Оглавление

Предисловие 3
1. Общая характеристика ПЛИС корпорации Microsemi 6
1.1. Обзор и преимущества Flash-технологии 6
1.2. Малопотребляющие ПЛИС семейств ProASIC3 и IGLOO 8
1.3. Система-на-кристалле SmartFusion 13
1.4. Система-на-кристалле SmartFusion2 и ПЛИС IGLOO2 16
2. Состав и установка пакета САПР Libero SoC 21
2.1. Комплектация пакета САПР Libero SoC 21
2.2. Получение инсталлятора и установка пакета Libero SoC 22
2.3. Получение файла лицензии 30
3. Проектирование для ПЛИС, изготовленных по Flash-технологии. Среда Libero SoC 33
3.1. Общие характеристики и интерфейс среды Libero SoC 33
3.2. Этапы проектирования в среде Libero SoC 36
3.3. Способы описания устройств в Libero SoC 38
3.3.1. Схемный редактор SmartDesign 38
3.3.2. HDL-редактор 43
3.3.3. Каталог IP-ядер 45
3.4. Разработка устройств на ПЛИС SmartFusion2 и IGLOO2 49
3.4.1. Подсистемы MSS и HPMS 49
3.4.2. Инструмент SystemBuilder 50
3.4.3. Инструмент Configure MSS 55
3.5. Синтез проектов в среде Libero SoC (Synopsys Sinplify) 58
3.6. Моделирование устройств в среде Libero SoC (Modelsim) 62
3.7. Размещение и трассировка проекта 70
3.8. Статический временной анализ и временное моделирование 74
3.9. Программирование ПЛИС с помощью программатора FlashPro 79
3.10. Внутрисхемная отладка проектов для ПЛИС в Libero SoC 82
3.10.1. Инструмент SmartDebug 82
3.10.2. Инструмент Identify 87
4. Среда SoftConsole 95
4.1. Настройки взаимодействия, место в Design Flow 95
4.2. Интерфейс пользователя 98

4.3. Процесс отладки 102
5. Примеры проектов в среде Libero SoC 106
5.1. PCIe Control Plane Lab 106
5.2. IGLOO2 Flash*Freeze Lab 134
5.3. SmartFusion2 ARM Cortex-M3 Lab 139
Список литературы 158

Предисловие

Корпорация Microsemi была основана в 1960 году и на сегодняшний является одним из крупнейших производителей аналоговых, цифро-аналоговых, цифровых и дискретных компонентов высокой надежности, в том числе радиационно-стойких.

3 ноября 2010 года в состав корпорации Microsemi вошла корпорация Actel, став её структурным подразделением Microsemi SoC Products Group (подразделение «систем-на-кристалле»). Продукцией Actel/Microsemi SoC PG являются программируемые логические интегральные схемы (ПЛИС, FPGA) со встроенным программируемым постоянным запоминающим устройством (ППЗУ, Flash) или однократно-программируемым постоянным запоминающим устройством (ОППЗУ, Antifuse), а также ПЛИС с ППЗУ и ОППЗУ с повышенной радиационной стойкостью.

Для тех, кто знаком с продукцией Actel/Microsemi, не секрет, что продукция корпорации примечательна в первую очередь за счёт следующих преимуществ:

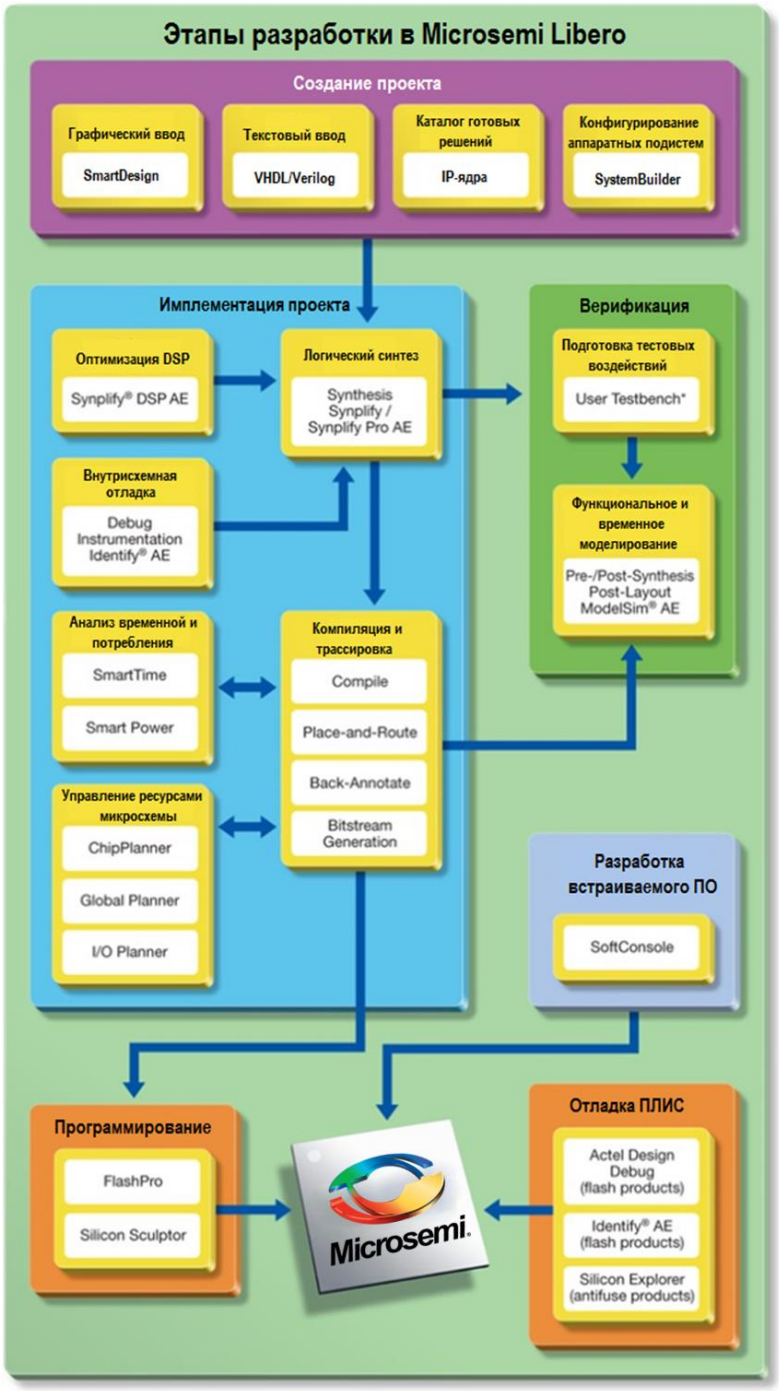
- Малое энергопотребление;
- Высокая надёжность;
- Безопасность.

