SITARA™ ARM® PROCESSORS

BOOt camp

Building Blocks for PRU Development

Module 1

PRU Hardware Overview

This session covers a hardware overview of the PRU-ICSS Subsystem.

Author: Texas Instruments®, Sitara™ ARM® Processors

Oct 2014

Creative Commons Attribution-ShareAlike 3.0 (CC BY-SA 3.0)

creative commons

You are free:

to Share – to copy, distribute and transmit the work

to Remix – to adapt the work

to make commercial use of the work

Under the following conditions:

Attribution – You must give the original author(s) credit

Share Alike - If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

With the understanding that:

Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.

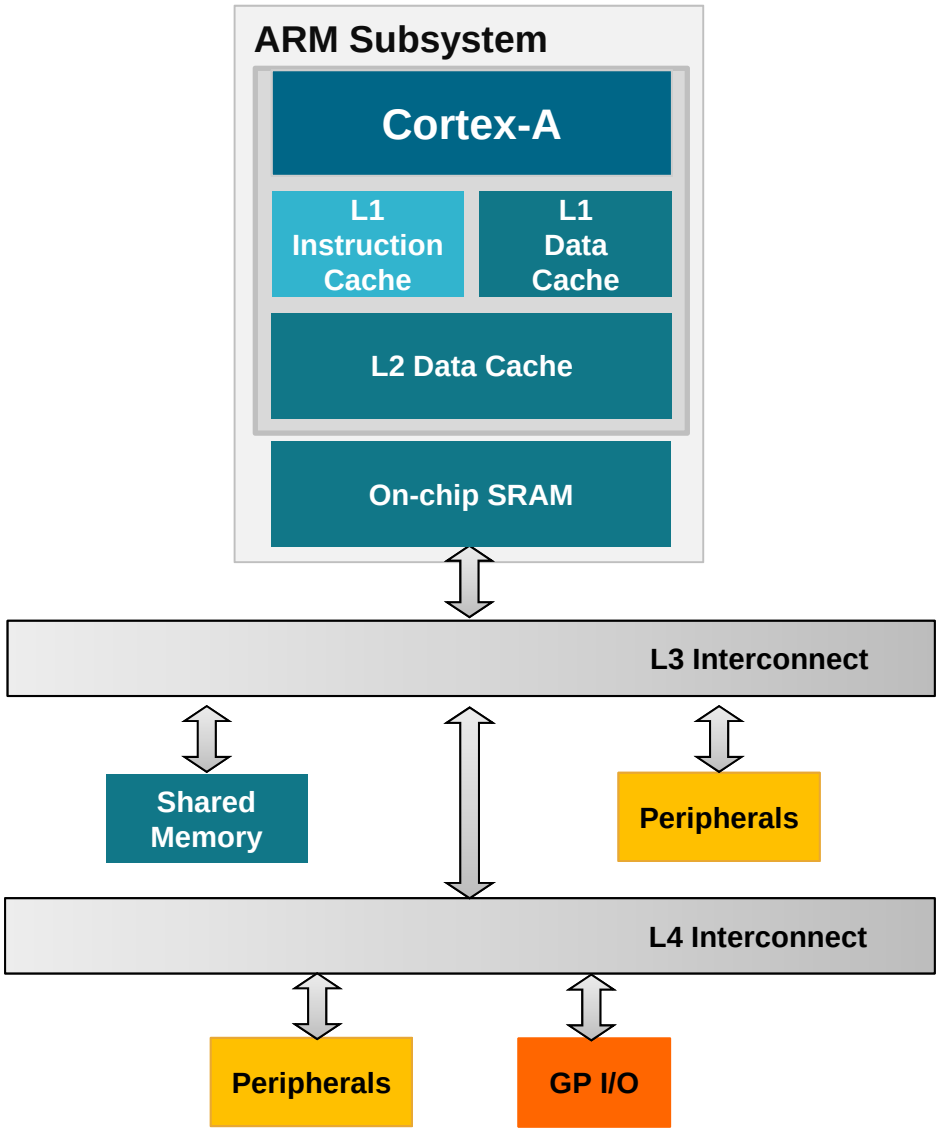
Public Domain — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.

Other Rights — In no way are any of the following rights affected by the license:

Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

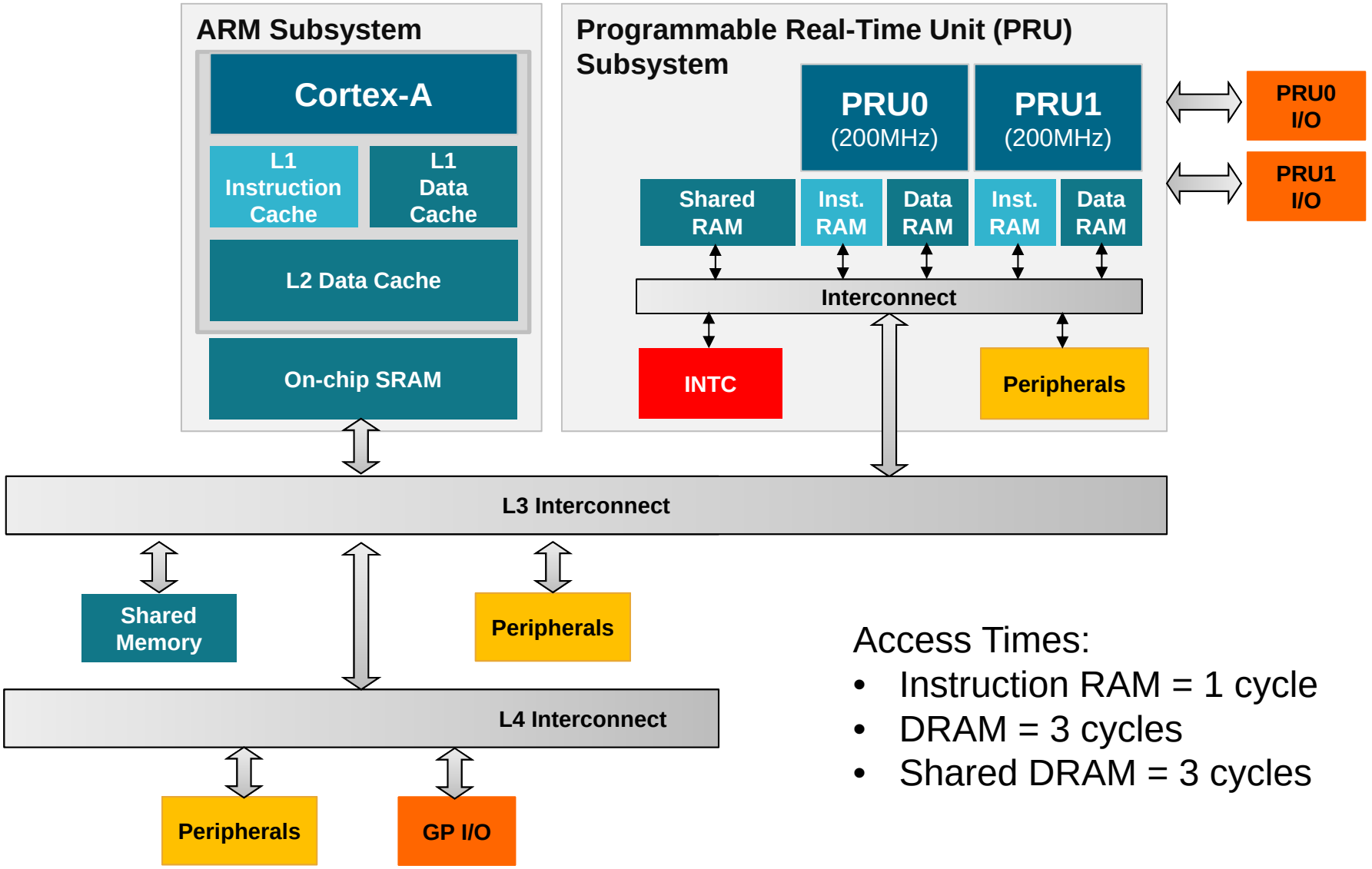
CC BY-SA

ARM SoC Architecture



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

ARM + PRU SoC Architecture

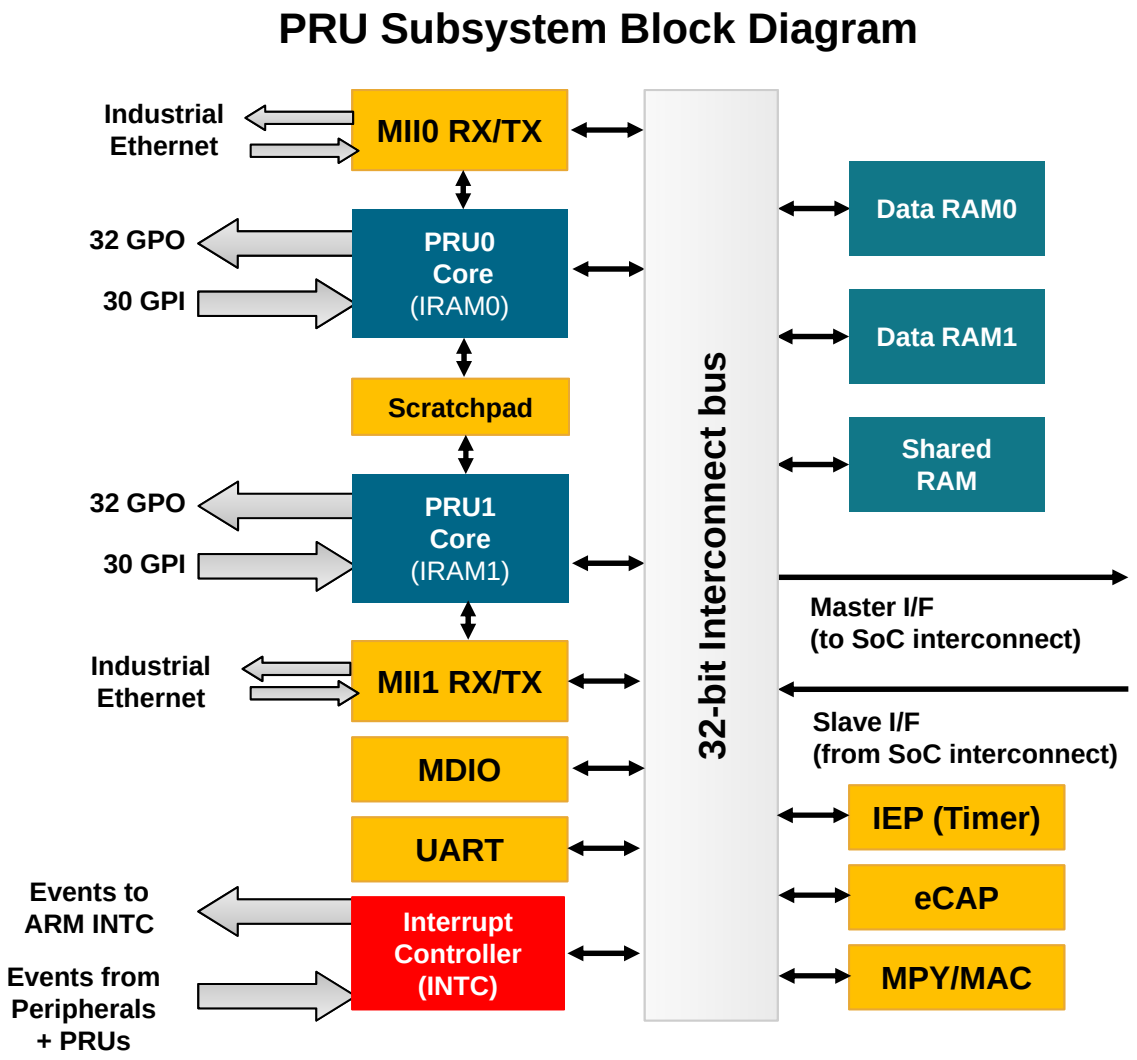


Access Times:

- 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



Features & Benefits

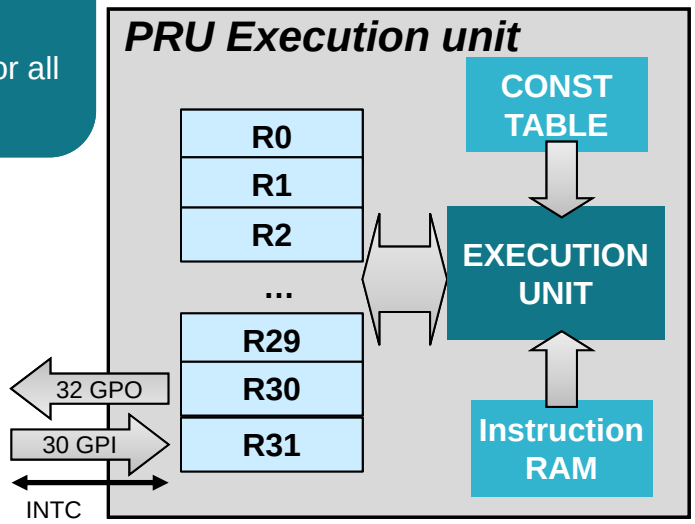
Feature	Benefit
Each PRU has dedicated instruction and data memory and can operate independently or in coordination with the ARM or the other PRU core	Use each PRU for a different task; use PRUs in tandem for more advanced tasks
Access all SoC resources (peripherals, memory, etc.)	Direct access to buffer data; leverage system peripherals for various implementations
Interrupt controller for monitoring and generating system events	Communication with higher level software running on ARM; detection of peripheral events
