自作(CPU+OS)
つくってみた！

@筑波大学キャンパスOJT 秋学期成果発表会

作成したもの

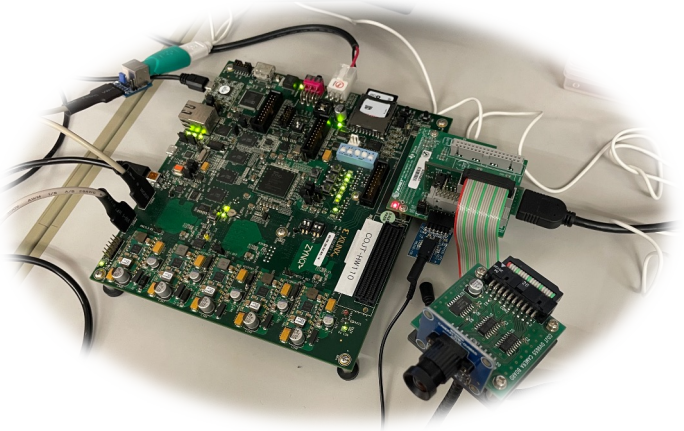
✓ 自作CPU

- ISA : RISC-V (rv32i)
- 5段パイプライン

  → 命令フェッチ / デコード / 実行 / リード / ライト
  IF　　ID　　EX　　RD　　WR
- 50MHzで動作

✓ 自作OS

- 自作CPU上で動作



RISC-V ？

RISC-V
自由でオープンな RISC 命令セットアーキテクチャ
Instruction Set Architecture (ISA)

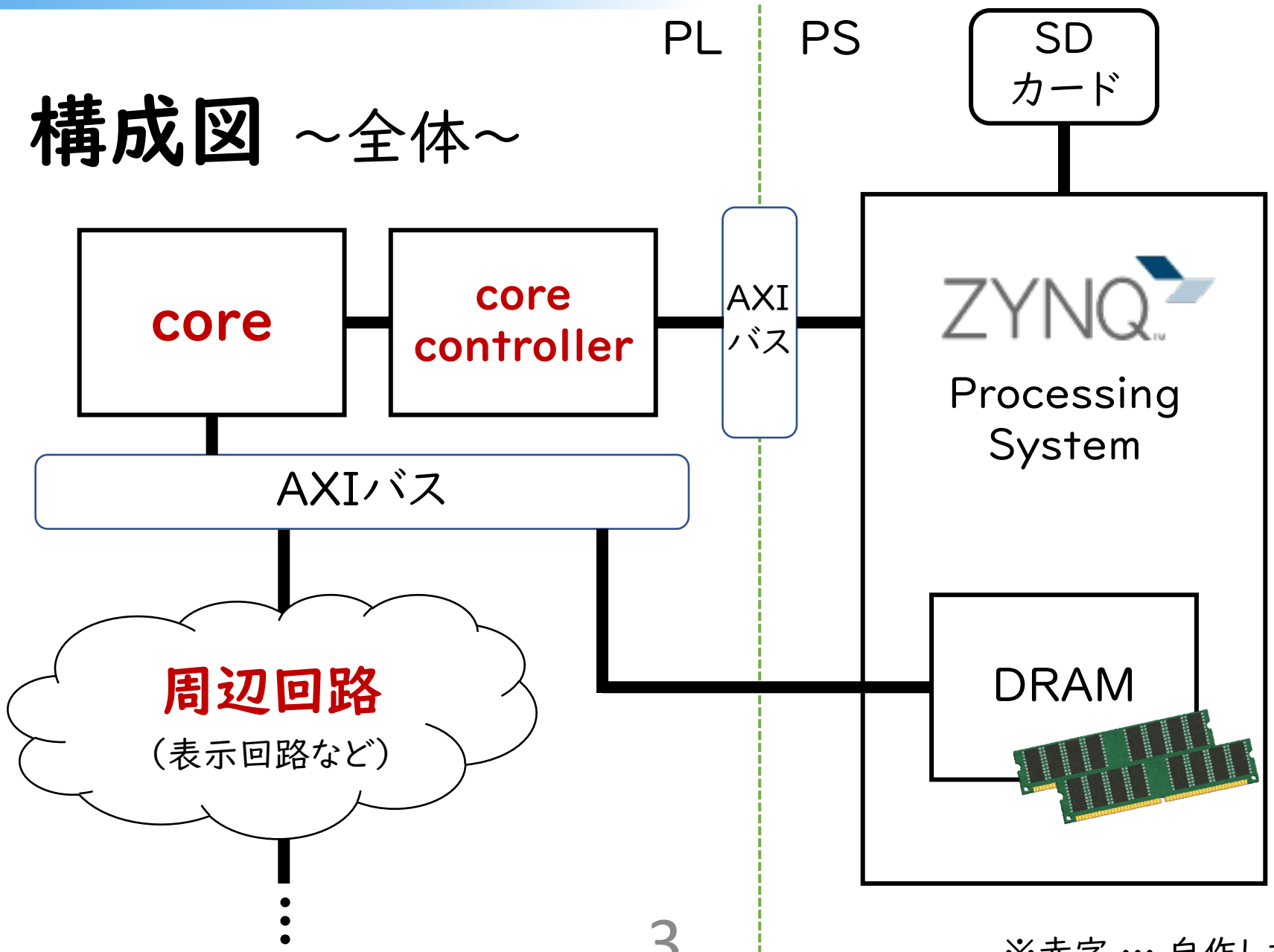
オープンなコラボレーションによるイノベーション
領域と業界を横断して設計の自由を提供
半導体業界を融合する戦略的プラットフォーム

https://riscv.org/japan/ より

- 仕様が全て公開されていて扱いやすい
- GNU Toolchain など周辺環境の充実度が魅力的　etc...