По технологии производства ПЛИС корпорации Microsemi можно разделить на две группы: ПЛИС, изготовленные по Flash-технологии и ПЛИС изготовленные по Antifuse-технологии. Преимущество обеих технологий над третьей технологией мира FGPA — SRAM — в меньшем потреблении, большей степени интеграции микросхем, большей степени надёжности и энергонезависимости конфигурации (нет необходимости во внешней или внутренней конфигурационной ПЗУ) и, в том числе, сложности её считывания из запрограммированного устройства. Добавьте к этому высокую производительность, готовность по включению питания и интеграцию с процессором и периферийными устройствами в пределах одного кристалла, и получите идеальное решение для устройств различных областей применения.

Система автоматизированного проектирования (далее САПР) Libero SoC (далее Libero) предназначена для создания проектов, логического синтеза, моделирования, трассировки и генерации файла прошивки под все ПЛИС, выполненные по Flash-технологии. Libero обеспечивает эффективное проектирование устройств на ПЛИС с применением подхода построения систем на уровне блоков и модулей с активным применением макро-элементов и готовых функциональных блоков (intellectual property, IP-ядра). Наглядная схема разработки и дружественный интерфейс пакета позволяют освоить работу с ним в максимально короткие сроки и сократить время от старта проекта до выхода готового изделия. Гибкая настройка среды и подключение сторонних инструментов позволят персонализировать процесс разработки и использовать привычные для разработчика средства.

Libero включает в себя продвинутые средства для управления системой ввода/вывода, временными характеристиками и энергопотреблением разрабатываемого устройства, в том числе графические среды задания физических и временных ограничений, временного анализа и отчёт по потребляемой мощности. Так как разработчик может до загрузки конфигурации в ПЛИС узнать приблизительное энергопотребление, значительно сокращается время разработки систем, чувствительных к этому параметру (например, автономные устройства, работающие от батареи).

Также средства Libero предлагают автоматическую оптимизацию по соотношению производительность/потребление, что позволяет без лишних усилий оптимизировать проект. Схематически маршрут проектирования (Design Flow) показан на рисунке.



Существует два варианта лицензии для Libero — бесплатная Libero Gold, платная Libero Platinum. Ограничение на данный момент состоит лишь в том, что в Libero Platinum недоступны наиболее ёмкие и продвинутые кристаллы последних семейств. Большинство проектов может быть полностью разработано в варианте Gold.

В данном руководстве будет описана установка и работа в среде Libero, для закрепления материала будет предложено выполнить ряд заданий и лабораторных работ. Начинается описание с общих характеристик и актуальной линейки Flash-ПЛИС, выпускаемых корпорацией

Microsemi. Далее, в главе 2, будет проведён обзор комплектации САПР, процесс получения программы-установщика и установка пакета. Глава 3 посвящена детальному описанию разработки устройств в Libero с применением встроенных и внешних инструментов, особенностям разработки под отдельные семейства ПЛИС и комплексной отладке разрабатываемого устройства. В главе 4 будут рассмотрены конкретные примеры проектов в среде Libero SoC, а также будут предложено выполнить самостоятельную работу для закрепления материала.

