



# Celoxica

>: cutting a LONG story SHORT //

## CeloxicaのC言語ベースハードウェア設計支援 ツール「DK」、ならびに「ソフトウェア・コンパイル・ システム・デザイン」設計手法のご案内

日本セロックシカ株式会社

〒240-0005 横浜市保土ヶ谷区神戸町134  
横浜ビジネスパーク ウエストタワー11F

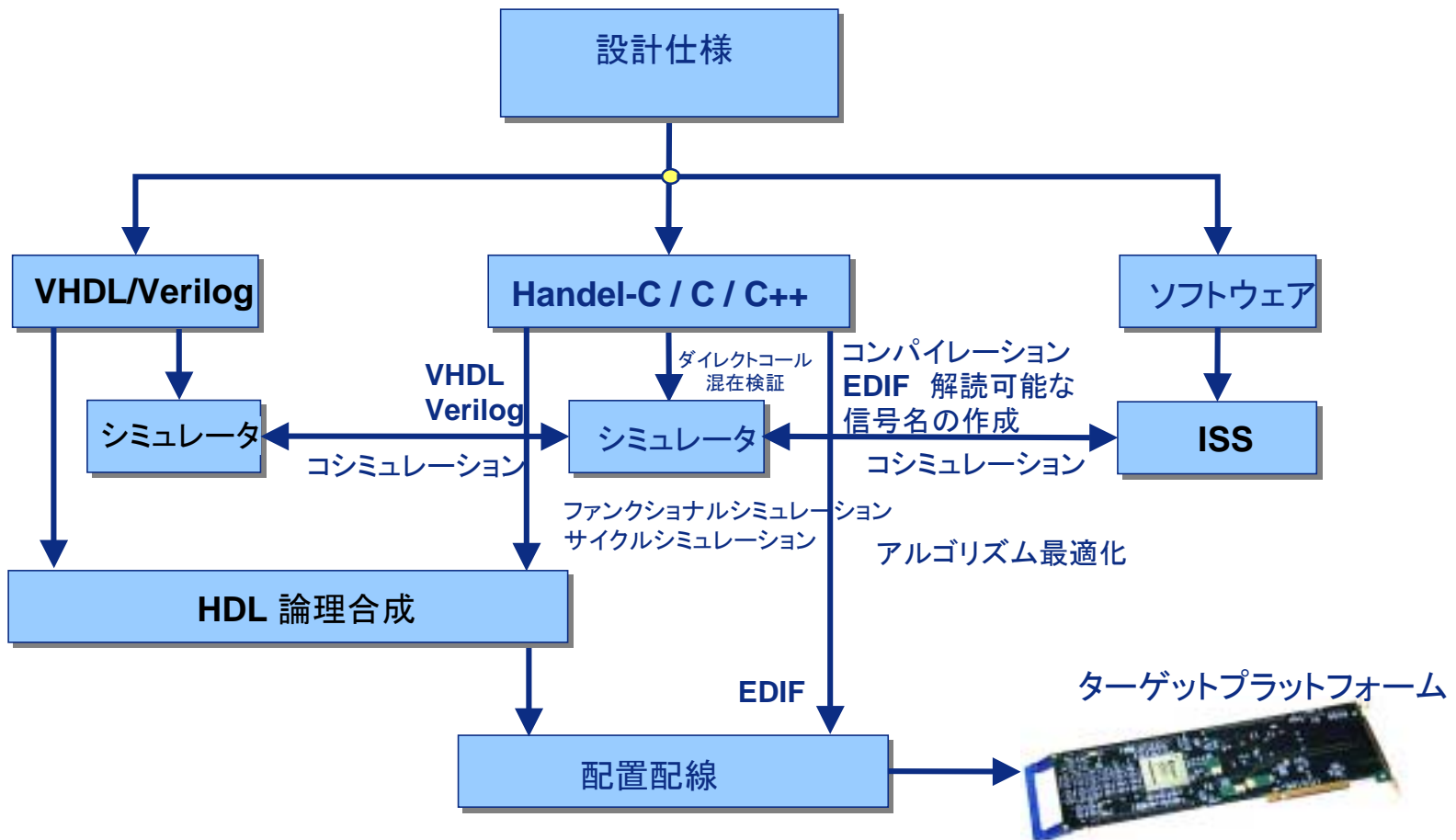
TEL:045-331-0218 FAX:045-331-0433

E-mail:sales.japan@celoxica.com

# Celoxicaのビジョン、目標

- 従来製品や、リ・コンフィギャラブル製品の開発スピードの加速
- 設計の早期段階での検証を可能にする“ソフトウェア・コンパイル・システム・デザイン”設計手法の提供
- 製品・業務内容
  - ANSI-Cの拡張言語“Handel-C”をベースにした統合設計環境「DK デザインスイート」
  - 高速プロトタイピング用開発ボード : RC100, RC1000
  - IP(Handel-C)、設計サービス、コンサルティング
- 高抽象度でアルゴリズム性の強いシステム・デザインへのソリューションを提供
  - ソフトウェア、フォーカス型のデザインには…
    - ソフトウェア/ハードウェアのトレードオフ
  - ハードウェア、フォーカス型のデザインには…
    - 高速プロトタイピング
    - プラットホーム・インディペンデントな設計環境

