Восьмая независимая
научно-практическая конференция
«Разработка ПО 2012»

1 - 2 ноября, Москва

2012 CEE-SEC(R)
Software Engineering Conference in Russia

Трансформация программного обеспечения в микросхему: рутина или творчество?

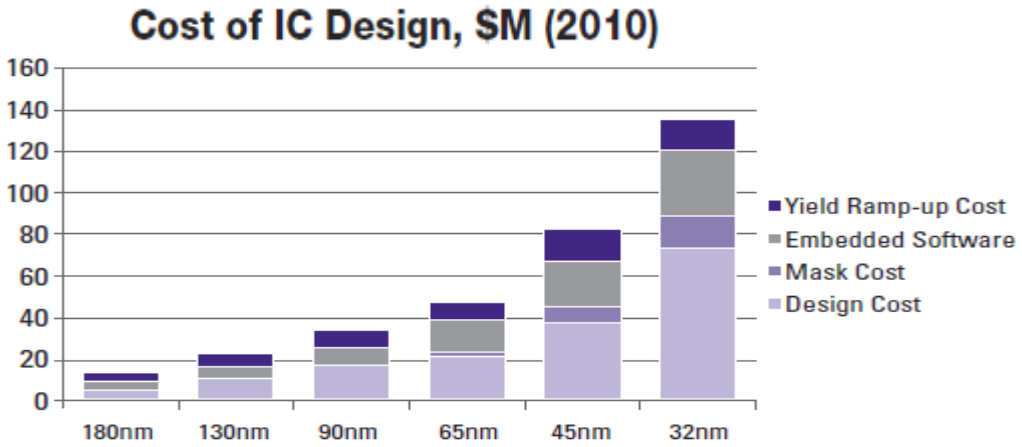
Леонид Пурто

SPIRIT DSP
Voice & Video Engine Experts

www.spiritdsp.com

www.spiritnavigation.com

Сколько стоит разработка микросхемы?

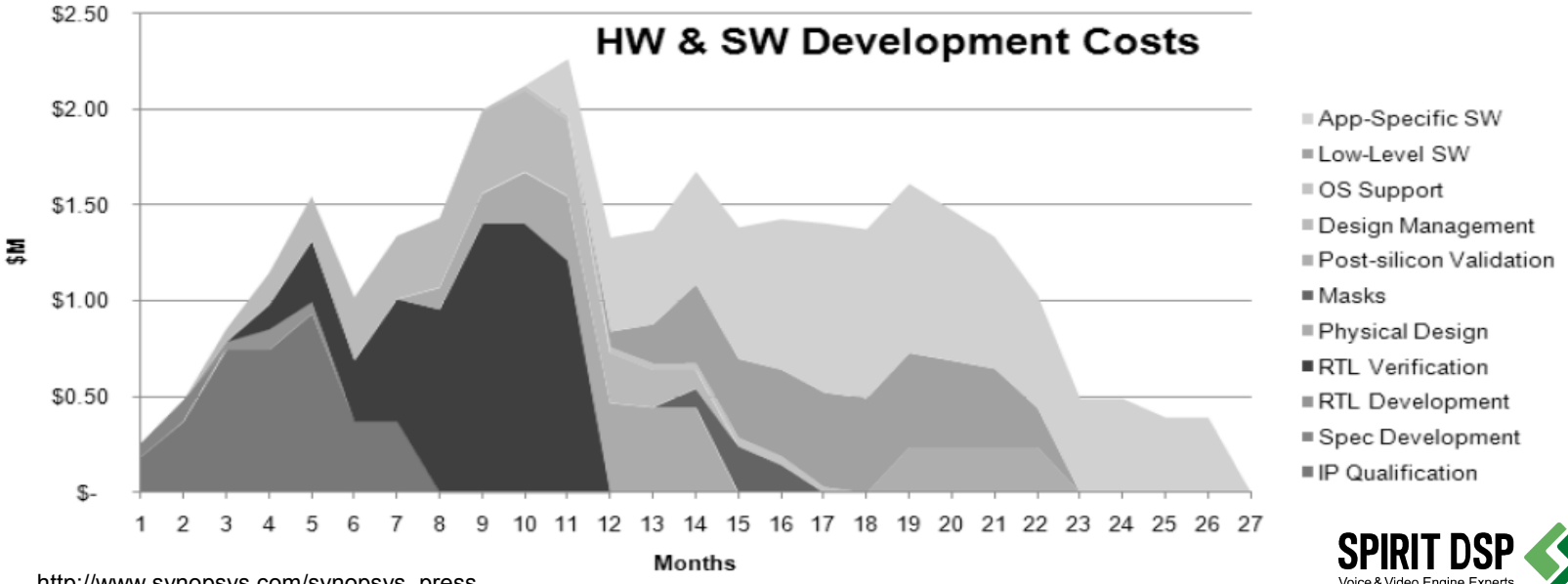


http://www.gsaglobal.org/forum/2011/3/articles_keystone.asp

Пример (Synopsys 2011г):

