Linuxを搭載したM32Rアーキテクチャ研究開発用プラットフォーム

高田 浩和、作川 守、坂本 圭、山本 整†、稲岡 一弘††、近藤 弘郁、清水 徹

三菱電機株式会社　システムLSI事業化推進センター、
†三菱電機株式会社　情報技術総合研究所、
††三菱電機セミコンダクタ・アプリケーション・エンジニアリング株式会社

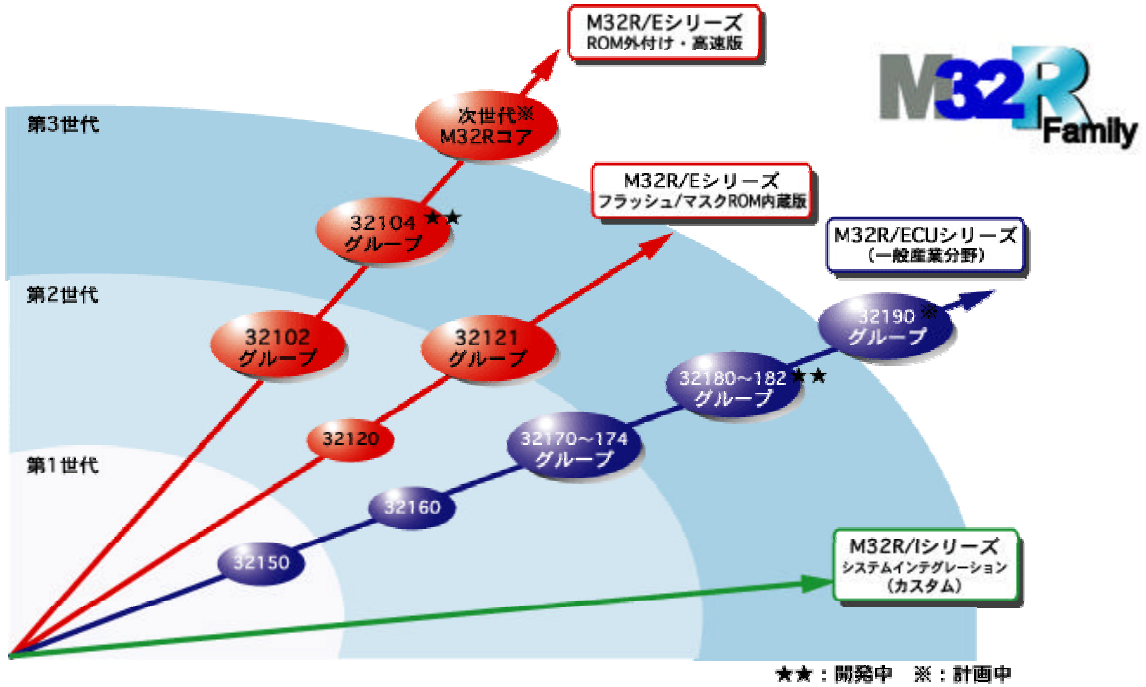
発表内容

- M32Rプロセッサの概要
- M32R組み込みシステム・プラットフォームの開発
  - M32Rソフトマクロ・コア
  - 研究開発用プラットフォーム (評価ボードMappi)
- Linux/M32Rの開発
  - Linuxカーネル移植
  - GNUツール拡張／ライブラリ移植
  - ソフトウェア・パッケージの作成

- デモンストレーション

M32Rファミリ RISCマイコン

- 組み込み用32ビットRISCプロセッサ
  - 三菱電機オリジナル32ビットRISCアーキテクチャ
  - デジタルコンシューマ機器、車載、産業機器への組み込み応用
    - デジタルスチルカメラ(DSC)、デジタルビデオカメラ(DVC)、携帯電話(PDC)など