# Handel-C言語ベース統合設計環境 「DKデザインスイート」 設計フロー



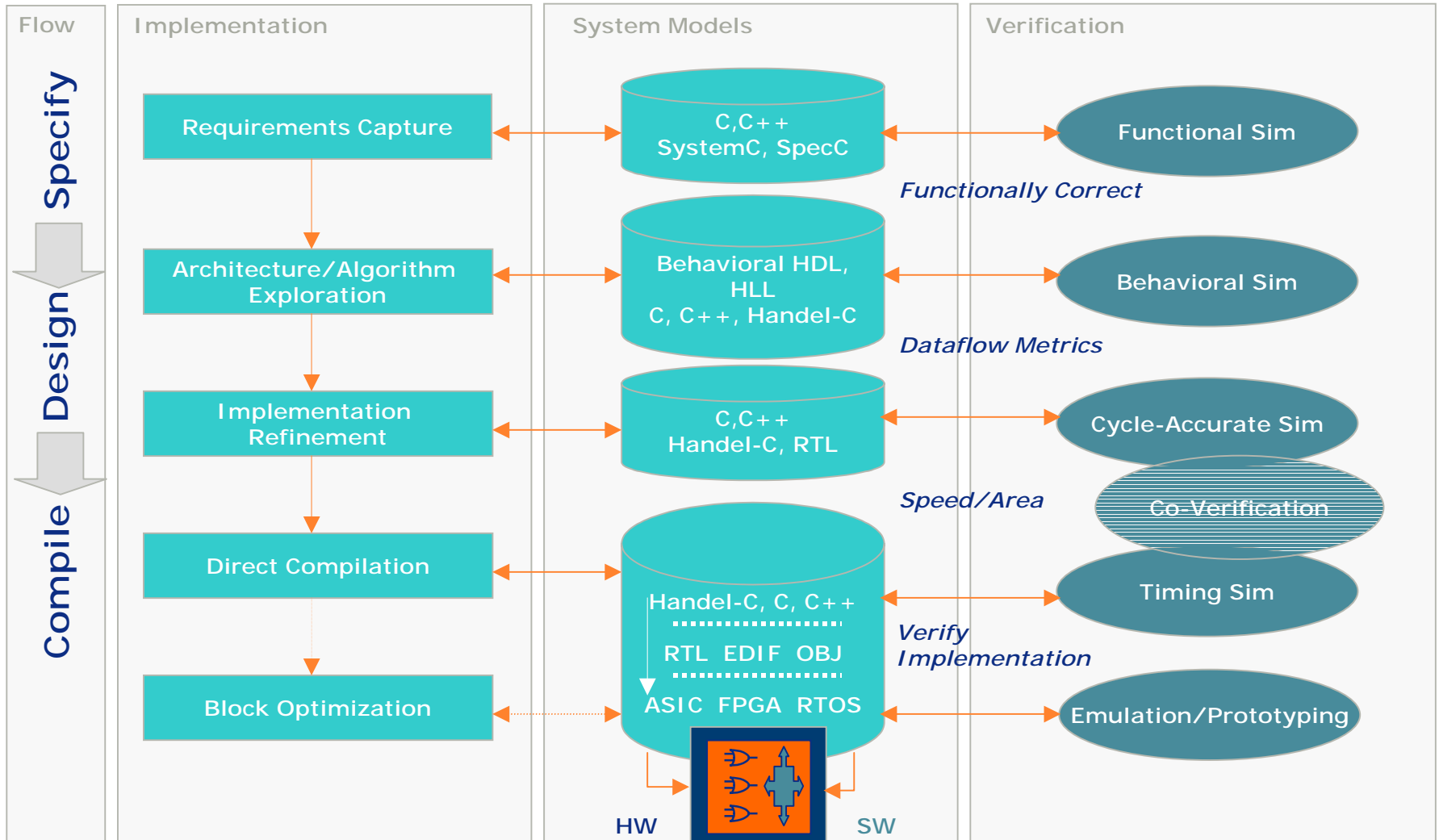
# どのような設計手法が効果的なのか？

- ＞ ハードウェア設計方式はブレードボードとロジック・アナライザを利用した手法から発展してきた。
  - 構造化指向
  - RTLの設計言語
  - 設計リソースに対する詳細な制御（面積、スピード、パワー…）