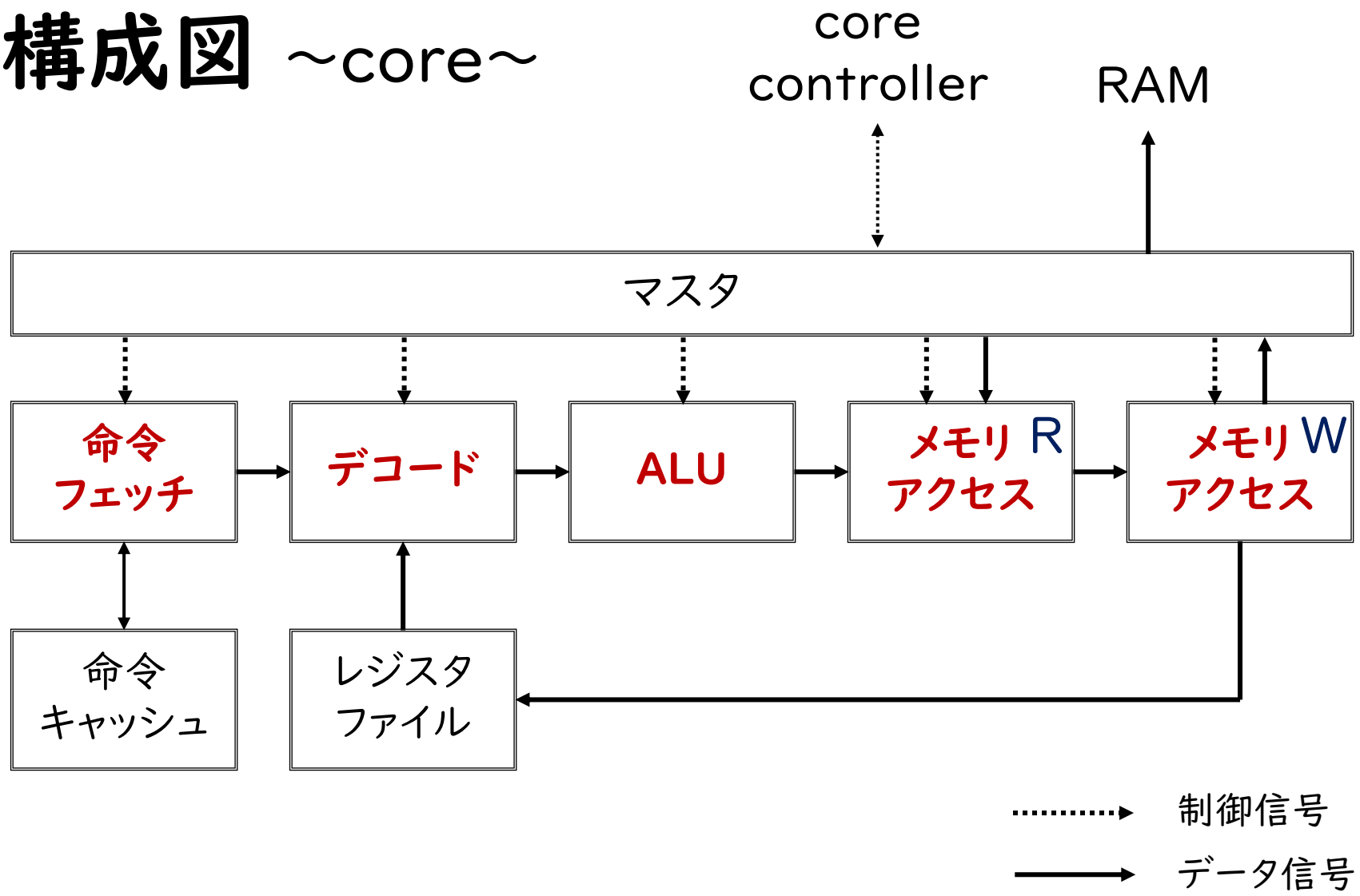
→ **初めての自作CPUに適している**

構成図 ～全体～

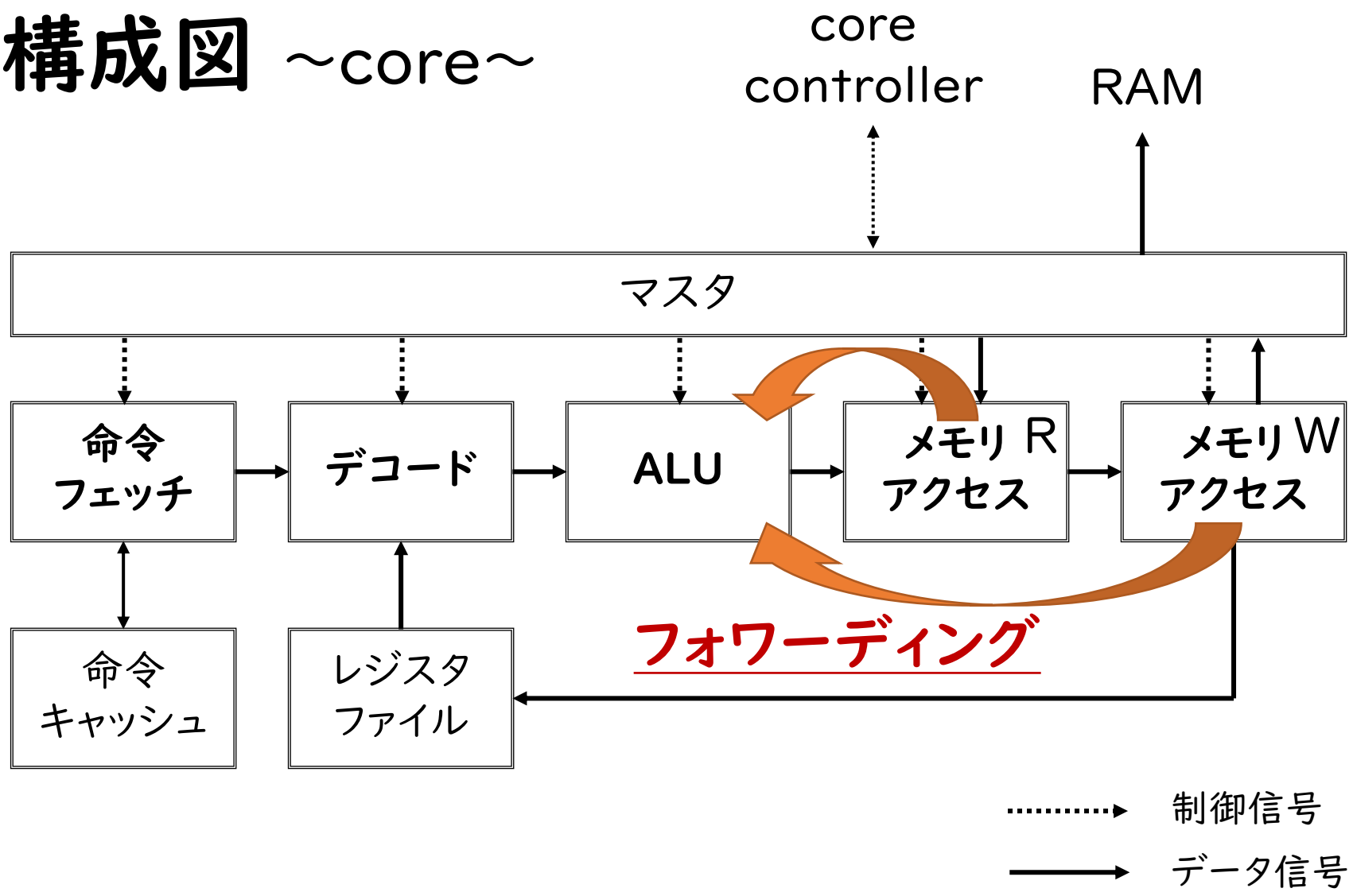


※赤字 … 自作した箇所

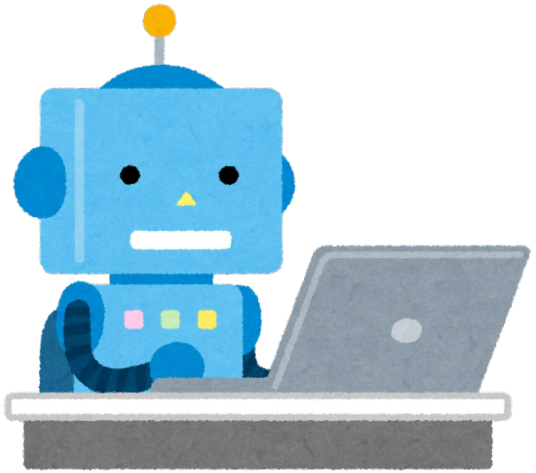
構成図 ～core～


core
controller
RAM
マスタ
命令
フェッチ
デコード
ALU
メモリ R
アクセス
メモリ W
アクセス
命令
キャッシュ
レジスタ
ファイル
制御信号
データ信号

構成図 ～core～


core
controller
RAM
マスタ
命令
フェッチ
デコード
ALU
メモリ R
アクセス
メモリ W
アクセス
命令
キャッシュ
レジスタ
ファイル
フォワーディング
制御信号
データ信号

