L2 cache: Minimum latency of 8 cycles
- Access to on-chip SRAM: 20 cycles
- Access to shared memory over L3 Interconnect: 40 cycles

# ARM + PRU SoC Architecture

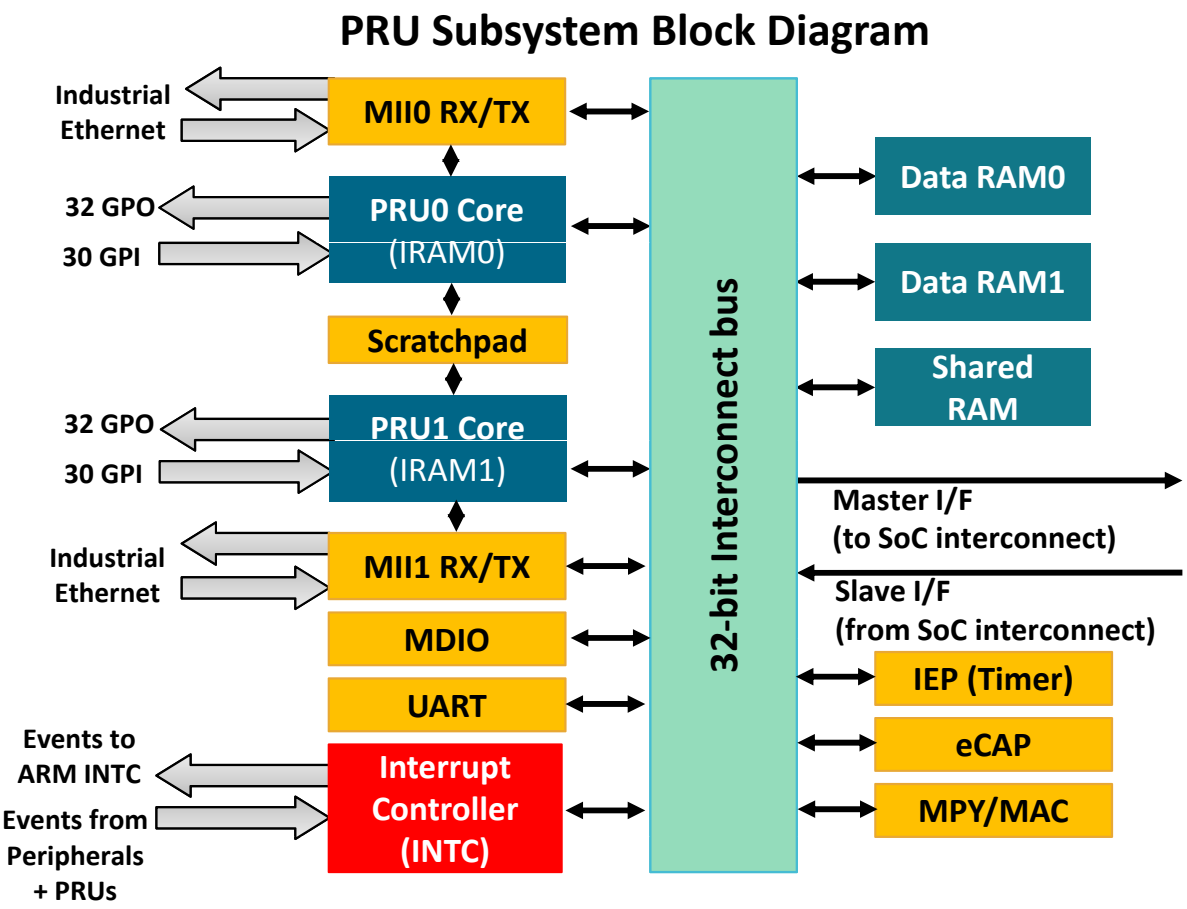



Access times:

- Instruction RAM = 1 cycle
- DRAM = 3 cycles
- Shared DRAM = 3 cycles

# Programmable Real-Time Unit (PRU) Subsystem

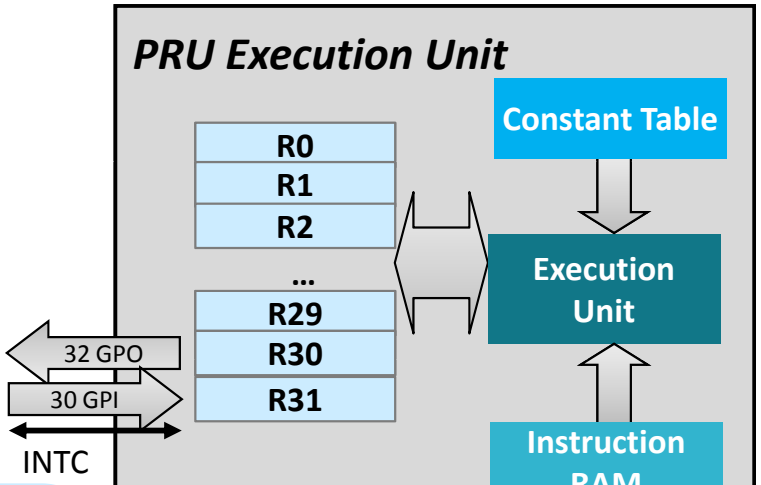
- Programmable Real-Time Unit (PRU) is a low-latency microcontroller subsystem.
- Two independent PRU execution units:
  - 32-Bit RISC architecture
  - 200MHz;  5ns per instruction
  - Single cycle execution; No pipeline
  - Dedicated instruction and data RAM per core
  - Shared RAM
- Includes Interrupt Controller for system event handling
- Fast I/O interface: Up to 30 inputs and 32 outputs on external pins per PRU unit



# PRU Submodules

**Building Blocks for PRU Development: PRU Hardware Overview**

# PRU Functional Block Diagram




## Constant Table

Saves PRU cycles by providing frequently used peripheral base addresses

## Execution Unit

- Logical, arithmetic, and flow control instructions
- Scalar, no Pipeline, Little Endian
- Single-cycle execution

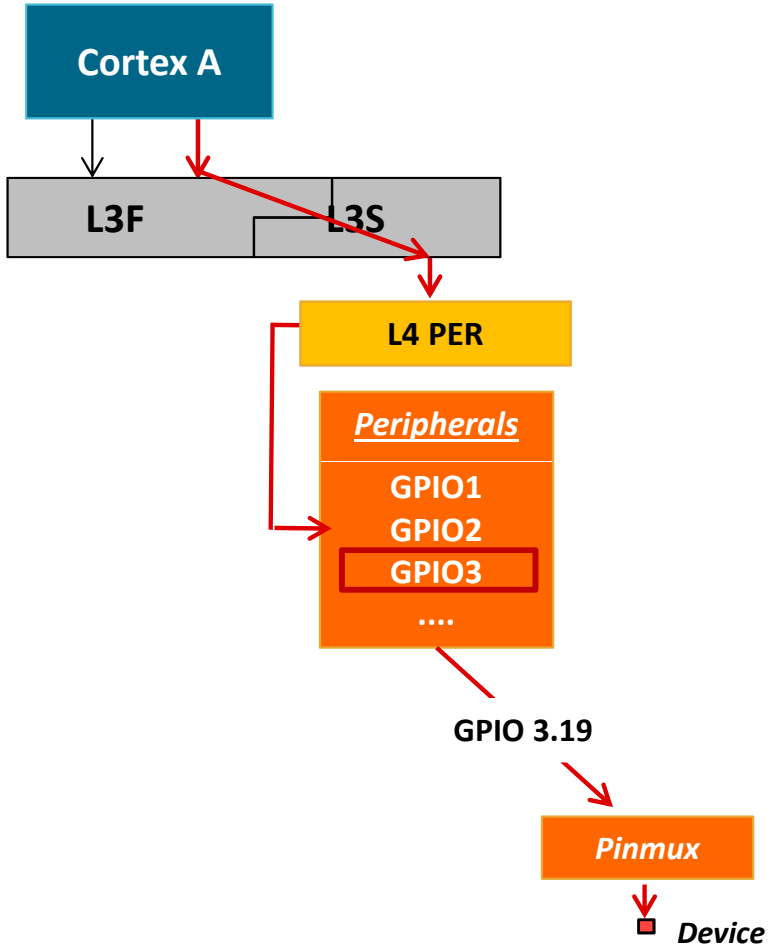
## Instruction RAM

- Typical size is a multiple of 4KB (or 1K instructions)
- Can be updated with PRU reset