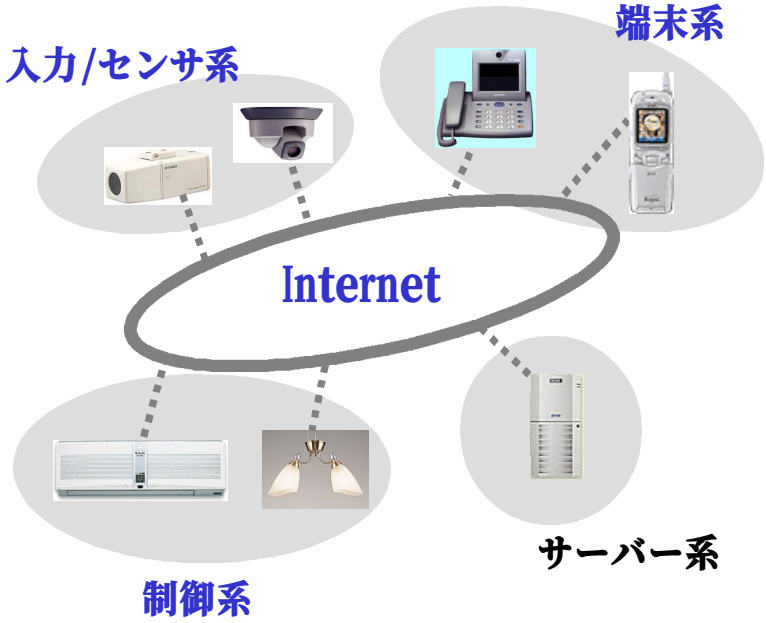


M32R研究開発用プラットフォーム

- Linuxソフトウェア・プラットフォームの開発

背景

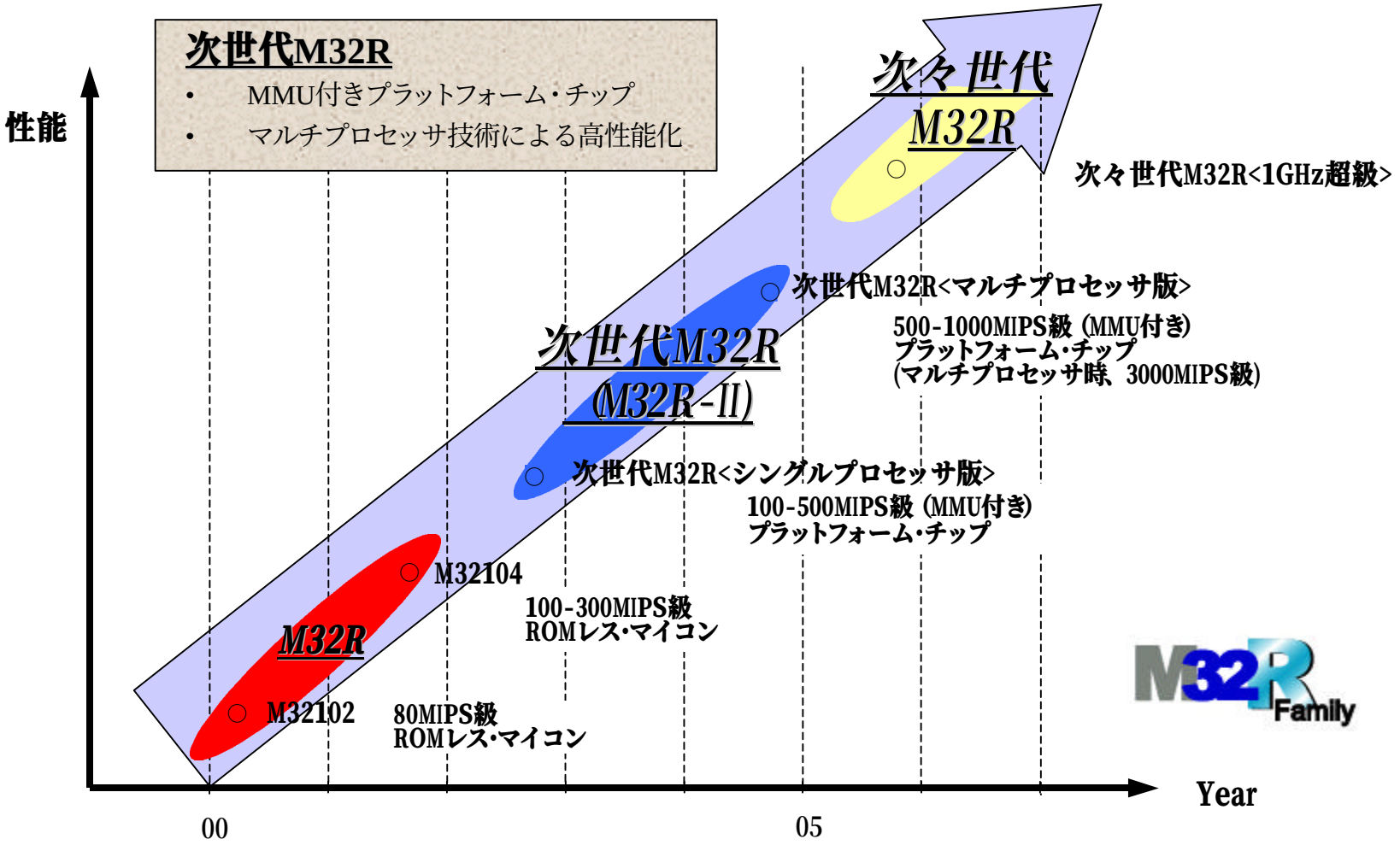
- システムLSIの高集積化
  ⇒ システム・オン・チップ(**SoC**)
- 組み込みシステムの高機能化・高性能化、ネットワーク化
  ⇒ システム構築におけるソフトウェア開発の比重が増大