1. Общая характеристика ПЛИС корпорации Microsemi

1.1. Обзор и преимущества Flash-технологии

В настоящее время производители ПЛИС используют три основных технологических базы для производства своих микросхем, это:

- SRAM (static random access memory, статическое оперативное запоминающее устройство);
- Flash, EEPROM (electrically erasable programmable read-only memory, электрически стираемое перепрограммируемое постоянное запоминающее устройство);
- Antifuse (технология восстановления перемычек).

Корпорация Microsemi выпускает ПЛИС, выполненные по двум последним технологиям — Flash и Antifuse. Основные преимущества этих технологий определяют адресность продукции корпорации и её основные особенности. Во-первых, это энергопотребление. ПЛИС, выполненные по SRAM технологии дают достаточно сильный импульс тока при включении питания, также им необходимо время и ток потребления для загрузки конфигурации из внешней конфигурационной ПЗУ. Flash и Antifuse ПЛИС лишены этих недостатков и готовы к работе сразу после включения питания (рис. 1.1).

Рис.1.1. Потребляемый ток при включении питания SRAM и Flash ПЛИС

После того, как ПЛИС включена и функционирует, её потребляемая мощность состоит из статической и динамической составляющих. Статическая составляющая определяется токами утечки транзисторов, когда устройство находится в неактивном режиме. Динамическая составляющая определяется потребляемой мощностью в активном режиме. SRAM ячейка имеет существенные токи утечки и, вследствие, гораздо большую статическую потребляемую мощность, а вкупе с режимами запуска и конфигурирования, SRAM ПЛИС потребляет намного больше энергии, чем Flash ПЛИС. Всё существеннее это ощущается при переходе на меньший технологический процесс. При этом ячейка Flash имеет гораздо меньшие размеры, что способствует большей степени интеграции микросхем (рис.1.2).

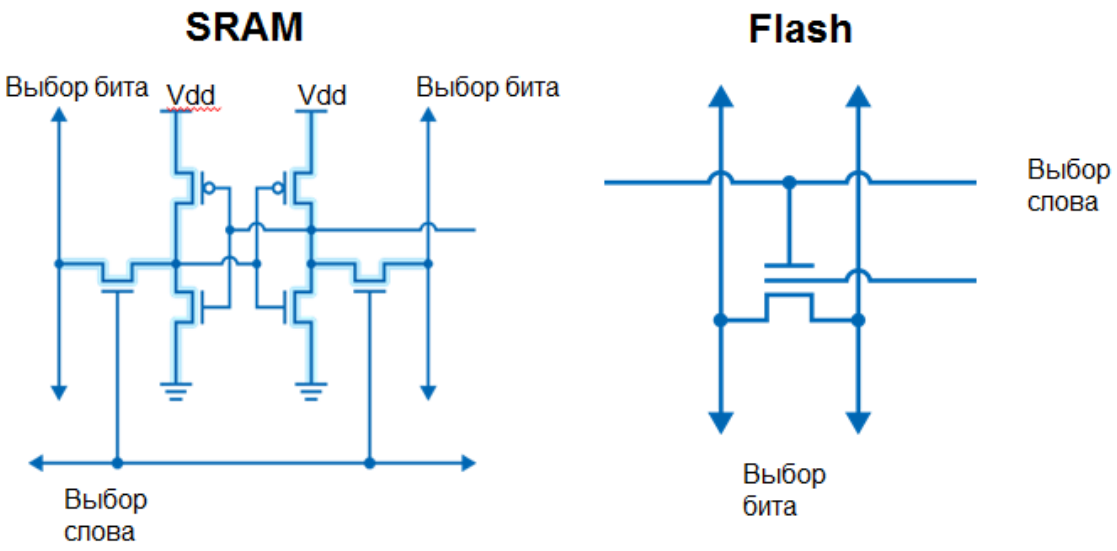


Рис. 1.2. Ячейки SRAM и Flash

Уменьшение размеров ПЛИС и отсутствие внешней конфигурационной ПЗУ позволяют значительно сэкономить место на печатной плате и улучшить массогабаритные показатели готового устройства.

В таблице 1.1 приведено сравнение трёх технологий по основным параметрам, важным при реализации проектов.

Таблица 1.1. Сравнение основных технологий производства ПЛИС

	SRAM	**Antifuse**	**EEPROM/Flash**
Перепрограммируемость	Да (в системе)	Нет	Да (в системе или оффлайн)
Скорость перепрограммирования (включая стирание)	Быстро	—	Медленнее
Энергозависимость конфигурации (необходимость программирования при включении питания)	Да	Нет	Нет (но это возможно)
Требуется внешний конфигурирующий файл	Да	Нет	Нет
Подходит для прототипирования	Да	Нет	Да
Готовность сразу после включения	Нет	Да	Да
Защита конфигурации и IP-ядер	Только с использованием специальных подходов	Очень хорошая	Очень хорошая
Размер конфигурационной ячейки	Большая (шесть транзисторов)	Очень маленькая	Средняя/маленькая (два транзистора)
Энергопотребление	Среднее	Низкое	Ниже, чем SRAM
Радиационная стойкость	Нет	Да	Выше, чем у SRAM

Несложно заметить, что кроме энергопотребления, Flash и Antifuse технологии имеют преимущества по ряду других показателей. Например, защита интеллектуальной собственности и пользовательских «ноу-хау», которые достаточно легко считать из внешней конфигурационной памяти SRAM ПЛИС и технологически обусловленная устойчивость к радиационным эффектам Antifuse и Flash ПЛИС. Все перечисленные достоинства делают ПЛИС Microsemi SoC незаменимыми в ряде приложений с высокими требованиями к потребляемой мощности, надёжности и безопасности.