A wireless headset design by a large semiconductor company, performed in a mainstream 65nm technology

In total the development cost for the project is estimated as $31,650,000.



http://www.synopsys.com/synopsys_press

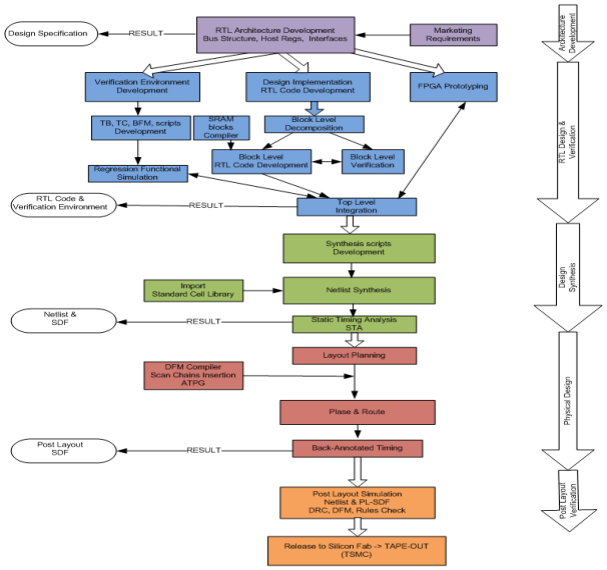
Факторы, определяющие стоимость разработки микросхемы

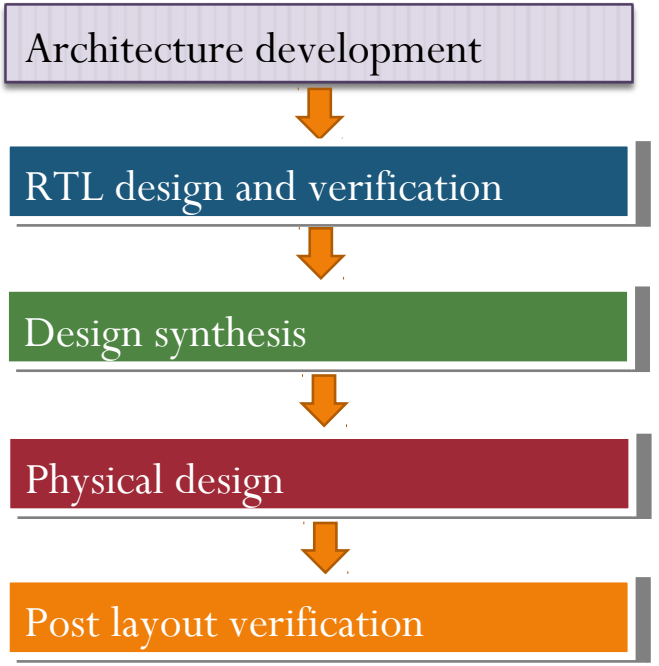
- Чем глубже в “субмикрон”, тем выше стоимость разработки
- Стоимость проектов даже для одного техпроцесса может отличаться на порядок
- Для оценки стоимости проекта Global Semiconductor Alliance (GSA) предлагает “IP ROI calculator” с сотнями вопросов
- Разнообразие проектов не отменяет превалирования расходов по двум основным статьям (на них приходится 2/3 от всех затрат):
  - Разработка программного обеспечения (встраиваемого и сопроводительного)
  - Разработка описания микросхемы на уровне регистровых передач (RTL)

Маршрут проектирования микросхемы

- Разработка ASIC = разработка аппаратной части (hardware IP) + встраиваемого программного обеспечения
- Маршрут проектирования аппаратной части: архитектура → RTL → синтез → физический дизайн → сэмплы

IC design flow:

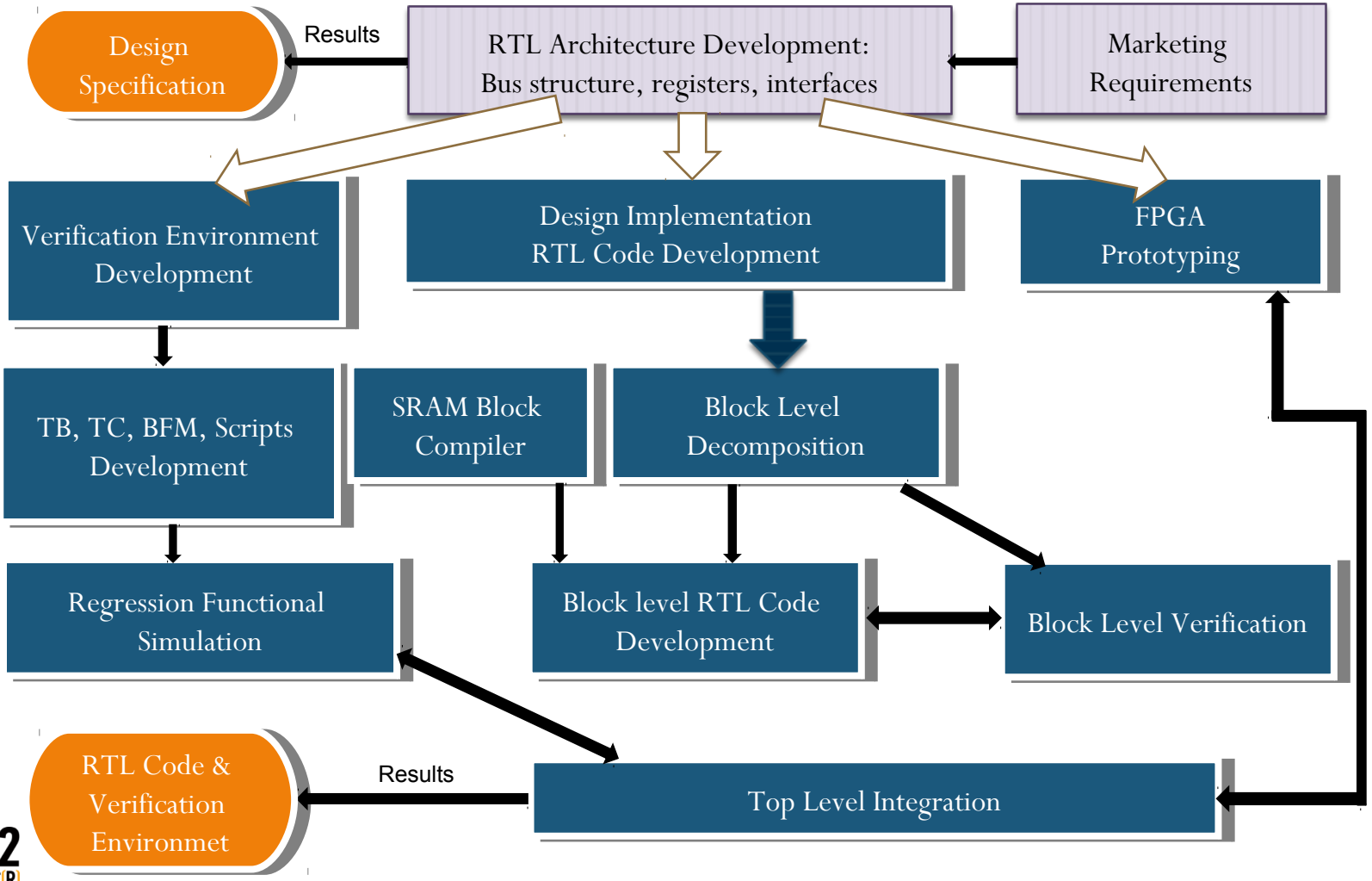




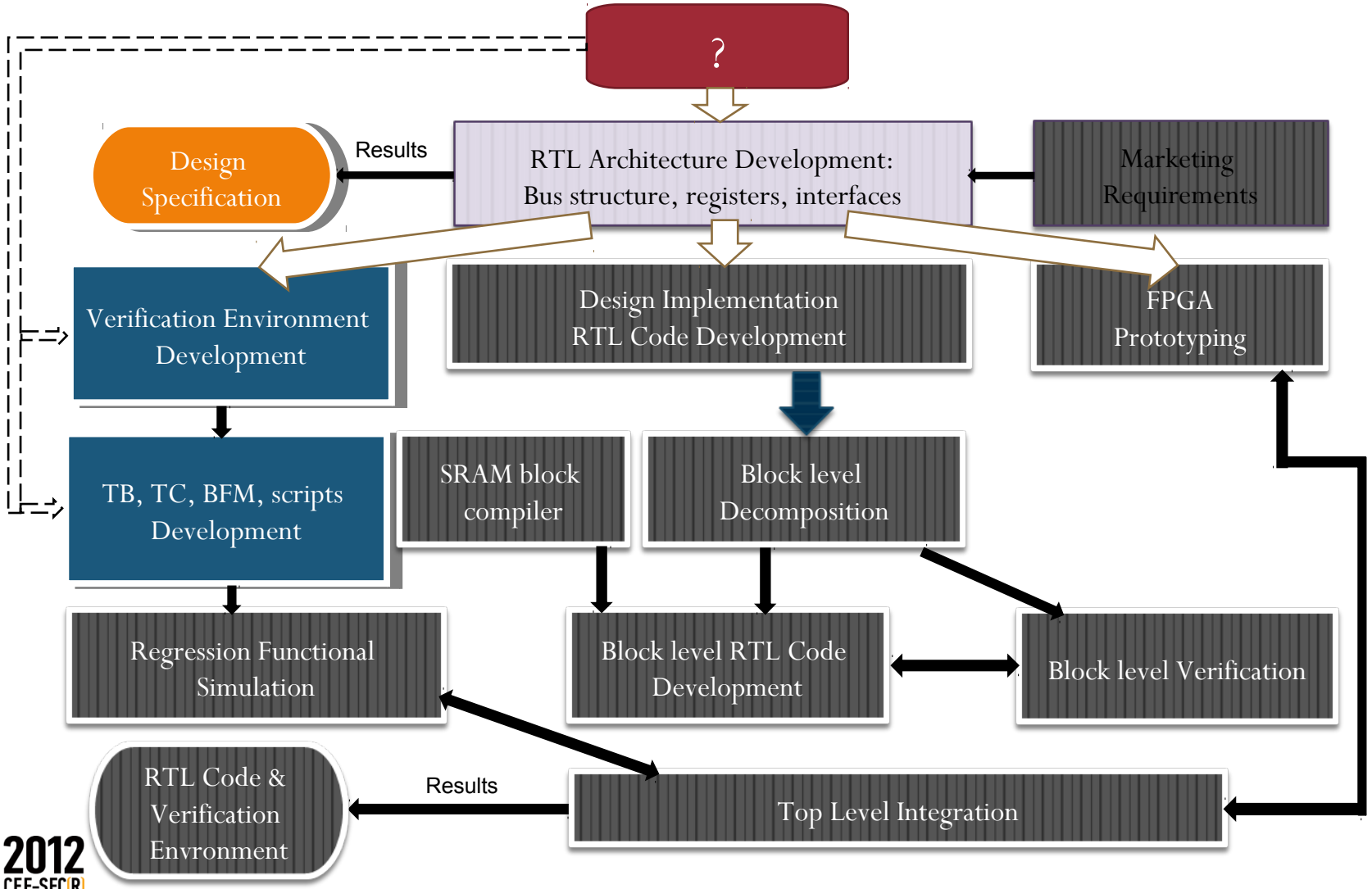
2012 CEE-SEC(R)
Software Engineering Conference in Russia

SPIRIT DSP
Voice & Video Engine Experts

Разработка архитектуры и RTL-дизайн



Кто ставит задачу RTL-дизайнеру



Разработка ПО и RTL

- Разработка ПО и RTL-описания – наиболее трудоемкие и ответственные составляющие проекта по созданию микросхемы
- RTL – это тоже ПО, но написанное на особом низкоуровневом языке – языке описания аппаратуры (VHDL, Verilog, …)
- Что же тогда язык высокого уровня в RTL?