目的

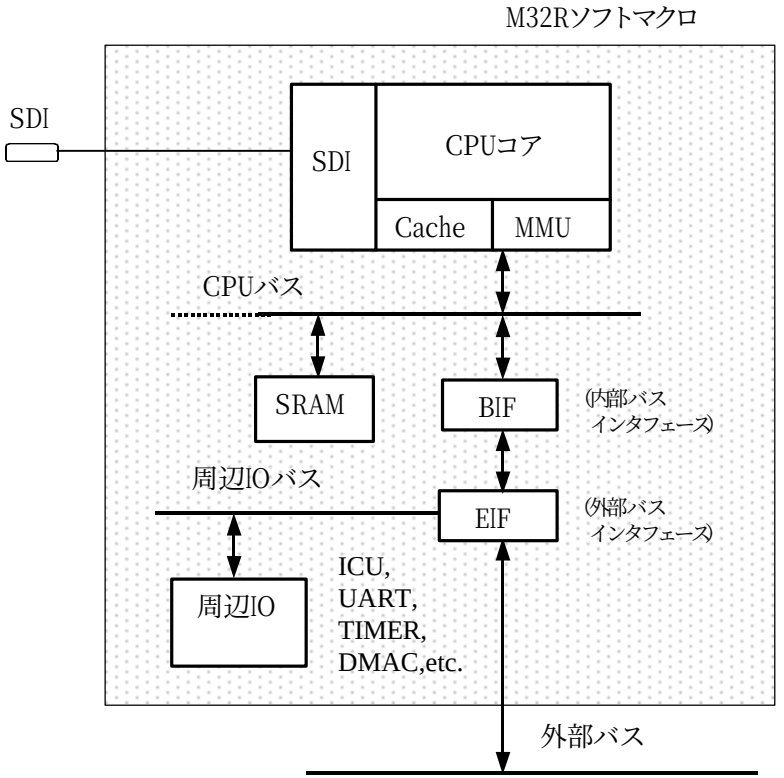
- **M32R用GNU/Linux開発環境の構築**
  デファクトなソフトウェア・プラットフォーム(Linuxなど)を利用した効率的なソフトウェア開発環境を実現
- **次世代M32Rプロセッサ・コアの開発**
  コンパクト、低消費で高性能なプロセッサ・コア

M32Rプロセッサ・コア ロードマップ



M32Rソフトマクロ・コア

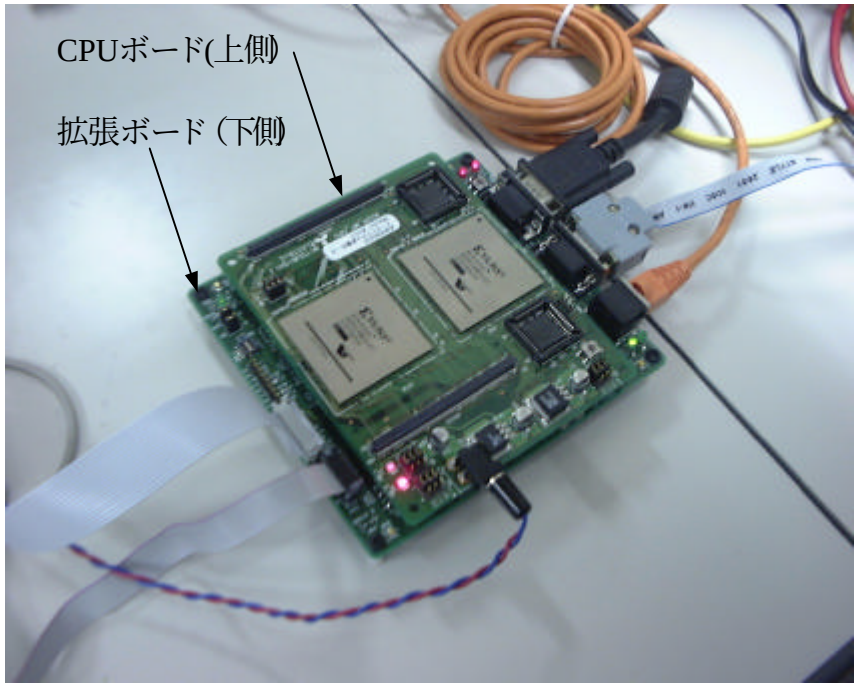
- **M32Rソフトマクロ**
  - 32ビットRISCプロセッサ
  - M32R-II ISA
    - 16ビット/32ビット命令形式
    - 117命令/ 6アドレッシングモード
    - 汎用レジスタ 32ビット×16
    - ２命令並列実行
    - DSP機能命令をサポート
- 特長
  - Verilog-HDLで記述
  - フル・シンセサイザブル
    - プロセステクノロジに依存しない
    - コンパクトなコア
      ⇒ FPGAにもマッピング可能