Развитие в данных направлениях и удовлетворение потребностей заказчиков в соответствующий областях ведётся ещё с времени до вхождения корпорации Actel в состав Microsemi. Накопленный опыт и наследие находят применение в новых семействах ПЛИС и систем-на-кристалле Microsemi SoC. Развитие семейств ПЛИС корпорации вплоть до новейших на момент написания данного руководства семейств представлено на рис. 1.3.

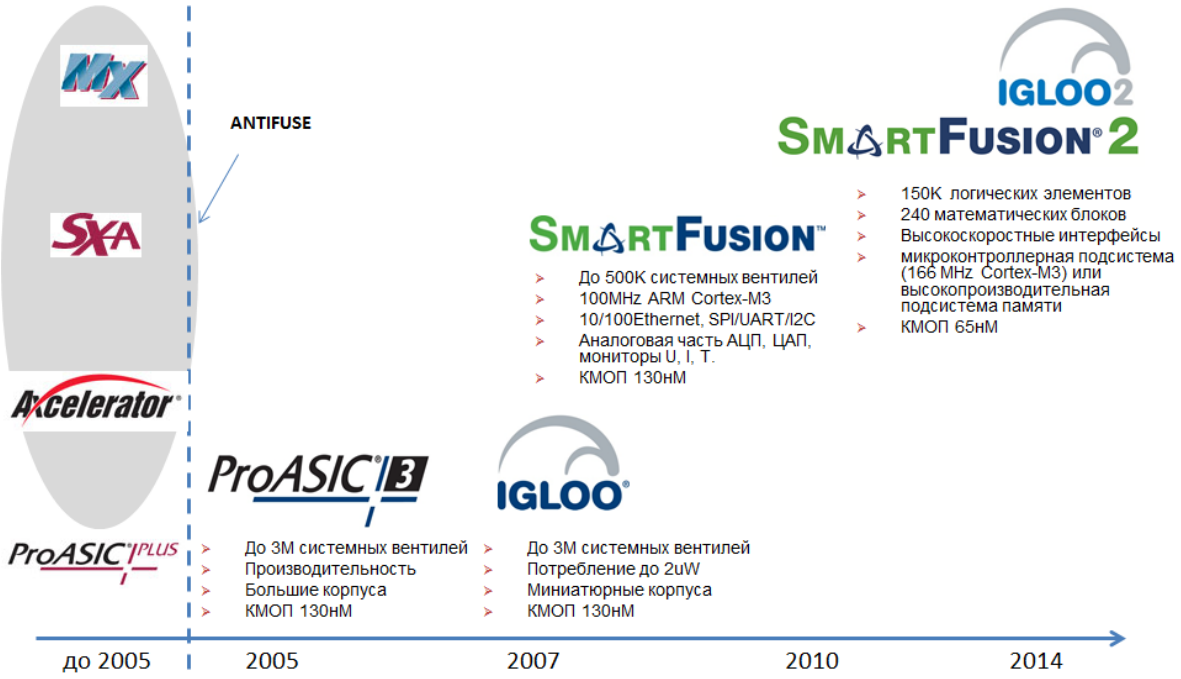


Рис. 1.3. Развитие ПЛИС корпорации Microsemi

В дальнейших разделах подробнее будет рассказано об актуальной линейке Flash ПЛИС, для работы с которыми предназначена САПР Libero SoC. Наиболее полную и подробную информацию о всех семействах ПЛИС Microsemi можно получить на сайте корпорации по ссылке: http://www.microsemi.com/products/fpga-soc/fpga-and-soc.

1.2. Малопотребляющие ПЛИС семейств ProASIC3 и IGLOO

Актуальная линейка Flash ПЛИС корпорации Microsemi начинается с семейств ProASIC3 и IGLOO. Данные семейства давно серийно выпускаются и уже успели зарекомендовать себя во

многих проектах. В рамках этих двух семейств предлагаются как высокопроизводительные полнофункциональные FPGA, так и миниатюрные малопотребляющие микросхемы (рис. 1.4).