デモ



アピールポイント

✓フルスクラッチで**自作(CPU+OS)を達成**
- それなりに動きます

✓この一年間で**作った回路全てを使用！**
- 表示回路
- キャプチャ回路
- サウンド回路
- 描画回路

出来なかったこと

✓ キャッシュ
- 命令キャッシュはOK
- データキャッシュは未着手のまま…

✓ 割り込み
- CLINT / PLIC 実装の負担が大きく断念

もっとやりたいこと

✓ 単体動作の実現
- プログラムのロード部分をARMコアに頼っている
- UART or SDカードからの起動ができるように…

✓ ネットワーク通信
- 簡単なHTTPサーバが建ったら面白い

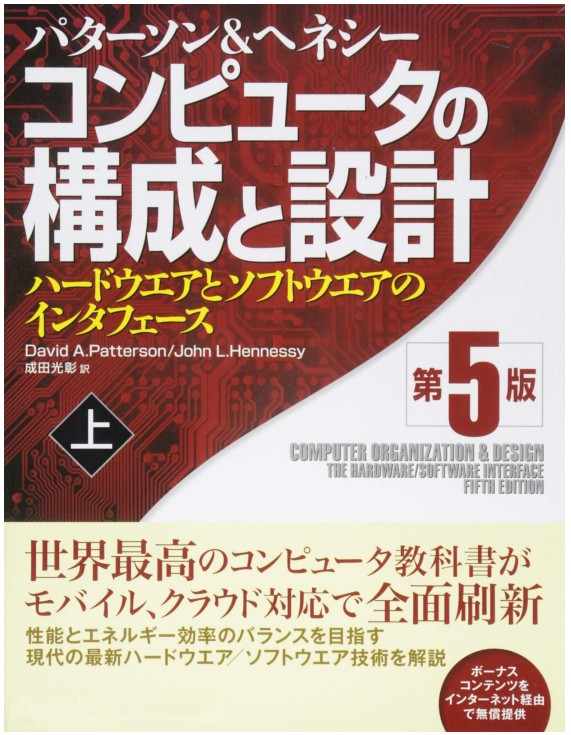