## Special Registers (R30 and R31)

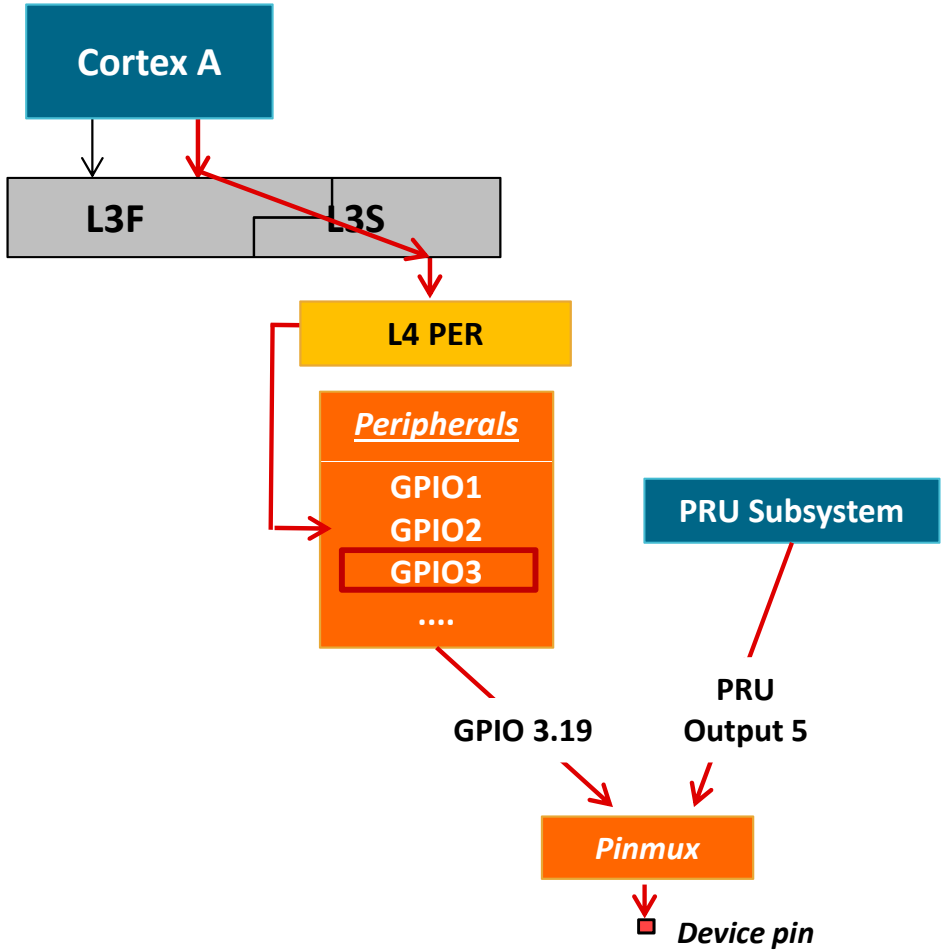
- R30
  - Write: 32 GPO
- R31
  - Read: 30 GPI + 2 Host Int status
  - Write: Generate INTC Event