Dedicated, fast input and output pins	Input/output interface implementation; detect and react to I/O event within two PRU cycles
Small, deterministic instruction set with multiple bit-manipulation instructions	Easy to use; fast learning curve

Now let’s go a little deeper…

PRU Functional Block Diagram

General Purpose Registers

- All instructions are performed on registers and complete in a single cycle
- Register file appears as linear block for all register to memory operations



Constant Table

- Ease SW development by providing freq used constants
- Peripheral base addresses
- Few entries programmable

Execution Unit

- Logical, arithmetic, and flow control instructions
- Scalar, no Pipeline, Little Endian
- Register-to-register data flow
- Addressing modes: Ld Immediate & Ld/St to Mem

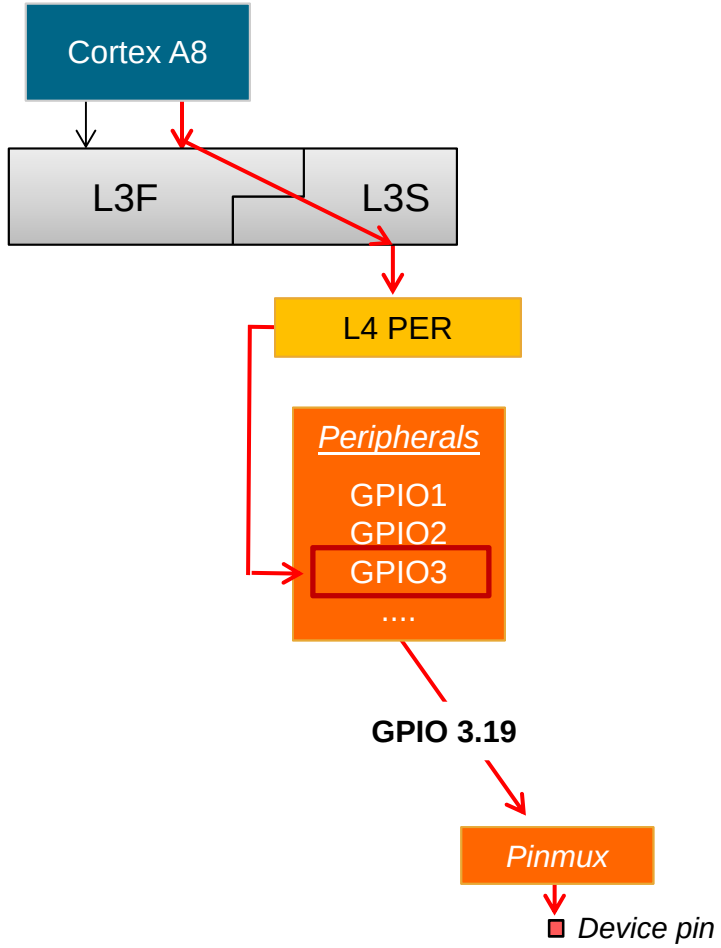
Special Registers (R30 and R31)

