# 特定用途向けプロセッサ/DSP設計開発ツール ASIP Designerのご紹介

## RISC-V Days Tokyo 2022 Autumn

日本シノプシス合同会社
スタッフアプリケーションエンジニア 伴野 充

2022年11月17日

# アジェンダ

- シノプシスプロセッサポートフォリオ
- ASIP Designerのご紹介
- RISC-Vプロセッサの開発
- ツール連携機能のご紹介
- まとめ

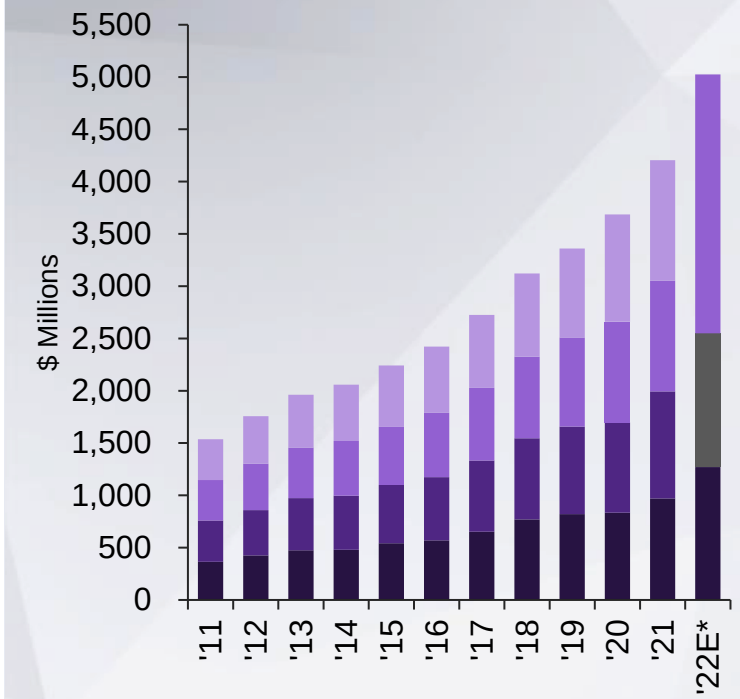
# Synopsys Today:シリコンからソフトウェアへ

**FY21 売上:**
**~$4.2B**

**従業員数:**
**16,660+**

**取得特許:**
**~3,430**

**オフィス拠点:**
**124**





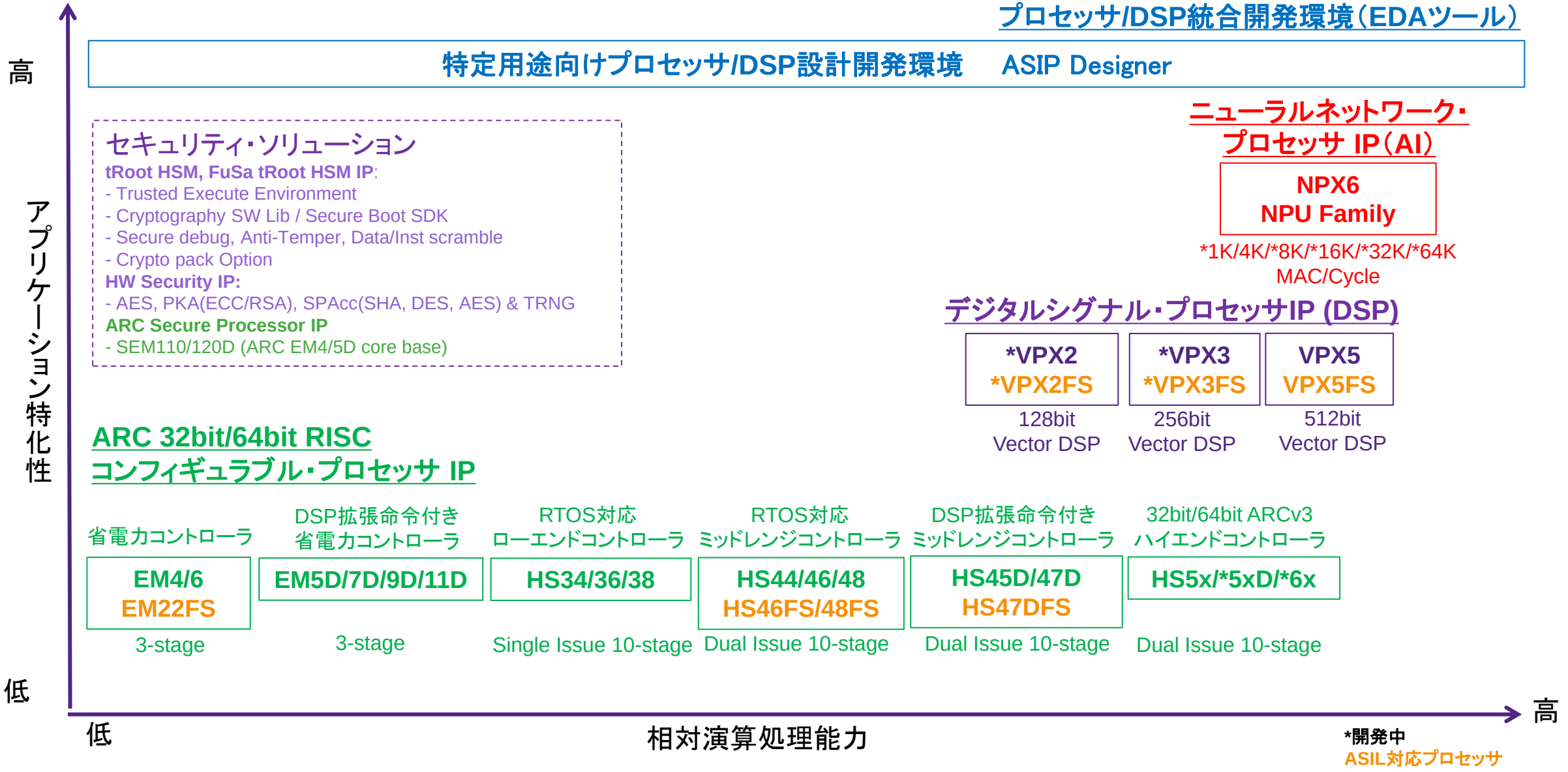
**Market / Region**

業界ナンバーワンの（EDA)電子設計自動化ツールとサービス

幅広いIPポートフォリオと＃1のインターフェースIP、アナログ、組込みメモリ、物理IPおよびプロセッサ

ガードナーのアプリケーションセキュリティ・テスト向け“Magic Quadrant” のリーダー

# シノプシス・プロセッサ製品一覧



# ASIP Designerのご紹介

# ドメインに特化したプロセッサの時代

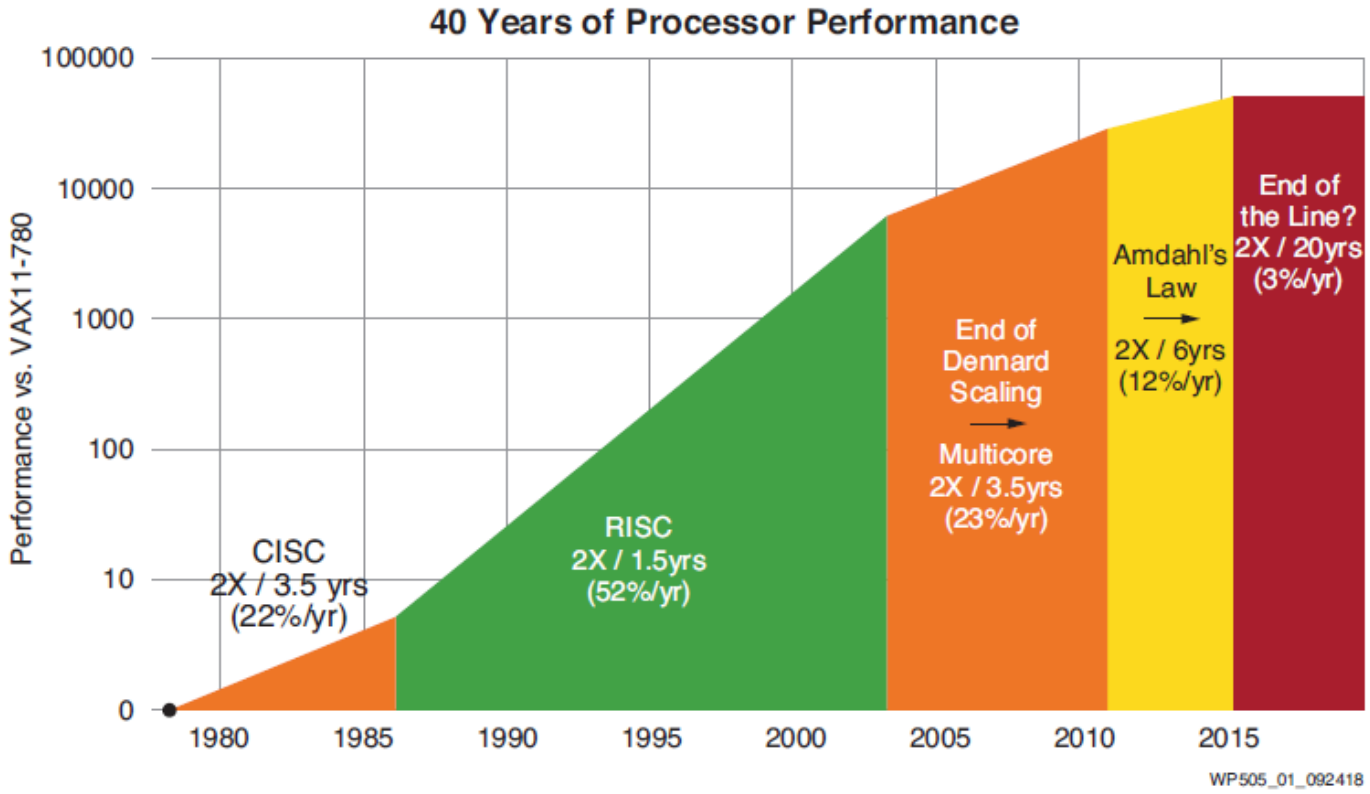



*Figure 1:* **Processor Performance vs. Time**

Source: Xilinx Whitepaper 505 – Versal ACAP
(reference to J. Hennessy, D. Patterson, *Computer Architecture: A Quantitative Approach* (6th Edition, 2019)