ソースコード

Yuta I 004/COJTHW-CPUProj

Yuta I 004/COJTHW-SiCOS

参考資料

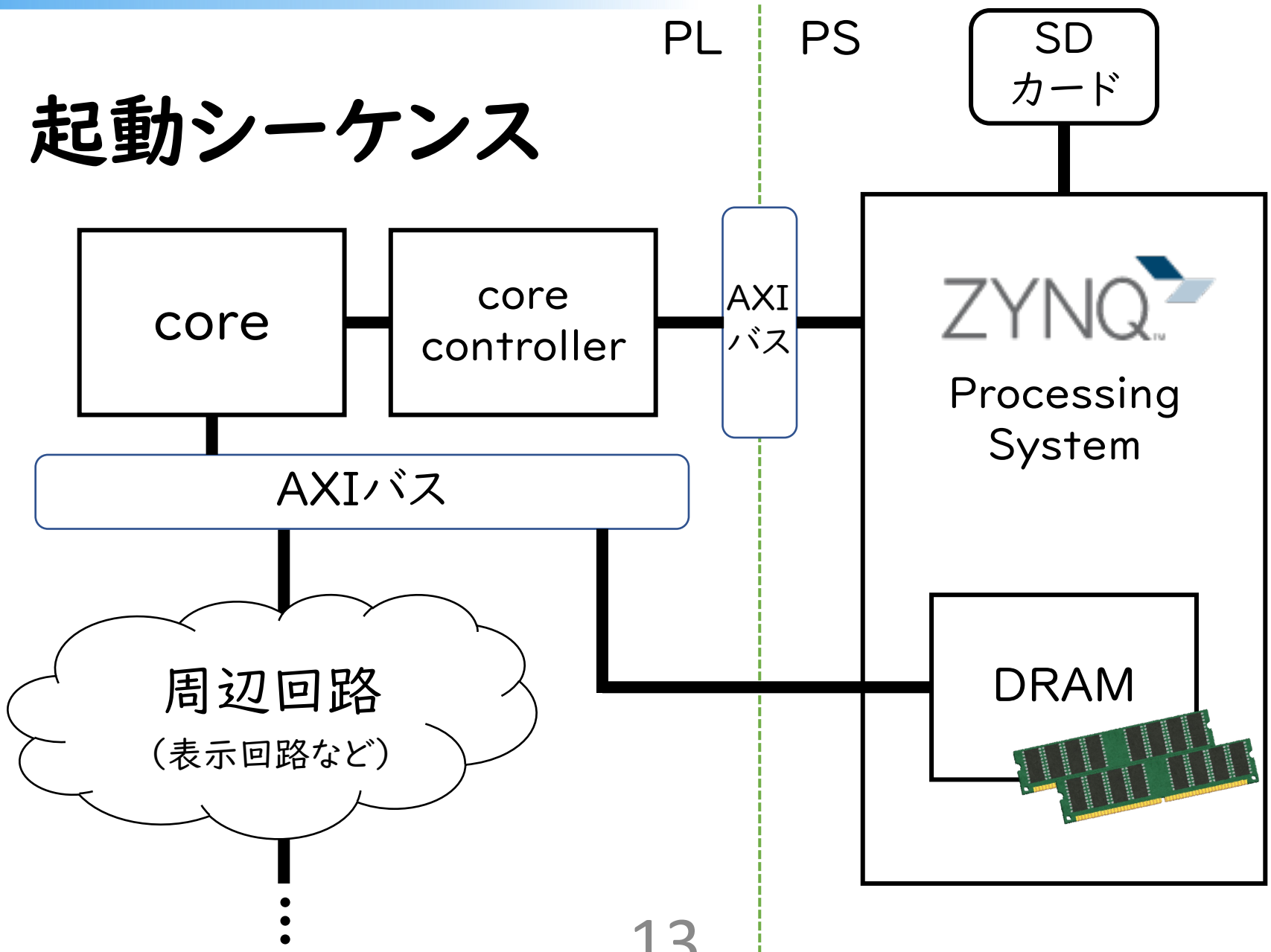
オープンアーキテクチャのススメ
RISC-V 原典



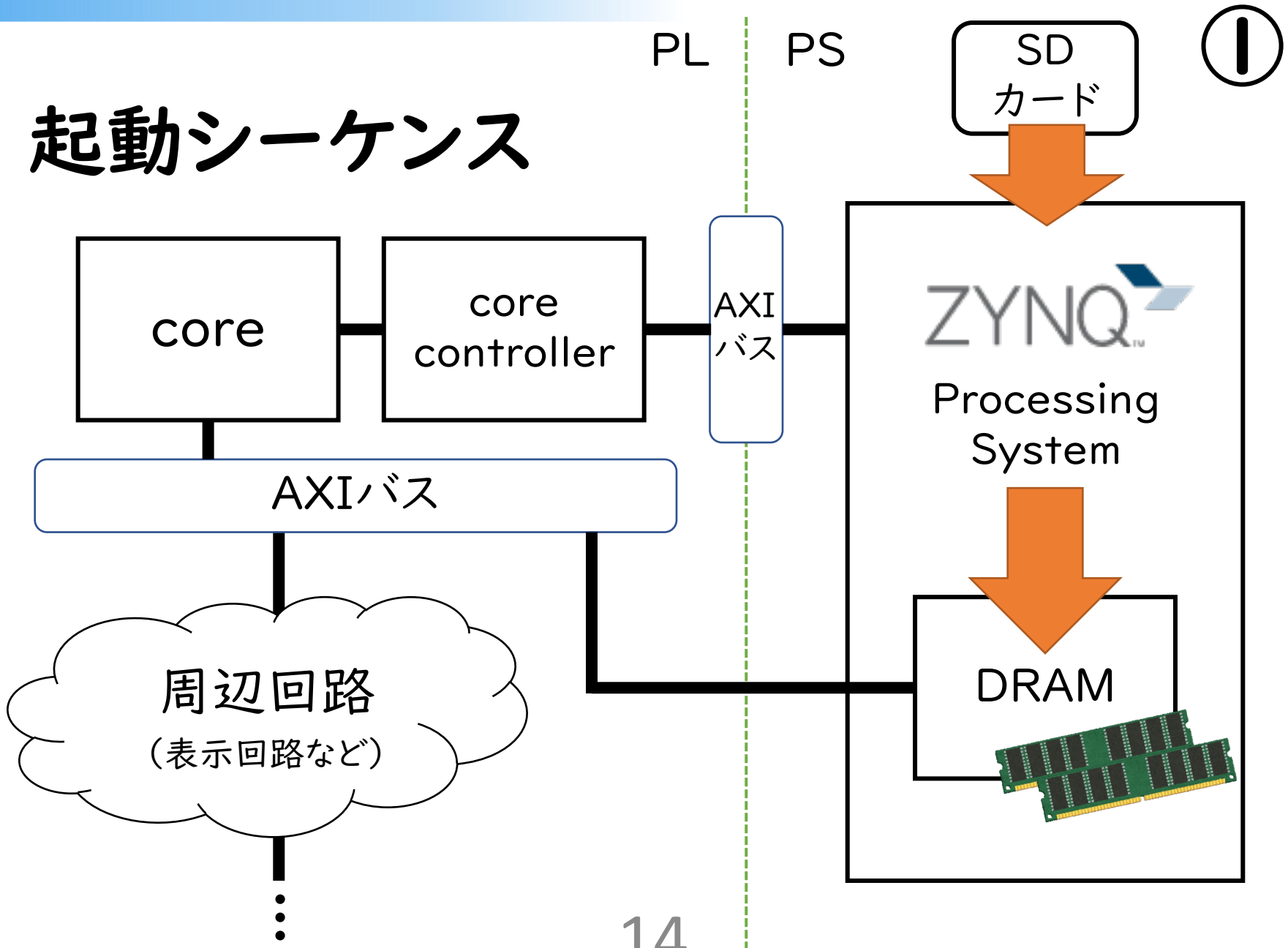
コンピュータの構成と設計
第5版

もっと詳しく!

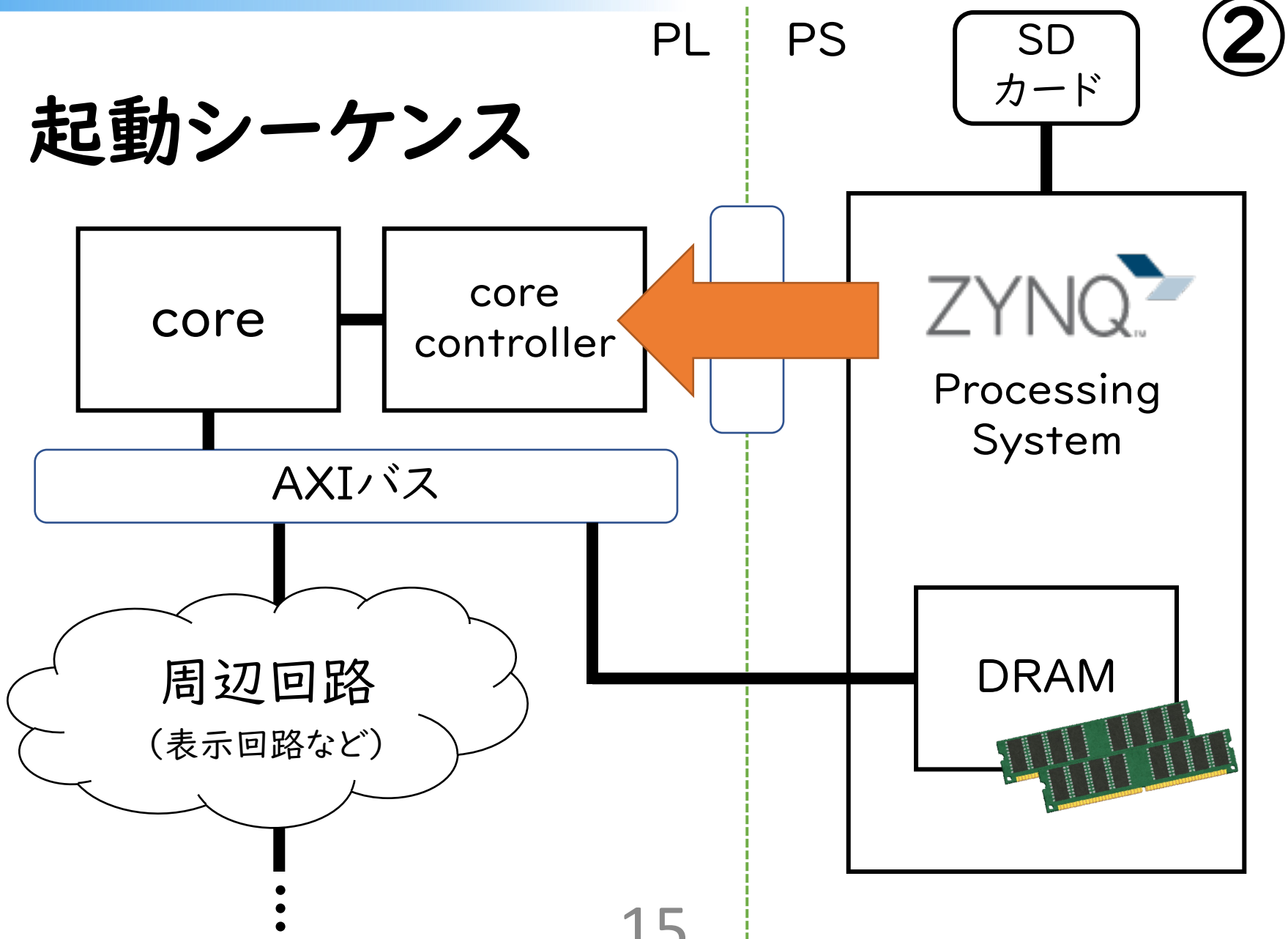
起動シーケンス


PL
PS
SD
カード
core
core
controller
AXI
バス
ZYNQ
Processing
System
AXIバス
周辺回路
（表示回路など）
DRAM