Алгоритмы не разрабатывают на низкоуровневых языках

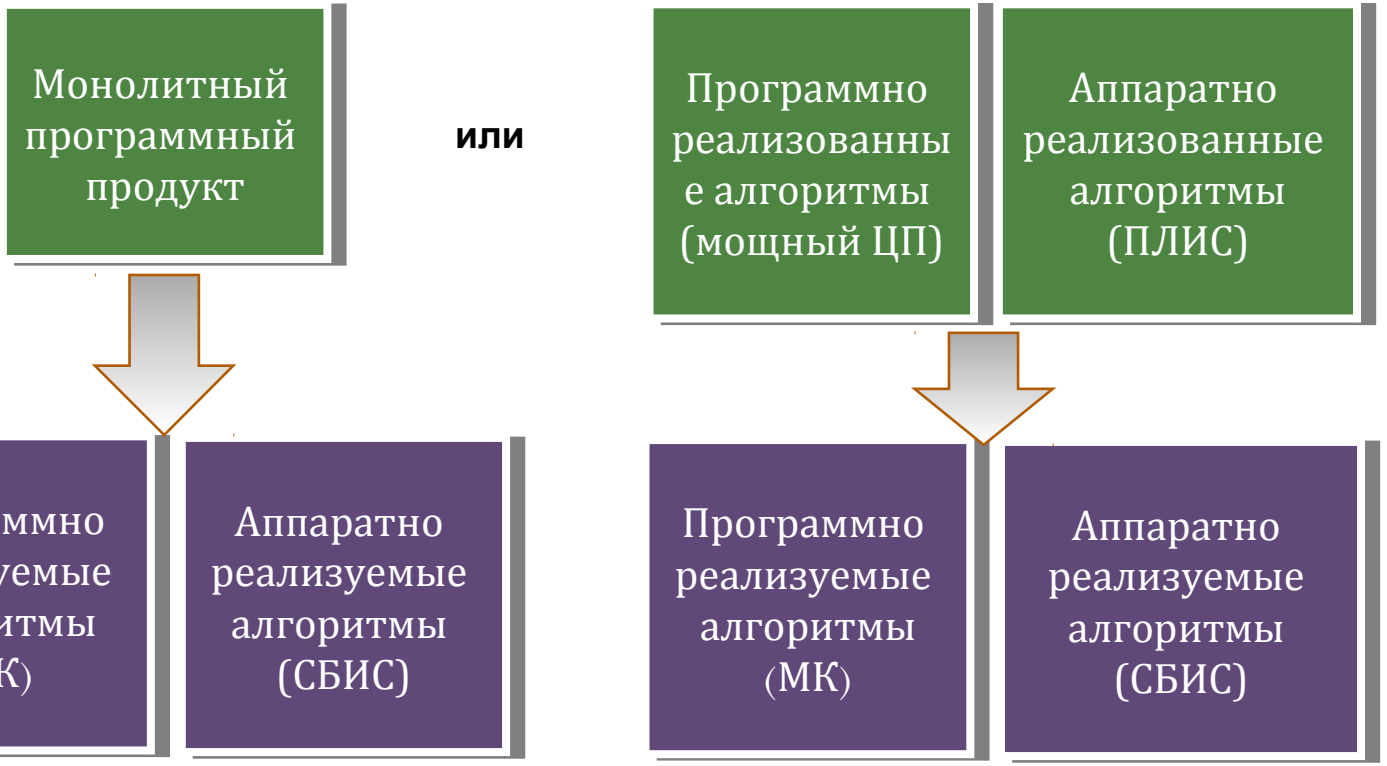
- Классический маршрут разработки ПО с серьезной мат. обработкой: Matlab → C/C++ → asm
- Оптимальный маршрут разработки RTL аналогичен: Matlab → ?++ → RTL (VHDL/Verilog/etc)
  - "?++" не обязателен?
- Тогда “крик души” Mentor Graphics адресован вам!
- Mentor Graphics, “**High-Level Synthesis Blue Book**”, глава “**A broken design flow**”: “Suddenly, what was initially a straightforward process from specification to implementation becomes a nightmarish iterative cycle.“

Почему "крик души" от Mentor Graphics до сих пор не услышан?

- Почему не используют высокоуровневый синтез (high-level synthesis)?
- Кто все-таки озадачивает RTL-дизайнеров?

Сложившиеся маршруты проектирования на основе математически сложного ПО

Существующий проект:



Новый проект:

Техническое задание RTL-дизайнеру (Design Architect) ставит программист!