- 1995 – 2003:

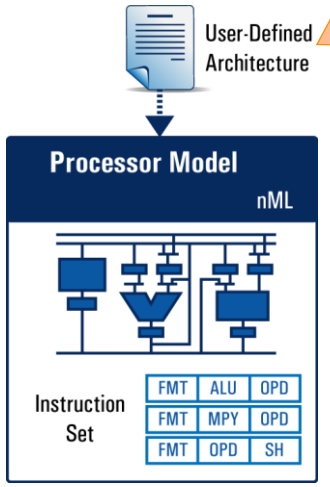
  プロセス技術による性能（より高い周波数だが同じ電力消費）およびメモリアクセスの最適化（Dennard Scaling）
- 2003: Multicore

  タスクの並列化制限に到達（Amdahlの法則）
- 2015: Domain-Specific Processors

  ヘテロジニアス・マルチコア設計への移行

# ASIP Designer

- アーキテクチャ構造記述言語(nML)により、
  - **アプリケーションに特化した命令セットを持つ専用プロセッサ/ DSP を自動生成する設計環境**




**nML記述**

```
action {
  stage E1:
    vecd =
     vsub(vecr=V[r],vecs=V[t])@vec;
    V[t] = vect = vabs(vecd) @vabs;
  }
  syntax : "vadiff v"t ",v"r ",v"t;
  image  : t::r;
```

**ASIP設計開発ツール**
**“ASIP Designer”**

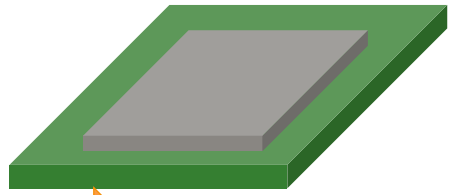
**命令セット**
**シミュレータ(ISS)**

**SystemC ベース**
**System Simulator**

**C/C++ コンパイラ**
**アセンブラ/リンカー**

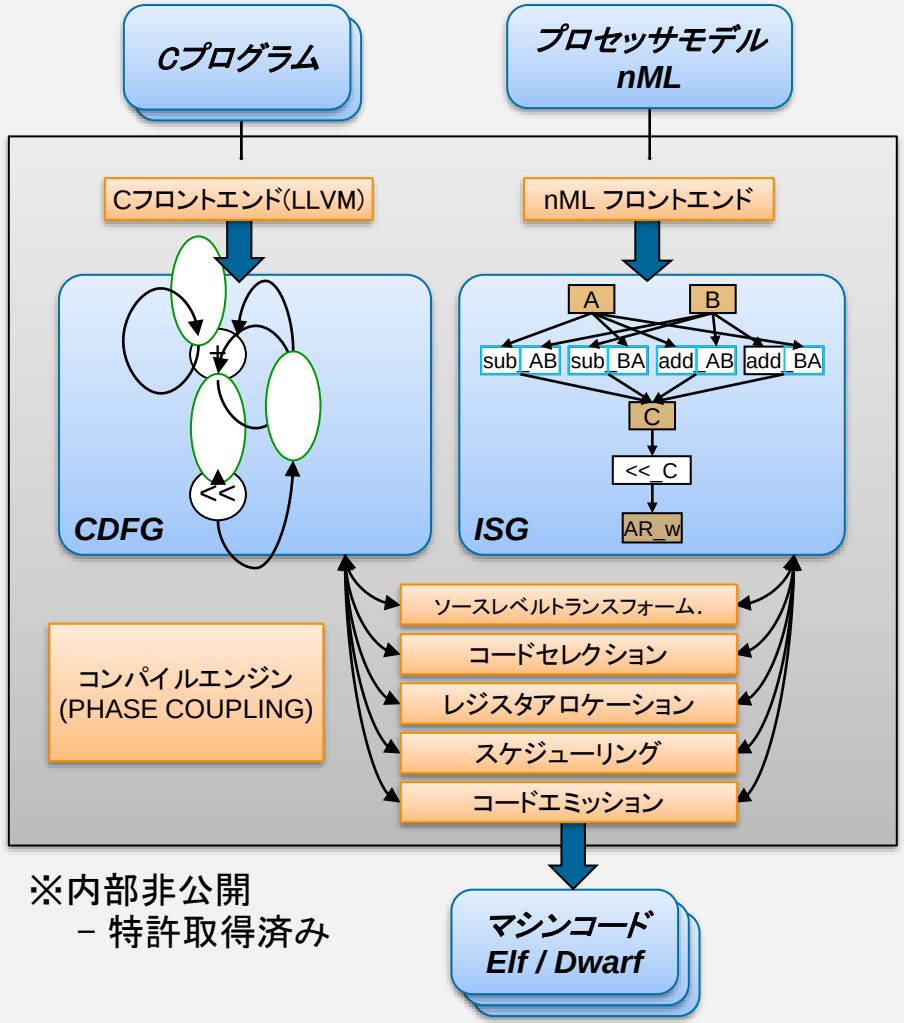
**デバッガ/プロファイラ**

**Verilog / VHDL**
**RTL コード**



**ASIC / FPGA**

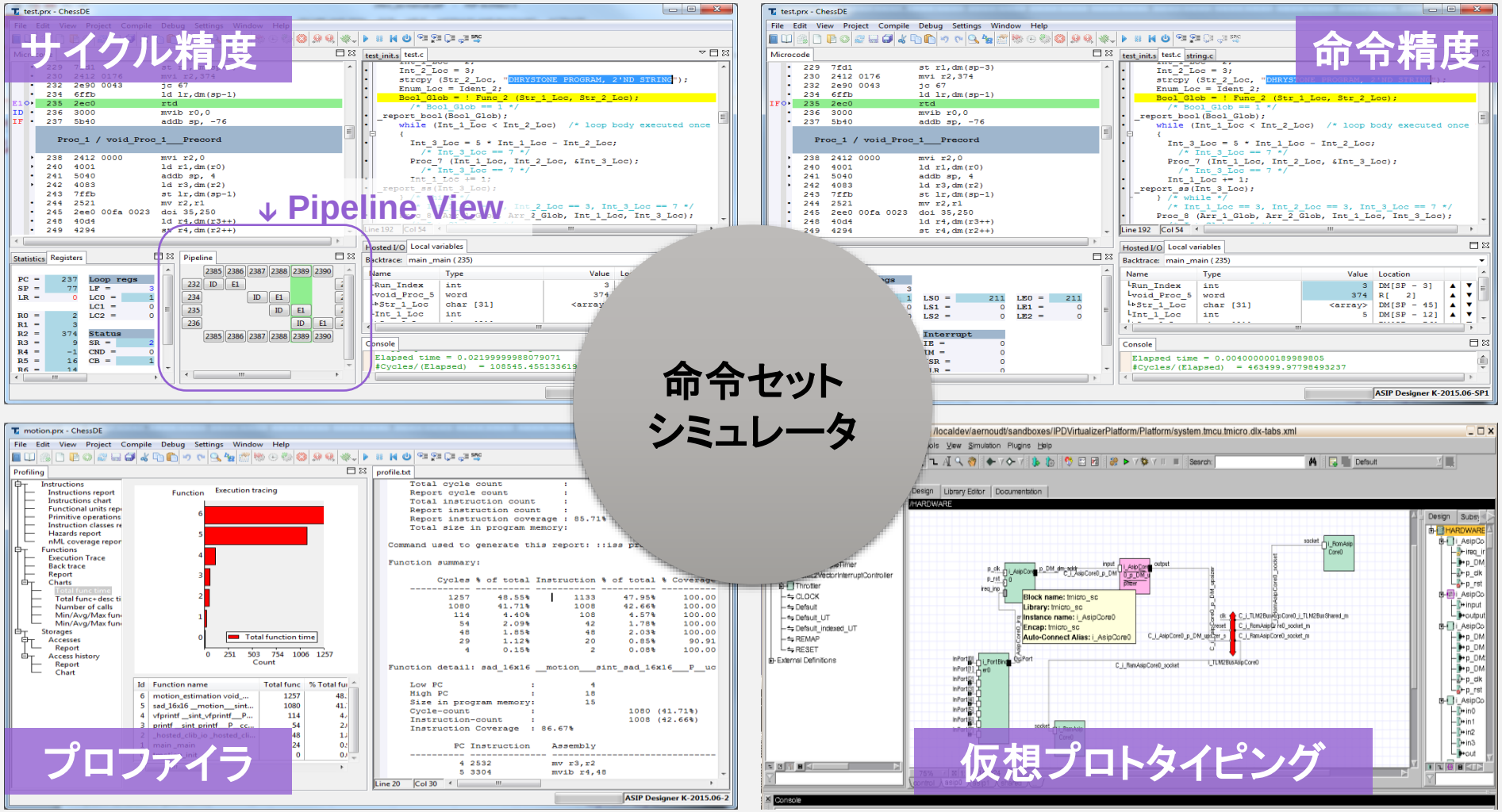
# Graph-Based C コンパイラ “Chess”