# Fast I/O Interface



# Fast I/O Interface

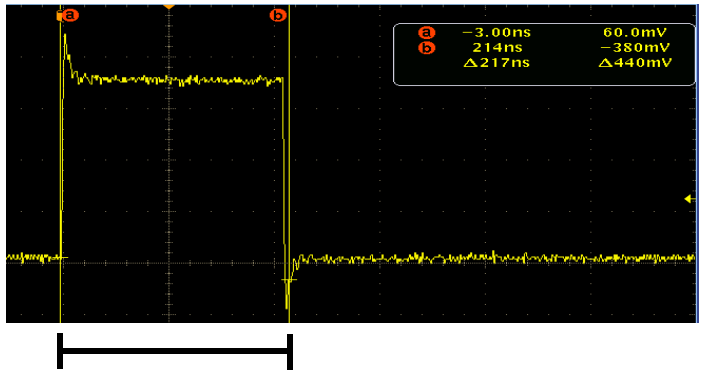
- Reduced latency through direct access to pins:
  - Read or toggle I/O within a single PRU cycle.
  - Detect and react to I/O event within two PRU cycles.
- Independent General Purpose Inputs (GPIs) and General Purpose Outputs (GPOs):
  - PRU R31 directly reads from up to 30 PRU GPI pins.
  - PRU R30 directly writes up to 32 PRU GPO pins.
- Configurable I/O modes per PRU core:
  - GP input modes:
    - Direct connect
    - 16-bit parallel capture
    - 28-bit shift
  - GP output modes:
    - Direct connect
    - Shift out



# GPIO Toggle: Bench Measurements

## ARM GPIO Toggle:

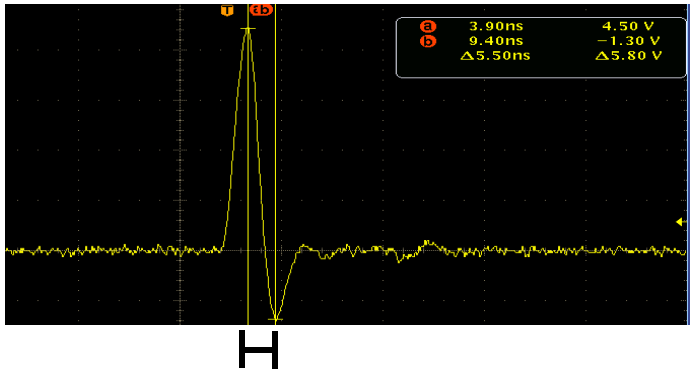
```
int main(){

    // Configure GPIO module, pinmuxing, etc.

    // Toggle system-level GPIO 3.19 from ARM core
    BitToggle(GPIO_INSTANCE_ADDRESS+GPIO_SETDATAOUT,
              GPIO_INSTANCE_ADDRESS+GPIO_CLEARDATAOUT);

    while();
}

unsigned long BitToggle(unsigned long val1, unsigned long val2){
    asm(
        " mov r2, #0x00080000" "\n\t"
        "str     r2,[r0]" "\n\t"          // Set GPIO 3.19
        "str     r2,[r1]" "\n\t"          // Clear GPIO 3.19
    );
    return val1;
}
```




~200ns

## PRU IO Toggle:

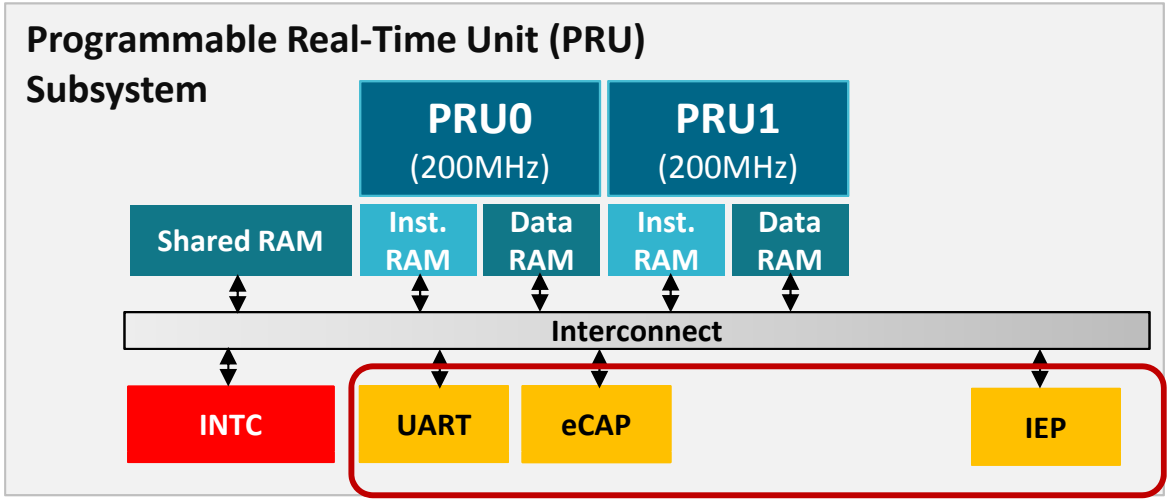
```
.origin 0
.entrypoint PRU_GPIO_TOGGLE


PRU_GPIO_TOGGLE:

    // Set PRU GPO 5
    SET       R30, R30, 5

    // Clear PRU GPO 5
    CLR       R30, R30, 5

    HALT
```




