

DSM(ドメイン・スペシフィック・モデリング)ツール を利用したESLツールの開発

プロファウンド・デザイン・テクノロジー(株)

tsukamoto@profound-dt.co.jp

2012/7/6

会社紹介

- 社名 : プロファウンド・デザイン・テクノロジー株式会社
- 事業内容 : ソフトウェアおよびハードウェアの開発、設計、販売
およびコンサルティング
- 所在地 : 横浜市中区扇町1-1-25 キンガビル4F
- 代表 : 代表取締役 塚本泰隆
- 設立 : 2011年9月1日

内容

- SystemCの現状
- グラフィカル入カツール Breakfast
~バーチャル・プラットフォームから高位合成まで~
- DSMツールで楽々開発
- まとめ

SystemCの現状

- バーチャル・プラットフォームが役に立つことはわかってきた
 - ◆ ソフトウェアの先行開発など
- 高位合成が役に立つことはわかってきた
 - ◆ 設計期間の短縮
 - ◆ 高速なシミュレーションによる十分な検証

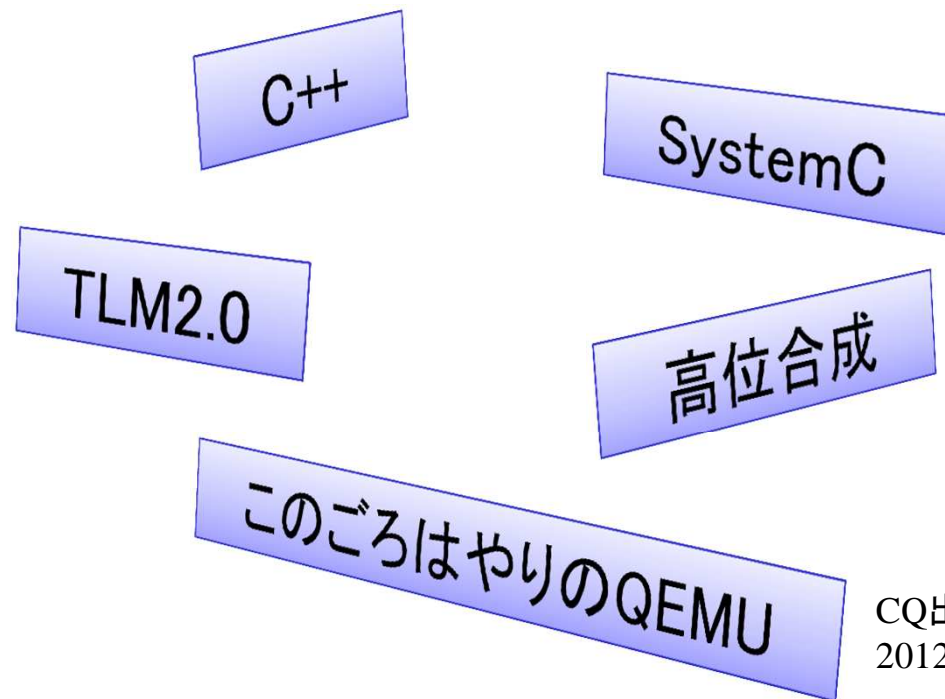


現在は次のステップへ

いかに多くの人に使ってもらおうか

大きな壁

- SystemCおよびESLに関しては、学習に時間がかかる



CQ出版社の雑誌「インターフェース」
2012年5月号でQEMUの特集あり

いったい、どれだけ勉強すればいいの？

大きな壁

例

SystemCの難しい箇所、実はお決まりの記述が多い

↓
エキスパートのソースコードをコピー&ペーストすればいい

↓
少し変えると……



↓
コンパイルエラーやセグメント例外が……



↓
他人のソースコードをコピー&ペーストしてきたから
原因がなかなか判明せず……

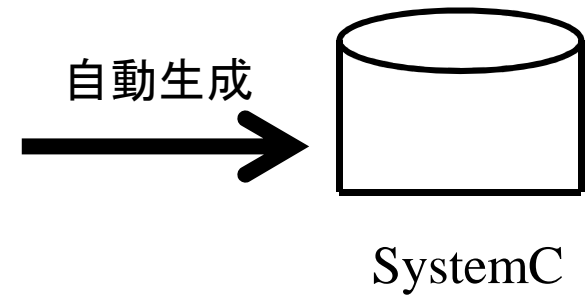
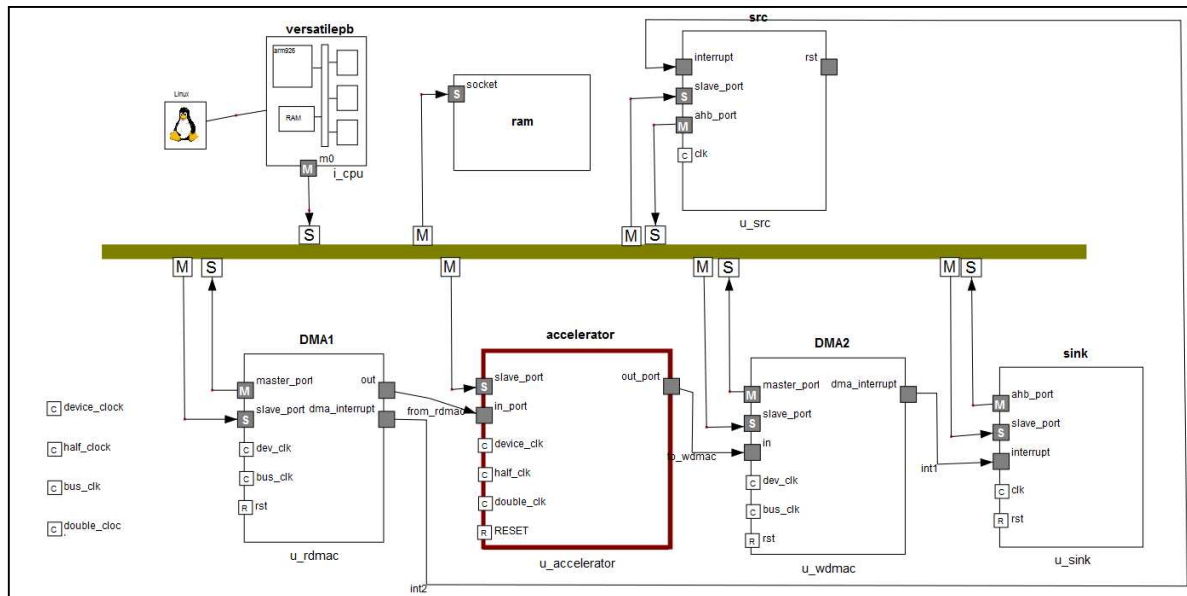
内容

- SystemCの現状
- **グラフィカル入カツール Breakfast**
~バーチャル・プラットフォームから高位合成まで~
- DSMツールで楽々開発
- まとめ

グラフィカル入カツールは必須

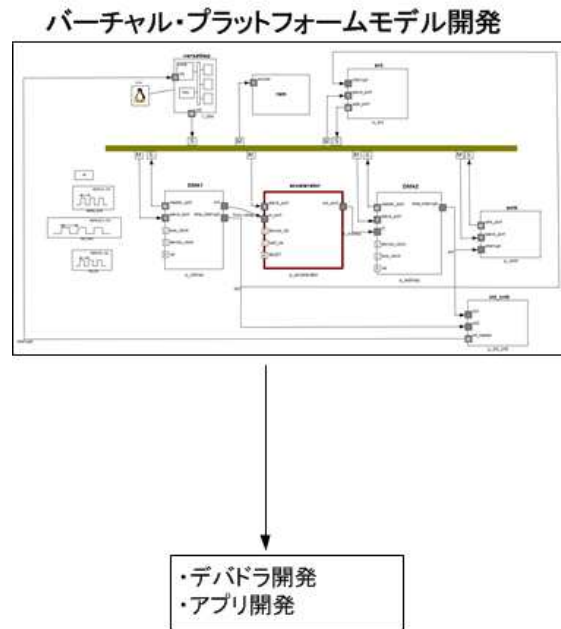
- お決まりの記述部分は絵で入力
- お決まりのコードは完全自動生成

無駄な学習時間やミスを撲滅

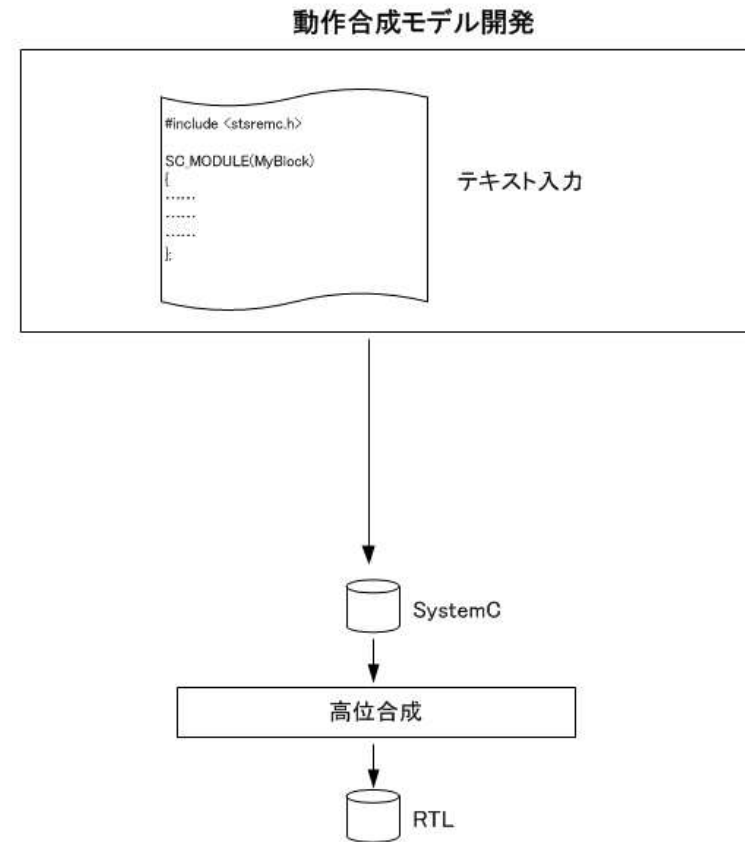


ここまではよくある話。今回は何が違うのか...