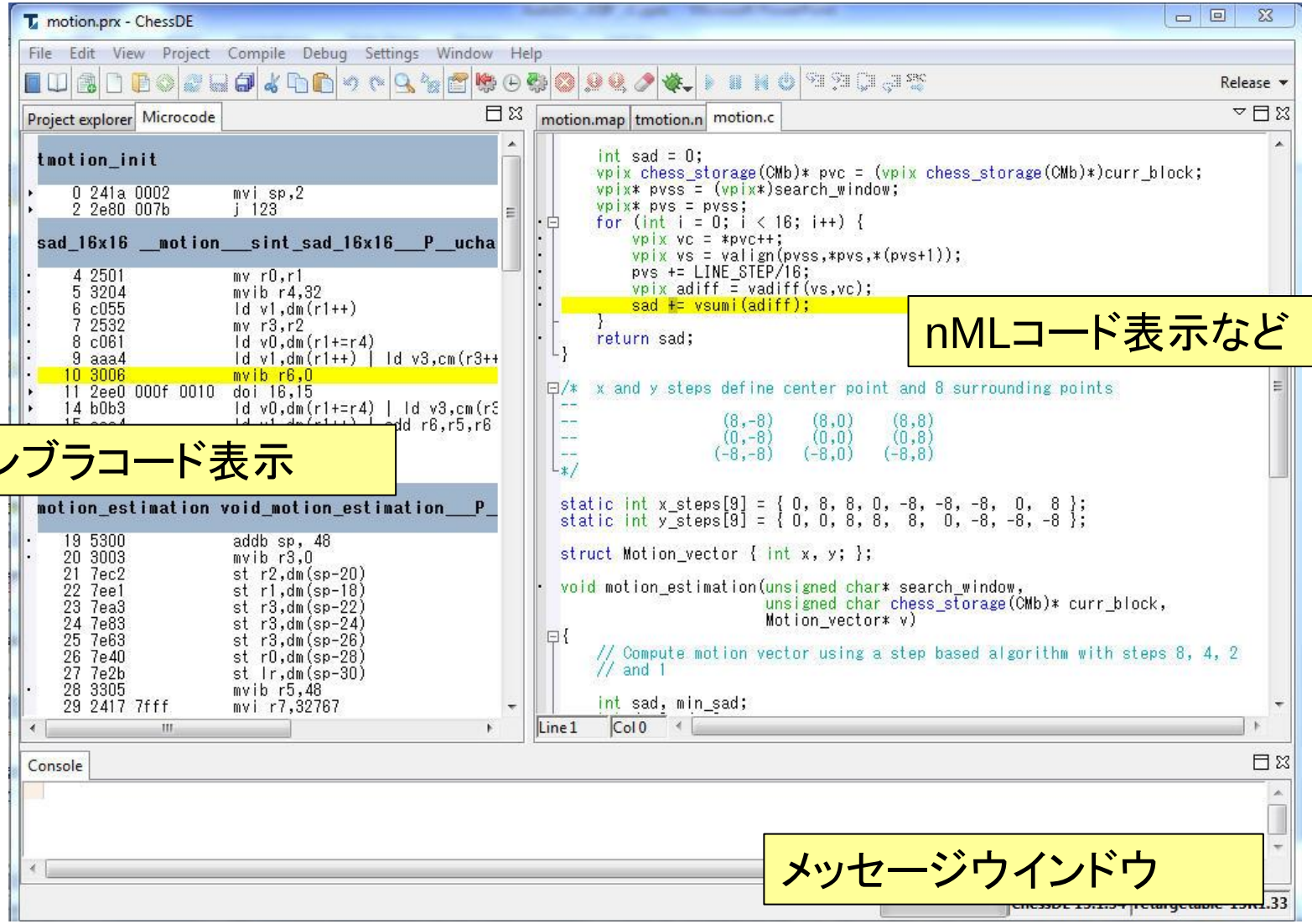

※内部非公開
– 特許取得済み

- フロントエンド
  - C → コントロールデータフローグラフ
  - nML → インストラクションセットグラフ
- コンパイルフェーズ
  - CDFGから ISGへマッピング
  - 独自“Graph”アルゴリズム
- ISGに含まれる構成情報
  - ハードウェアリソース
  - データタイプ
  - 接続情報
  - 命令エンコード情報
  - 命令レベルの並列化情報
  - 命令パイプライン情報

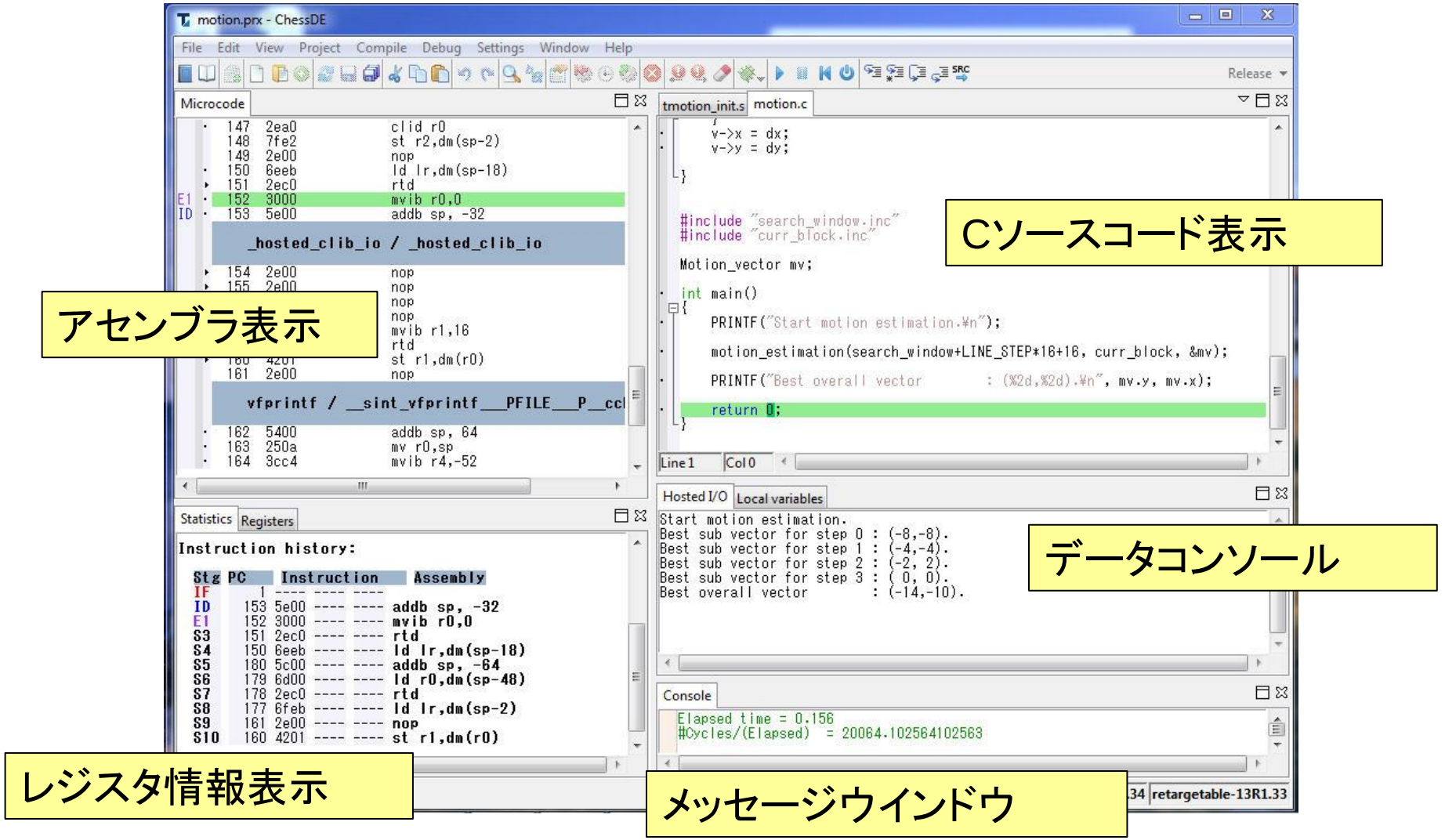
# ASIP Designer - 命令セットシミュレータ



# ASIP Designer生成 ー ASIPデバッガ



# ASIP Designer生成 ー ソフトウェアデバッガ



# ソフトウェア・プロファイリング機能

## 命令プロファイリング

```
Function summary:

      Cycles % of total Instruction % of total % Coverage
------------ ---------- ----------- ---------- ----------
        1620     51.01%        1548     52.60%     86.67%
        1258     39.61%        1133     38.50%     90.38%
         131      4.12%         124      4.21%     94.74%
          63      1.98%          49      1.66%     77.78%
          56      1.76%          56      1.90%    100.00%
          38      1.20%          27      0.92%     70.97%
           7      0.22%           4      0.14%     75.00%

Function detail: sad_16x16__motion__sint_sad_16x16__P_u

    Low PC                :          4
    High PC               :         18
    Size in program memory:         15
    Cycle-count           :                    1620 (51.01%)
    Instruction-count     :                    1548 (52.60%)
    Instruction Coverage  : 86.67%

           PC Instruction    Assembly
---------- ---------- ----------------------------------------
            4 2501           mv r0,r1
            5 3204           mvib r4,32
            6 c055           ld v1,dm(r1++)
            7 2532           mv r3,r2
            8 c061           ld v0,dm(r1+=r4)
            9 aaa4           ld v1,dm(r1++) | ld v3,cm(r3++)
           10 3006           mvib r6,0
           11 2ee0 000f 0010 doi 16,15
           14 b0b3           ld v0,dm(r1+=r4) | ld v3,cm(r3
           15 aac4           ld v1,dm(r1++) | add r6,r5,r6
           16 2ec0           rtd
           17 2506           mv r0,r6
           18 2e00           nop

Function detail: motion_estimation void_motion_estimation___P_

    Low PC                :         19
    High PC               :        122
    Size in program memory:        104
    Cycle-count           :                    1258 (39.61%)
    Instruction-count     :                    1133 (38.50%)
    Instruction Coverage  : 90.38%

                                                  Relative cycle use in simulation
------------------------------------------------ ------------------------------
_sint_sad_16x16___P__uchar___P **************************
nff___PFILE___P__cchar___Pvoi **
_P__cchar
ed_clib_io

Exe-count      Cycles Wait states Relative cycle use within func
--------- ---------- ----------- ------------------------------
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***
       36        108           0 *********
      576        576           0 ********************************
      576        576           0 ********************************
       36         36           0 ***
       36         36           0 ***
       36         36           0 ***

_PMotion_vector
```