- ＞ ソフトウェア設計アプローチは対象ハードウェア・アーキテクチャに詳細に対応したプログラミングにより発展してきた。
  - 機能指向
  - 高位レベル言語
  - リソース（メモリー、クロック周波数）のコントロールの必要性は少ない

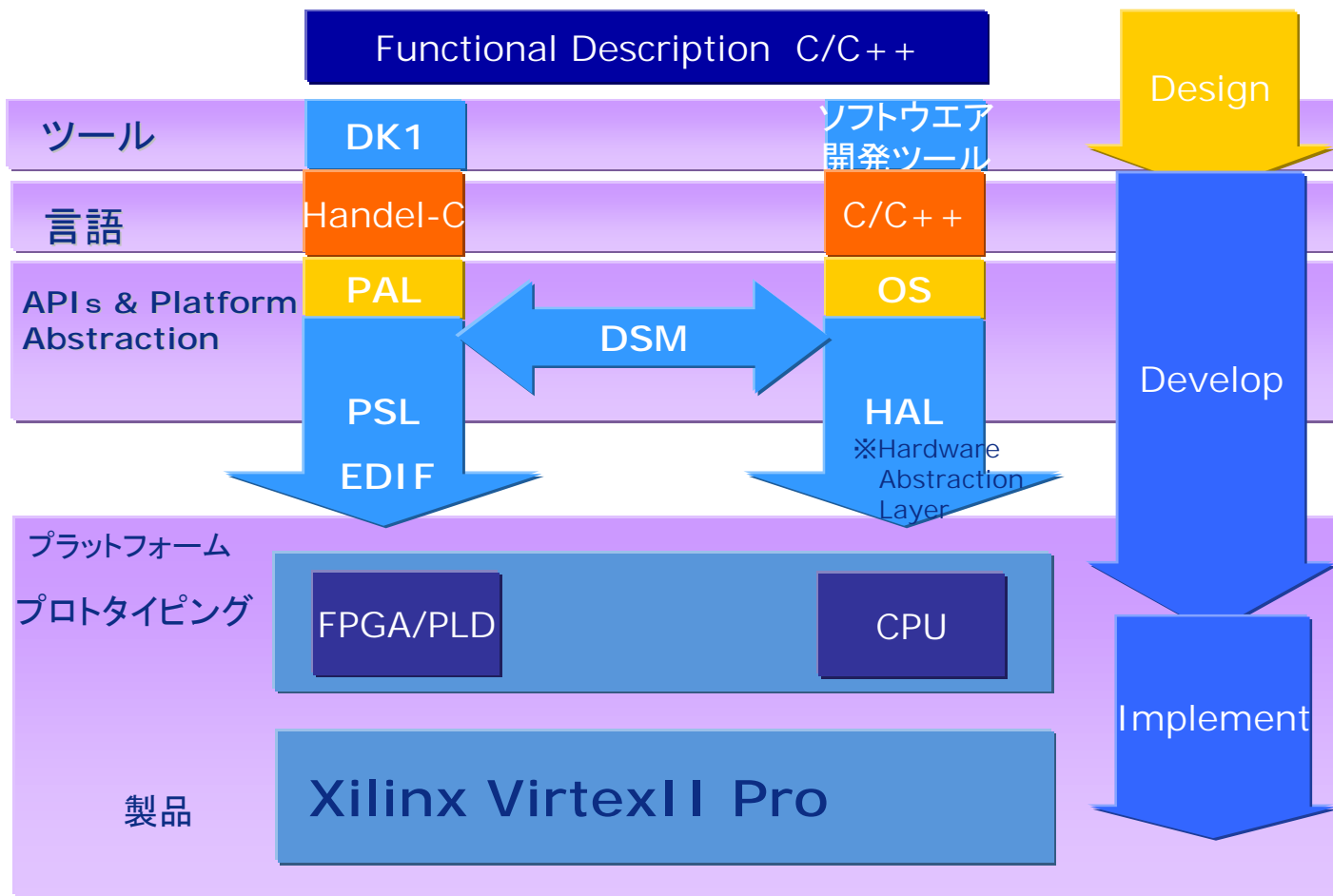


- ＞ 新規プロセッサ及びFPGAテクノロジーを効果的に利用
  - ソフトウェアとハードウェアの設計文化を融合化
  - 従来のソフト/ハードの境界を越えた設計作業