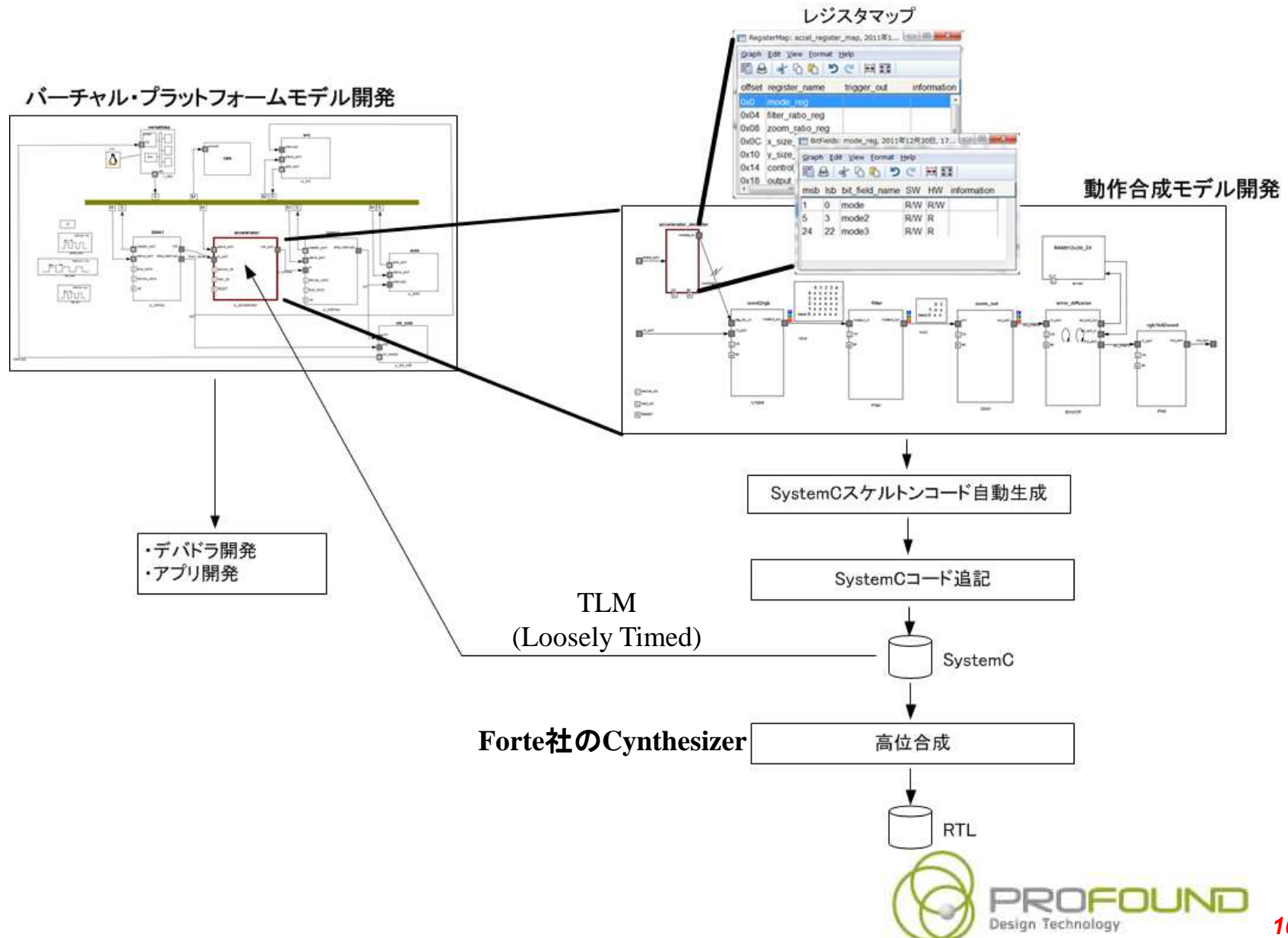
これまでのグラフィカル入カツール



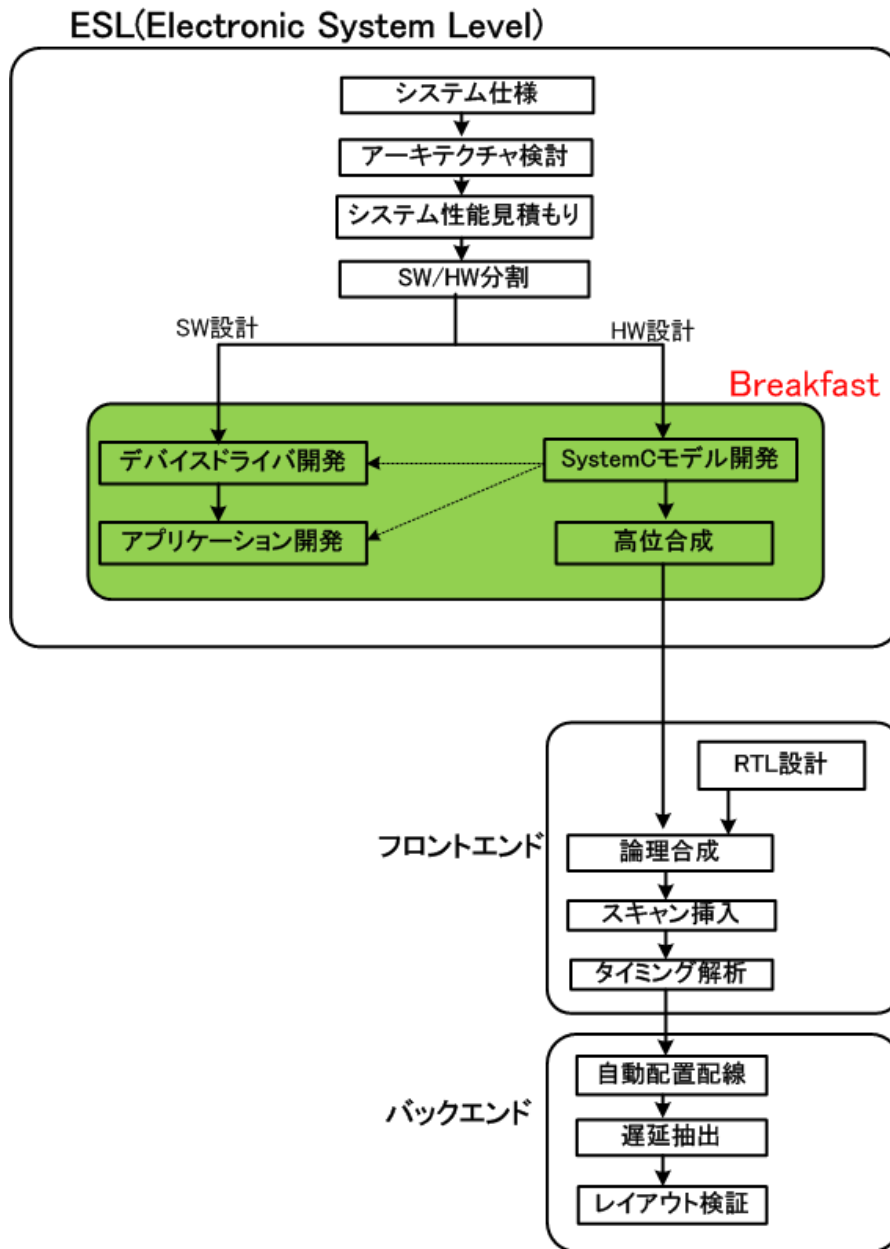
壁



プロファウンドのESLツールBreakfast(開発中)



Breakfastの守備範囲



- ・性能評価
- ・ソフトウェア先行開発
- ・高位合成

Breakfast

Breakfastの特徴

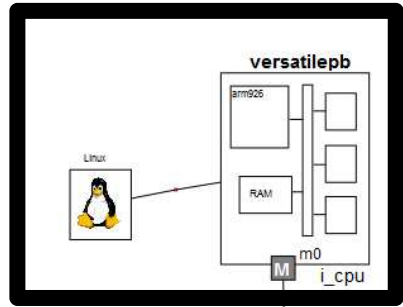
- バーチャル・プラットフォーム(VPF)の構築が容易
 - ◆ CPUモデルはQEMU(フリーソフト)を使用
 - ◆ QEMU(CPUモデル)とSystemCモデルの接続が容易
 - ✓ 割り込み接続も容易
 - ◆ SystemC TLM2.0に関する**深い知識は不要**
 - ◆ 抽象度はLT(Loosely Timed)

- 高位合成用SystemC記述が容易
 - ◆ モジュール間I/F、スレッド間I/FのSystemC記述は自動生成
 - ◆ 画像処理用ラインバッファのSystemC記述は自動生成
 - ◆ 設計者は**アルゴリズム本体のコーディングに集中**