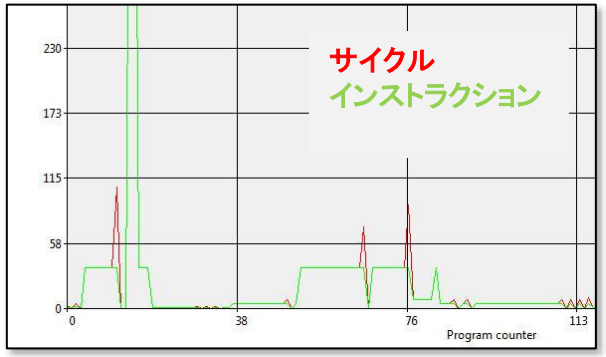
## 命令＆サイクル・カウント

```
ISS command      = <PROCDIR>/../iss/tmotion_ca
ISS mode         = Cycle accurate

Cycle count      = 3176
Instruction count = 2943

PC =   130

SP =    98
Stack area: DMb[    2 ..  2049], growing up
Maximum stack pointer value =   146

Instruction history:

 Stg  PC   Instruction   Assembly
 IF   130  ---- ----  ----
 ID   180  5c00 ----  ---- addb sp, -64
 E1   179  6d00 ----  ---- ld r0,dm(sp-48)
 S3   178  2ec0 ----  ---- rtd
 S4   177  6feb ----  ---- ld lr,dm(sp-2)
 S5   161  2e00 ----  ---- nop
 S6   160  4201 ----  ---- st r1,dm(r0)
 S7   159  2ec0 ----  ---- rtd
 S8   158  3101 ----  ---- mvib r1,16
 S9   157  2e00 ----  ---- nop
 S10  156  2e00 ----  ---- nop
```

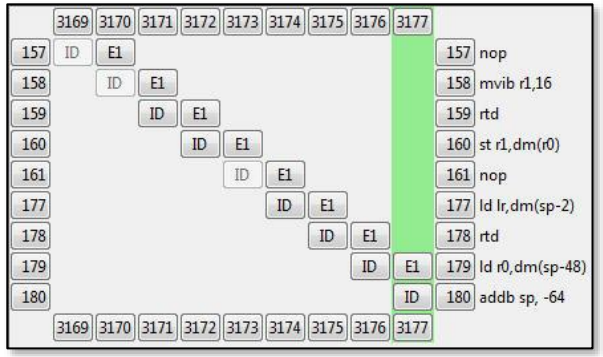
## nMLカバレッジ

```
tmotion  (0.266224)
 .alu_instr  (0.0654762)
  .alu_rrr  (0.142857)
   +alu_op  (0.142857)
     -add  (1)
     -addc  (0)
     -sub  (0)
     -subb  (0)
     -and  (0)
     -or  (0)
     -xor  (0)
  .compare_rr  (0.25)
   +compare_op  (0.25)
     -lt  (1)
     -ltu  (0)
     -le  (0)
     -leu  (0)
     -gt  (0)
     -gtu  (0)
     -ge  (1)
     -geu  (0)
  .equal_rr  (0)
  .alu_rr  (0)
  .minmax_rrr  (0)
  .select_rrr  (0)
 .shift_instr  (0.333333)
  .shift_rrr  (0.333333)
```

## 命令＆サイクル・チャート



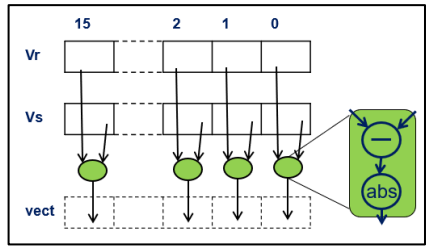

## パイプラインビュー



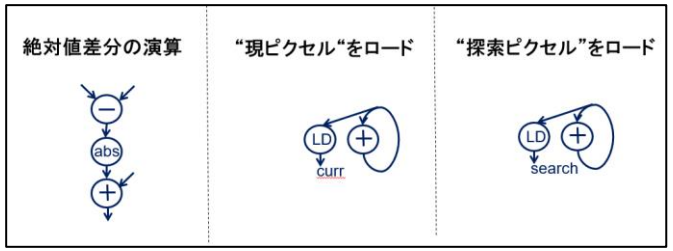
# アーキテクチャ最適化例(動き予測)

## Design1:　一般的なRISCアーキテクチャ

```
Profiling information for ::iss generated by Checkers Q-2020.03#7e5ed72dc8#200717 on 0
Program being simulated:

    D:/ASIP_Demo/tmotion/design1/motion/Release/motion

    Total cycle count           :                78025
    Report cycle count          :                78025
    Total instruction count     :                69803
    Report instruction count    :                69803
    Report instruction coverage : 98.80%
    Total size in program memory:                  193

Command used to generate this report: ::iss profile save D:/ASIP_Demo/tmotion/design1/motion/instruction_report.txt -type function_details -user_cycle_count Off -source

Function summary:

      Cycles % of total  Instruction % of total % Coverage  Function                                                                  Relative cycle use in simulation
---------- ----------- ------------ ----------- ---------- ------------------------------------------------------------------------ ------------------------------------------------
     76356      97.86%        68388      97.97%     100.00% sad_16x16 __sint_sad_16x16___P__uchar___P__uchar                          **************************************************
      1417       1.82%         1193       1.71%     100.00% motion_estimation void_motion_estimation___P__ucha
       114       0.15%          108       0.15%     100.00% vfprintf __sint_vfprintf___PFILE___P__cchar___Pvoi
        54       0.07%           42       0.06%     100.00% printf __sint_printf___P__cchar
        48       0.06%           48       0.07%     100.00% _hosted_clib_io _hosted_clib_io
        29       0.04%           20       0.03%      90.91% main _main
         4       0.01%            2       0.00%     100.00% tmotion_init tmotion_init

Function detail: sad_16x16 __sint_sad_16x16___P__uchar___P__uchar

    Low PC                :         4
    High PC               :        22
    Size in program memory:        18
    Cycle-count           :                76356 (97.86%)
    Instruction-count     :                68388 (97.97%)
    Instruction Coverage  : 100.00%

    PC         Instruction    Assembly                                                  Exe-count      Cycles Wait states Relative cycle use within function
------------ ------------ ------------------------------------------------------------ ---------- ---------- ----------- ------------------------------------------------
           4 3104         mvib r4,16                                                            36         36           0
           5 3205         mvib r5,32                                                            36         36           0
           6 3000         mvib r0,0                                                             36         36           0
           7 2edc 0015    do r4,21                                                              36         72           0
           9 3003         mvib r3,0                                                             36         36           0
          10 2edc 0014    do r4,20                                                             576       1152           0 ***
          12 2e00         nop                                                                  576        576           0 *
          13 4a56         lbu r6,dm(r1++)                                                     9216       9216           0 ****************************
          14 4a97         lbu r7,dm(r2++)                                                     9216       9216           0 ****************************
          15 05b7         sub r6,r6,r7                                                        9216       9216           0 ****************************
          16 0fb3         ge r6,r3                                                            9216       9216           0 ****************************
          17 2f01         jcr 1                                                               9216      16500           0 ************************************************
          18 059e         sub r6,r3,r6                                                        5574       5574           0 ****************
          19 2e00         nop                                                                 5574       5574           0 ****************
          20 0006         add r0,r0,r6                                                        9216       9216           0 ****************************
          21 004d         add r1,r1,r5                                                         576        576           0 *
          22 2eb8         rt                                                                    36        108           0
```