- Программисты модифицируют ПО, ставят задачу RTL-дизайнеру
- Следствия:
  - Неграмотное, расплывчатое ТЗ
  - Невозможность отладки
- Практикуемые формы ТЗ:
  - Документы, диаграммы, блок-схемы (в формате Word)
  - Matlab/Simulink
  - Исходные алгоритмы на C

Поняв частности, перейдем к сути

Наиболее болезненные операции при трансформации ПО в СБИС:

- Разделение исходного алгоритма на две части: аппаратные акселераторы и программное обеспечение, запускаемое на интегрируемом программном ядре и/или на хост-процессоре
- Моделирование аппаратной и программной части как единого целого
- Сложность, если не сказать невозможность отладки аппаратной части после выпуска опытных образцов

Наиболее популярные маршруты проектирования RTL

- Разработка RTL кода на основе текстового и графического описания алгоритмов
- Прямой синтез RTL из Matlab (Matlab/Simulink HDL coder (Mathworks), Synphony (Synopsys), …)
- Прямой синтез RTL из C (Synfora (Synopsys), CatapultC (Calypto, ранее Mentor Graphics))
- Перевод исходного C-кода в RTL-ориентированную форму с последующим ручным или автоматическим получением RTL

Перевод исходного C-кода в RTL-ориентированную форму

- Переход от "untimed coding style" к "loosely-timed coding style“ (параллельные процессы, сигналы, зоны синхронизации)
- Оптимизация регистровой и “компилируемой” памяти (введение типов int1 .. int63 и прообразов блоков SRAM)
- Определение шин, принципов взаимодействия программной и аппаратной части, внешних интерфейсов
- Введение двух “золотых референсных моделей”: Simple и True
  - Simple – untimed coding style, простые типы. Является золотой референсной моделью для проверки модели True
  - True – RTL-ориентированная модель, “битэкзектная” по отношению к модели Simple. Является золотой референсной моделью для проверки RTL

Варианты RTL-ориентированных форм

- В соответствии со стандартом SystemC
- В соответствии с собственными правилами
- Смешанный вариант

SystemC - достоинства

1. IEEE Standard
2. Реализует большинство возможностей языков VHDL/Verilog
3. Совместим с современными компиляторами C++
4. Поддержан рядом САПР, подобно VHDL/Verilog может быть напрямую синтезирован в netlist.
5. Прост в использовании, нагляден
6. Удобен в совместном моделировании RTL-ориентированного кода и ПО
7. Поддержан библиотекой Universal Verification Methodology (UVM)

SystemC – недостатки (с позиции синтеза netlist)

1. Не поддержан основными САПР либо поддержан усеченно
2. Не знаком большинству RTL-дизайнеров
3. Не полностью реализует базовые возможности языков описания аппаратуры
4. Не поддерживает низкоуровневые ограничения – constraints (ряд САПР предлагает решение вне рамок SystemC)
5. Как следствие – враждебное отношение подавляющего большинства RTL-дизайнеров

SystemC – недостатки (с позиции “золотой модели” и отладки на функциональном уровне)

1. Значение переменных-сигналов практически невозможно смотреть стандартным отладчиком
2. Моделирование переменных-сигналов принципиально дольше моделирования неблокируемых переменных (в 10-200 раз медленнее операций с типами данных “Algorithmic C” из пакета CatapultC)
3. Неудобство моделирования на этапе проработки архитектуры (до появления детальной спецификации RTL):
    1. Неудобство моделирования интерфейсов, спецификация которых еще не уточнена
    2. Отсутствие инициализации модулей, необходимость вводить сигнал Reset и т.д.

SystemC – дополнительные негативные последствия синтезируемости

Использование SystemC и в качестве золотой модели, и как языка для высокоуровневого синтеза в конечном счете “убивает” золотую модель:

1. Исходный код становится нечитаемым и непригодным для отладки алгоритмов
2. Катастрофически замедляется моделирование
3. Возможны проблемы совмещения RTL (SystemC) и остального отлаживаемого ПО (программного кода вне функции sc_start())

RTL-ориентированный код в соответствии с собственными правилами

1. Вместо systemc.h используем mc_int.h, ac_int.h (типы данных “Algorithmic C” из пакета CatapultC)
2. Не используем блокирующий интерфейс сигналов. Вместо этого прорабатываем последовательность вызовов в соответствии с направлением потоков данных
3. Модули и методы (SC_MODULE и SC_METHOD) как в SystemC. Функции, регистрируемые в SC_METHOD, вызываются из интегрирующей функции, определяющей межмодульные взаимодействия в микросхеме.
4. Событийность (чувствительность к изменениям на входных портах) реализуется в минимальном объеме
5. Положительные и отрицательные фронты имитируются на минимально достаточном уровне