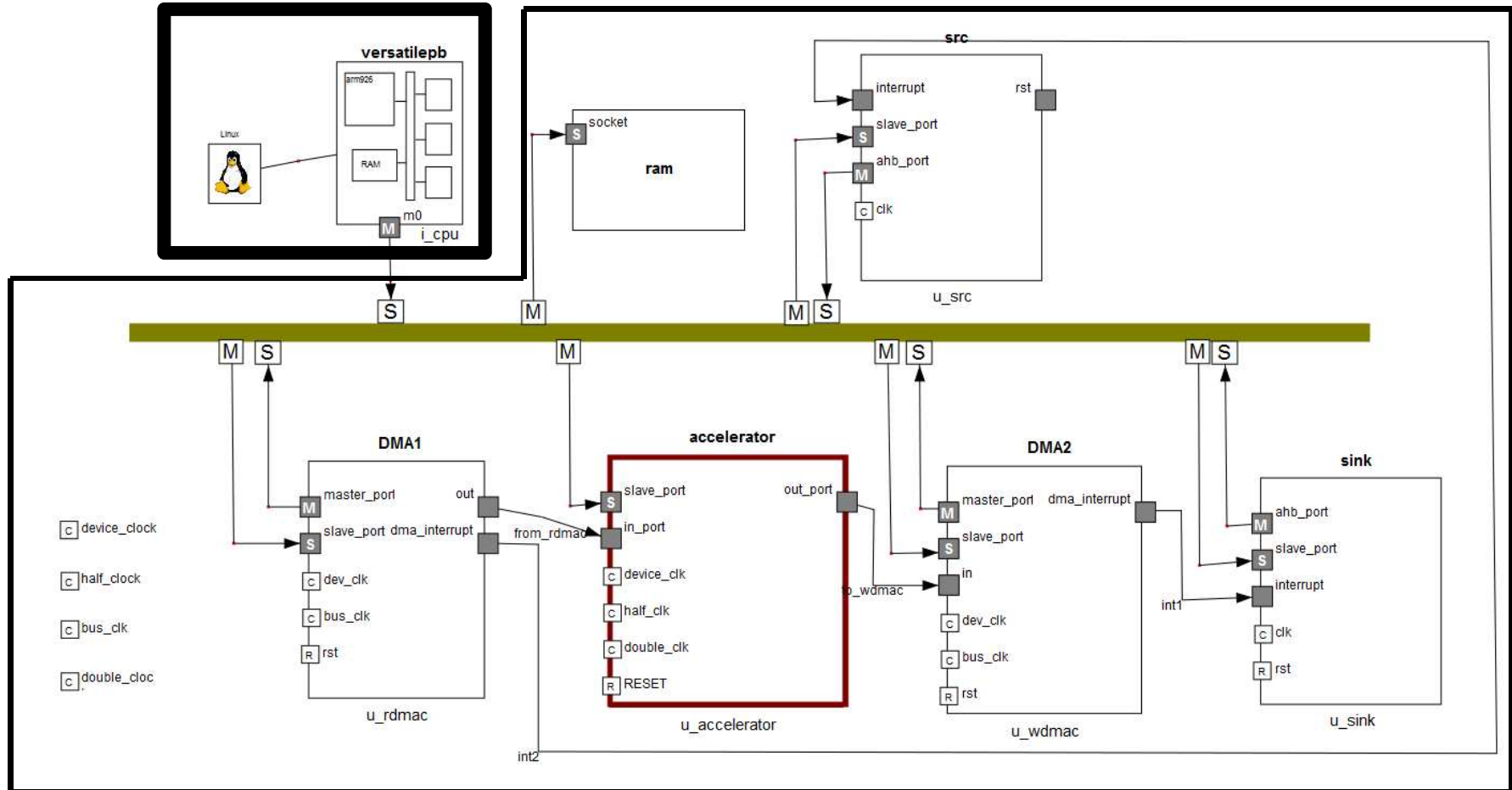
- 高位合成用SystemCモデルをVPFにおいて利用可能
 - ◆ VPFではクロック記述なし
 - ◆ **レジスタI/F(アドレスデコーダ)記述は高位合成とVPFにおいて共通**

QEMUとSystemCの接続(1/3)

QEMU(CPUモデル)



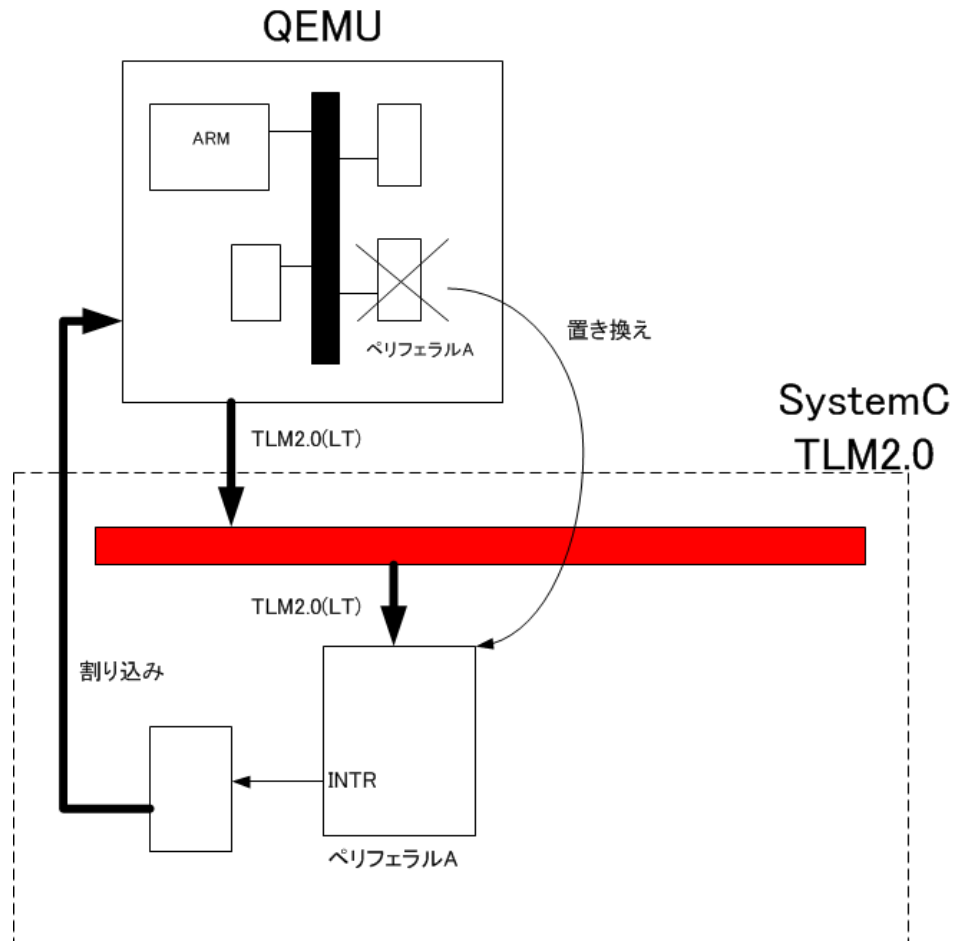
SystemC TLM2.0



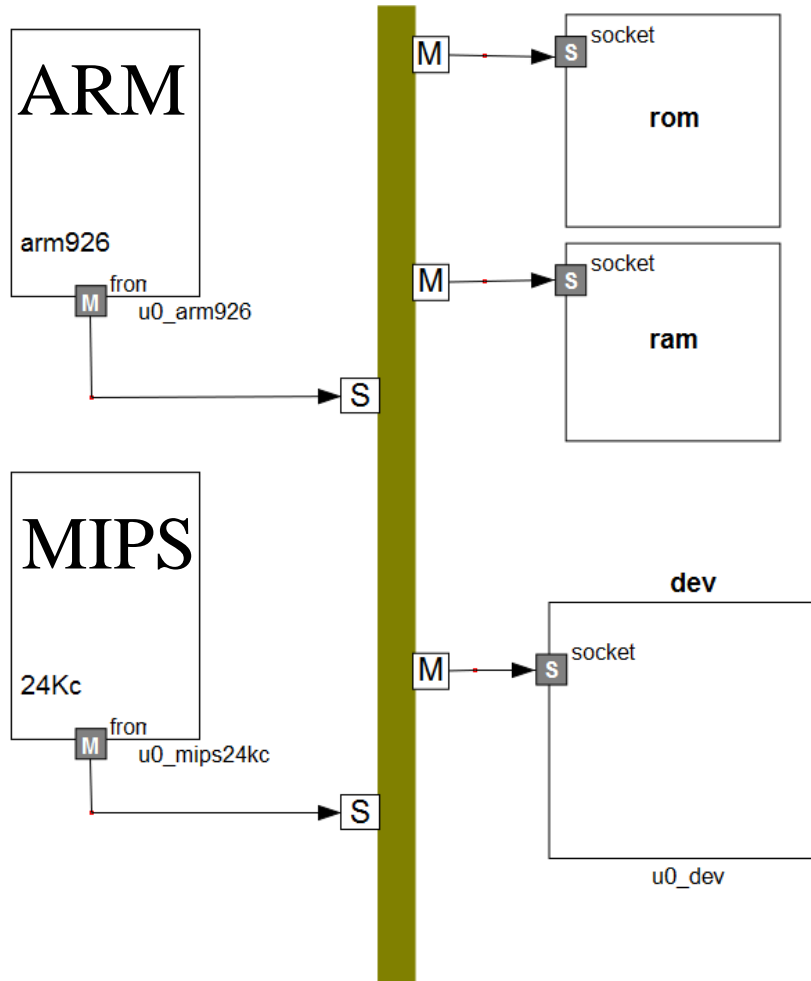
QEMUとSystemCの接続 : GreenSocsやプロセス間通信は使わず

QEMUとSystemCの接続(2/3)