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

Instruction RAM

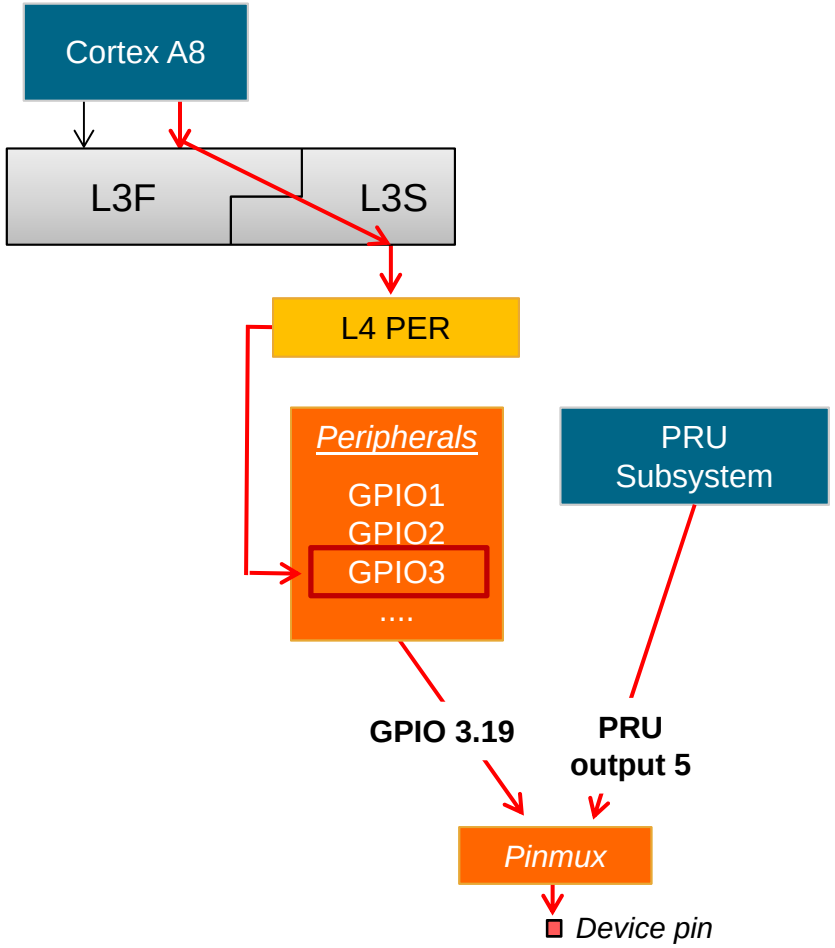
- Typical size is a multiple of 4KB (or 1K Instructions)
- Can be updated with PRU reset

Fast I/O Interface



Fast I/O Interface

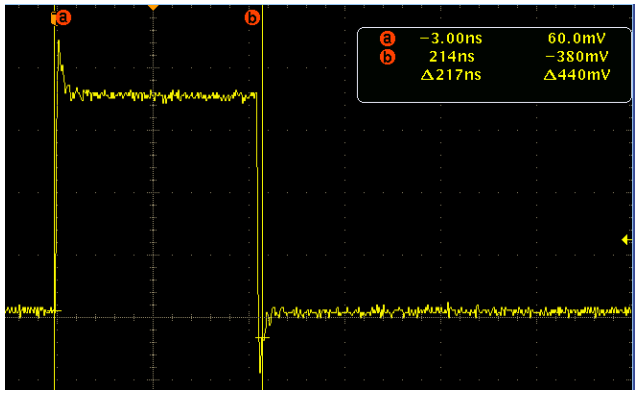
- Reduced latency through direct access to pins
  - Read or toggle I/O within a single PRU cycle
  - Detect and react to I/O event within two PRU cycles
- Independent general purpose inputs (GPIs) and general purpose outputs (GPOs)
  - PRU R31 directly reads from up to 30 GPI pins
  - PRU R30 directly writes up to 32 PRU GPOs
- 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