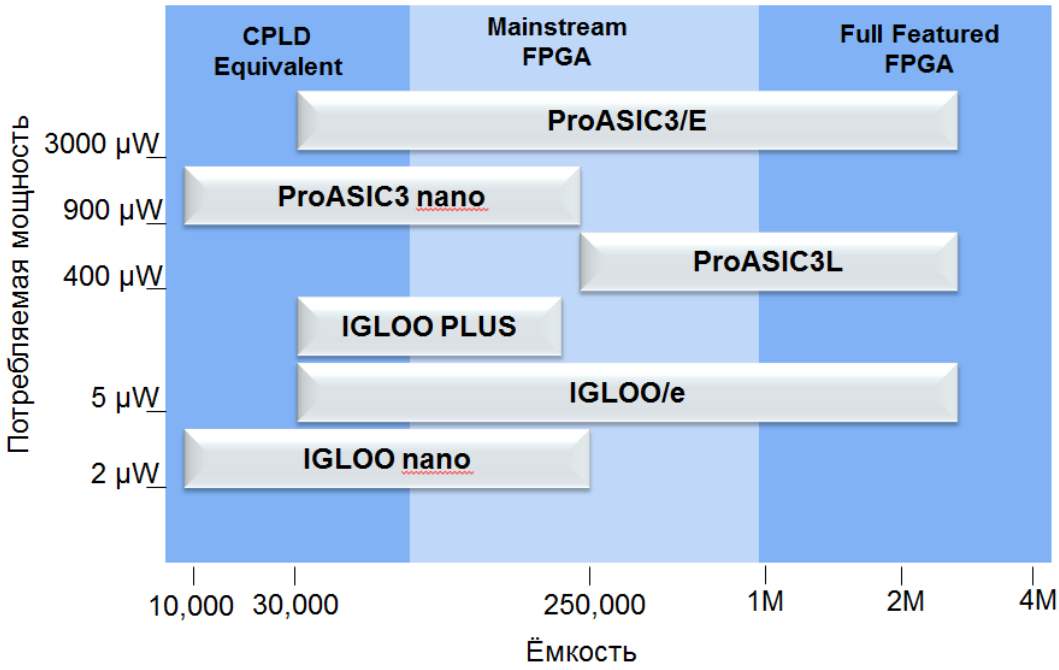


Рис. 1.4. Сравнительная диаграмма семейств ProASIC3 и IGLOO

Таким образом, при выборе микросхемы соблюдается баланс между производительностью и потребляемой мощностью. Высокопроизводительные малопотребляющие ProASIC3 или сверхмалопотребляющие производительные IGLOO.

ProASIC3 - это семейство ПЛИС, которое обеспечивает прорыв в энергопотреблении, цене, производительности, плотности, а также в характеристиках для современных наиболее требовательных приложений высокого уровня. Устройства ProASIC3 поддерживают процессор ARM Cortex-M1, предлагая преимущества программируемости и недорогого периода от начала разработки изделия до выхода его на рынок. Устройства ProASIC3 основаны на энергонезависимой Flash-технологии и поддерживают от 10 тыс. до 3 млн. вентилей и до 620 высокопроизводительных вводов/выводов.

Семейство высокопроизводительных малопотребляющих ПЛИС ProASIC3 включает три подсемейства:

- ProASIC3/E;
- ProASIC3L;
- ProASIC3nano.

Базовое подсемейство ProASIC3/E — высокая производительность и малая потребляемая мощность, ProASIC3L — подсемейство с пониженным энергопотреблением и ProASIC3nano — подсемейство с минимальными размерами корпусов (минимальный — 6x6 мм).

Архитектура семейства ProASIC3 представлена на рис. 1.5

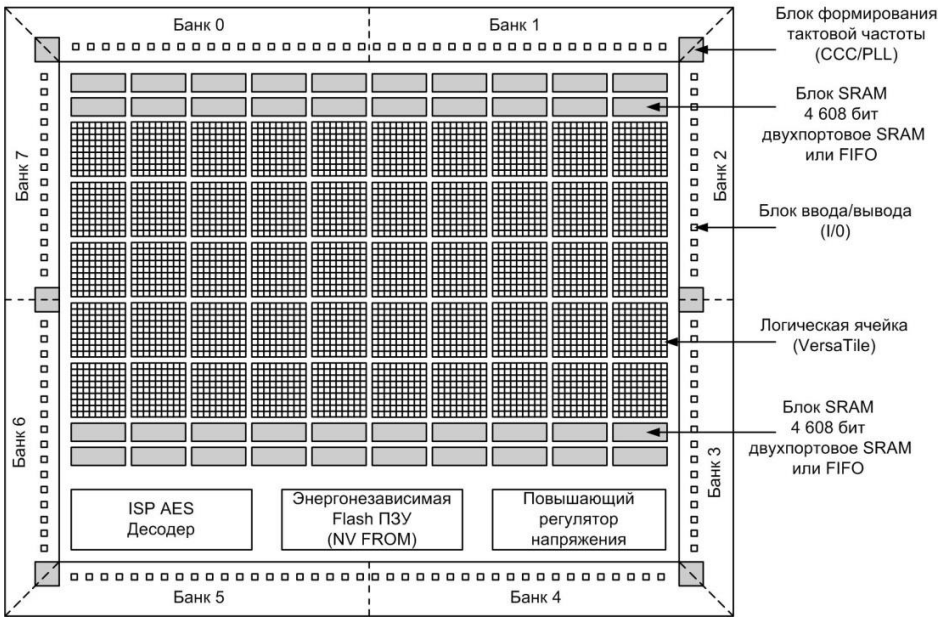


Рис. 1.5. Архитектура ProASIC3

Основным логическим элементов матрицы является универсальная логическая ячейка VersaTile. Каждая ячейка может быть сконфигурирована как логическая функция трёх переменных и синхронный D-триггер (рис. 1.6).