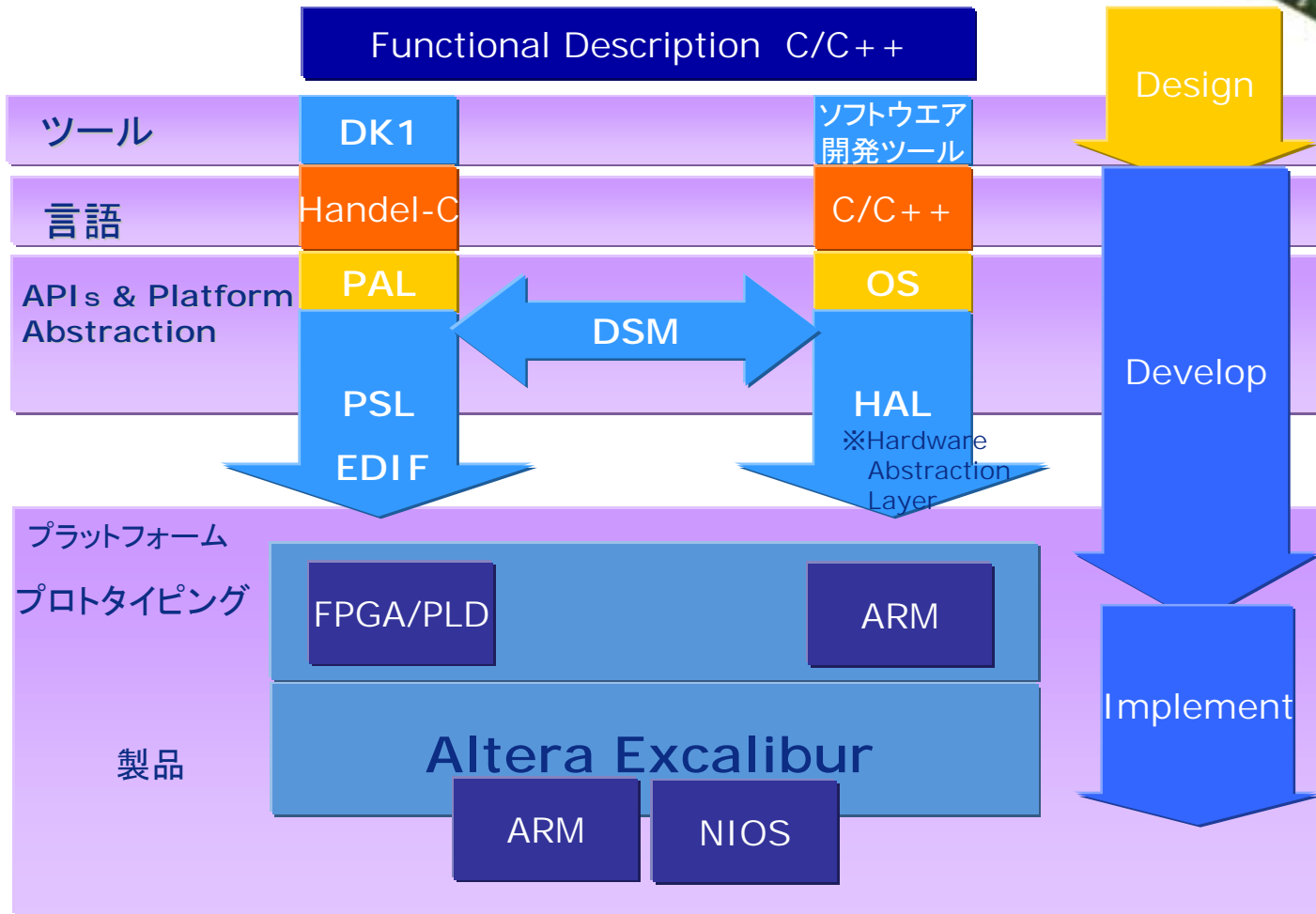
# “ソフトウェア・コンパイル・システム・デザイン” フロー



# VirtexII Pro向けソリューション



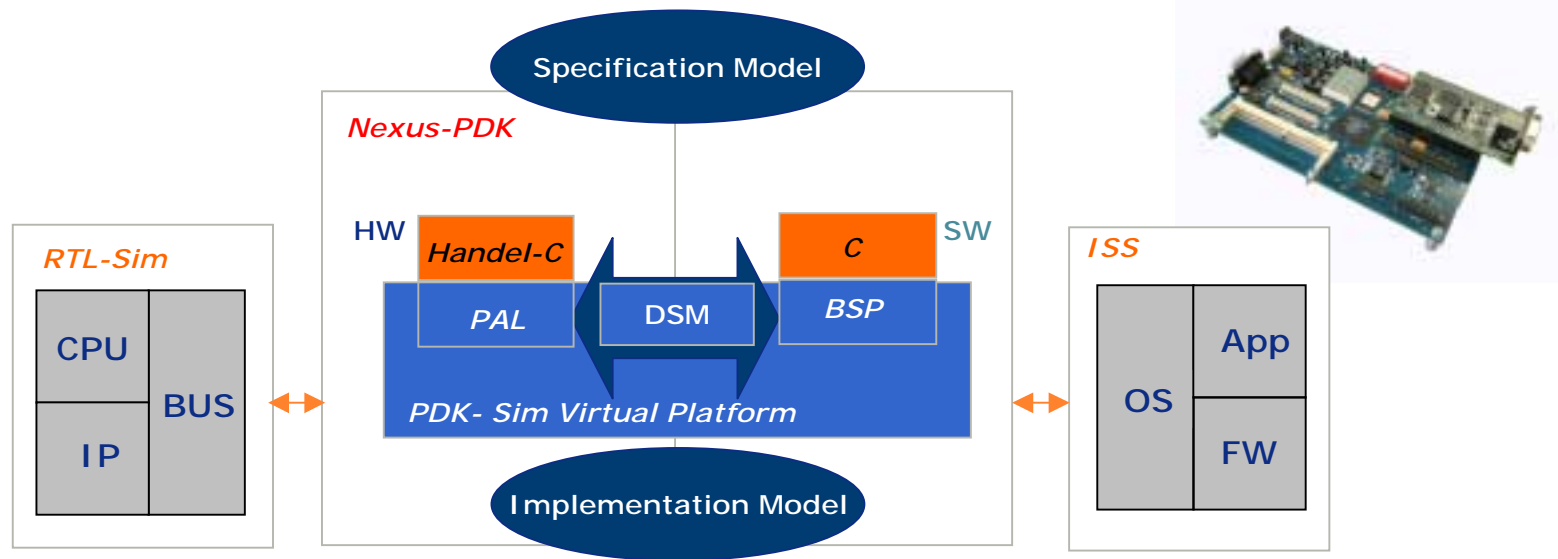
# SoPC向けソリューション



# DK デザインスイート Ver1.1 特長

- > 柔軟性に富んだ、コ・シミュレーション環境
  - 他社製品とのオープンなインテグレーション環境
  - デザイン検証を強く意識したパッケージ
- > ターゲット・ハードウェアとアプリケーション・デザイン間のシンプル・インテグレーションを実現
  - 各種I/Oのインターフェイスを設計者自身が設計コストを削減
  - 多くのハードウェア・プラットフォームへの迅速な対応が可能
    - > アプリケーションの用途が、多岐にわたる
    - > ハードウェア・プラットフォームの品種が非常に多い
    - > ターゲット・ハードウェアへの移植性を圧倒的に向上
    - > Etc...
- > プロセッサに対する過度の負担を、ハードウェアに移行
  - より高速な処理、プロセッサ負担の軽減によるバンドワイドな対応が可能
  - ソフトウェア・アルゴリズムのハードウェア化に対する迅速な対応が可能

# DK デザインスイート Ver1.1 — Nexus-PDK



## > DKシミュレーション環境— Nexus-PDK

- *Handel-C*シミュレーション環境
  - > Handel-C シミュレーション
- コシミュレーション環境
  - > ANSI-C, C++, VHDL, VerilogHDL, System C