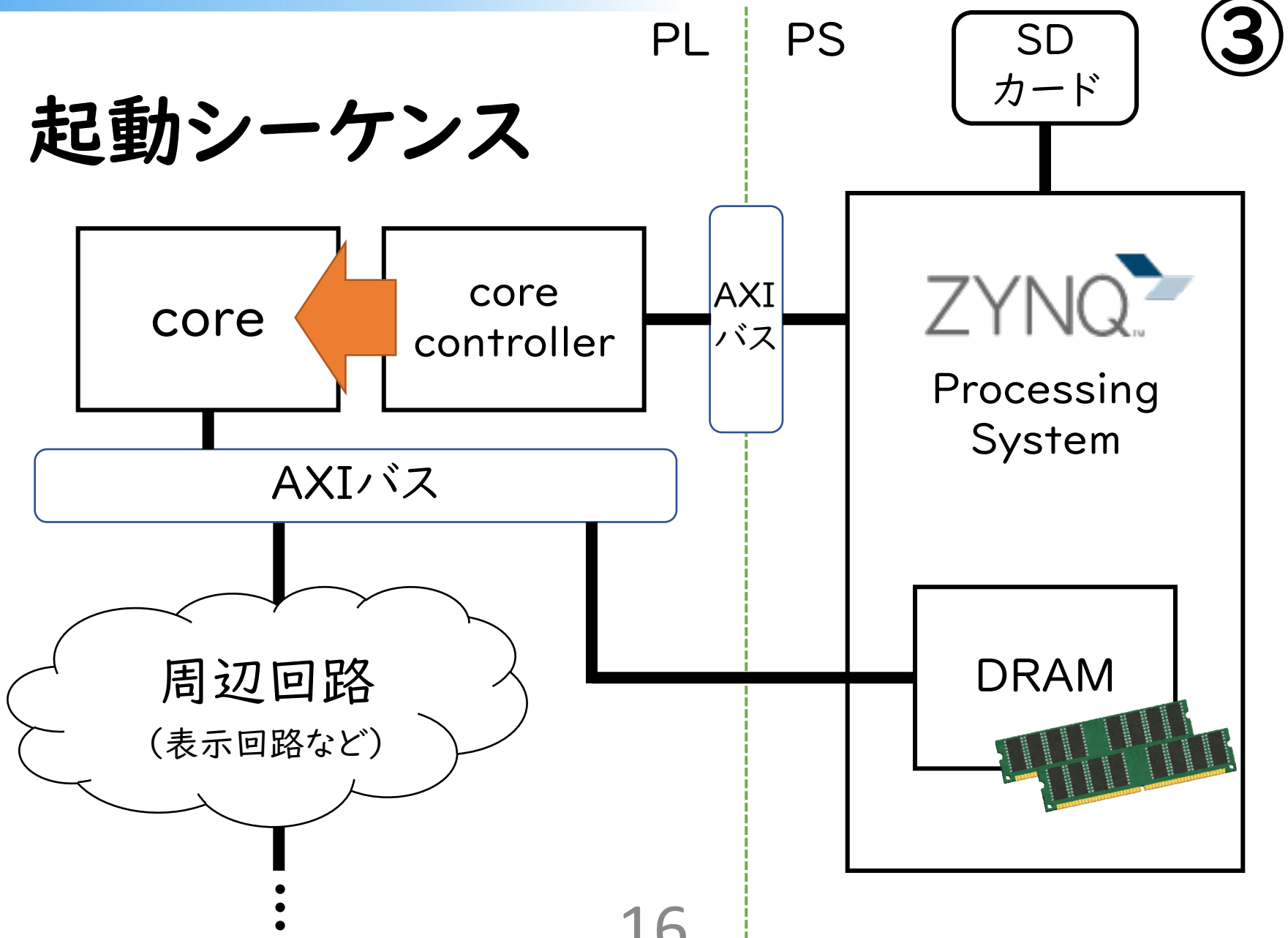
起動シーケンス


PL
PS
SD
カード
①
core
core
controller
AXI
バス
ZYNQ
Processing
System
AXIバス
周辺回路
（表示回路など）
DRAM

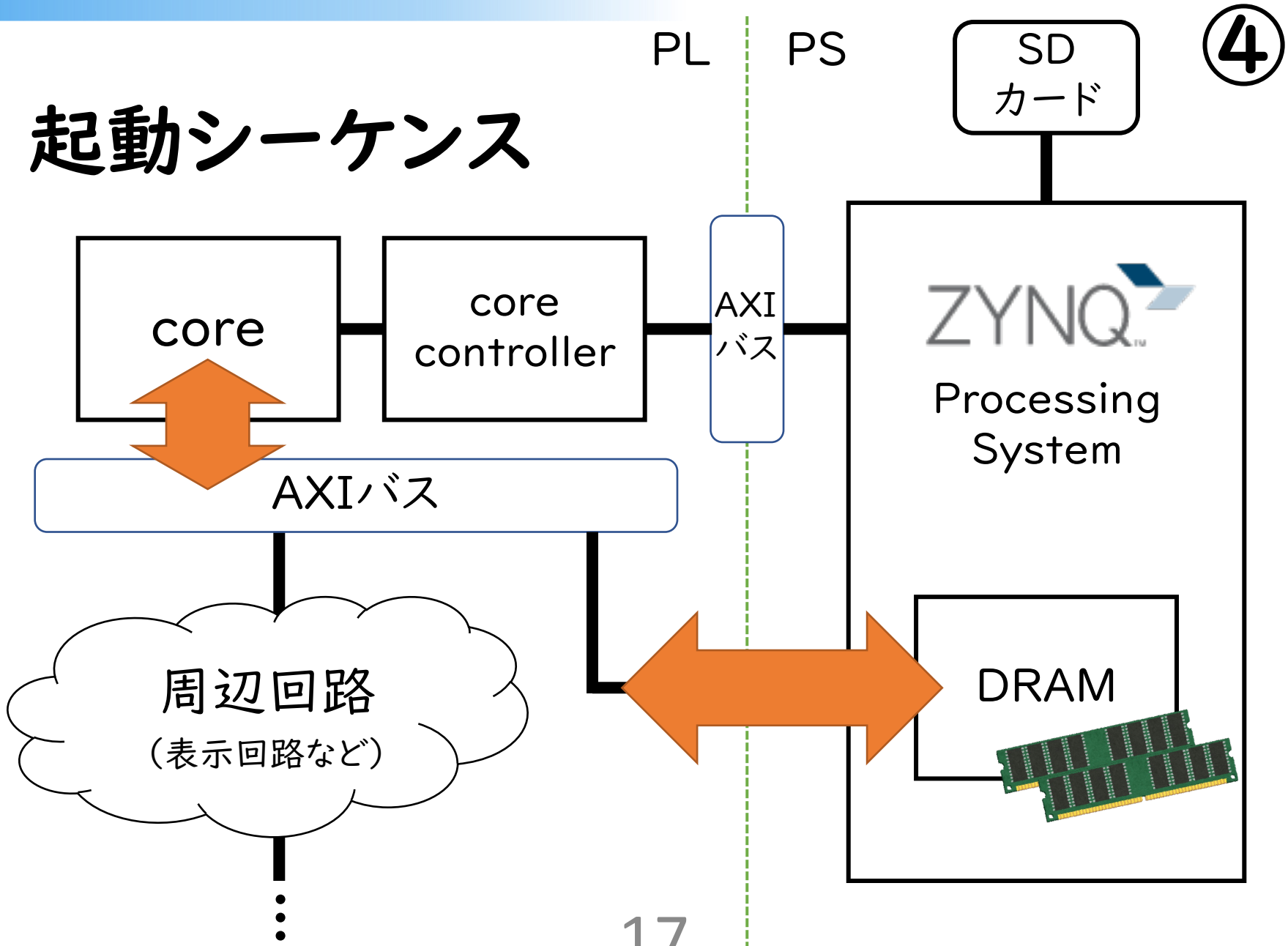
起動シーケンス


PL
PS
SD
カード
②
core
core
controller
ZYNQ
Processing
System
AXIバス
周辺回路
（表示回路など）
DRAM