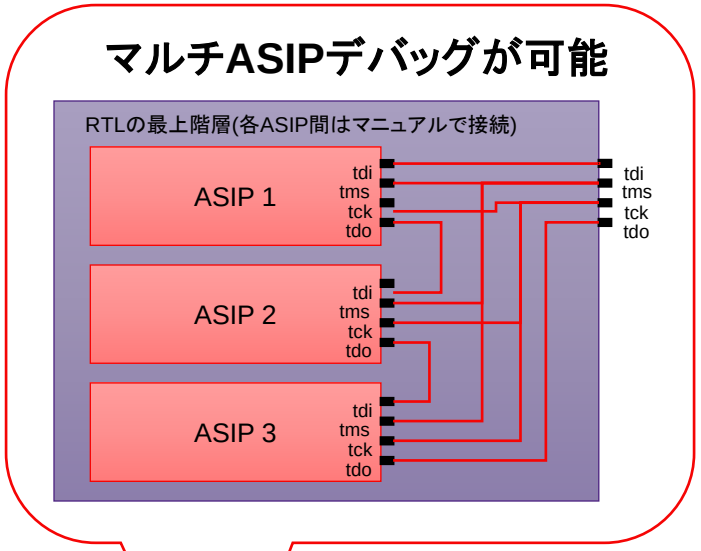

+



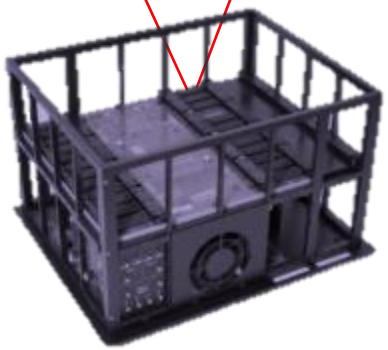

## Design5:　SIMD＋3スロットVLIW

```
Profiling information for ::iss generated by Checkers Q-2020.03#7e5ed72dc8#200717 on 0
Program being simulated:

    D:/ASIP_Demo/tmotion/design5/motion/Release/motion

    Total cycle count           :                 3237
    Report cycle count          :                 3237
    Total instruction count     :                 3011
    Report instruction count    :                 3011
    Report instruction coverage : 98.83%
    Total size in program memory:                  196

Command used to generate this report: ::iss profile save D:/ASIP_Demo/tmotion/design5/motion/instruction_report.txt -type function_details -user_cycle_count Off -source

Function summary:

      Cycles % of total  Instruction % of total % Coverage  Function                                                                  Relative cycle use in simulation
---------- ----------- ------------ ----------- ---------- ------------------------------------------------------------------------ ------------------------------------------------
      1728      53.38%         1656      55.00%     100.00% sad_16x16 __sint_sad_16x16___P__uchar___PCMb__ucha                        **************************
      1257      38.83%         1133      37.63%     100.00% motion_estimation void_motion_estimation___P__ucha ******************
       114       3.52%          108       3.59%     100.00% vfprintf __sint_vfprintf___PFILE___P__cchar___Pvoi *
        54       1.67%           42       1.39%     100.00% printf __sint_printf___P__cchar
        48       1.48%           48       1.59%     100.00% _hosted_clib_io _hosted_clib_io
        29       0.90%           20       0.66%      90.91% main _main
         4       0.12%            2       0.07%     100.00% tmotion_init tmotion_init

Function detail: sad_16x16 __sint_sad_16x16___P__uchar___PCMb__uchar

    Low PC                :         4
    High PC               :        25
    Size in program memory:        22
    Cycle-count           :                1728 (53.38%)
    Instruction-count     :                1656 (55.00%)
    Instruction Coverage  : 100.00%

    PC         Instruction    Assembly                                                  Exe-count      Cycles Wait states Relative cycle use within function
------------ ------------ ------------------------------------------------------------ ---------- ---------- ----------- ------------------------------------------------
           4 2501         mv r0,r1                                                              36         36           0 ***
           5 3204         mvib r4,32                                                            36         36           0 ***
           6 c055         ld v1,dm(r1++)                                                        36         36           0 ***
           7 2532         mv r3,r2                                                              36         36           0 ***
           8 c061         ld v0,dm(r1+=r4)                                                      36         36           0 ***
           9 aaa4         ld v1,dm(r1++) | ld v3,cm(r3++) | valign v0,r0,v0,                   36         36           0 ***
          10 c083         vsad r5,v0,v3                                                         36         36           0 ***
          11 3006         mvib r6,0                                                             36         36           0 ***
          12 c061         ld v0,dm(r1+=r4)                                                      36         36           0 ***
          13 2ee0 0011 000e doi 14,17                                                           36        108           0 **********
          16 aaa4         ld v1,dm(r1++) | ld v3,cm(r3++) | valign v0,r0,v0,                  504        504           0 ************************************************************
          17 b0d3         ld v0,dm(r1+=r4) | add r6,r5,r6 | vsad r5,v0,v3                     504        504           0 ************************************************************
          18 c181         ld v3,cm(r3++)                                                        36         36           0 ***
          19 c104         valign v0,r0,v0,v1                                                    36         36           0 ***
          20 01ae         add r6,r5,r6                                                          36         36           0 ***
          21 c083         vsad r5,v0,v3                                                         36         36           0 ***
          22 01ae         add r6,r5,r6                                                          36         36           0 ***
          23 2ec0         rtd                                                                   36         36           0 ***
          24 2506         mv r0,r6                                                              36         36           0 ***
          25 2e00         nop                                                                   36         36           0 ***
```

# RTLコード生成と実機デバッグ環境の構築




- nML(PDG)記述からRTLコードを生成
  - Verilog/VHDL RTLコード
  - JTAG Key/JTAG key2をサポート
  - Design Compiler/ Simplify用スクリプト
  - **生成プロセッサ（RTL）のロイヤリティ不要**



HAPS-80 FPGAボード
シリーズ

Xilinx USB2

Digilent JTAG HS2

Macraigor

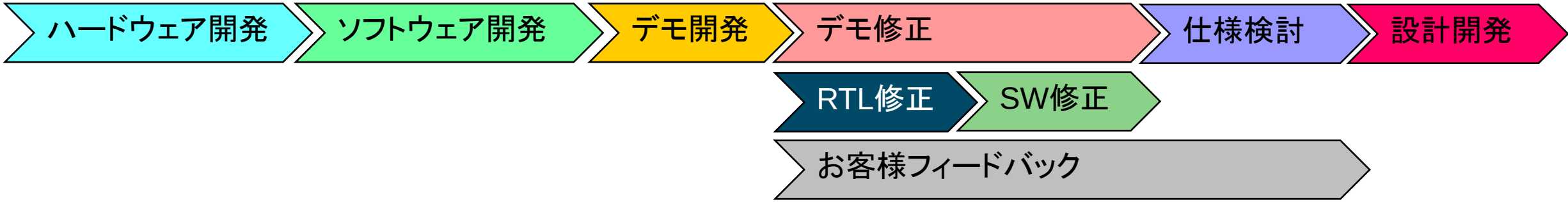
Lauterbach

FTDI USB Hi-Speed to MPSSE cable.
Segger J-Link Ultra cable.

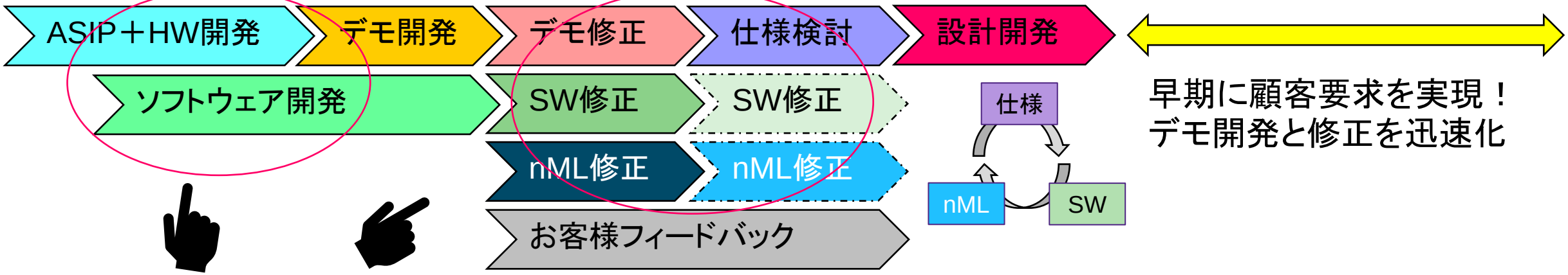
ソフトウェア
デバッガ

# ASIP Designer – HW/SW協調設計

一般的な設計開発手法




ASIPによる設計開発手法



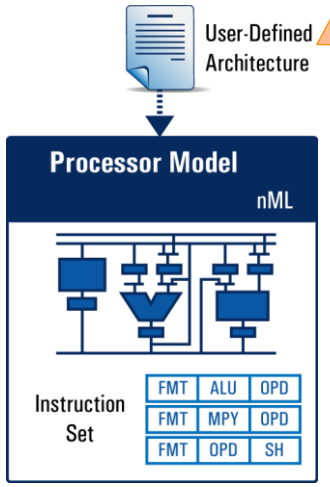

早期に顧客要求を実現！
デモ開発と修正を迅速化

コンパイラ・イン・ザ・ループ！

# RISC-Vプロセッサの開発

# ASIP Designer

- アーキテクチャ構造記述言語(nML)により、
  - **アプリケーションに特化した命令セットを持つ専用プロセッサ/ DSP を自動生成する設計環境**




**nML記述**

```
action {
  stage E1:
    vecd =
     vsub(vecr=V[r],vecs=V[t])@vec;
    V[t] = vect = vabs(vecd) @vabs;
  }
  syntax : "vadiff v"t ",v"r ",v"t;
  image  : t::r;
```