## > Platform Abstraction Layer(PAL)

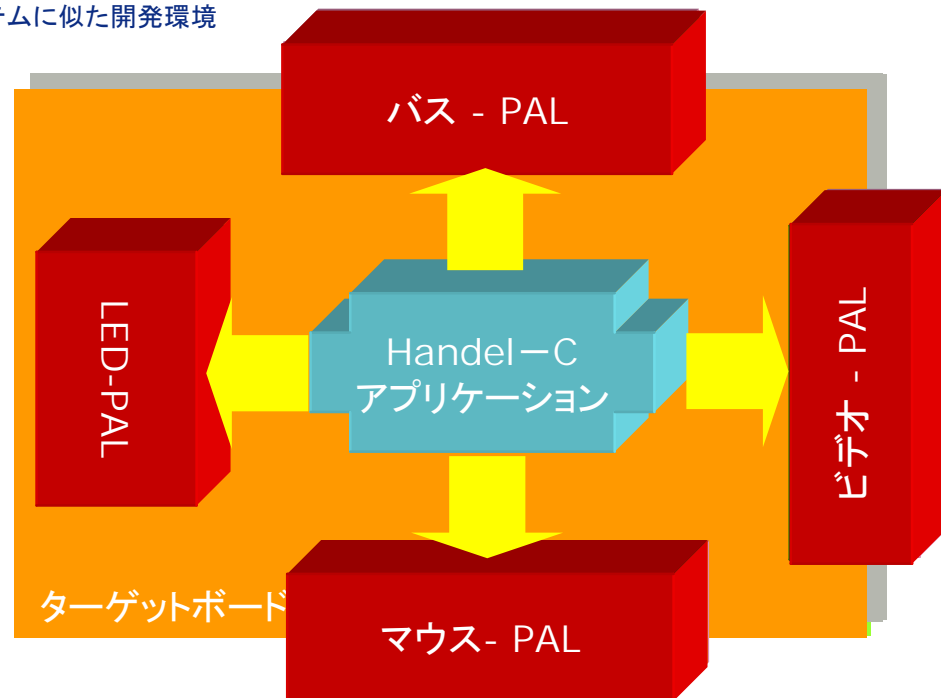
- ハードウェア設計用APIライブラリ
  - > 各種ドライバー ——— PS2、7セグメントLED、USB、etc...
- バーチャル・シミュレーション環境

## > Data Streaming Manager(DSM)

- プロセッサとハードウェアのインターフェイスをとるAPIライブラリ

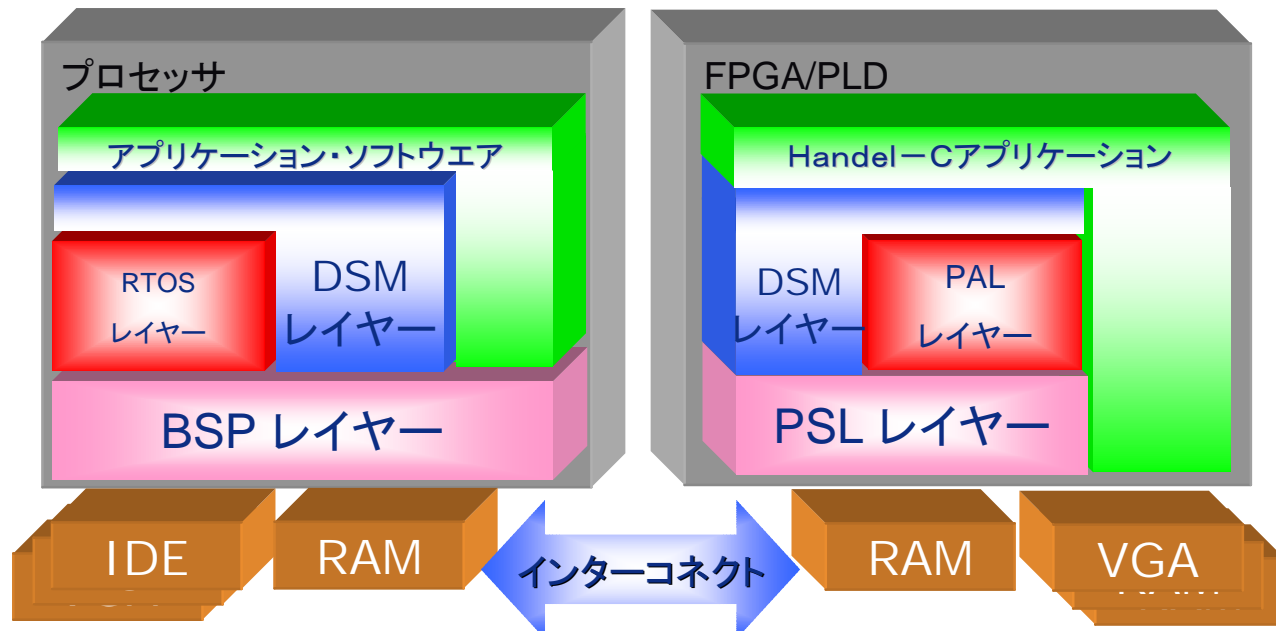
# ターゲットボードに依存しない下位レイヤーを構築するAPI群—“Platform Abstraction Layer”