- QEMU内のペリフェラルをSystemCモデルとして外に出すことも可能。
 - ◆ ただしSystemCモデルの作成は必要

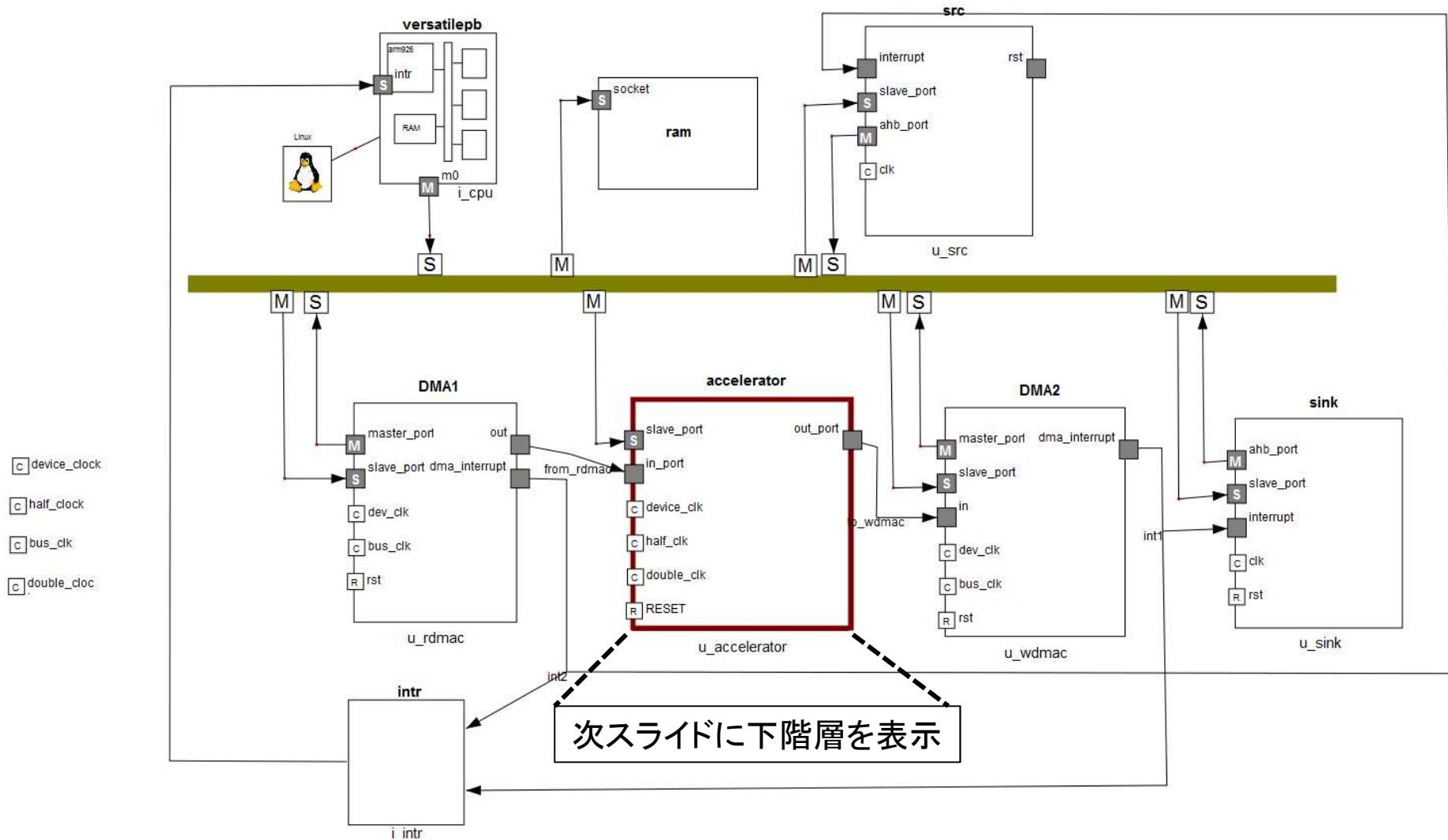


QEMUとSystemCの接続(3/3)

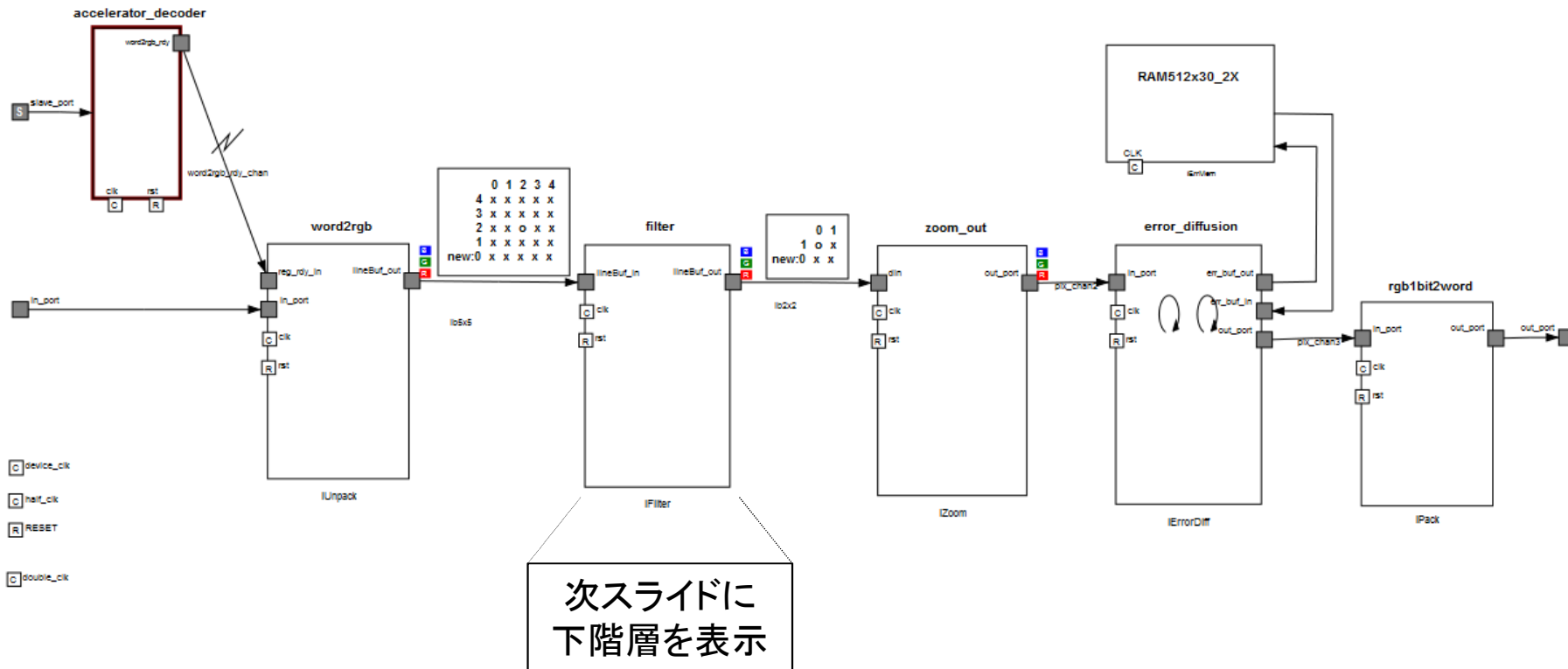


マルチコアにも対応

高位合成向けグラフィカル入力(1/3)



高位合成向けグラフィカル入力(2/3)



高位合成向けグラフィカル入力(3/3)

4:line_buffer: Object: interface_type in port

IF_class_name: lb5

direction: in out

data_type: rgb<8>

data_width: 24

x_width: 5

y_width: 5

pixel_delay: 2

line_delay: 2

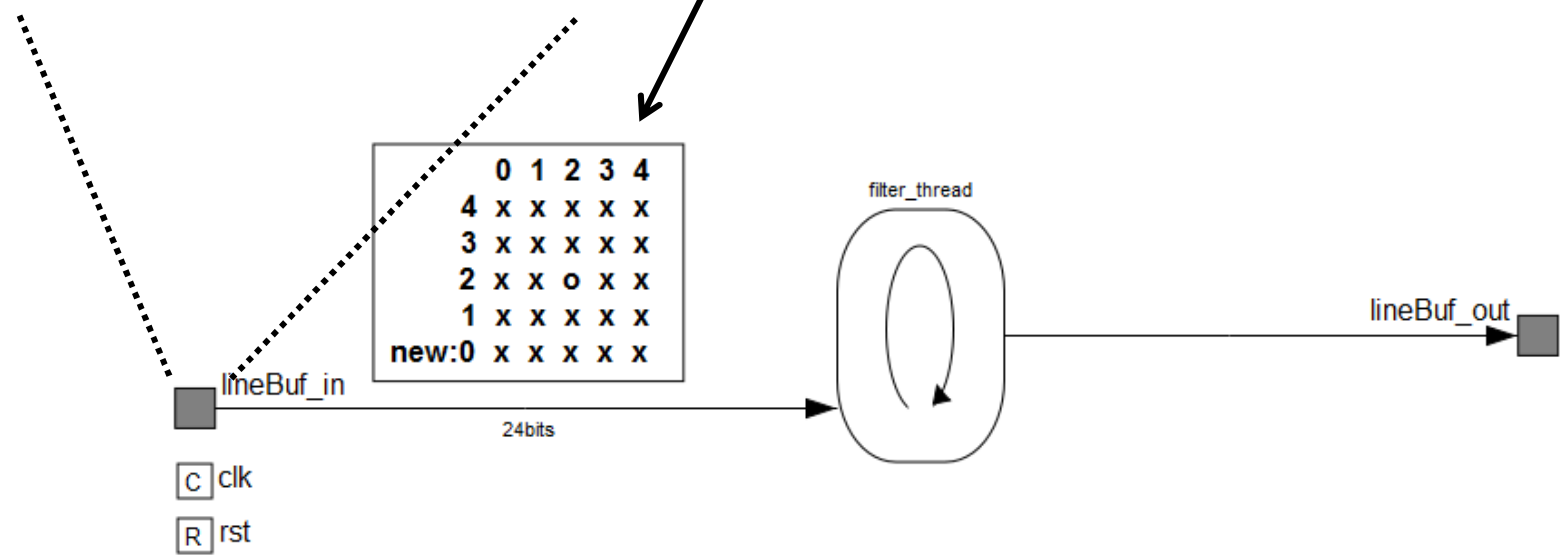
bundle_memory:

dummy_pixel_constant_value: 0

dummy_pixel: 0

ここを入力すると、5x5のシンボルが自動的に表示される

- ラインバッファ回路の自動生成 (Cynthesizerの機能)
- 上階層にも同じシンボルが自動的に表示される
- ドキュメントとしても有益



自動生成されるSystemCコード例

```
ParamT p;  
sc_uint<1> m_reg_rdy_in;  
rgb<8> WorkingSet[5][5];  
rgb<8> m_lineBuf_out ;  
  
reg_rdy_in.wait_trig( m_reg_rdy_in );  
  
while(1){  
    p = param_i.read();  
    lineBuf_in.set_size( p.y_size, p.x_size);  
    lineBuf_out.set_size( p.y_size, p.x_size);  
    lineBuf_in.start_tx();  
    lineBuf_out.start_tx();  
    while( !lineBuf_in.y_done() ) {  
        while( !lineBuf_in.x_done() ) {  
            lineBuf_in.get(workingSet);  
  
            calc( WorkingSet, m_lineBuf_out);  
  
            lineBuf_out.put( m_lineBuf_out);  
        }  
        lineBuf_in.next_y();  
        lineBuf_out.next_y();  
    }  
    lineBuf_in.end_tx();  
    lineBuf_out.end_tx();  
}
```