~5ns = ~40x Faster

# Integrated Peripherals

- Provide reduced PRU read/write access latency compared to external peripherals
- Local peripherals do not need to go through external L3 or L4 interconnects
- Can be used by PRU or by the ARM as additional hardware peripherals on the device
- Integrated peripherals:
  - PRU UART
  - PRU eCAP
  - PRU IEP (Timer & DigIO)



# PRU Memory Map

- PRU local memory map

**Table 5. Local Data Memory Map**

| Start Address | PRU0 | PRU1 |
|---|---|---|
| 0x0000_0000 | Data 8KB RAM 0[1] | Data 8KB RAM 1[1] |
| 0x0000_2000 | Data 8KB RAM 1[1] | Data 8KB RAM 0[1] |
| 0x0001_0000 | Data 12KB RAM2 (Shared) | Data 12KB RAM2 (Shared) |
| 0x0002_0000 | INTC | INTC |
| 0x0002_2000 | PRU0 Control Registers | PRU0 Control Registers |
| 0x0002_2400 | Reserved | Reserved |
| 0x0002_4000 | PRU1 Control | PRU1 Control |
| 0x0002_4400 | Reserved | Reserved |
| 0x0002_6000 | CFG | CFG |
| 0x0002_8000 | UART 0 | UART 0 |
| 0x0002_A000 | Reserved | Reserved |
| 0x0002_C000 | Reserved | Reserved |
| 0x0002_E000 | IEP | IEP |
| 0x0003_0000 | eCAP 0 | eCAP 0 |
| 0x0003_2000 | MII_RT_CFG | MII_RT_CFG |
| 0x0003_2400 | MII_MDIO | MII_MDIO |
| 0x0003_4000 | Reserved | Reserved |
| 0x0008_0000 | System OCP_HP0 | System OCP_HP1 |

- PRU global memory map

**Table 6. Global Memory Map**

| Offset Address | PRU-ICSS |
|---|---|
| 0x0000_0000 | Data 8KB RAM 0 |
| 0x0000_2000 | Data 8KB RAM 1 |
| 0x0001_0000 | Data 12KB RAM 2 (Shared) |
| 0x0002_0000 | INTC |
| 0x0002_2000 | PRU0 Control |
| 0x0002_2400 | PRU0 Debug |
| 0x0002_4000 | PRU1 Control |
| 0x0002_4400 | PRU1 Debug |
| 0x0002_6000 | CFG |
| 0x0002_8000 | UART 0 |
| 0x0002_A000 | Reserved |
| 0x0002_E000 | IEP |
| 0x0003_0000 | eCAP 0 |
| 0x0003_2000 | MII_RT_CFG |
| 0x0003_2400 | MII_MDIO |
| 0x0003_4000 | PRU0 8KB IRAM |
| 0x0003_8000 | PRU1 8KB IRAM |
| 0x0004_0000 | Reserved |

- SoC memory map

**Table 2-4. L4 Fast Peripheral Memory Map (continued)**

| Device Name | Start_address (hex) | End_address (hex) | Size | Description |
|---|---|---|---|---|
| PRU_ICSS | 0x4A30_0000 | 0x4A37_FFFF | 512KB | PRU-ICSS Instruction/Data/Control Space |
| | 0x4A38_0000 | 0x4A38_0FFF | 4KB | Reserved |

# PRU Read Latencies: Local vs Global Memory Map

The PRU directly accessing internal MMRs (Local MMR Access) is faster than going through the L3 interconnects (Global MMR Access).