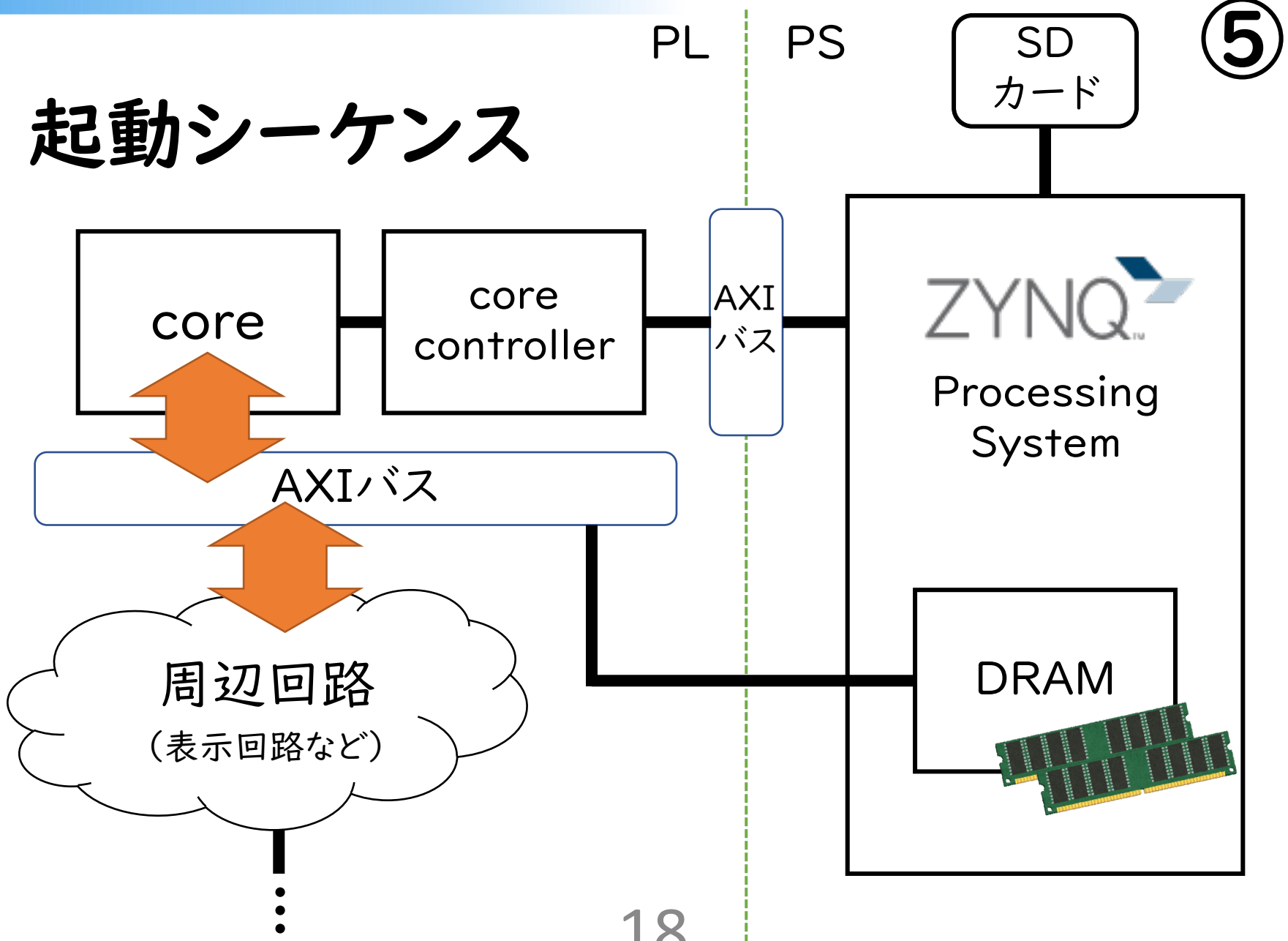
起動シーケンス


PL
PS
SD
カード
③
core
core
controller
AXI
バス
ZYNQ
Processing
System
AXIバス
周辺回路
（表示回路など）
DRAM

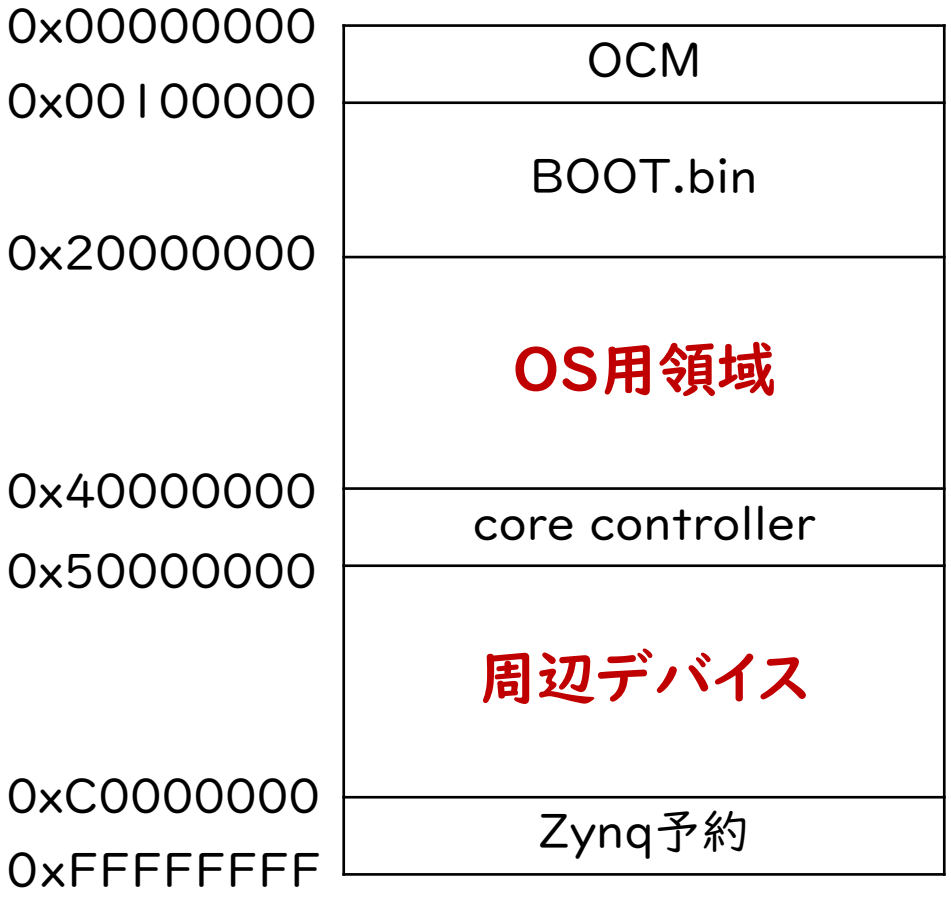
起動シーケンス


PL
PS
SD
カード
④
core
core
controller
AXI
バス
ZYNQ
Processing
System
AXIバス
周辺回路
（表示回路など）
DRAM

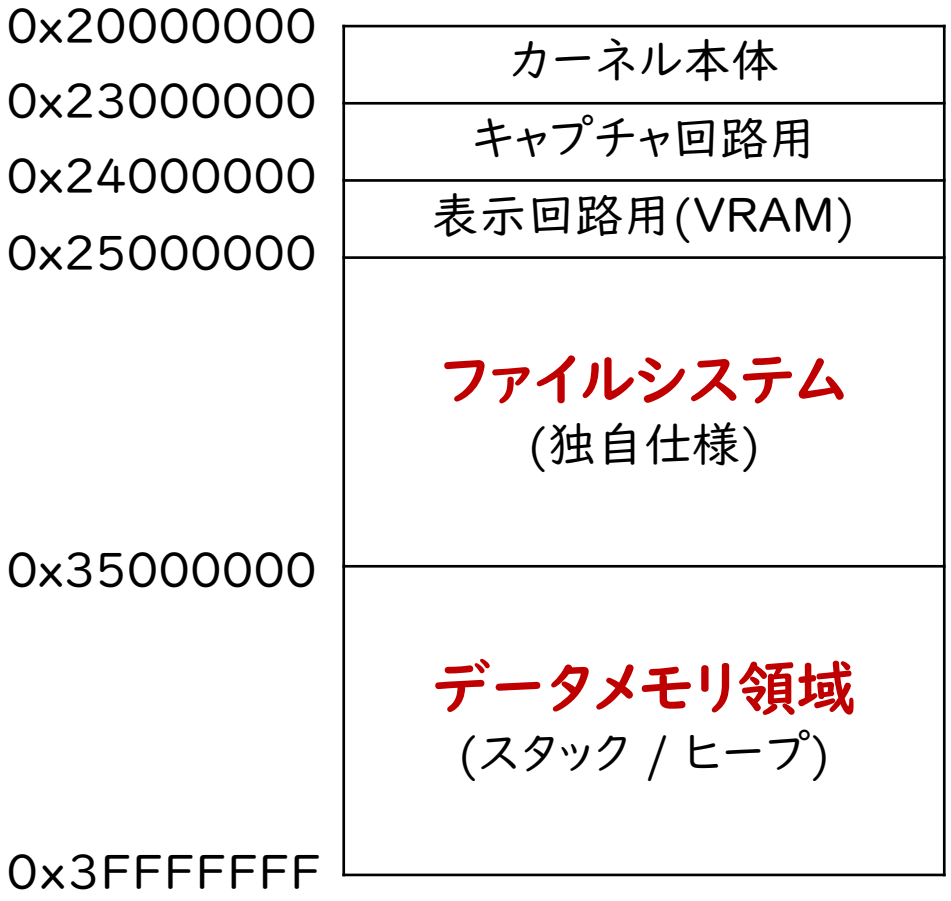
起動シーケンス


PL
PS
SD
カード
⑤
core
core
controller
AXI
バス
ZYNQ
Processing
System
AXIバス
周辺回路
（表示回路など）
DRAM