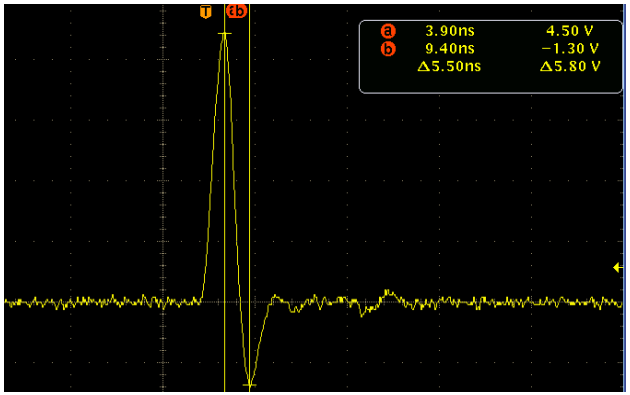
```
int main(){

    // 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:

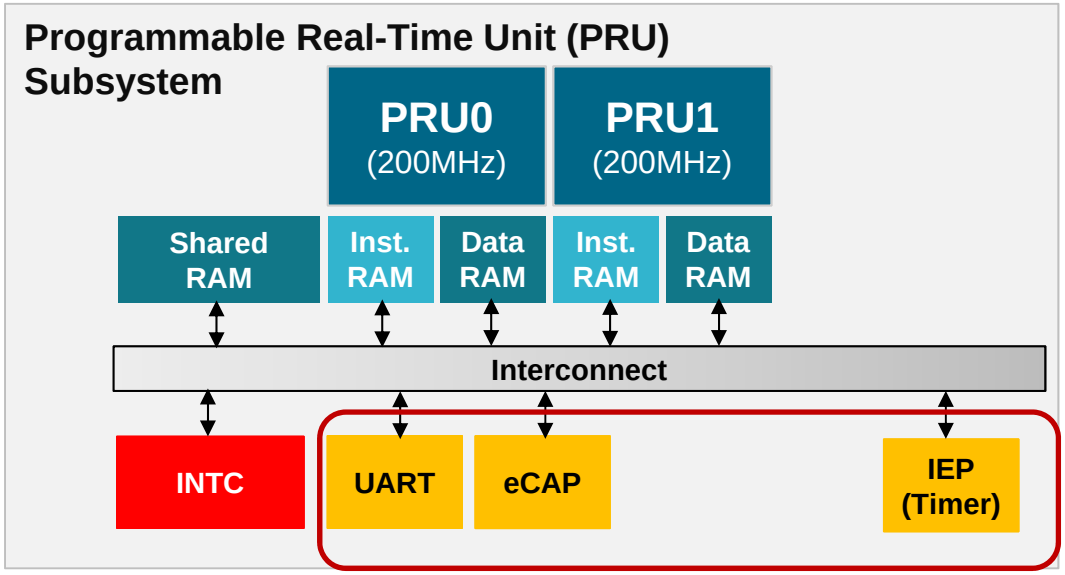
```
.origin 0
.entrypoint PRU_GPIO_TOGGLE


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 don't 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)



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 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 *(PRU cycles @ 200MHz)*	Global MMR Access *(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

Sitara Device Comparison

Features	AM18x	AM335x	AM437x	
	PRUSS	PRU-ICSS1	PRU-ICSS1	PRU-ICSS0
Number of PRU cores	2	2	2	2
Max Frequency	CPU freq / 2	200 MHz	200 MHz	200 MHz
IRAM size (per PRU core)	4 KB	8 KB	12 KB	4 KB
DRAM size (per PRU core)	512 B	8 KB	8 KB	4 KB
Shared DRAM size	0 KB	12 KB	32 KB	0 KB
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	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
GPI Pins (PRU0, PRU1)	30, 30	17, 17	13, 0	20, 20
GPO Pins (PRU0, PRU1)	32, 32	16, 16	12, 0	20, 20
MPY/MAC	N	Y	Y	Y
Scratchpad	N	Y (3 banks)	Y (3 banks)	N
INTC	1	1	1	1
Peripherals	n/a	Y	Y	Y
UART	0	1	1	1
eCAP	0	1	1	not pinned out
IEP	0	1	1	not pinned out
MII_RT	0	2	2	not pinned out
MDIO	0	1	1	not pinned out

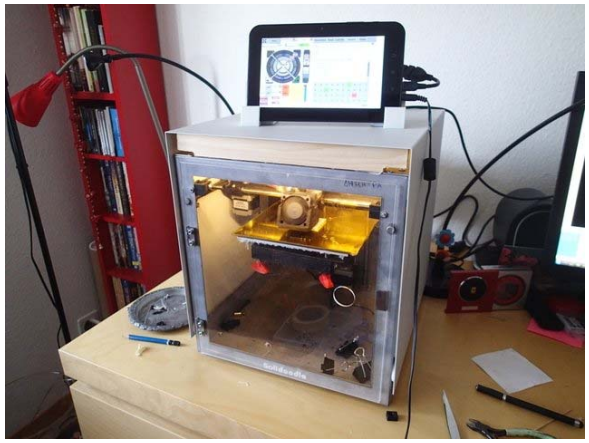
Examples of how people have used the PRU…