| | Local MMR Access | Global MMR Access |
|---|---|---|
| | *PRU cycles @ 200MHz* | *PRU cycles @ 200MHz* |
| PRU R31 (GPI) | 1 | N/A |
| PRU CTRL | 4 | 36 |
| PRU CFG | 3 | 35 |
| PRU INTC | 3 | 35 |
| PRU DRAM | 3 | 35 |
| PRU Shared DRAM | 3 | 35 |
| PRU ECAP | 4 | 36 |
| PRU UART | 14 | 46 |
| PRU IEP | 12 | 44 |

**Note:** Latency values listed are “best-case” values.

# PRU Memory Access FAQ

**Q:** *Why does my PRU firmware hang when reading or writing to an address external to the PRU Subsystem?*

**A:** The **OCP master port** is in standby and **needs to be enabled** in the PRU-ICSS CFG register space, SYSCFG[STANDBY_INIT].

**Table 5. Local Data Memory Map**

| Start Address | PRU0 | PRU1 |
|---|---|---|
| 0x0000_0000 | Data 8KB RAM 0 (1) | Data 8KB RAM 1 (1) |
| 0x0000_2000 | Data 8KB RAM 1 (1) | Data 8KB RAM 0 (1) |
| 0x0001_0000 | Data 12KB RAM2 (Shared) | Data 12KB RAM2 (Shared) |
| 0x0002_0000 | INTC | INTC |
| 0x0002_2000 | PRU0 Control Registers | PRU0 Control Registers |
| 0x0002_2400 | Reserved | Reserved |
| 0x0002_4000 | PRU1 Control | PRU1 Control |
| 0x0002_4400 | Reserved | Reserved |
| 0x0002_6000 | CFG | CFG |
| 0x0002_8000 | UART 0 | UART 0 |
| 0x0002_A000 | Reserved | Reserved |
| 0x0002_C000 | Reserved | Reserved |
| 0x0002_E000 | IEP | IEP |
| 0x0003_0000 | eCAP 0 | eCAP 0 |
| 0x0003_2000 | MII_RT_CFG | MII_RT_CFG |
| 0x0003_2400 | MII_MDIO | MII_MDIO |
| 0x0003_4000 | Reserved | Reserved |
| 0x0008_0000 | System OCP_HP0 | System OCP_HP1 |

# PRU “Interrupts”

- The PRU does not support asynchronous interrupts.
  - However, specialized h/w and instructions facilitate efficient polling of system events.
  - The PRU-ICSS can also generate interrupts for the ARM, other PRU-ICSS, and sync events for EDMA.
- From UofT CSC469 lecture notes, “*Polling is like picking up your phone every few seconds to see if you have a call. Interrupts are like waiting for the phone to ring.*
  - *Interrupts win if processor has other work to do and event response time is not critical*
  - *Polling can be better if processor has to respond to an event ASAP*”
- Asynchronous interrupts can introduce jitter in execution time and generally reduce determinism. The PRU is optimized for highly deterministic operation.