評価用ターゲットボード(Mappi)

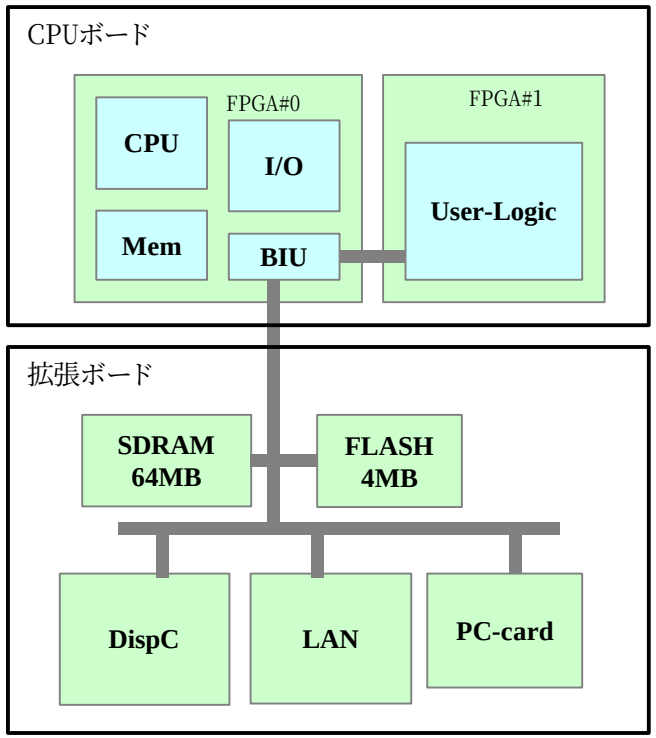
- Linuxを動作させるのに必要最小限のハードウェアを実装
- プロセッサとして、M32Rソフトマクロ・コアをマッピングしたFPGAを使用

→ 機能変更・拡張・デバッグが容易



※ CPUボード(FPGA：Xilinx XCV2000E x2)
拡張ボード(LAN、SDRAM、DispC、PCカードスロットなど)

ブロック図



M32R用Linuxソフトウェア・プラットフォーム

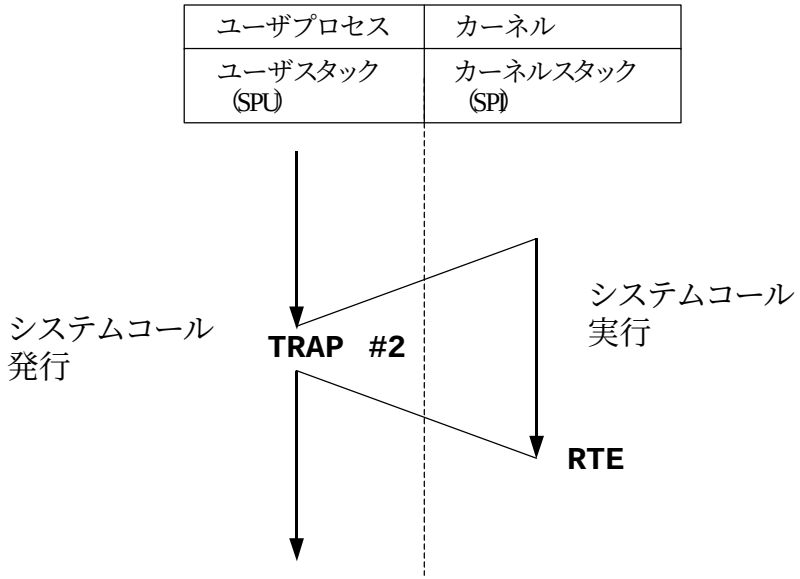
- Linux/M32R
  - M32R用GNU/Linux環境
    - スタンドアロン環境(ramdisk)、NFSRoot環境
  - SMP対応カーネル
- Linux/M32Rの開発 (2000年～)
  - 開発の流れ
    - Linuxカーネルの移植
    - GNUツール(GCC, Binutils) の拡張
    - M32Rプラットフォームの開発 (評価ボード開発)
    - GNU Cライブラリの移植
    - セルフツール環境整備、ルートファイルシステム整備