実際は、コメント文やForte社専用のディレクティブも生成される

設計者は、この関数の中身を記述するだけでOK

あとはCynthesizerを使ってRTLを自動生成

シミュレーション速度(1/2)

- スライド16~18のバーチャル・プラットフォーム(ARM)にてLinuxが起動(起動時間:約20秒)

```
[tukamoto@cent56-64 demo_new]$ make sim_B_f50_z50_b0_TLM
make[2]: 'build_sim_image' に対して行うべき事はありません.
Executing simulation: bdw_work/sims/sim_B_f50_z50_b0_TLM 50 50 0

SystemC 2.2.0 --- Sep 26 2011 19:24:18
Copyright (c) 1996-2006 by all Contributors
ALL RIGHTS RESERVED
NOTE: Forte Design Systems Hub Simulation Platform : version 4.2.1

Warning: (W505) object already exists: top.u_src.command. Latter decl
In file: ../../../../src/sysc/kernel/sc_object.cpp:196
mach_name=versatilepb
vl.c machine->init()
[]

[ 1.777095] registered taskstats version 1
[ 1.780264] rtc-pl031 dev:e8: setting system clock to 2012-02-27 03:53:17 UTC
(1330314797)
[ 1.786544] Initializing network drop monitor service
[ 1.789757] Freeing init memory: 112K
Loading, please wait...
[ 1.839366] udeu[39]: starting version 164
[ 1.859424] input: AT Raw Set 2 keyboard as /devices/fpga:06/serio0/input/imp
ut0
ut0
[ 2.010964] SCSI subsystem initialized
[ 2.038239] PCI: enabling device 0000:00:0c.0 (0100 -> 0103)
[ 2.041936] sym0: <895a> rev 0x0 at pci 0000:00:0c.0 irq 27
[ 2.047156] sym0: No NURAM, ID 7, Fast-40, LVD, parity checking
[ 2.056757] sym0: SCSI BUS has been reset.
[ 2.060000] scsi0 : sym-2.2.3
Begin: Loading essential drivers ... done.
Begin: Running /scripts/init-premount ... done.
Begin: Mounting root file system ... Begin: Running /scripts/local-top ... done.
Begin: Waiting for root file system ... [ 5.059121] sym0: unknown interrupt(s
) ignored, ISTAT=0x5 DSTAT=0x80 SIST=0x0
[ 5.065712] scsi 0:0:0:0: Direct-Access QEMU QEMU HARDDISK 0.15 PQ
: 0 ANSI: 5
[ 5.072471] scsi target0:0:0: tagged command queuing enabled, command queue d
ePTH 16.
```

<実行環境>

- ・実行マシン : Intel Core i7-2600 3.4GHz
- ・OS : CentOS 5.7 64ビット版(Windows7 上でVMWare Player を使用)

シミュレーション速度(2/2)

- 画像処理回路を制御するCプログラムを、スライド16~18のバーチャル・プラットフォーム(ARM)にてLinux上で実行 (sim時間:約3秒)



画像処理回路の出力画像
(160×220ピクセル)

画像処理回路への入力画像
(320×440ピクセル)

各種デバッグ

The screenshot displays a QEMU virtual machine environment. On the left, a terminal window shows the Linux boot process, including system messages and user login. A yellow callout box labeled "Linux起動画面" (Linux boot screen) points to this terminal. On the right, a "sc_main.cpp - Source Window" shows C++ code for a SystemC program. A yellow callout box labeled "SystemCのデバッグ" (SystemC debugging) points to the source code. Below the source window, a GDB debugger window is open, showing the source code with a breakpoint set at line 37. A yellow callout box labeled "アプリケーションのデバッグ" (Application debugging) points to the GDB window. At the bottom left, another GDB window shows the kernel source code for "linux-2.6-2.6.32/debian", with a yellow callout box labeled "Linuxカーネルソースのデバッグ" (Linux kernel source debugging) pointing to it.

Linux起動画面

SystemCのデバッグ

Linuxカーネルソースのデバッグ

アプリケーションのデバッグ

内容

- SystemCの現状
- グラフィカル入カツール Breakfast
~バーチャル・プラットフォームから高位合成まで~
- DSMツールで楽々開発
- まとめ

Breakfastのプロトタイプは3週間で開発

- Breakfastのプロトタイプは仕様を決めてから、3週間程度で開発
 - ◆ グラフィカル入カツールをゼロから開発するとなると通常は数人年かかる
 - ◆ DSMツールを利用し、Breakfastのプロトタイプ開発期間を大幅短縮

DSMって何？

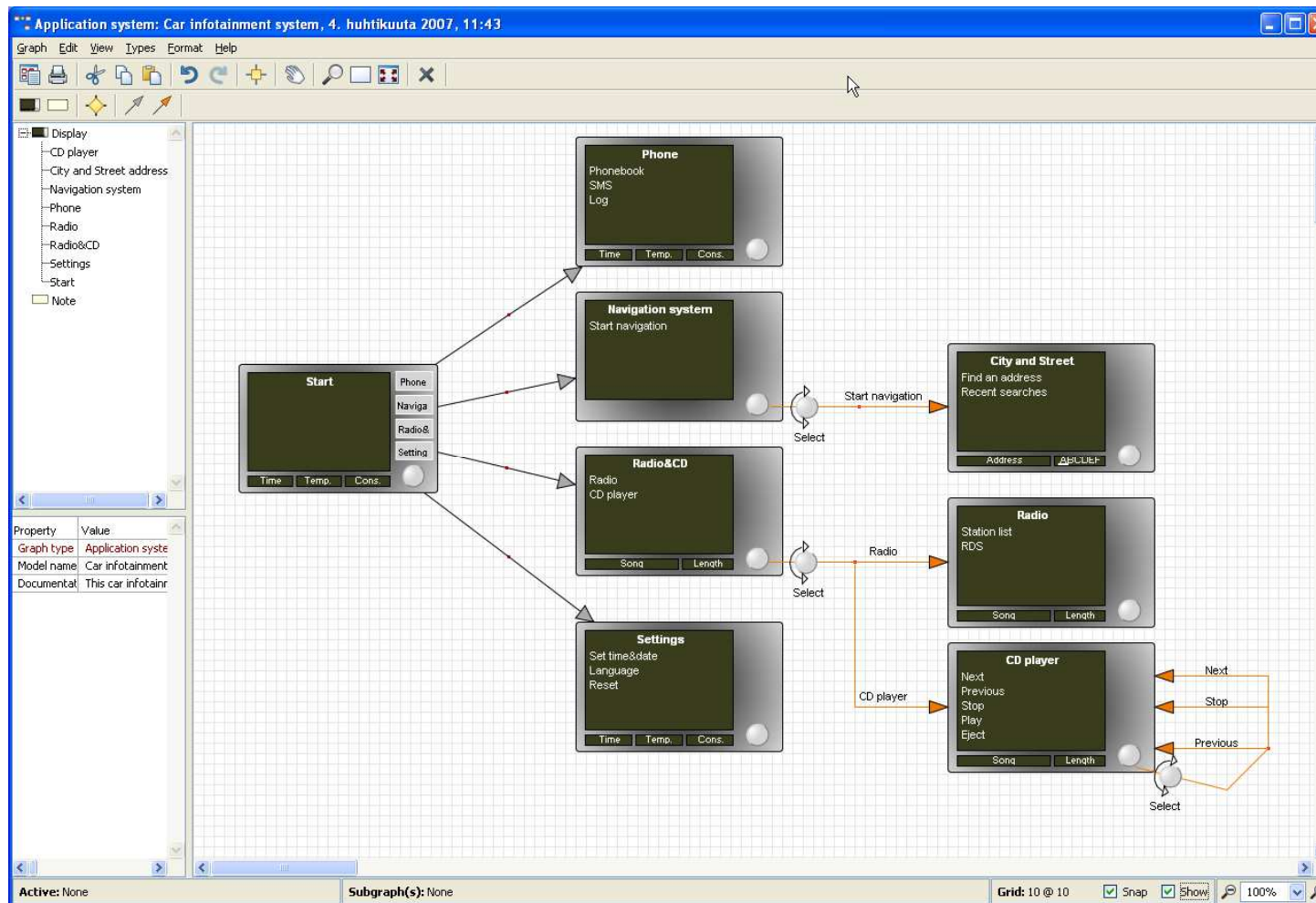
DSMとは？

DSM = ドメイン・スペシフィック・モデリング

- Domain Specific Modelingの略
 - ◆ Domain Specific → 「**特定の分野に特化した**」という意味
 - ◆ Modeling → ざっくり言うと「**絵を使った抽象化**」のこと
 - ◆ モデルからソースコードやドキュメントを自動生成
- MDD(Model Driven Development:モデル駆動開発)の一種
 - ◆ MDDは様々な呼ばれ方をしているが、本質的にはすべて同じ
 - model-driven architecture
 - model-driven engineering
 - model-based software development
 - model-based design
 - model-integrated computing
 - domain-specific modeling