# PRU Subsystem Feature Comparison

| Features | AM18x/ OMAPL138 | AM335x | AM437x | | AM571x | AM572x (SR1.1 / SR2.0*) |
|---|---|---|---|---|---|---|
| | PRUSS | PRU-ICSS1 | PRU-ICSS1 | PRU-ICSS0 | 2 x PRU-ICSS | 2 x PRU-ICSS |
| Number of PRU cores | 2 | 2 | 2 | 2 | 2 | 2 |
| Max frequency | CPU freq / 2 | 200 MHz | 200 MHz | 200 MHz | 200 MHz | 200 MHz |
| IRAM size (per PRU core) | 4 KB | 8 KB | 12 KB | 4 KB | 12 KB | 12 KB |
| DRAM size (per PRU core) | 512 B | 8 KB | 8 KB | 4 KB | 8 KB | 8 KB |
| Shared DRAM size | -- | 12 KB | 32 KB | -- | 32KB | 32KB |
| General purpose input (per PRU core) | Direct | Direct; or 16-bit parallel capture; or 28-bit shift | Direct; or 16-bit parallel capture; or 28-bit shift; or 3ch EnDat 2.2; or 9ch Sigma Delta | Direct; or 16-bit parallel capture; or 28-bit shift; or 3ch EnDat 2.2; or 9ch Sigma Delta | Direct; or 16-bit parallel capture; or 28-bit shift; or 3ch EnDat 2.2; or 9ch Sigma Delta | Direct; or 16-bit parallel capture; or 28-bit shift |
| General purpose output (per PRU core) | Direct | Direct; or Shift out | Direct; or Shift out | Direct; or Shift out | Direct; or Shift out | Direct; or Shift out |
| GPI Pins (PRU0, PRU1) | 30, 30 | 17, 17 | 13, 0 | 20, 20 | 0 / 21*, 21 | 21, 21 |
| GPO Pins (PRU0, PRU1) | 32, 32 | 16, 16 | 12, 0 | 20, 20 | 0 / 21*, 21 | 21, 21 |
| MPY/MAC | N | Y | Y | Y | Y | Y |
| Scratchpad | N | Y (3 banks) | Y (3 banks) | N | Y (3 banks) | Y (3 banks) |
| CRC16/32 | 0 | 0 | 2 | 2 | 2 | 0** / 2 |
| INTC | 1 | 1 | 1 | 1 | 1 | 1 |
| Peripherals | n/a | Y | Y | Y | Y | Y |
| UART | 0 | 1 | 1 | 1 | 1 | 1 |
| eCAP | 0 | 1 | 1 | no connect | 1 | 1 |
| IEP | 0 | 1 | 1 | no connect | 1 | 1 |
| MII_RT | 0 | 2 | 2 | no connect | 2 | 2 |
| MDIO | 0 | 1 | 1 | no connect | 1 | 1 |
| Simultaneous protocols | 1 | 1 | 2*** | | 2 | 2 |

* AM571x PRU-ICSS1 does not pin out the PRU0 core GPIs/GPOs.

** AM572x SR1.1 does not include CRC16/32.

*** 2nd protocol limited to EnDAT/Profibus/BISS/HIperphase DSL or serial-based protocols.

# Example PRU Applications

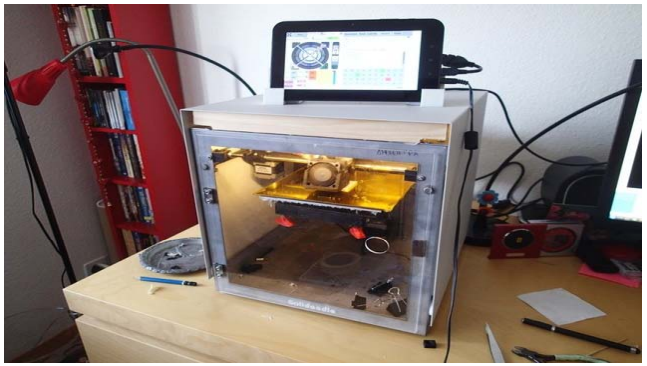
**Building Blocks for PRU Development: PRU Hardware Overview**

# Use Case Examples

- Industrial Protocols
- ASRC
- 10/100 Switch
- Smart Card
- DSP-like functions
- Filtering
- FSK Modulation
- LCD I/F
- Camera I/F
- RS-485
- UART
- SPI
- Monitor Sensors
- I2C
- Bit banging
- Custom/Complex PWM
- Stepper motor control

Not all use cases are feasible on PRU:

- Development complexity
- Technical constraints (i.e., running Linux on PRU)



**Development Complexity**

# For More Information

- Visit the PRU-ICSS Wiki: http://processors.wiki.ti.com/index.php/PRU-ICSS
- Download the PRU tools:
  - PRU Software Package — http://www.ti.com/tool/pru-swpkg
  - PRU CGT (Code Gen Tools) — *available through CCSv6 app center*
  - Linux drivers for interfacing with PRU — *available in Processor SDK*
- Order the PRU Cape: http://www.ti.com/tool/PRUCAPE
- For questions regarding topics covered in this training, visit the support forums at the TI E2E Community website: http://e2e.ti.com