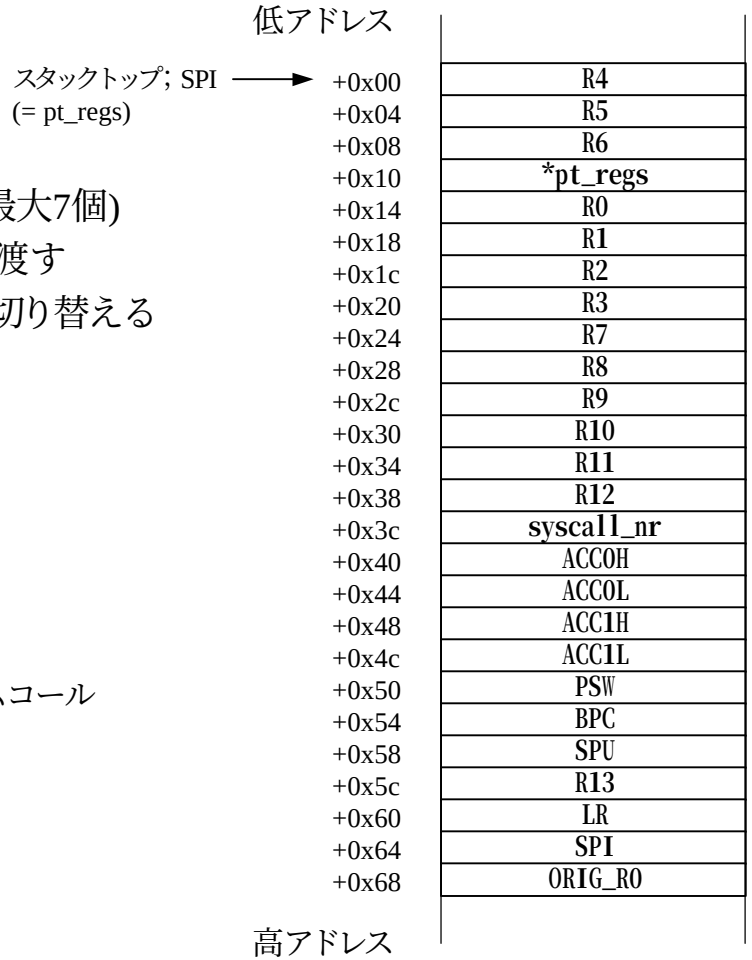
Linuxカーネルの移植

- Linuxカーネル
  - M32R用カーネル
    - 開発当初は　2.2系カーネルで開発をスタート(v2.2.16~)
    - その後、2.4系カーネルに移行(最新版 v2.4.18)、SMP対応
- アーキテクチャ・ポート
  - アーキテクチャ依存部分の移植
    - include/asm-m32r/，arch/m32r/
  - 何が問題となったか
    - LinuxカーネルではGCCの拡張機能を多用 (inline関数、asm関数)
    - ツールの完成度(ツールを並行して開発・デバッグ)
  - どのように開発を進めたか
    - ヘッダファイルの整備がある程度進まないとコンパイルすらできない
      ⇒　stubルーチンを用意して、徐々に開発
    - まずはスケジューラから(GNUシミュレータを活用(MMU非対応版))

システムコールI/F

- システムコールI/F
  - システムコール：TRAP #2
    R7：システムコール番号
    R0 ~ R6：パラメータ0 ～パラメータ6　(最大7個)
  - pt_regsを暗黙のスタックパラメータとして渡す
  - CLRPSW命令により明示的にスタックを切り替える