- > ハードウェア・プラットフォームに依存しない開発環境の提供
  - ハードウェア・プラットフォーム間の移植性を強化
  - ハードウェアの詳細設計を、アプリケーション設計者が考慮する必要のない環境
    - > アプリケーション設計者から見た、共通のインターフェイス仕様 ⇒ PAL
  - 開発期間短縮に貢献
    - > オペレーティング・システムに似た開発環境
      - アクセス関数
      - リンク・ライブラリ
      - Etc...



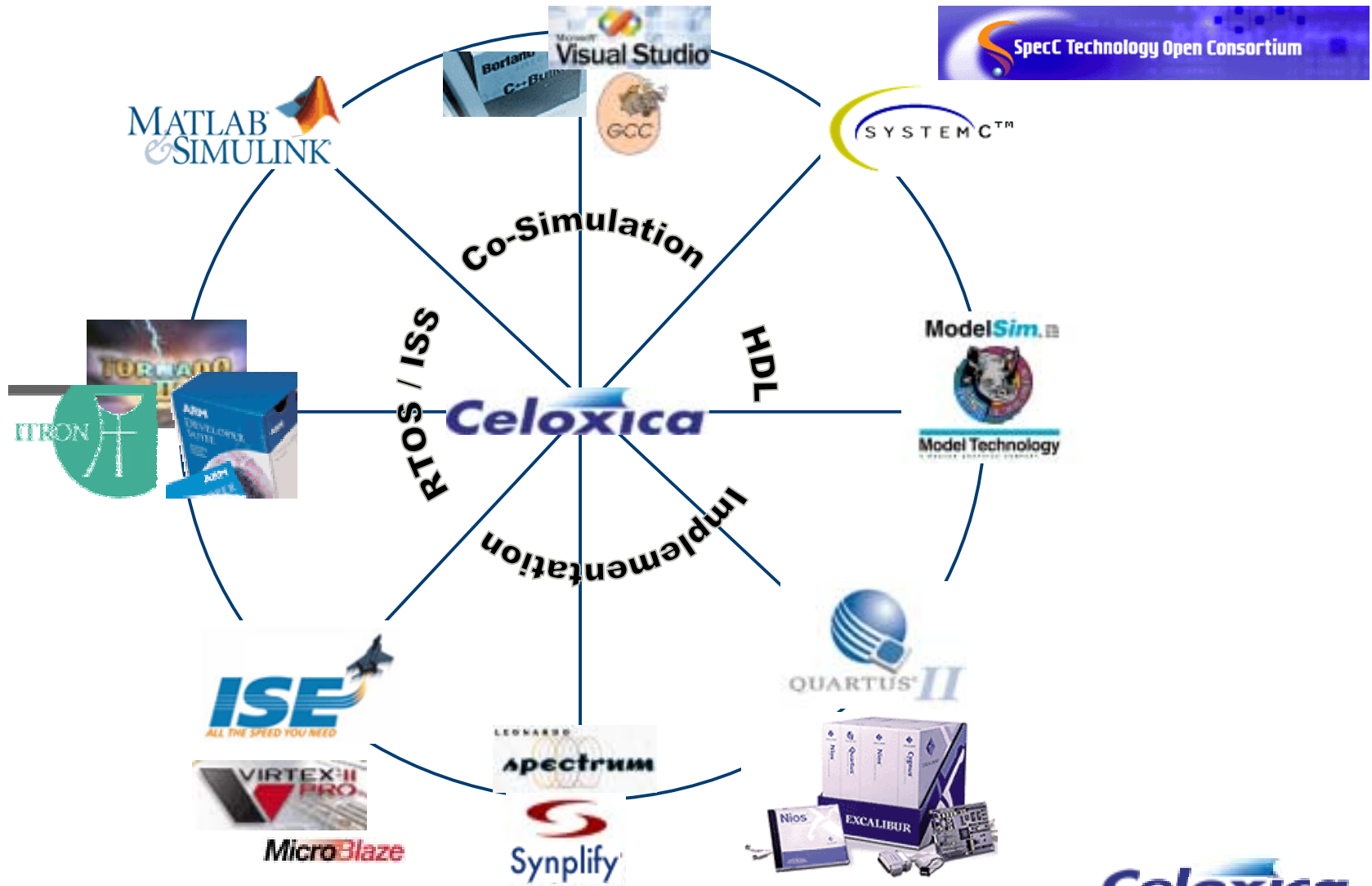
# ハードウェア・ソフトウェアのインターフェイスを容易にする オープンAPIー“Data Streaming Manager(DSM)”