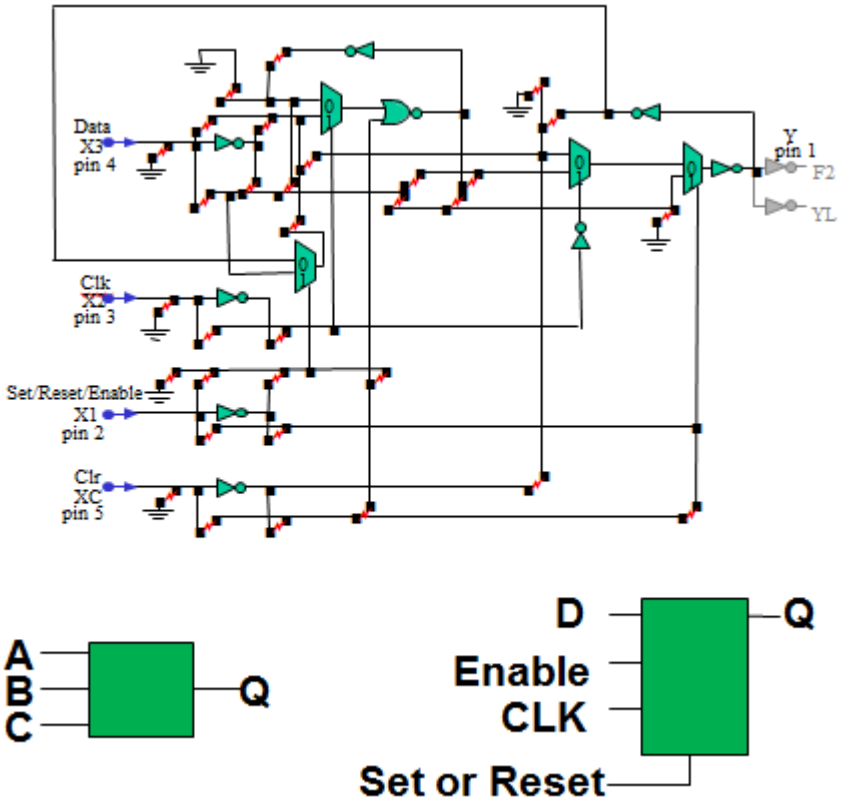


Рис. 1.6. Универсальная логическая ячейка VersaTile

Основные характеристики подсемейств сведены в таблицу 1.2.

Таблица 1.2. Характеристики подсемейств ProASIC3

	ProASIC3/E	**ProASIC3nano**	**ProASIC3L**
Поддержка ARM	M1 ProASIC3/E		M1 ProASIC3L
Обзор	Малопотребляющее недорогое решение FPGA	Самое дешевое решение с расширенными возможностями ввода/вывода	Баланс между малым энергопотреблением, производительностью и низкой стоимостью
Системные вентили	15,000 - 3,000,000	10,000 - 250,000	250,000 - 3,000,000
Макс. количество пользовательских вводов/выводов	620	71	620
Потребление мощности	3 мВт	3 мкВт	330 мкВт

Микросхемы семейства IGLOO позиционируется как ПЛИС с минимальным энергопотреблением и низкой стоимостью. Семейство также включает в себя три подсемейства:

- IGLOO/E;
- IGLOO+;
- IGLOOnano.

ПЛИС семейства IGLOO полностью изготовлены по FLASH технологии и разработаны в соответствии с современными требованиями по потребляемой мощности и размерам, предъявляемыми к микросхемам, использующимся в портативной электронике. Созданные на базе архитектуры семейства FPGA ProASIC3, с использованием низкопотребляющей FLASH технологии с рабочим напряжением от 1.2В до 1.5В, семейства IGLOO могут успешно использоваться в промышленной аппаратуре, обеспечивая минимальную потребляемую мощность.

Микросхемы семейства IGLOO имеют объем до 3 миллионов системных вентилей, двухпортовое статическое ОЗУ с объемом памяти 504 кбита, до 6 встроенных ФАПЧ с умножителем частоты, и до 616 пользовательских входов/выходов. В проектах может быть использован 32-х битный ARM процессор Cortex-M1 без лицензионных взносов и гонораров с микросхемами M1 IGLOO. Разработанный специально для использования в программируемой логике процессор Cortex-M1 предлагает баланс между производительностью и объемом.

Основные характеристики семейства IGLOO сведены в таблицу 1.3.