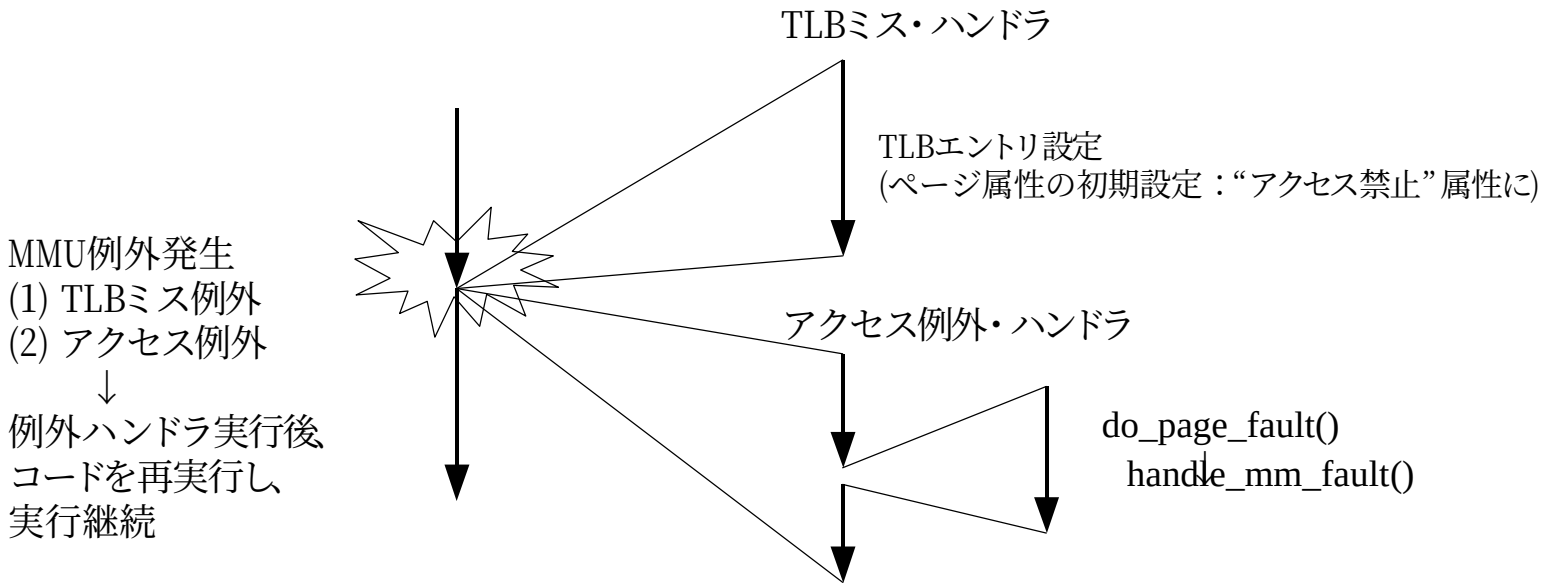
メモリ管理

- Linuxのメモリ管理 (ページング)
  - MMU例外を利用し、デマンドローディング処理／
    コピー・オン・ライト処理を実現　⇒　MMUが必須

- M32R-IIコア
  - MMU
    - ソフトウェアによりTLBを更新　(cf. MIPS)
    - FPGA版コアのTLBエントリ数は命令／データ各16エントリ
    - 4KB／ページ(Linux/M32R)、Largeページ(4MB) をサポート
  - キャッシュ
    - 命令・データ分離キャッシュ
    - 物理キャッシュ(physically indexed physically tagged cache) のため、
      キャッシュのフラッシュ処理が不要

メモリ管理：デマンドローディング処理

- 2つの例外ハンドラに処理を分割
- TLBミスハンドラの軽量化 (←TLBミスの頻度は比較的高い)
  - TLBミスハンドラではPTEの設定のみ行う
  - アセンブリ・コード化し、フル・コンテキストの退避を行わない



GNU開発環境：GNUツールの整備

- GNUツール(GCC, Binutils) の拡張
  - GCC (gcc-2.95系、gcc-3.0系)、Binutils (2.11.92系)
    - Cygnus GNUPro (組み込み用**`m32r-elf`**ツール) をベースに拡張
  - ELFのダイナミックリンク機能に対応
    - PICコード生成、共有ライブラリ･サポート
    - BFDライブラリの拡張
  - C言語ABI (Application Binary Interface) は変更せず
  - エンディアン･サポート(リトルエンディアンの新規サポート)
- クロスツール
  - Linux/x86版クロスツール(ターゲット：**`m32r-linux`**)
- セルフツール環境の整備
  - gcc, binutils, bash, sed, awk, perl, tcl, …

ELFダイナミックリンク機能への対応

- PICコード生成、共有ライブラリ・サポート
  - コンパイル・オプション：-shared -fPIC
- GOT (Global Offset Table)
  - GOTは R12レジスタ相対でアクセス
  - **BL(**branch&link)命令の実行により PC値を取得

```
    :
; PROLOGUE
push  lr
bl    .+4
ld24  r12,#_GLOBAL_OFFSET_TABLE_
add   r12,lr
    :
```

- PLT (Procedure Linkage Table) の実装
  - サブルーチン・コール時の分岐先シンボル参照はGOTの間接参照で行う (IA-32方式)
    - PLTエントリのコードフラグメントの書き換えを行わないため、特定の命令キャッシュ・ラインのインバリデート機能が不要
    - RELA形式のリロケーションを採用

ライブラリ移植

- GNU Cライブラリの移植
  - glibc-2.2.3, glibc-2.2.5
  - ダイナミック・リンカ (ld-linux.so)
    - 共有ライブラリ(ダイナミックリンク・ライブラリ)が使用可能
  - LinuxThreadライブラリの実装 (Pthread)
    - ユーザレベル排他制御
      - tasシステムコール(test and set)
      - Lamportのアルゴリズムによる実装
- 開発の流れ
  - スタティックリンク版 hello.c (newlib版→glibc版)
  - ダイナミックリンク版 hello.c, busybox, ...