Достоинства собственных правил

1. Легкость отладки (все переменные просматриваемы отладчиком, код лучше структурирован)
2. Скорость моделирования повышается в десятки раз
3. Легкость совмещения RTL-ориентированного и обычного кода (вплоть до untimed)
4. Легкость опционального включения/выключения модулей (реконфигурации проекта)
5. Бóльшая понятность RTL-дизайнерам

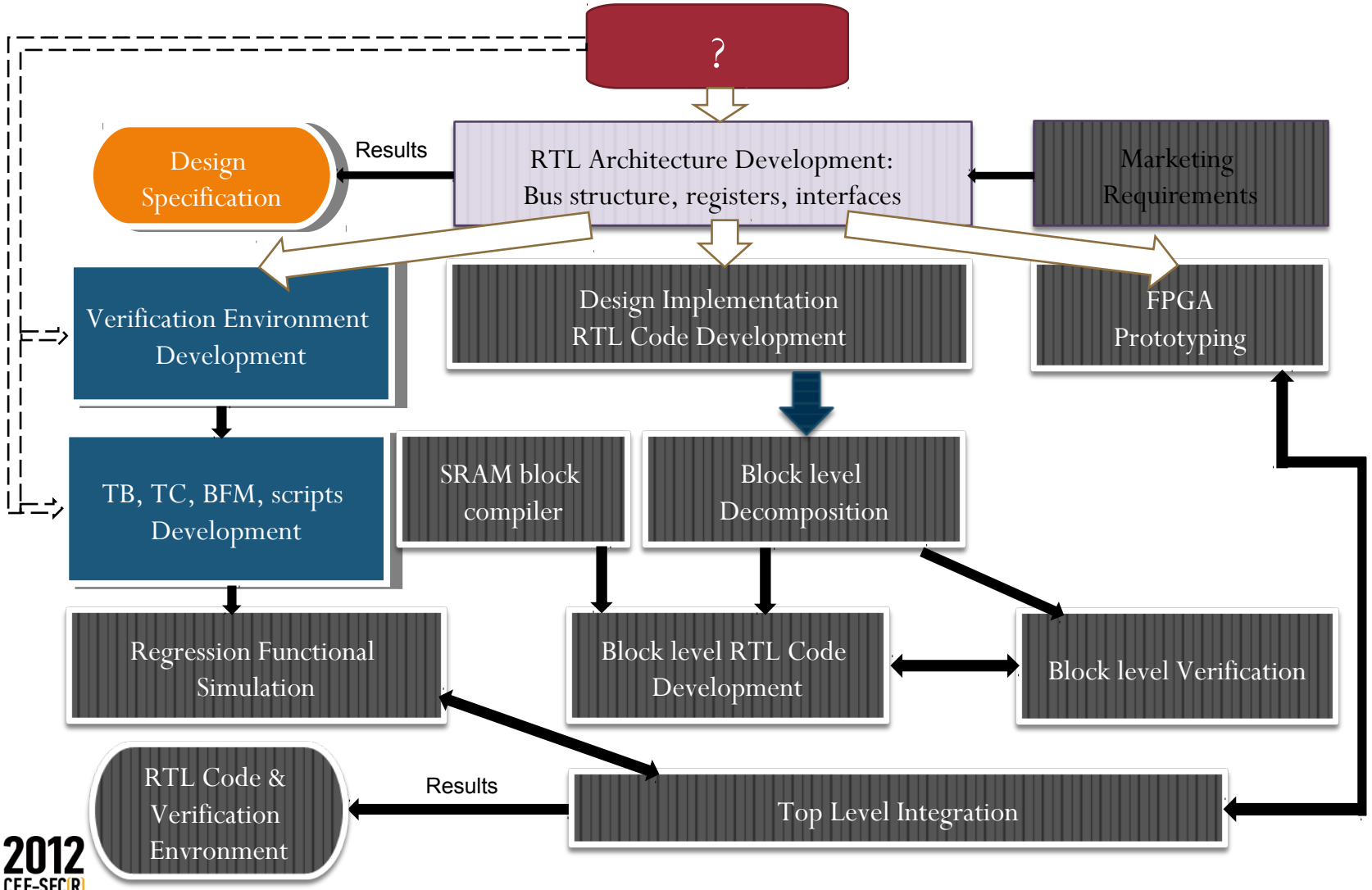
Недостатки собственных правил

1. Несоответствие стандарту
2. Несинтезируемость
3. Отсутствие циклоаккуратности
4. Меньшая пригодность при верификации (может быть решена через SystemVerilog DPI)