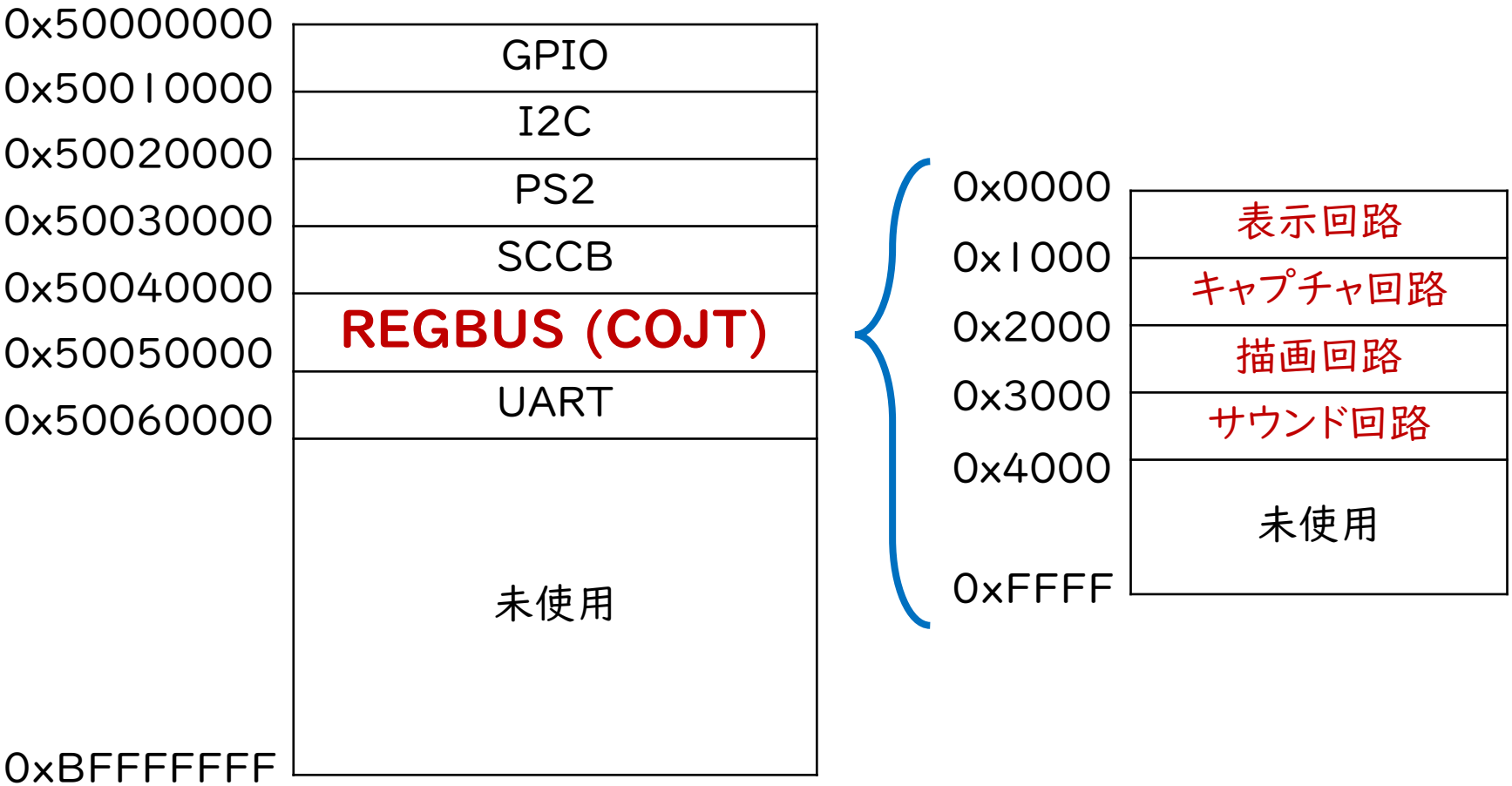
メモリマップ ～全体～


0x00000000
0x00100000
0x20000000
0x40000000
0x50000000
0xC0000000
0xFFFFFFFF
OCM
BOOT.bin
OS用領域
core controller
周辺デバイス
Zynq予約