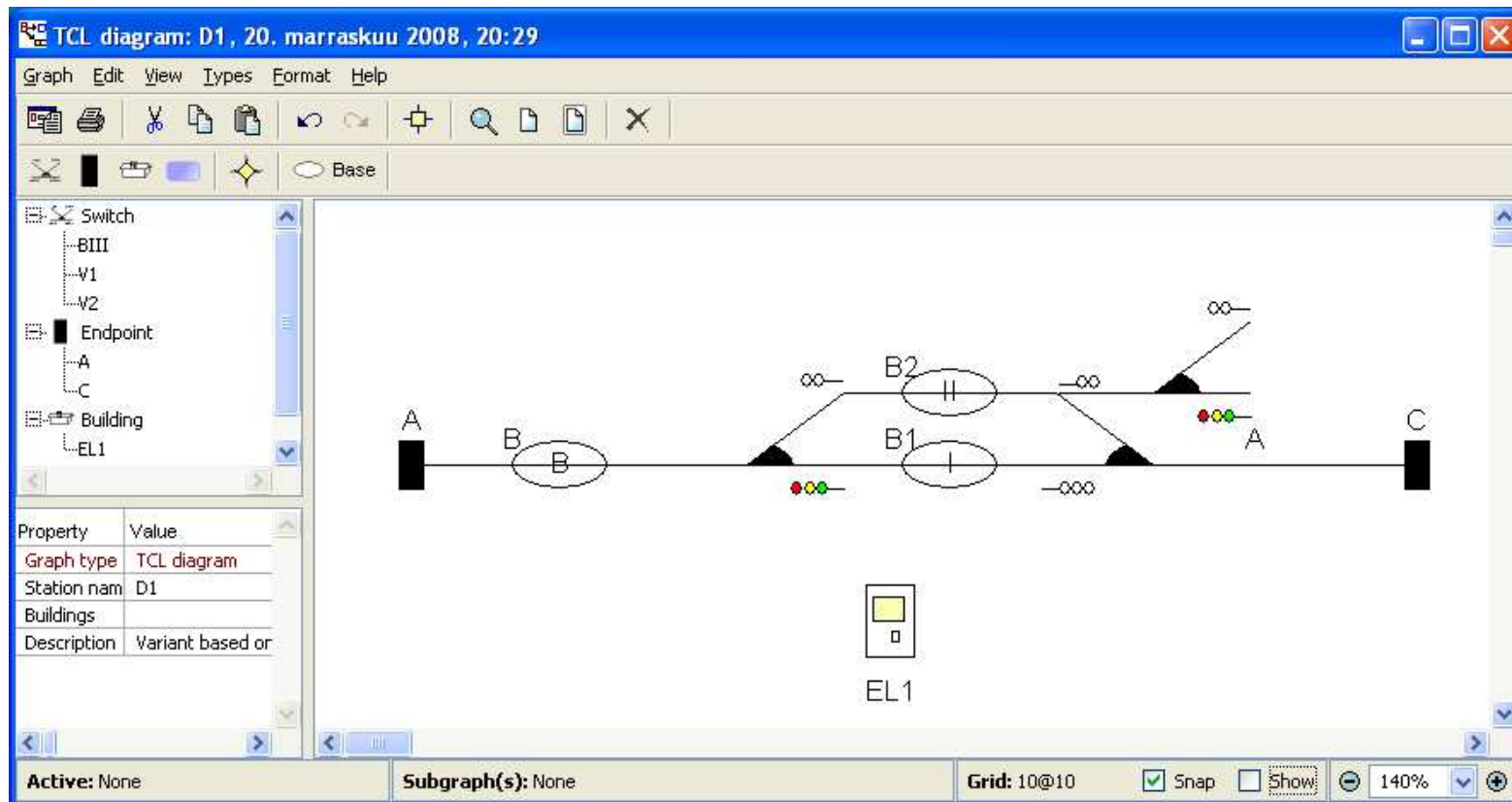
DSMの例(1/7)

自動車のヒューマン・マシン・インターフェイス（カーナビ）



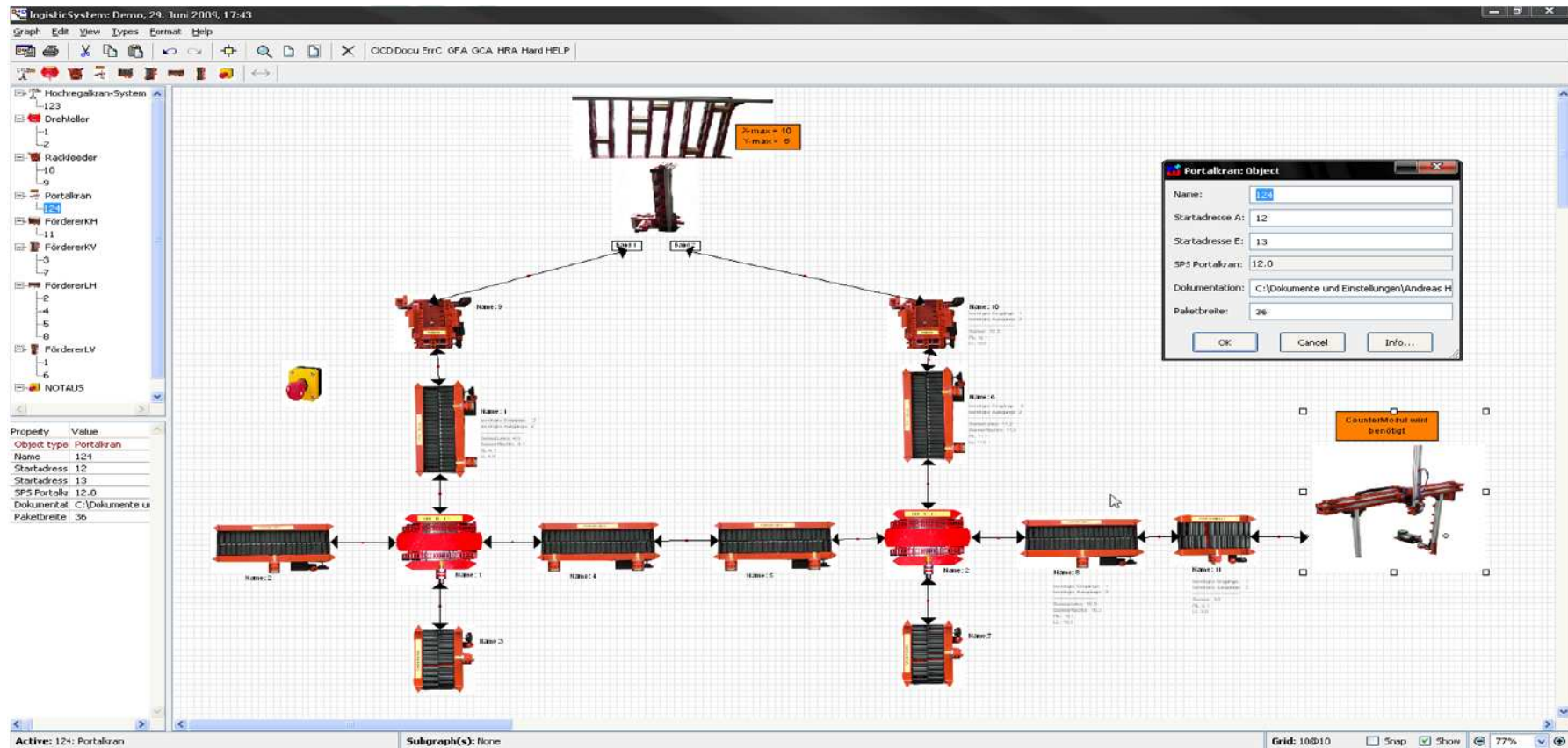
DSMの例(2/7)

列車の連結制御システム



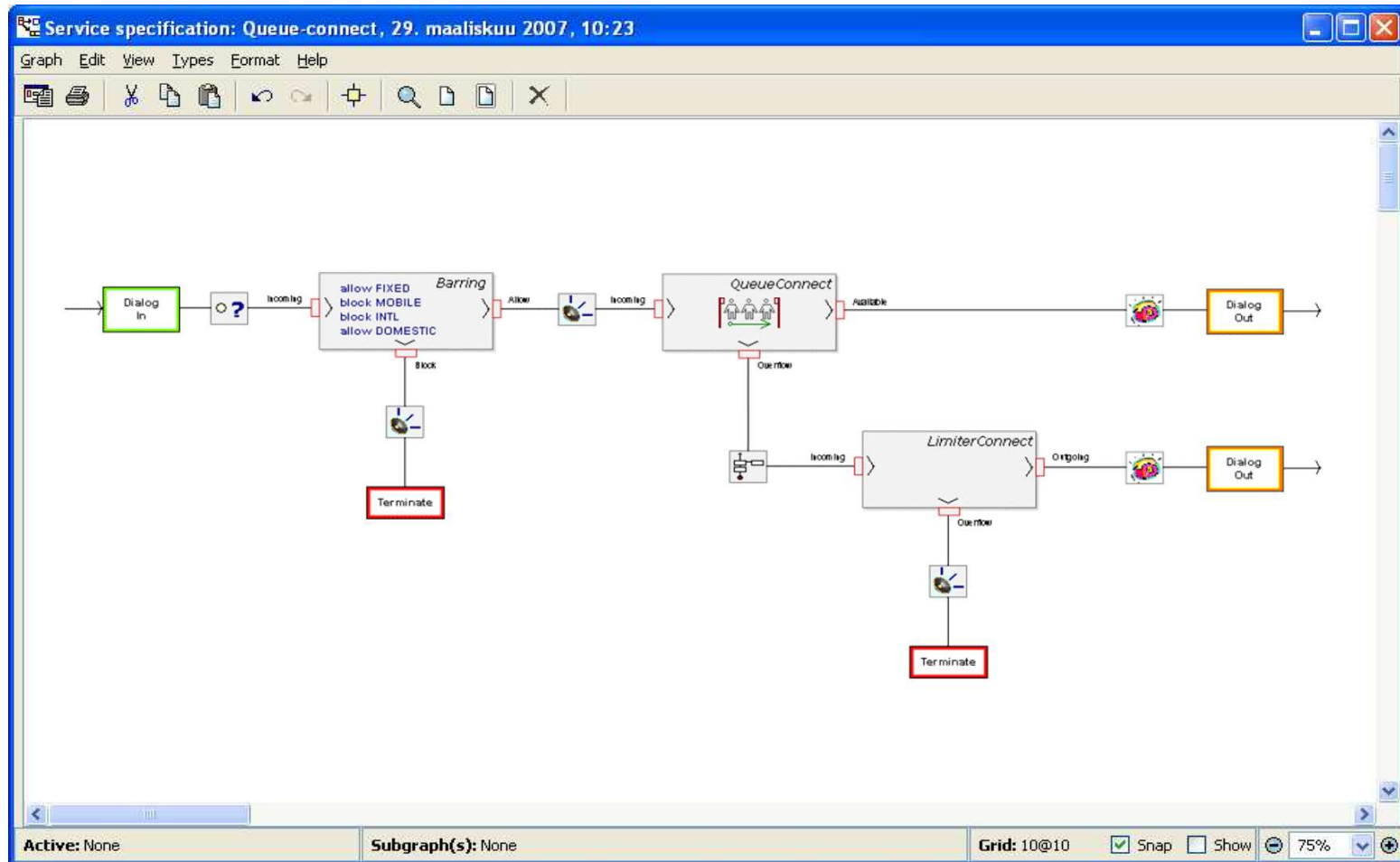
DSMの例(3/7)