Use Cases 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

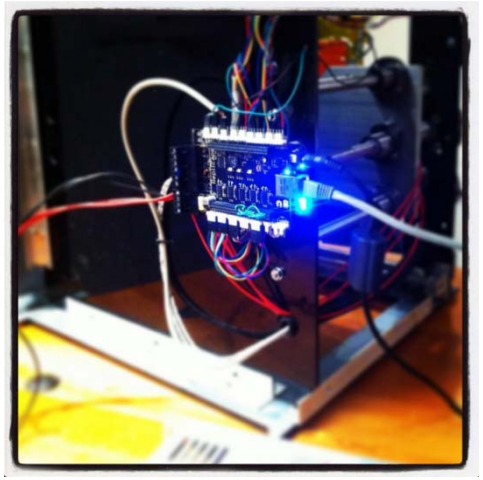
- Development complexity
- Technical constraints (i.e. running Linux on PRU)



Development Complexity

Replicape 3D Printer

- Replicate 3D Printer uses AM335x on BeagleBone
  - Cortex-A8 runs Linux, networking, HMI, model processing
    - Host apps written in Python
  - PRU controls step and direction of 5 stepper motors
    - App written in PRU assembly
- A8 calculates data, PRU communicates with motors
  - Shared region of DDR reserved for A8/PRU communication
  - Data consist of pin/delay timing tuples (8 bytes each)
- Sequence:
  1. GPIO pins are set – one or more of the 32-bit GPIO banks set with a predefined mask
  2. Delay is applied (# of 200MHz instructions)
  3. After sequence completes, PRU sends a signal to the host indicating that the segment is finished
  4. Host updates its memory usage for the PRU
- More info @ hipstercircuits.com



Thank you!

For more information about the PRU, visit:

Presentation Home – www.ti.com/sitarabootcamp

PRU-ICSS Wiki – http://processors.wiki.ti.com/index.php/PRU-ICSS

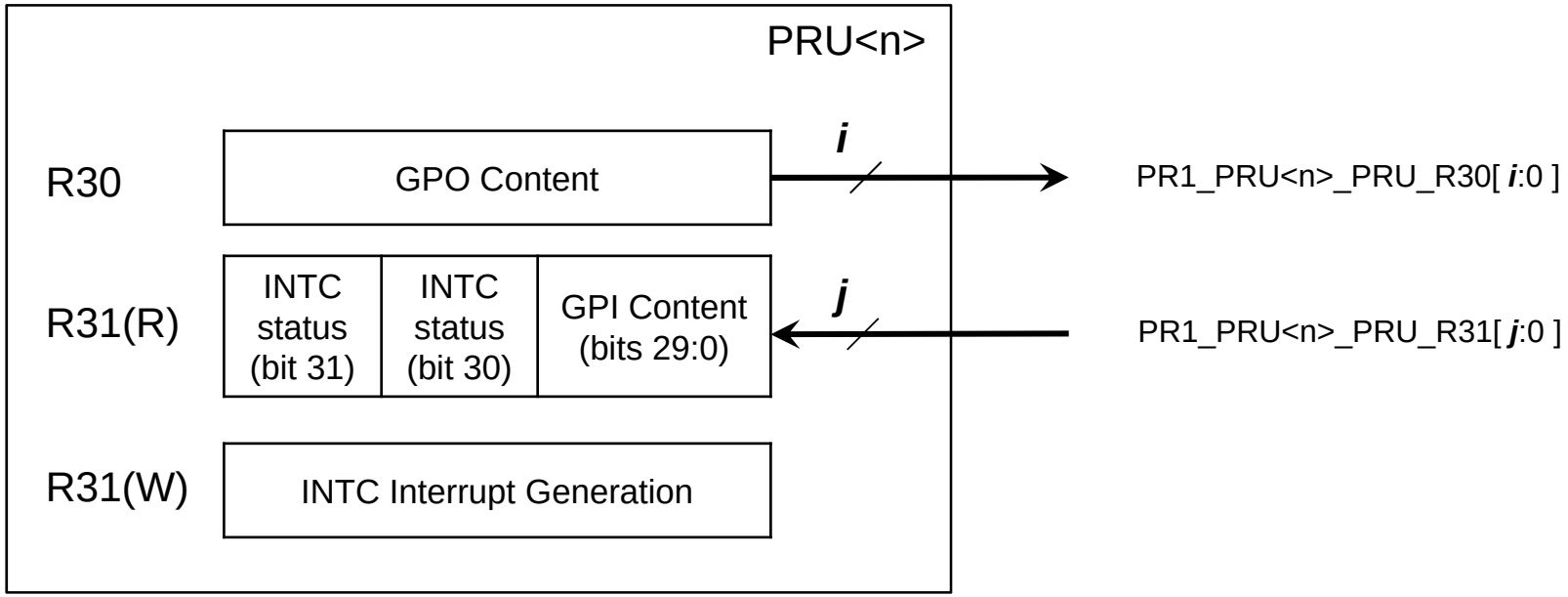
PRU Evaluation Hardware – http://www.ti.com/tool/PRUCAPE

Support – http://e2e.ti.com

Backup Slides

PRU Event/Status Register (R31)

- Writes: Generate output events to the INTC.
  - Write the event number (0 through 15) to PRU_R31_VEC[3:0] (R31 bits 3:0) and simultaneously set PRU_R31_VEC_VALID (R31 bit 5) to create a pulse to INTC.
  - Outputs from both PRUs are ORed together to form single output.
  - Output events 0 through 15 are connected to system events 16 through 31 on INTC.
- Reads: Return Host 1 & 0 interrupt status from INTC and general purpose input pin status.