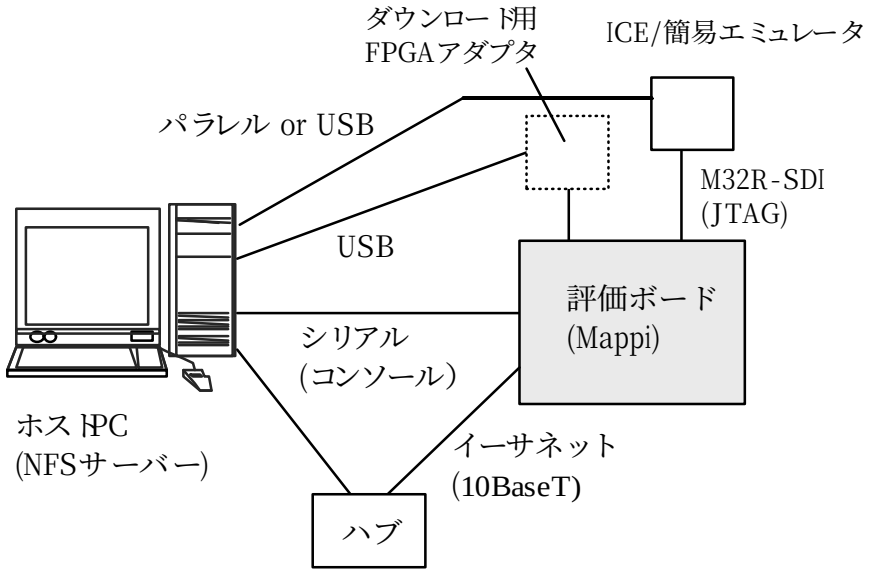
M32Rのオン・チップ・デバッグ機能

- SDI (Scalable Debug Interface)
  - M32Rファミリに共通な内蔵デバッグ機能のインタフェース仕様
  - JTAGインタフェースによるデバッグ制御
    - プログラムのダウンロード
    - モニタプログラムの実行
- 特長
  - チップやボード上にモニタプログラムを格納するためのメモリが不要
  - 高速なダウンロード
  - リアルタイムデバッグ機能のサポート(オプション)

デバッグ環境

- SDI対応GDB
  - リモートターゲット(m32rsdi) のサポート
  - SDIを用いたプログラムのダウンロード／実行／デバッグ
    - Cソースレベル
    - MMUによるアドレス変換　⇒　PCブレークの機能が必須
- デバッグの実際
  - カーネルのデバッグ
    - SDI対応GDB
    - その他：KD32R(改)、KGDB、GNUシミュレータ(MMU非対応)
  - アプリケーションのデバッグ
    - strace
    - gdbserver (ホストとethernet接続し、リモートデバッグ)