産業機器 (FA)



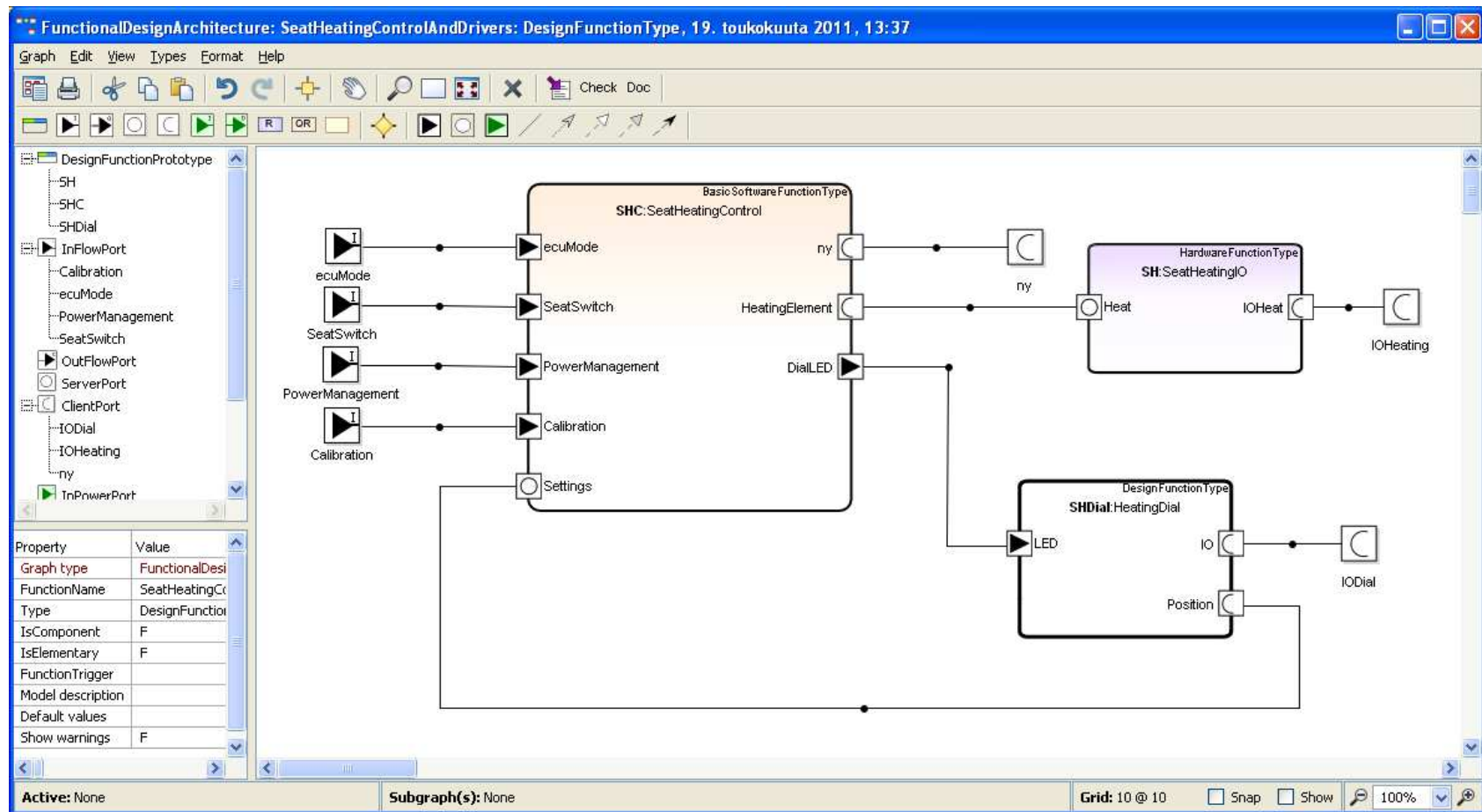
DSMの例(4/7)

電話通信サービス(IMS Service Creation)



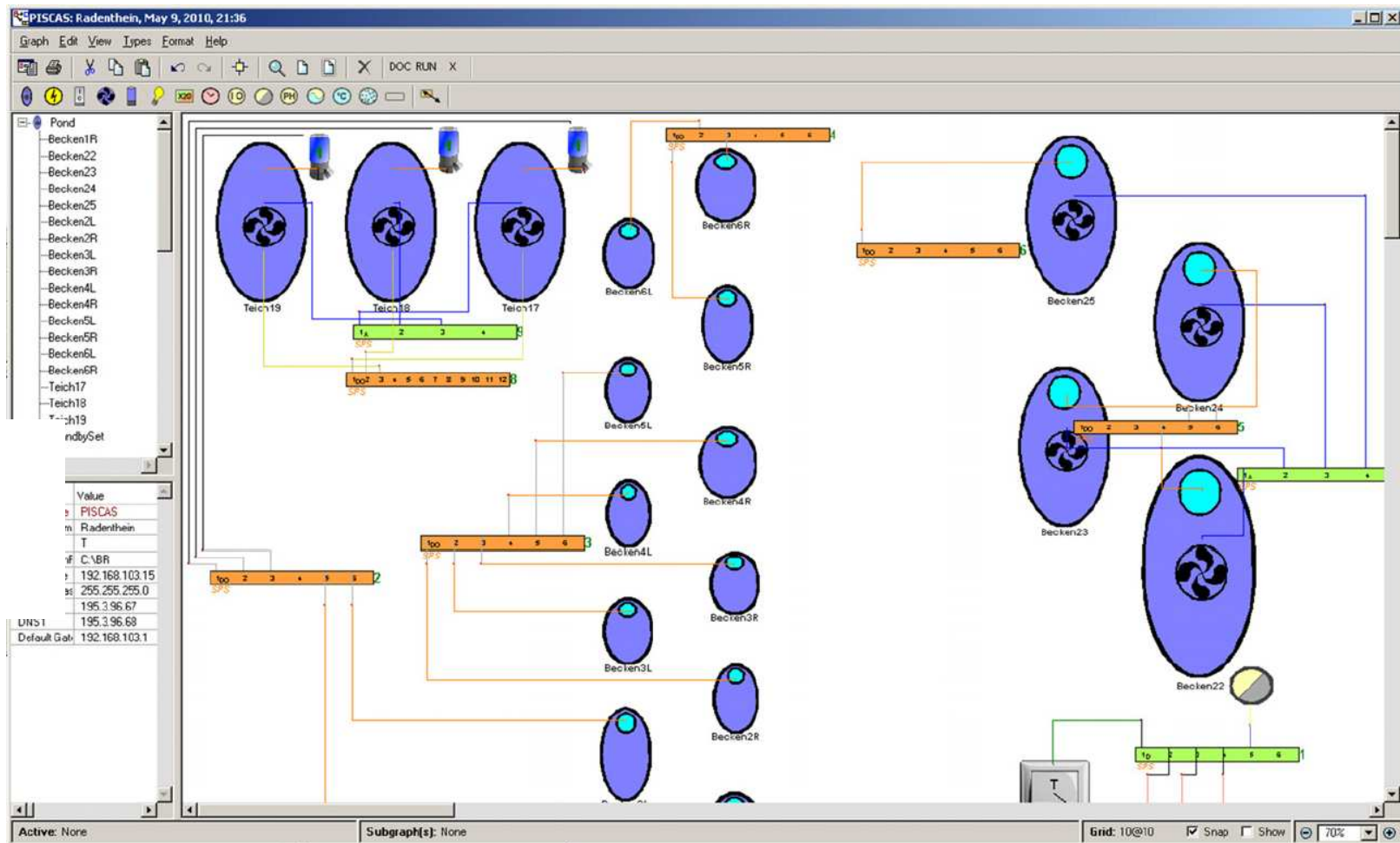
DSMの例(5/7)

車載向けアーキテクチャモデリング (AUTOSAR、EAST-ADLなど)