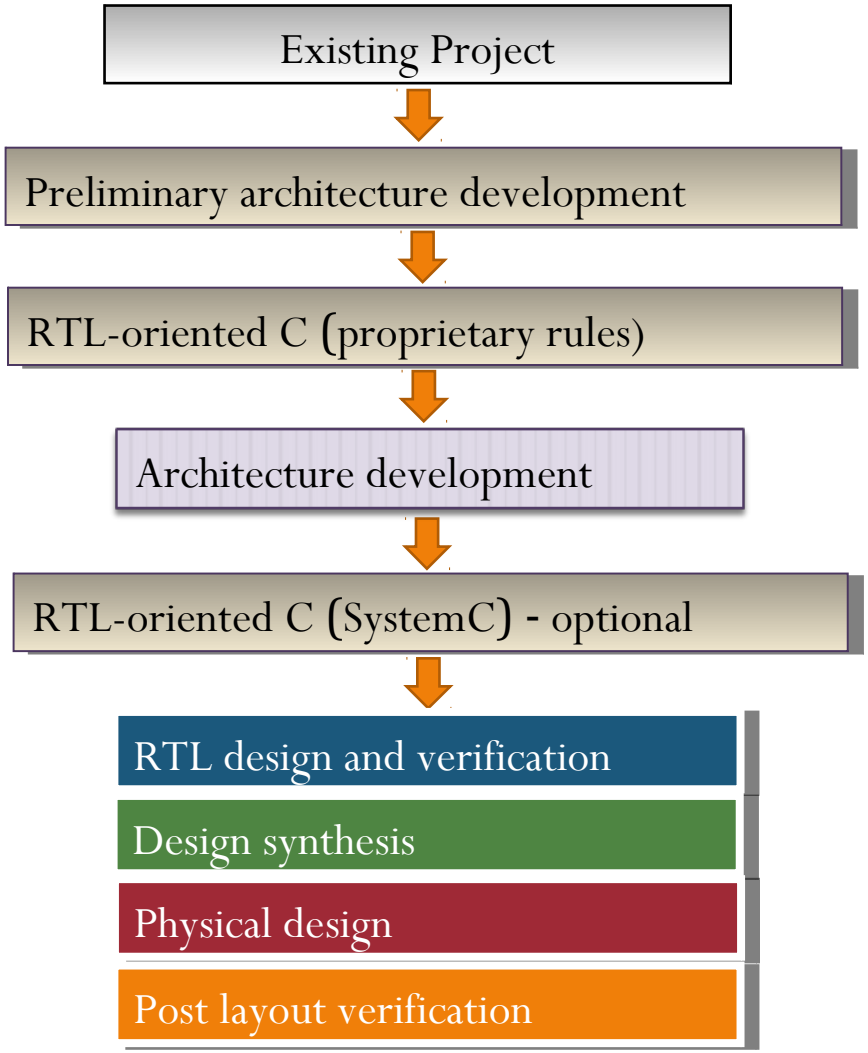
Смешанный вариант

- Смешанный вариант предполагает разбиение проекта на две части, одна из которых реализуется в соответствии с SystemC, другая в соответствии с собственными правилами
- При внешней привлекательности реализовать смешанный вариант крайне сложно из-за несовместимости сигналов SystemC и собственных правил, на основе которых работают эквивалентные неблокируемые переменные

Усовершенствованный маршрут проектирования RTL – что в начале?



Предлагаемые изменения в начале маршрута проектирования



Преимущества предлагаемого маршрута

1. Предлагаемый метод объединяет достоинства SystemC и подхода, основанного на собственных правилах
2. Ускоряет разработку проекта на SystemC
3. Переход от собственных правил к SystemC может быть совмещен с ревизией архитектуры

Метрики проекта “GPS/ГЛОНАСС приемник”

- Время разработки RTL-ориентированного кода одного аппаратного акселератора:
  - В рамках модели Simple - 2 дня
  - В рамках модели True – 7 дней
  - Перевод собственных правил в SystemC – 3 дня
- Время разработки Verilog-описания одного аппаратного акселератора на основе RTL-ориентированного C кода – 10 дней
- Время моделирования C кода на основе собственных правил: 1 секунда моделируемого времени за 2000 секунд на Core2 Duo 2.1 GHz

Подходите к проектированию микросхем творчески!

Спасибо

за внимание!

SPIRIT DSP

Voice & Video Engine Experts

www.spiritdsp.com

www.spiritnavigation.com