Таблица 1.3. Характеристики подсемейств IGLOO

Семейства IGLOO	**IGLOO/e**	**IGLOO nano**	**IGLOO PLUS**
Поддержка ARM	M1 IGLOO/e		
Обзор	Сверхмалопотребляющее программное решение	ПЛИС имеющая самые малые потребление мощности и габариты корпуса	ПЛИС со сверхмалым потреблением и расширенной конфигурацией портов ввода/вывода
Системные вентили	15,000 - 3,000,000	10,000 - 250,000	30,000 - 125,000
Макс. количество пользовательских вводов/выводов	620	71	212
Потребление мощности	5 мкВт	2 мкВт	5 мкВт

Своеобразной «визитной карточкой» семейства IGLOO является технология Flash*Freeze, обеспечивающая доступ к одноименному энергосберегающему режиму.

Технология Flash*Freeze, используемая в микросхемах IGLOO, позволяет легко входить и выходить из режима сверхнизкого потребления мощности, при котором оно составляет менее 2 мкВт, с сохранением данных как статического ОЗУ, так и регистров. Flash*Freeze технология позволяет легко управлять потреблением через специализированный вывод микросхемы (рис. 1.7) или используя внутреннюю схему управления тактированием. При этом не требуется отключать питание, порты ввода/вывода внешних устройств или тактовую частоту на системном уровне.

Рис. 1.7. Управляющий вывод Flash*Freeze в ПЛИС IGLOO

В режиме Flash*Freeze мощность потребления падает до 2 мкВт, при этом не требуется дополнительных устройств для отключения внешних сигналов с выводов или частоты, также сохраняется внутреннее состояние ПЛИС, содержимое статического ОЗУ и регистров. Включение и выключение режима Flash*Freeze занимает менее 1 мкс.

Кроме этого, IGLOO обладает сверхнизким потреблением (от 25 мВт) в активном режиме (режим холостого хода) обеспечивая функционирование в полном объеме портов ввода/вывода, статического ОЗУ, регистров и логических функций. Это позволяет устройствам IGLOO управлять системой управления питанием через внешние входы (например, сканировать входной сигнал клавиатуры) при минимальном расходе энергии.

1.3. Система-на-кристалле SmartFusion

Так называемые «системы-на-кристалле» ведут давнюю историю и уже успели стать неотъемлемой частью современной электронной аппаратуры разных областей применения. Растущая степень интеграции и оптимизация архитектуры микросхем программируемой логики позволяет увеличивать производительность и вкладывать в них всё больше и больше функциональных возможностей, сначала заменять множество разнесённой по печатной плате логики на одну микросхему CPLD, а теперь заменить связку ПЛИС-микроконтроллер-аналоговая часть на одну микросхему, включающую все узлы сразу. Такой подход в значительной степени увеличивает производительность систем в целом, так как взаимодействие в пределах одного кристалла значительно эффективнее.

Такой подход нашёл широкое применение и лёг в основу ряда устройств многих производителей. Первая «система-на-кристалле» корпорации Microsemi — SmartFusion — в полной мере является иллюстрацией изложенного выше подхода (рис. 1.8).

Комбинированная аналого-цифровая ПЛИС SmartFusion содержит в одной микросхеме программируемую логическую матрицу, аппаратный процессор ARM Cortex-M3 и программируемый аналоговый блок. Обеспечивает полностью настраиваемую конфигурацию, защиту вашего проекта и легкость в использовании. ПЛИС SmartFusion, выполненная полностью по Flash технологии, обеспечивает значительно большую гибкость по сравнению с традиционными решениями на аппаратных микроконтроллерах с фиксированным набором функций и имеет значительно меньшую стоимость по сравнению с решениями с использованием дорогостоящих софт-процессоров[1] в ПЛИС.

[1] Процессор, функциональность которого реализуется в матрице ПЛИС

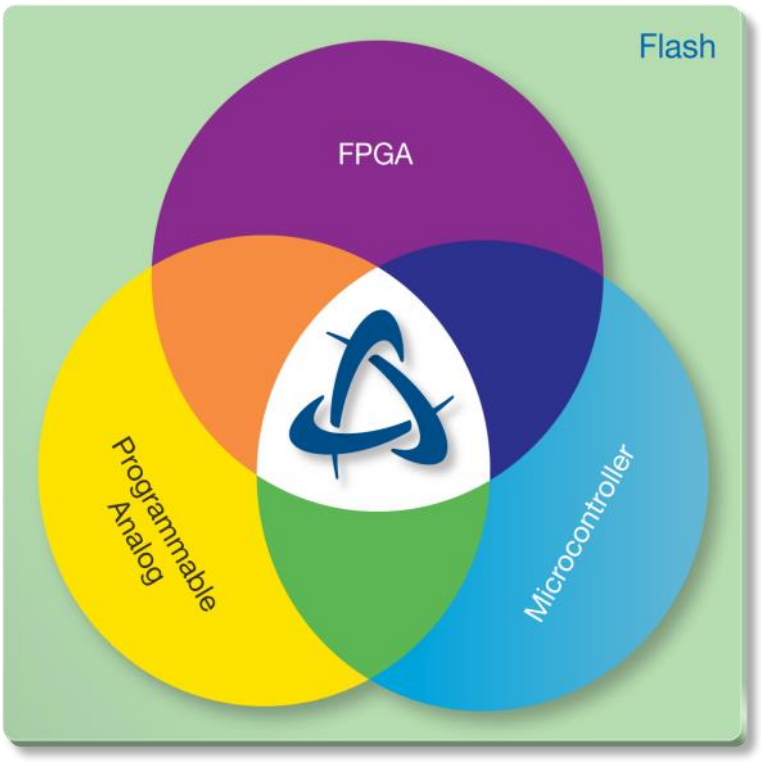


Рис. 1.8. Концепция SmartFusion

На рисунке 1.9 показана структурная схема ПЛИС семейства SmartFusion.