**ASIP設計開発ツール**
**“ASIP Designer”**

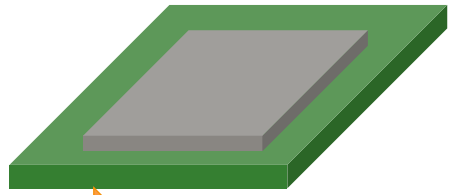
**命令セット**
**シミュレータ(ISS)**

**SystemC ベース**
**System Simulator**

**C/C++ コンパイラ**
**アセンブラ/リンカー**

**デバッガ/プロファイラ**

**Verilog / VHDL**
**RTL コード**



**ASIC / FPGA**

# ご提供可能なExampleプロセッサ

| 種類 | 説明 |
|---|---|
| マイクロコントローラ | |
| Tnano | 16-bit microcontroller, lightweight and configurable |
| Tmicro | 16-bit microcontroller, fully featured |
| DLX (family) | Variants of 32-bit microcontroller |
| Tmcu | 32-bit microcontroller |
| Trv32 (family) | 32-bit microcontroller featuring RISC-V ISA, 3 or 5 pipeline stages |
| Trv64 (family) | 64-bit microcontroller featuring RISC-V ISA, 3 or 5 pipeline stages |
| PD_Triop | 32-bit microcontroller with 64-bit address spaces |
| DSP及び並列演算エンジン | |
| Tdsp | 16/32-bit DSP |
| Tvec (family) | Variants of SIMD processor |
| Tvliw (family) | Variants of VLIW processor |
| 暗号処理エンジン | |
| Tsec | High throughput SHA/RSA/AES accelerator |
| Tcript | High throughput AES accelerator |
| その他 | |
| Tmotion | Accelerator of motion estimation kernel |
| Tcom8 | SIMD processor optimized for some communication kernels |
| FFTcore | Scalar implementation of complex FFT |
| Mxcore | Matrix processing ASIP for communication kernels |
| Primecore | SIMD implementation of prime-factor algorithm for FFT & DFT |
| Tgauss | Vectorization and memory management for image processing |
| Tvox | Accelerator for SLAM (simultaneous localization and mapping) |

Example使用に
一切の追加費用無し
“ツールライセンス費用のみ”

## 豊富なExample

- マイクロコントローラ
- DSPs
- SIMD、VLIW
- マルチスレッド
- セキュリティ（AES、SHA等）
- メモリI/F、AXIバス等

## Example使用のメリット

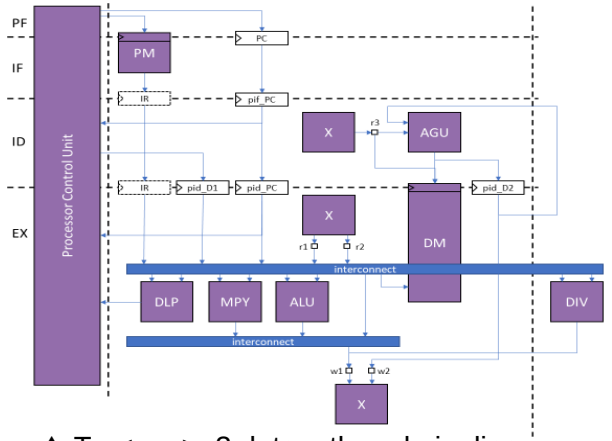
- 基本的な機能は既に設計開発済み
- 動作可能な例を参照可能
- モデリングの概念を効率的に習得
- nMLソースコードで提供されるため、ユーザーが自由に追加変更可能

# Trv(RISC-V ISA)モデル

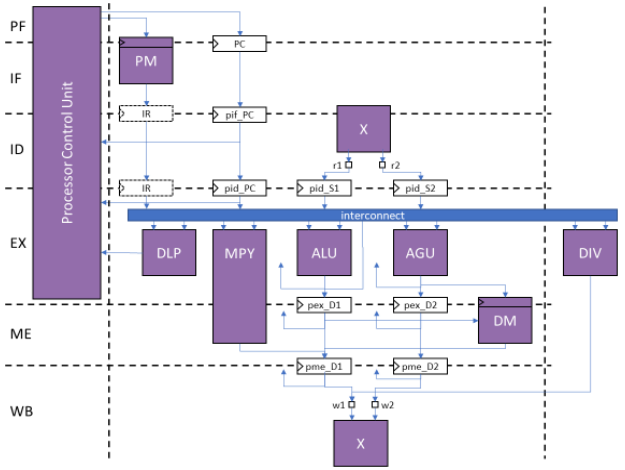
## 内容

- RISC-V ISAに最適化されたモデル
- ISA拡張のスターティングポイントとしての活用 (ASIP設計)

| | 32-bit データパス | 64-bit データパス |
|---|---|---|
| 3ステージ パイプライン | Trv32p3<br>Trv32p3x | Trv64p3<br>Trv64p3x |
| 5ステージ パイプライン | Trv32p5<br>Trv32p5x | Trv64p5<br>Trv64p5x |




▲ Trx<mm>p3 datapath and pipeline




▲ Trx<mm>p5 datapath and pipeline

## 機能 (2020.03)

- 対応ISA: RV64IM, RV32IM
  - 整数命令（Integer）
  - 乗算命令（Multiply）
- オプション拡張: Trv<mm>p<n>x
  - (標準) 圧縮命令(Compressed：IMC)
  - (カスタム) ゼロオーバヘッドループ
  - (カスタム) アドレス更新後のロード/ストア
- マイクロアーキテクチャの特長
  - 5段ステージパイプライン: IF, ID, EX, ME, WB
  - 3段ステージパイプライン: IF, ID, EX
  - 保護されたパイプライン（Protected pipeline）
    - 可能な場合はレジスタをバイパス
    - バイパスが出来ない時はStall (bubble)
  - ハードウェア乗算器
    - 64x64→128→select 64bit, 32x32 → 64 → select 32bit
  - 繰り返し処理の除算器

# RISC-V nML モデリングマニュアル

**ASIP Designer**

**Trv (RISC-V ISA) Models**

**Processor Manual**

S-2021.12

SYNOPSYS®

# 1

## Introduction

The TRV models are a family of ASIP DESIGNER example pr
Eight models are part of the example cores collection:

| Model | ISA | Description |
|---|---|---|
| TRV32P3 | RV32IM | 3-stage pipeline |
| TRV32P5 | RV32IM | 5-stage pipeline |
| TRV64P3 | RV64IM | 3-stage pipeline |
| TRV64P5 | RV64IM | 5-stage pipeline |
| TRV32P3X | RV32IMC | 3-stage pipeline, hardware loops, post-modify addressing |
| TRV32P5X | RV32IMC | 5-stage pipeline, hardware loops, post-modify addressing |
| TRV64P3X | RV64IMC | 3-stage pipeline, hardware loops, post-modify addressing |
| TRV64P5X | RV64IMC | 5-stage pipeline, hardware loops, post-modify addressing |

RV32 is the 32-bit RISC-V ISA. The TRV32 models have a 32-bit wide data path.

RV64 is the 64-bit RISC-V ISA. The TRV64 models have a 64-bit wide data path.

RVXXI is the basic instruction set. M adds multiplication, division and remainder instructions. C is the ISA extension for compressed instructions.

There are base and extended models. The extended models, ending with X, feature non-standard extensions. These have hardware support for two nested levels of zero-overhead loops. Their address generation unit supports post-modify addressing.

The P3 models have a 3-stage pipeline. The pipeline stages are fetch (IF), decode (ID), and execute (EX). Results are committed in the EX stage. Instructions can be stalled in the IF and the ID stage.

The P5 models have a 5-stage pipeline. The pipeline stages are fetch (IF), decode (ID), execute (EX),

# 3

## Getting Started

### 3.1 Steps to Build the Processor Model

The following steps build the processor model and processor-specific software libraries.

You can use the top-level batch project files:

- To build the processor model, processor support libraries, C libraries from the light-weight runtime stack, the cycle-accurate ISS with debug options, the Verilog HDL model and the support library for native compilation. Compile configuration `Release` for CHESS front-end.

```
chessmk model.prx -m
```

- To build the processor model, processor support libraries, C and C++ libraries from the light-weight runtime stack, the cycle-accurate ISS with debug options, the Verilog HDL model and the support library for native compilation. Compile configuration `Release_LLVM` for LLVM front-end.

```
chessmk model_llvm.prx -m
```

- To build the processor model, processor support libraries, C and C++
runtime stack, the cycle-accurate ISS with debug options, the Veril
library for native compilation. Compile configuration `Release_LLV`

```
chessmk model_llvm_full.prx -m
```

For typical use cases, we recommend to use the CHESS front-end first, i.e., it

Use the LLVM front-end for applications with a lot of control flow, or if

# 8

## Instruction Set

### 8.1 TRV32P3 and TRV32P5

The two models TRV32P3 and TRV32P5 implement the RV32IM ISA specification [3]. The list below shows the assembly syntax of all instructions. Some instructions have multiple alternative syntax, which the RISC-V ISA specification calls pseudoinstructions (for example: `beqz X, imm` is `beq x0, X, imm`).

```
addi X, X, imm      jal X, imm          rem X, X, X
add X, X, X         j imm               ret
andi X, X, imm      jr X                sb X, imm(X)
and X, X, X         lbu X, imm(X)       seqz X, X
auipc X, imm        lb X, imm(X)        sgtz X, X
beq X, X, imm       lhu X, imm(X)       sh X, imm(X)
beqz X, imm         lh X, imm(X)        slli X, X, imm
bgeu X, X, imm      li X, imm           sll X, X, X
bge X, X, imm       lui X, imm          sltiu X, X, imm
bgez X, imm         lw X, imm(X)        slti X, X, imm
bgtz X, imm         mulhsu X, X, X      sltu X, X, X
blez X, imm         mulhu X, X, X       slt X, X, X
```

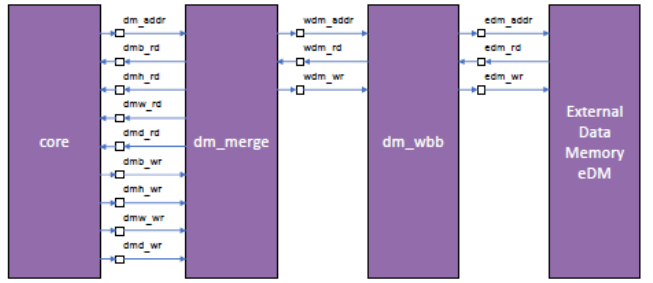
# 10

## IO Interfaces

The TRV models use three IO interfaces. Two are attached to the data memory interface. One is attached to the program memory interface.