- **DSMによるマイクロプロセッサとHandel-Cデザインのイージー・インテグレーションによる設計工程の加速**
  - > ソフトウェア・アルゴリズムをハードウェア化することによるシステム性能のボトルネックを解消
  - > 組み込みソフトウェア設計者がリコンフィギュラブルなハードウェア・リソースをコ・プロセッサとして使用できる環境を用意
  - > 高性能なハードウェアを開発する為のシステム・ソリューション



DSM開発フロー/設計レイヤー(ソフトウェア開発フローとの比較)

# 「DK」の 360° 自由度のある ツール・コネクティビティ



# HDLシミュレータとのコシミュレーション

The image displays a software development environment for HDL simulation, consisting of several windows:

- DK1 Design Suite [Stop] - [ttl7446\_test.c]**: The main IDE window showing a C program that interfaces with an HDL model. The code defines a `main` function that sets up variables for a 7446 decoder and a bus interface. It includes an `interface` declaration for `TTL7446` and a `bus_out` interface for displaying the decoder's segments.
- ModelSim SE/EE VHDL 5.4e**: A terminal window showing the command `vsim -do (run -all) ttl7446_wrapper` and its output, which lists the loading of various files and plugins for the simulation.
- wave - default**: A waveform viewer window showing a timing diagram with multiple signals. The signals are labeled with `ttl7446_wrapper` and segment numbers (0) through (5). The time axis ranges from 0 ns to 30 ns.
- Simulation Status**: A small window at the bottom left shows the simulation progress, including the current clock cycle (49) and the active thread (`main`).
- Digital Display**: A small window in the foreground shows a 7-segment display with the digit '0' lit up.

# ISSとのコシミュレーション

DK1

The image displays a software development environment with several windows. The main window shows a C program for calculating the Greatest Common Divisor (GCD) using a Euclidean algorithm. The code is as follows:

```
while (x != y) {  
    // Remove any remaining factors of 2  
    par {  
        while (even(x)) x = half(x);  
        while (even(y)) y = half(y);  
    }  
  
    // Now subtract the smallest number  
    // We halve the result immediately  
  
    if (x < y)  
        y = half (y - x);  
    else if (x != y) x = half (x - y);  
    else delay;  
}  
  
// x and y are now equal. so just return x  
while (n != 0) {  
    par {  
        n = n - 1;  
        x = twice(x);  
    }  
}
```

The environment includes a project explorer on the left, a code editor, a console window showing the execution output, and a system output monitor. The console output is:

```
enter a number: 84763426  
enter a number: 64532  
hardware: gcd: 2. cycles: 58  
software: gcd: 2. cycles: 401
```

The system output monitor shows the start of software execution and the command-line options. The ARM6\_1 console window shows the execution of the GCD algorithm.

# ハードウェア-ソフトウェア コシミュレーション & バーチャルプラットフォーム

DK1

The image displays a multi-windowed software development environment. The top-left window shows a Visual Studio IDE with a C++ source file named 'gunzip.cpp'. The code defines a 'GzipAgent' class with methods for 'GzipAgent(int argc, char\*\* argv)', 'GzipAgent::GzipAgent()', and 'GzipAgent::GzipAgent()'. The top-right window shows another Visual Studio IDE window with a C++ source file named 'gunzip.cpp', displaying code for 'GzipAgent' and 'GzipAgent::GzipAgent()'. The bottom-left window is titled 'DK1 Simulation' and shows a hardware simulation interface with a grid of LEDs and a keyboard. The bottom-right window is a terminal window titled 'F:\SPGC\Projects\Intel\ADK\Developer\Examples\Gunzip\Win32-Debug\gunzip.exe' showing the output of the 'gunzip.exe' program. The terminal output includes: 'GV Agent Simulation (c) Celoxica Ltd 2002', '>> Gunzip Agent <<', 'Please enter the name of the compressed file, e.g. test.gz', 'itest.gz', 'Successfully opened "test.gz"', 'Original (uncompressed) filename was "temp.doc". Do you wish to use this name for the decompressed file? (y/n)', 'y', 'Successfully opened "temp.doc"', 'Sending Data...', and a final dash '-'. The 'DK1' text is overlaid on the top-left corner of the image.