PRU-ICSS Enhanced GPIO Signals

GPI Signals

Function	Signal Name	PRU Reg Mapping
Direct Input Mode		
Data input	PRU<n>_GPI	pru<n>_r31 [29:0]
Parallel Capture Mode		
Data input	PRU<n>_DATAIN	pru<n>_r31 [15:0]
Clock	PRU<n>_CLOCK	pru<n>_r31 [16]
Shift In Mode		
Data input	PRU<n>_DATAIN	pru<n>_r31 [0]
Shift counter	PRU<n>_CNT_16	pru<n>_r31 [28]
Start bit detection	PRU<n>_GPI_SB	pru<n>_r31 [29]

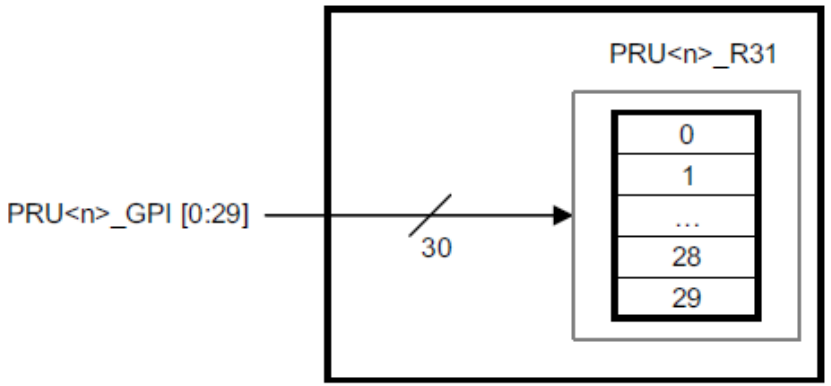
GPO Signals

Function	Signal Name	PRU Reg Mapping
Direct Output Mode		
Data output	PRU<n>_GPO	pru<n>_r30 [31:0]
Shift Out Mode		
Data output	PRU<n>_DATAOUT	pru<n>_r30 [0]
Clock	PRU<n>_CLOCK	pru<n>_r30 [1]
Load gpo_sh0	PRU<n>_LOAD_GPO_SH0	pru<n>_r30 [29]
Load gpo_sh1	PRU<n>_LOAD_GPO_SH1	pru<n>_r30 [30]
Enable shift	PRU<n>_ENABLE_SHIFT	pru<n>_r30 [31]