### 10.1 Data Memory Interface

This diagram shows the data memory interface chain:

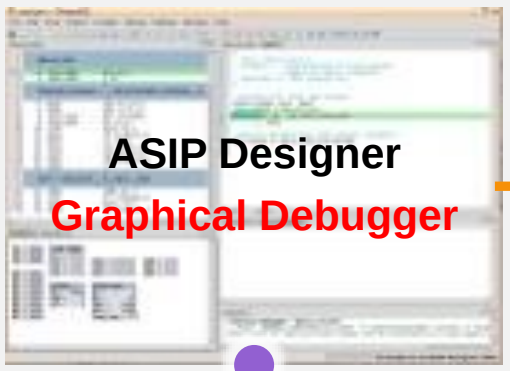


# ツール連携機能のご紹介

# ASIP Designerの統合化されたデバッグ環境

**ASIP Designer が生成する RTL**
- ASIPコア
- プロセッサデバッグコントローラ (PDC), JTAG
- APBバス, Zebu XTOR

デバッグPC




ISS
- Cycle-Accurate
- Instruction Accurate

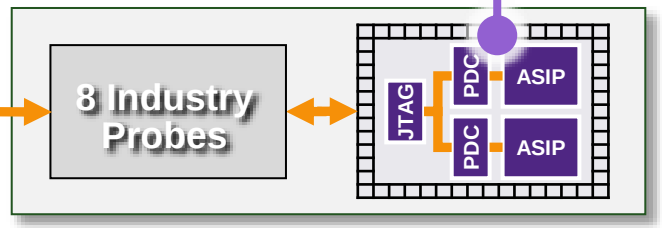
Debug Client SW (APIs)

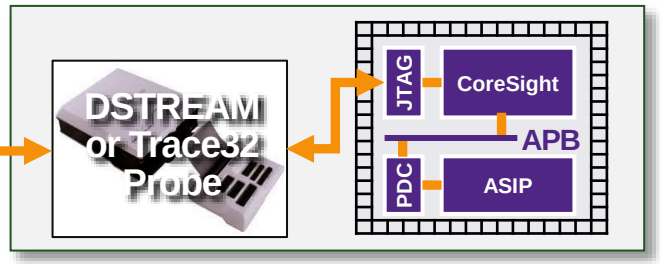
TCP/IP

ノート / デスクトップ PC

Jtalk Server

オンチップデバッグ (OCD) ハードウェア





TCP/IP


| Graphical Debugger | Debug Flow | No. Cores |
|---|---|---|
| ChessDE | ISS, OCD (single-core) | Single |
| ChessMP | ISS, OCD (multi-core) | Multi |
| Eclipse | ISS, OCD | Single |
| Virtualizer | Virtual prototyping (SystemC) | Multi |
| Verdi | RTL execution traces | Single |