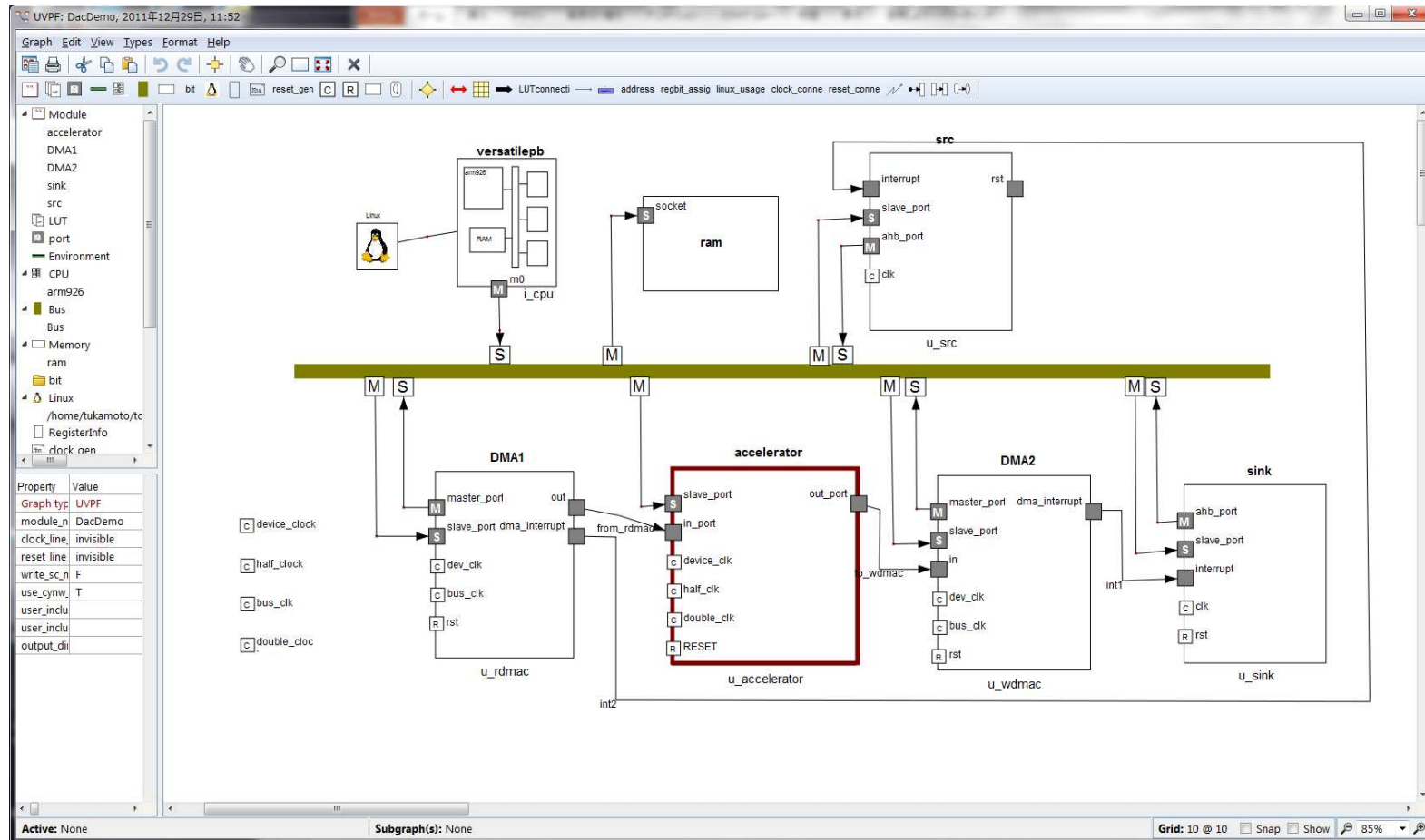
DSMの例(6/7)

養魚場の管理制御システム



DSMの例(7/7)

SoCのバーチャル・プラットフォーム(今回のBreakfast)

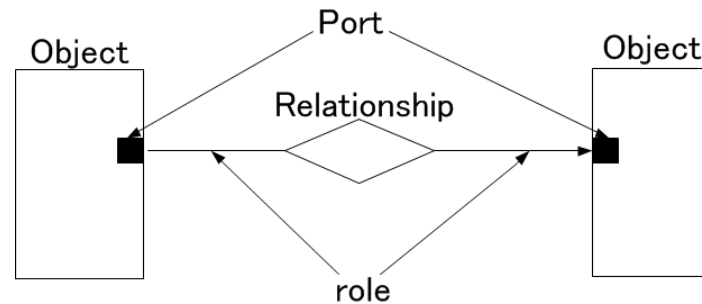


DSMツールMetaEdit+

◆ MetaEdit+とは？

- MetaCase社(フィンランド)のツール(日本代理店:富士設備工業(株))
- Domain Specific Modelingを支援するツール
 - 誤解を恐れずに言うと、「インテリジェントなお絵かきツール」
- すべての絵をグラフとしてとらえる
- グラフの構成要素

- Object
- Relationship
- Role
- Port



➤ 4つの特徴

- Object、Relationship、Role、Portに独自のシンボルを定義できる
- Object、Relationship、Role、Portに独自の各種プロパティを定義できる
- Object、Relationship、Role、Portの接続関係やプロパティの情報を取得できる

→ソースコード・ジェネレータなどの開発が可能

- オブジェクトの配置、配線接続、削除、移動などのお絵かきの基本機能は搭載済み

グラフの例

4:line_buffer: Object: interface_type in port

IF_class_name: lb5

direction: in out

data_type: rgb<8>

data_width: 24

x_width: 5

y_width: 5

pixel_delay: 2

line_delay: 2

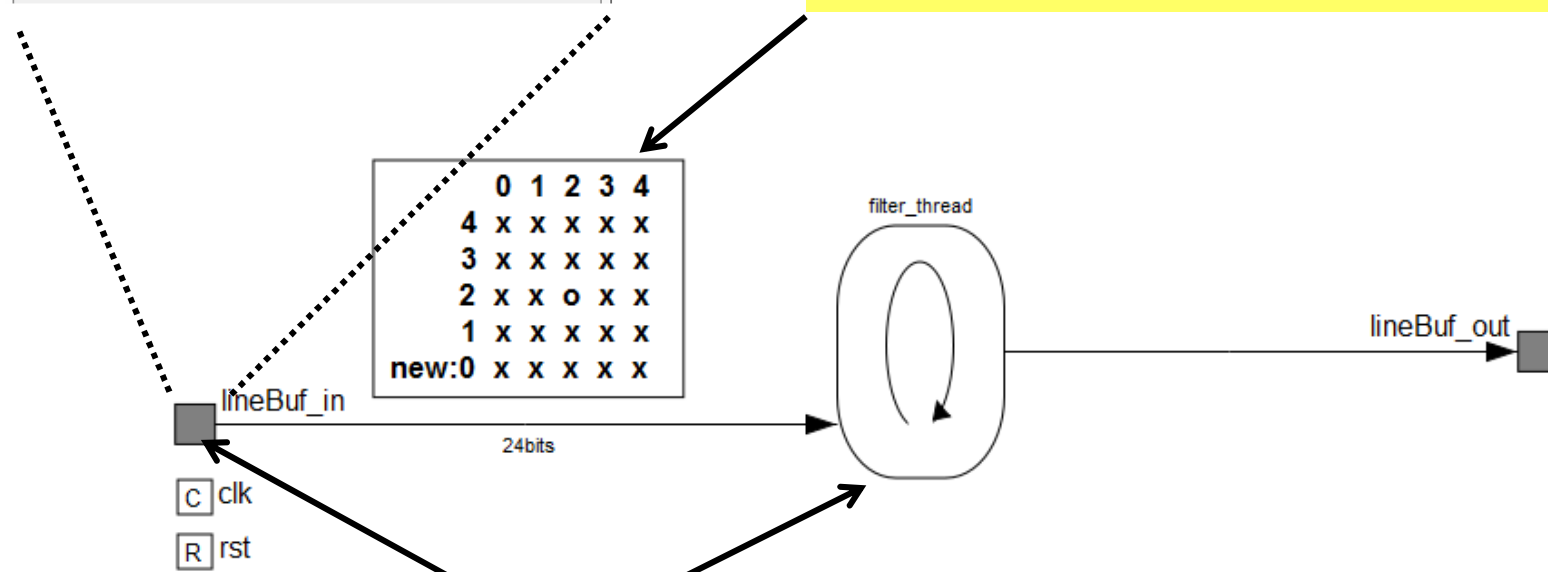
bundle_memory:

dummy_pixel_constant_value: 0

dummy_pixel: 0

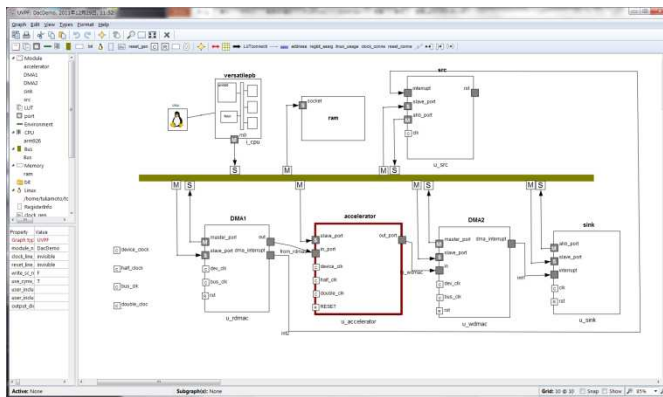
プロパティ(独自に簡単に定義できる)

リレーションシップ(形状は自由に定義できる)



オブジェクト(形状は自由に定義できる)

外部プログラムとの通信も可能



外部プログラム

- ・通信が可能
(SOAP準拠: Simple Object Access Protocol)
- ・外部プログラムから図形の描画なども可能
(描画用の関数が用意されている)

➤ MetaEdit+はいろいろな可能性を秘めている

内容

- SystemCの現状
- グラフィカル入カツール Breakfast
~バーチャル・プラットフォームから高位合成まで~
- DSMツールで楽々開発
- まとめ

まとめ

- プロファウンドのESLツールBreakfastを紹介
 - ◆ QEMUやSystemC TLM2.0に関する深い知識がなくても
バーチャル・プラットフォーム(VPF)の構築が可能
 - ◆ 高位合成用SystemCコードを使ってVPFによるソフト先行開発が可能
 - ◆ ラインバッファやモジュール間I/Fは絵で入力し、SystemCコードを自動生成

- DSMツールMetaEdit+を紹介
 - ◆ MetaEdit+を使ってBreakfastのプロトタイプを3週間で開発
 - ◆ MetaEdit+は様々な分野で利用できる可能性あり
 - ◆ 「どのように絵で表現すればよいか(どのように抽象化すればよいか)」
を考えることが重要