Direct Input / Output Modes

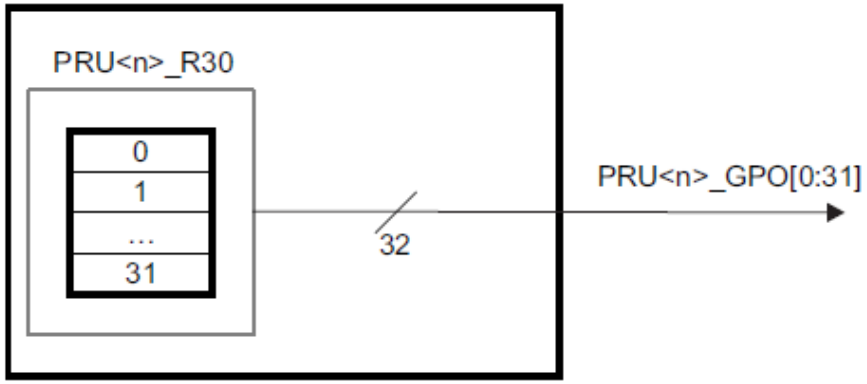
Direct Input

– PRU<n> R31[16:0] feed directly into the PRU



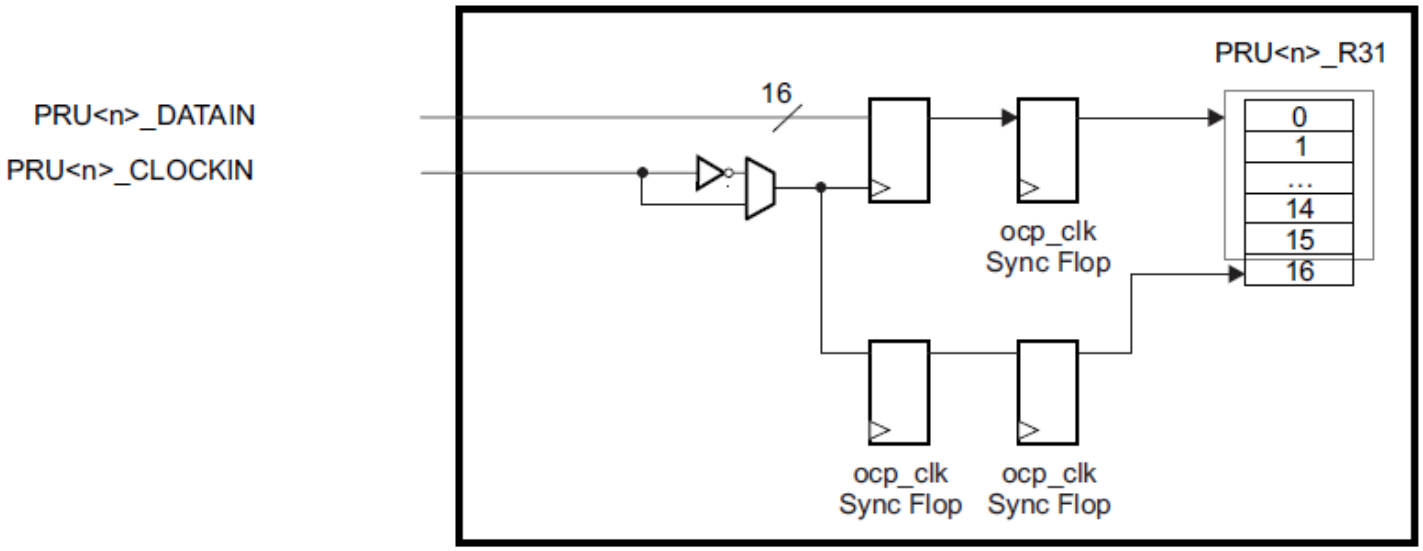
Direct Output

– PRU<n> R30[15:0] feed directly out of the PRU



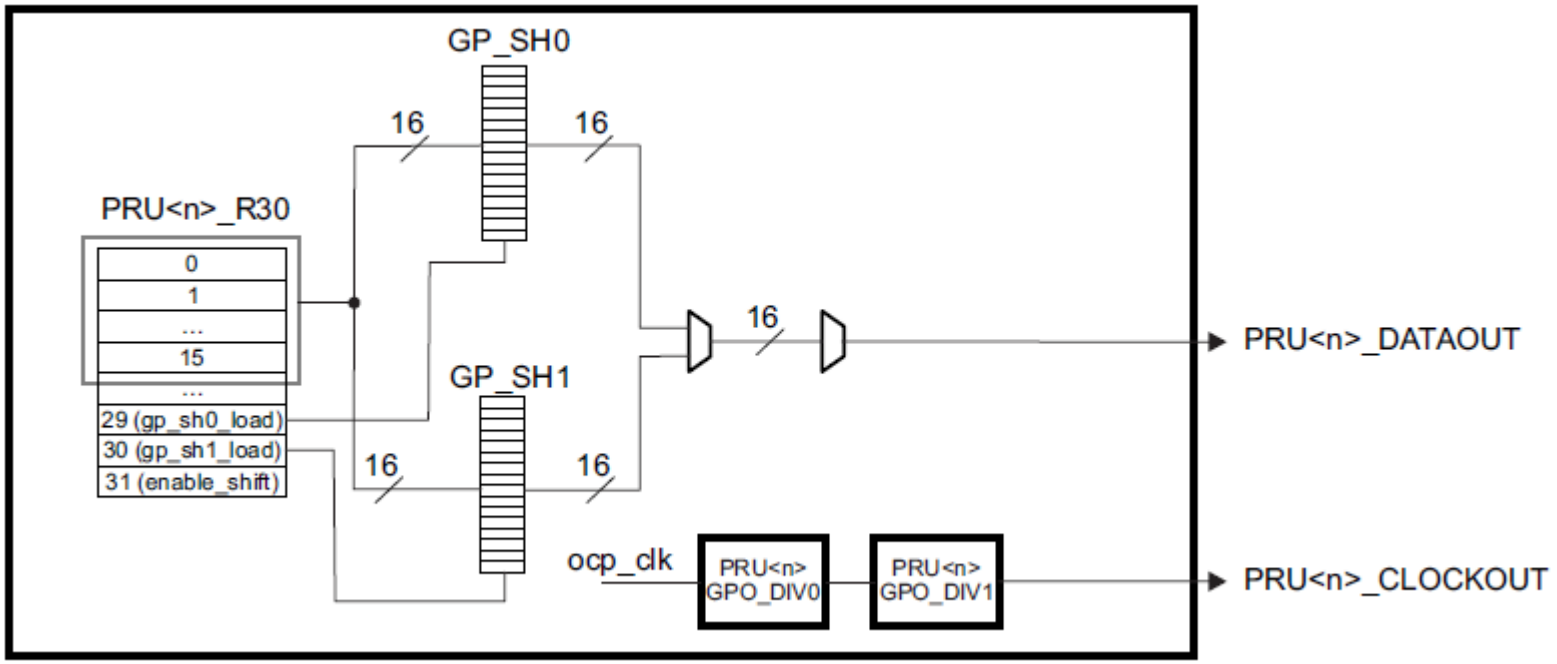
Shift In Mode

- PRU<n> R31[0] is sampled and shifted into a 28-bit shift register.
  - Shift Counter (Cnt_16) feature uses pru<n>_r31_status [28]
  - Start Bit detection (SB) feature uses pru<n>_r31_status [29]
- Shift rate controlled by effective divisor of two cascaded dividers applied to the 200MHz clock.
  - Each cascaded dividers is configurable through the PRU-ICSS CFG to a value of {1,1.5, …, 16}.



Shift Output Mode

- PRU<n> R30[0] is shifted out on every rising edge of the internal PRU<n>_CLOCK (pru<n>r30 [1]).
- Shift rate is controlled by the effective divisor of two cascaded dividers applied to the 200MHz clock. See Shift Input Mode.



Parallel Capture Mode

- PRU<n>_R31 [15:0] is captured by posedge or negedge of PRU<n>_CLOCK (pru<n>_r31_status [16]).