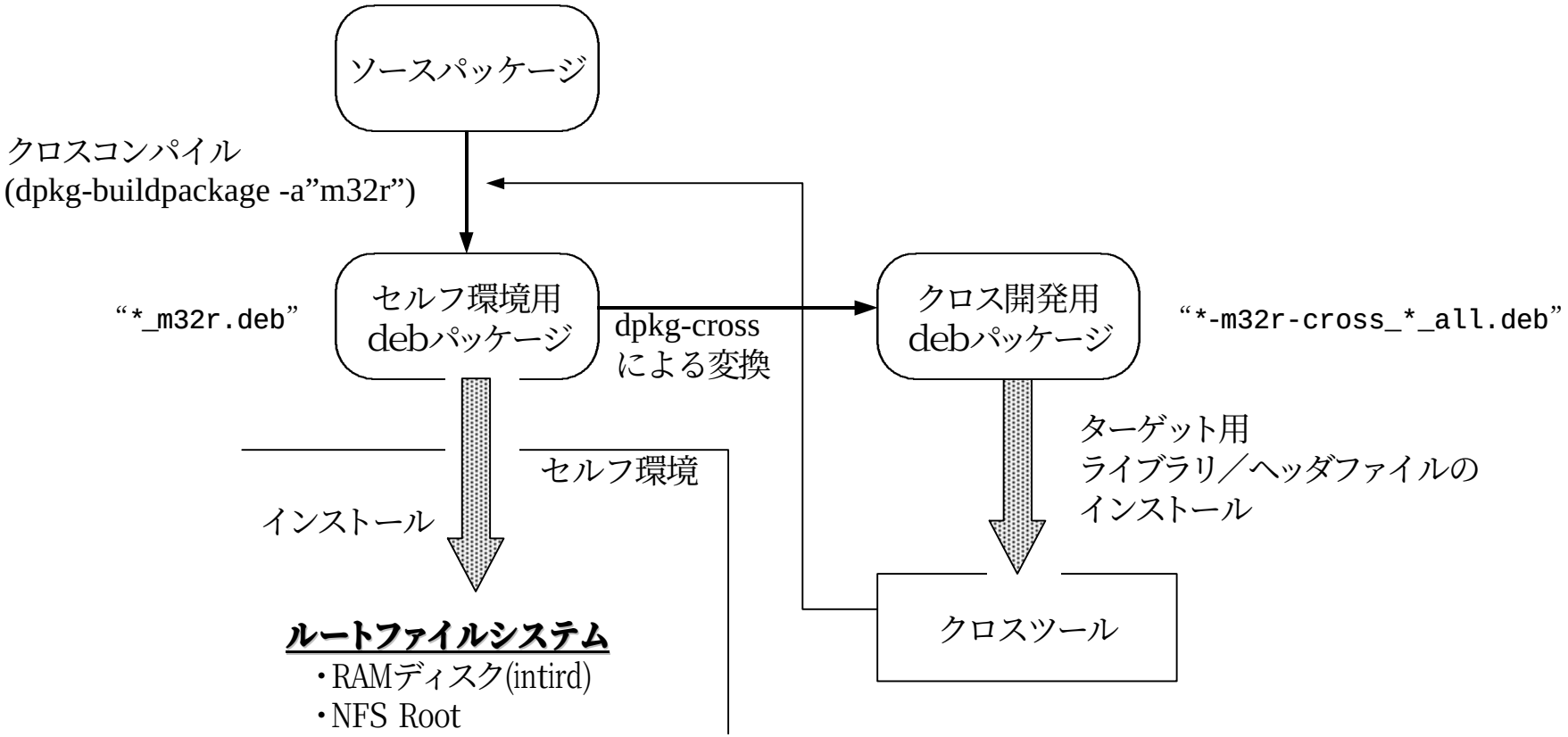
Linux/M32R開発環境



ソフトウェアパッケージのクロス開発

- ベースとなるディストリビューションとして
  Debian GNU/Linux を採用
  - 高度なパッケージ管理 (→開発時にも有効)
  - クロス開発のサポートがある
    - dpkg-buildpackage –a m32r –t m32r-linux
    - dpkg-cross
  - M32R用debパッケージ: bash, libc6, perl, etc. … 約100個
- クロス開発における問題点
  - ネイティブ環境とヘッダファイル／ライブラリのパスが
    異なるため、configure/makeがうまくいかない
    ⇒ Perl, Xサーバ／クライアントなど
  - ターゲット用ヘッダファイル、共有ライブラリの管理

クロス開発用パッケージの作成



今後の展開

- Linux/M32Rプラットフォーム
  - 性能評価とチューニング、安定化
  - 開発・機能拡張を継続
    - ミドルウェア、アプリケーション・プログラム開発環境の整備
    - カーネル・アップグレード(v2.5系カーネル)
      - MP性能向上、O(1)スケジューラ、プリエンプティブ・カーネル
  - プロセッサコア開発へのフィードバック
  - GNU/Linux開発環境・プラットフォームの公開

- 「M32Rソフトマクロ」提供プログラム
  - 研究・教育目的でのM32Rソフトマクロ(M32102製品版)の無償公開
    http://www.vdec.u-tokyo.ac.jp/CHIP/M32R/M32R_annai.html
    ※VDEC：大規模集積システム設計教育センター

まとめ

- Linux/M32R：M32R用GNU/Linux環境
  - M32Rアーキテクチャに Linuxを移植
  - M32Rソフトマクロ・コアをマッピングしたFPGA上で Linux(UP版・MP版) が動作
- ソフトマクロとFPGAの組み合わせ
  - ハードウェアとソフトウェアの協調設計・並行開発
- オープンソースの活用
  - 組み込みシステム開発に大きなインパクト

    ⇒　今後、組み込みOSとしてもLinuxが広く普及