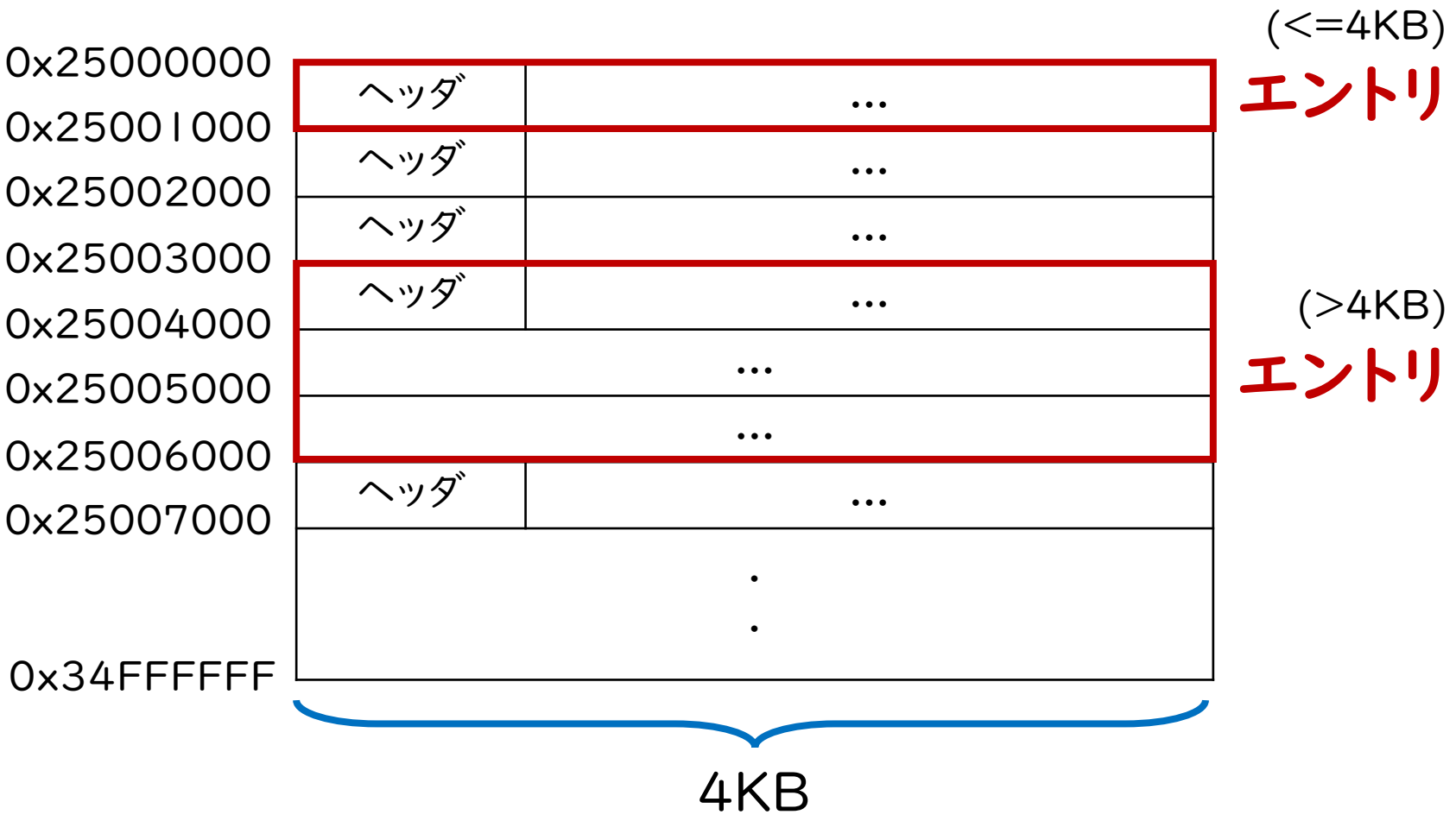
メモリマップ ～OS用領域～



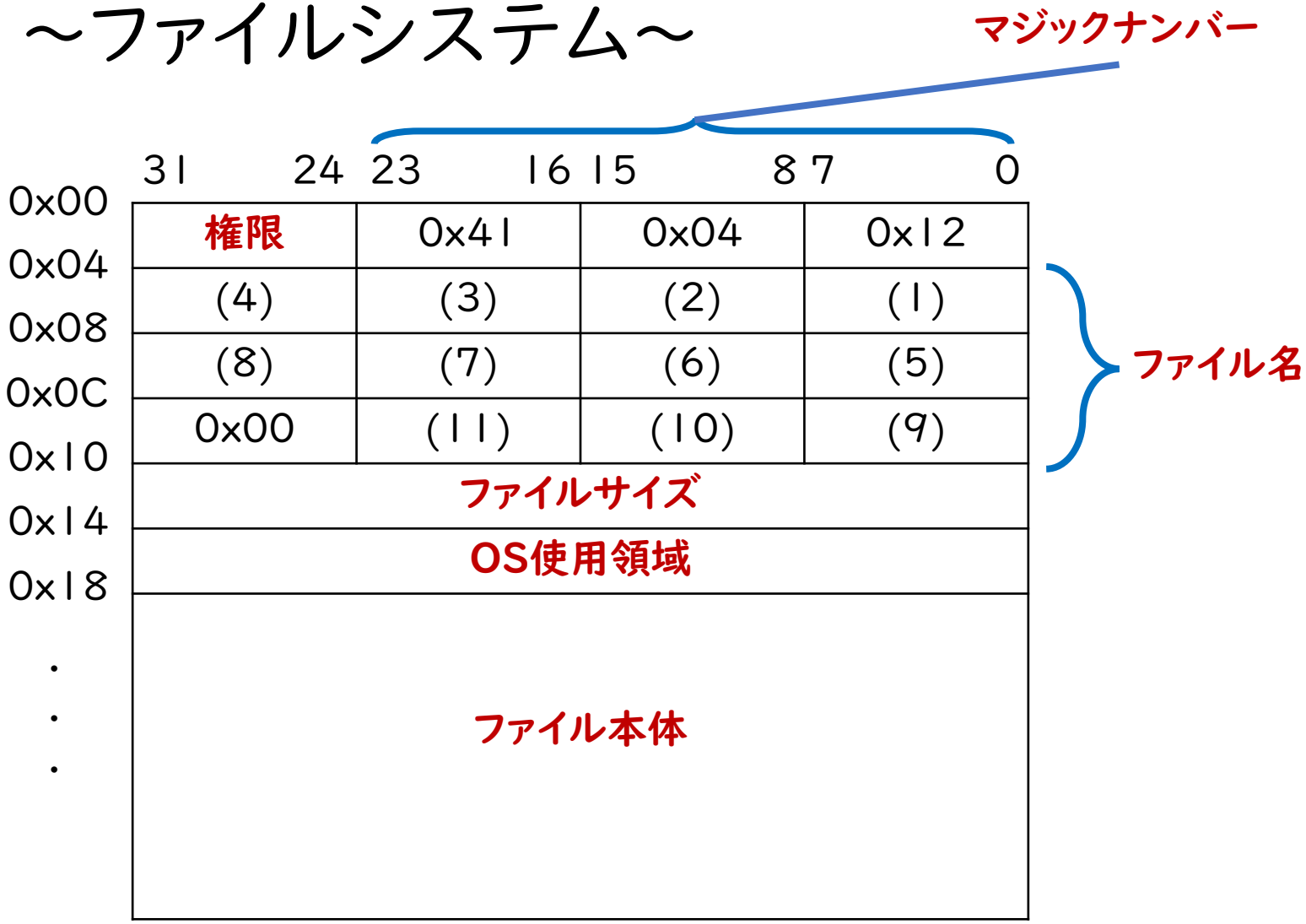
メモリマップ ～周辺デバイス～



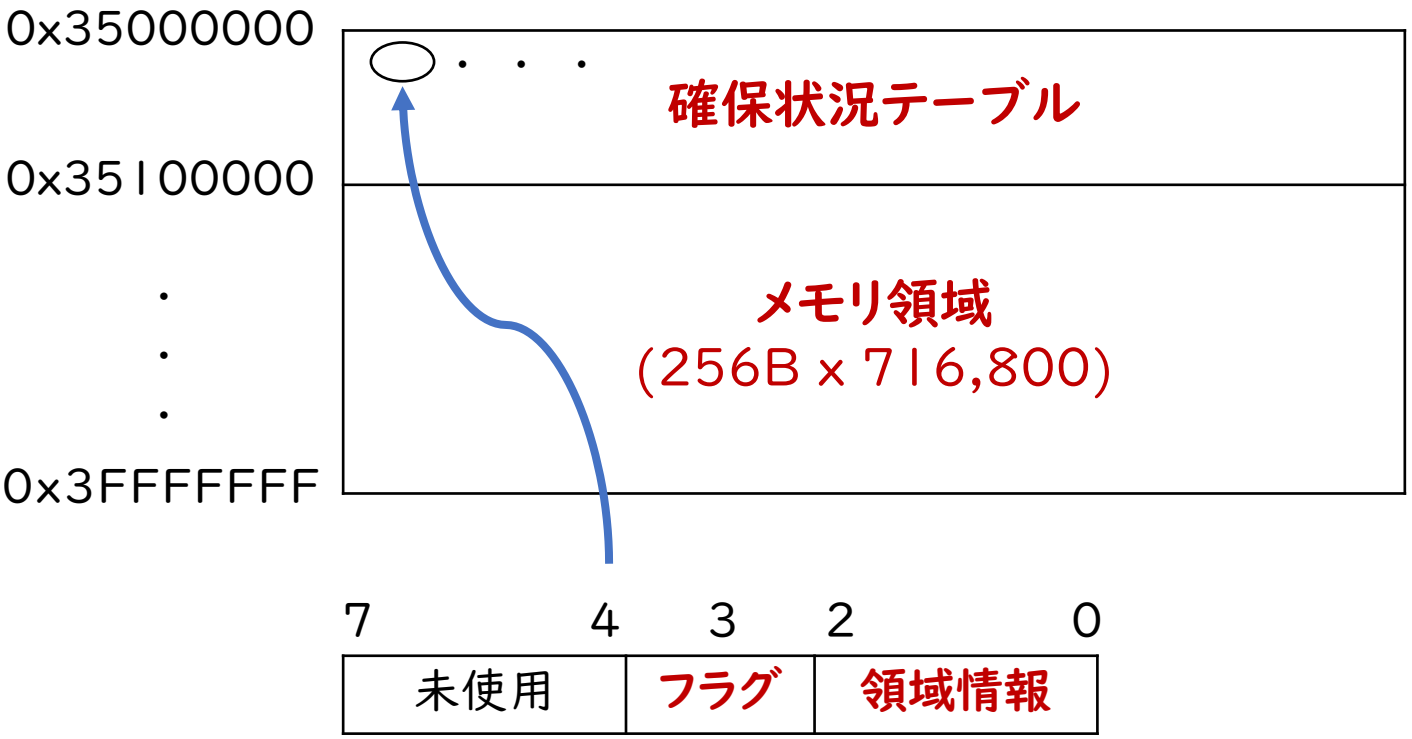
OS ～ファイルシステム～


(<=4KB)
エントリ
(>4KB)
エントリ
0x25000000
0x25001000
0x25002000
0x25003000
0x25004000
0x25005000
0x25006000
0x25007000
0x34FFFFFF
ヘッダ
…
4KB

OS ～ファイルシステム～


マジックナンバー
31 24 23 16 15 8 7 0
0x00
0x04
0x08
0x0C
0x10
0x14
0x18
権限
0x41
0x04
0x12
(4)
(3)
(2)
(1)
(8)
(7)
(6)
(5)
0x00
(11)
(10)
(9)
ファイル名
ファイルサイズ
OS使用領域
ファイル本体

OS ～メモリ管理～



ex) 0x35000000 → 0x3510000 ～ 0x351000FF
0x35000001 → 0x3510100 ～ 0x351001FF　を管理