**Jtalk Server**
- 10を超える主要なOCDプローブメーカで標準サポート
- OCDプローブのドライバにて動的に自動接続

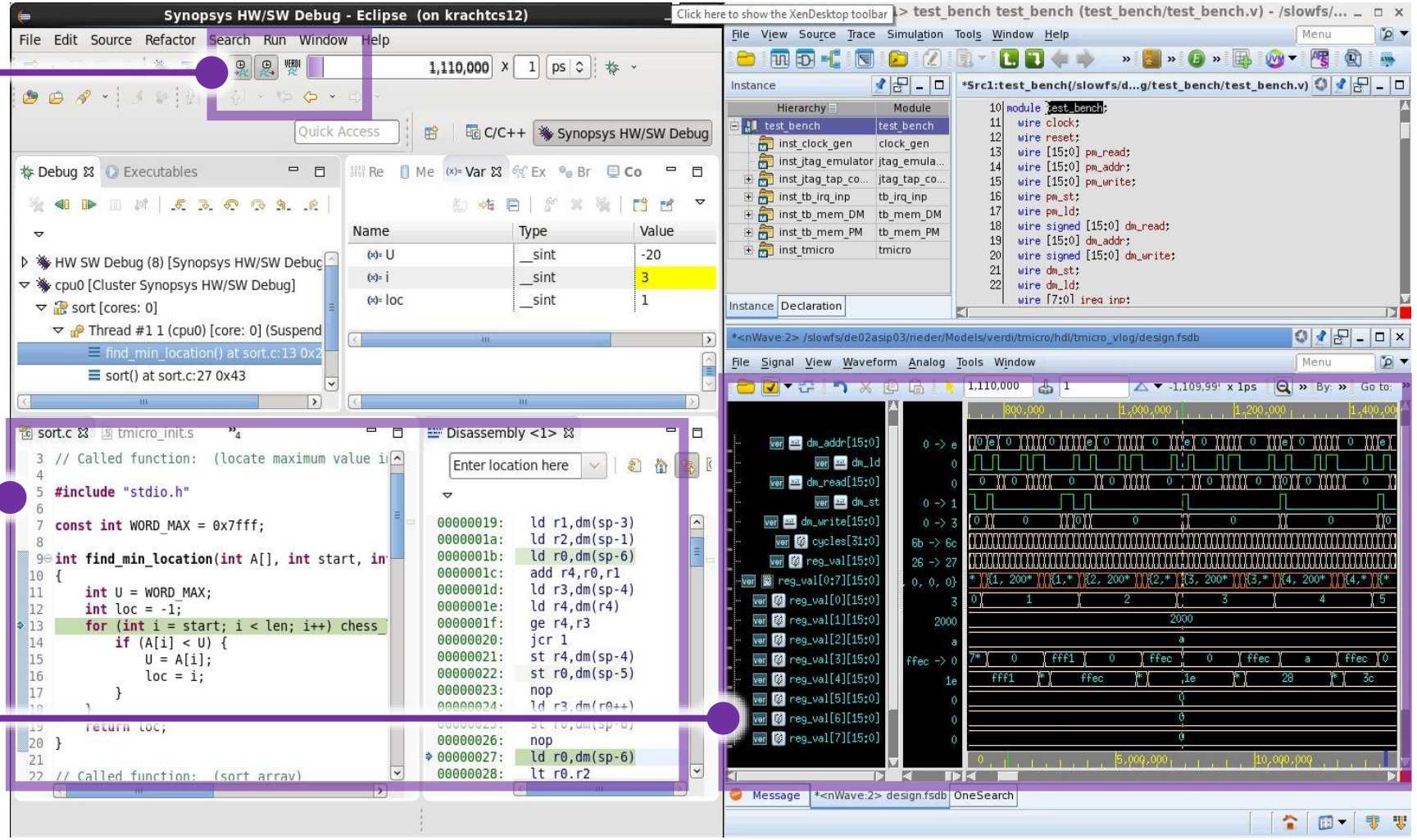
# Verdi HW/SW デバッガ と ASIP Designer

## Graphical Debugger Functionalities

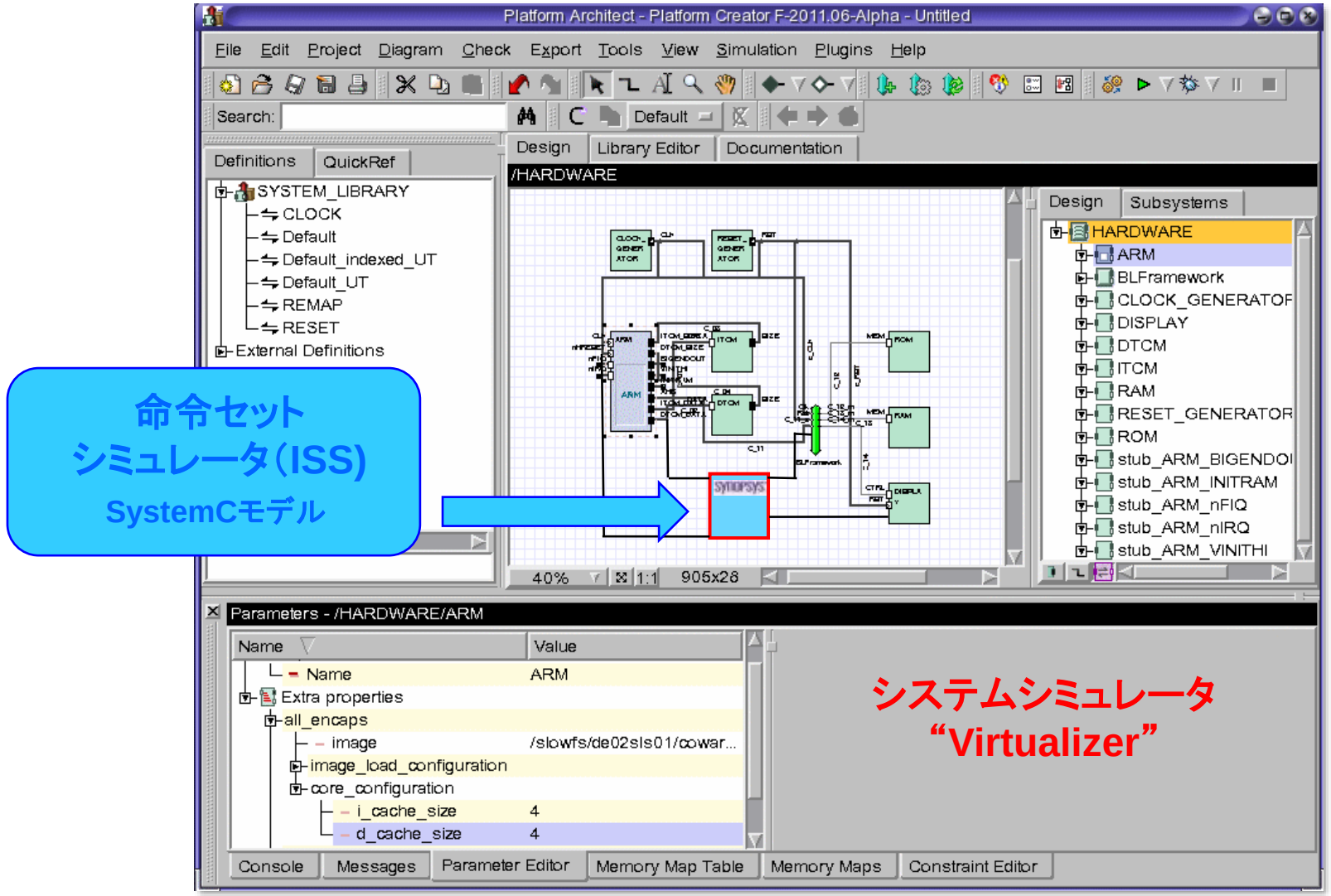
実行制御:
停止-ステップ実行と巻き戻し

ハードウェアビューと完全に同期されたCソースおよびアセンブリコードビュー

RTL波形ビュー:
時間の前後にスクロール



# システムシミュレータへプロセッサをエクスポート



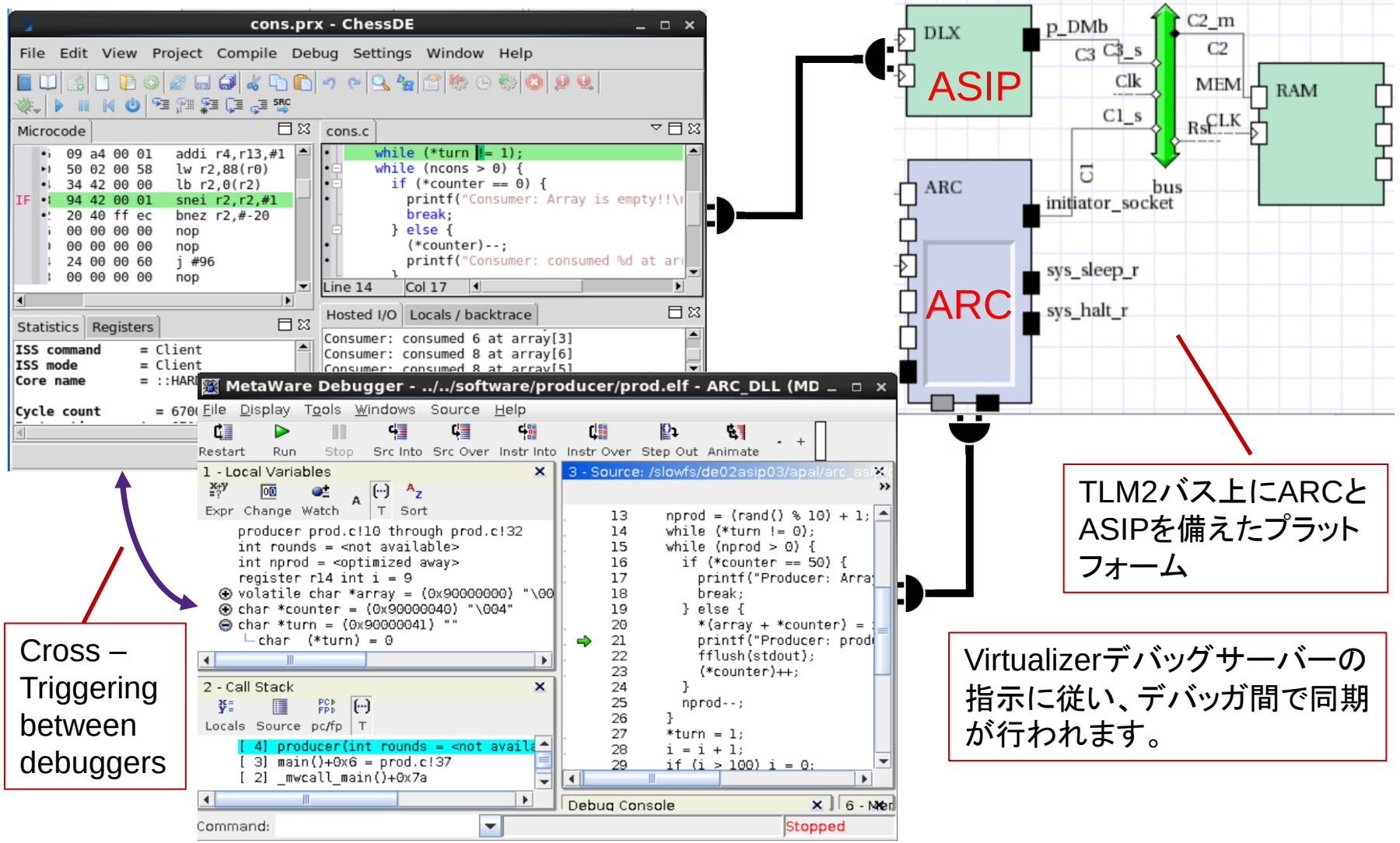
# システムレベル　シミュレーション & デバッグ

## ARC（ホスト）&ASIP（コプロ）マルチコアプラットフォームのシミュレーション＆デバッグ

- Virtualizer / Platform Architect環境におけるMetaWare（ARC)とChessDE（ASIP)SWデバッガー間の同時デバッグと同期（クロストリガー）の例

### シミュレーションモデル

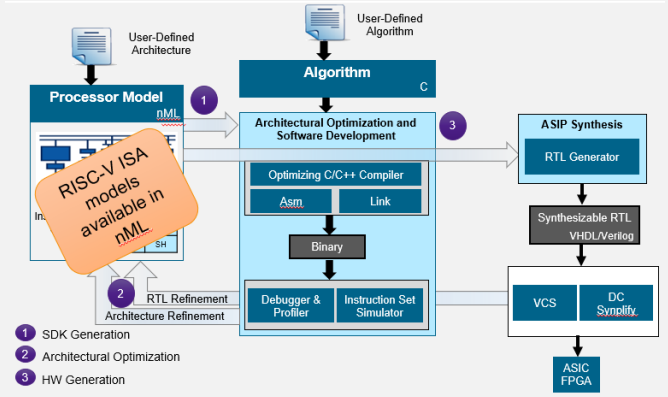
- 命令精度（SW開発用）とサイクル精度（HWアーキ検討またはドライバ開発用をASIP Designerで生成可能
- ARCはnSIM(命令精度)、nCAM(疑似サイクル精度)シミュレーションモデルをご用意
- その他のバス/メモリモデル等は別途弊社System Design Group（SDG）担当者へお問い合わせください

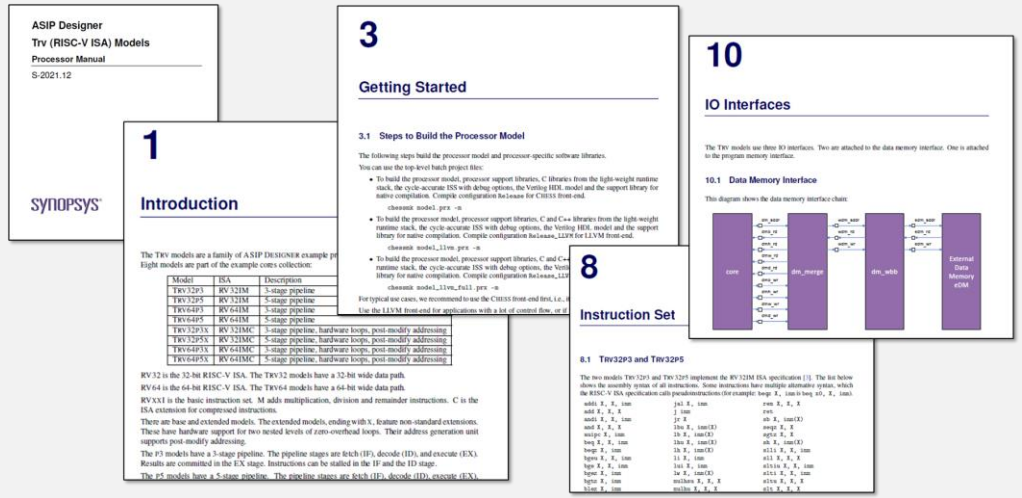


# まとめ

# ASIP Designer – ASIP デザインを容易に

- アプリケーションに特化したアーキテクチャによるPPAの大幅な改善
- RISC-V ISAベースのExampleモデルで即座にトライアルが可能
- 統合化されたデバッグ環境とシステムレベル設計
- **製品の差別化を実現する拡張可能かつロイヤリティフリーなRISC-Vプロセッサを開発！**
  - ご興味のある方はぜひご連絡ください
  - コンタクト先: jp-mkg_info@synopsys.com



SYNOPSYS®
Silicon to Software™

# Thank You