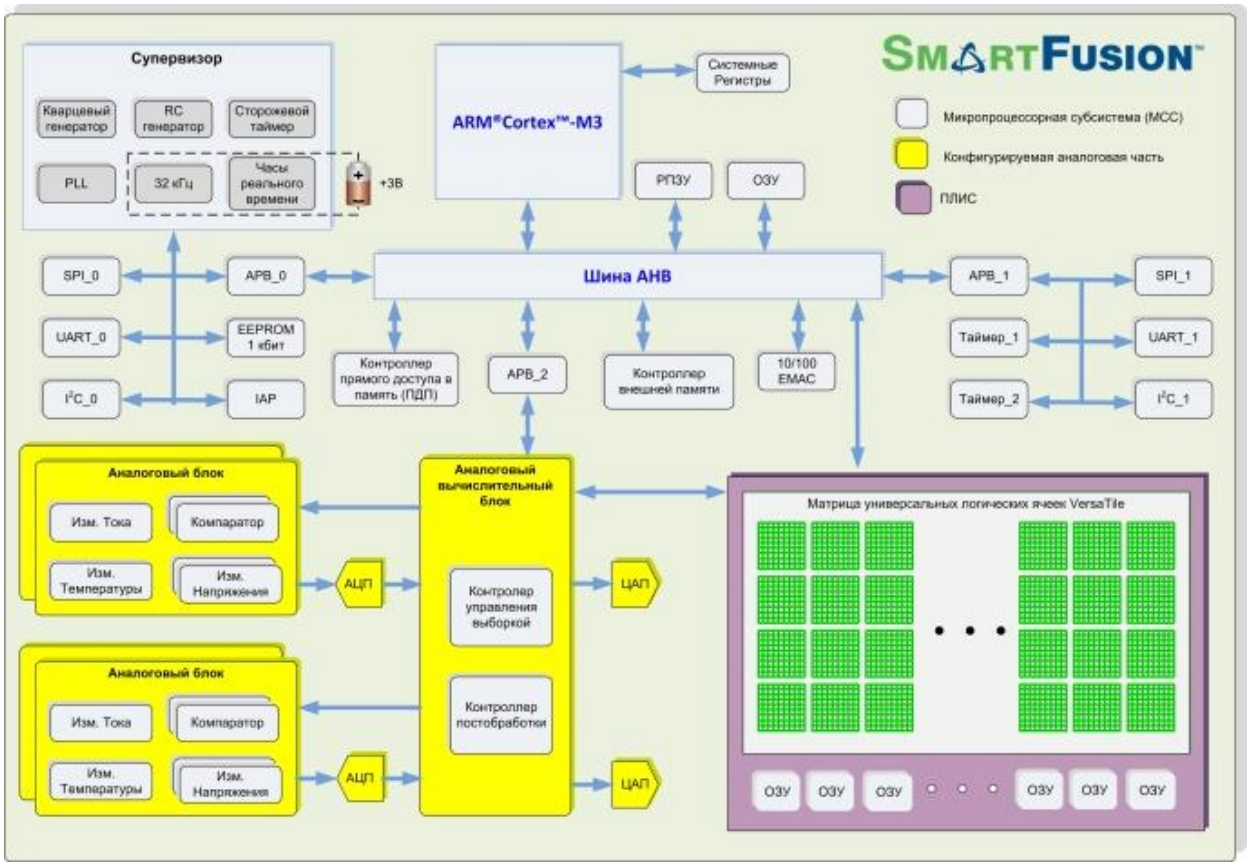


Рис. 1.9 Структура ПЛИС семейства SmartFusion

Отличительные черты данного семейства:

- Аппаратный 100 МГц 32-битный процессор ARM Cortex-M3;
- AHB шина, выполненная в виде многоуровневой матрицы соединений с пропускной способностью до 16 Гбит/с;
- 10/100 Ethernet MAC;
- По два контроллера SPI, I2C и UART;
- Два каскадируемых 32-битных таймера;
- До 512 кбайт Flash ПЗУ и до 64 кбайт СОЗУ;
- Контроллер внешней памяти;
- 8-канальный контроллер ПДП;
- Итегрированные ЦАП и АЦП;
- Интегрированные каналы измерения напряжения, тока и температуры;
- До 10 высокоскоростных 50 нс компараторов;
- Аналоговый вычислительный блок разгружает процессор от предварительной обработки аналоговых сигналов;
- До 47 аналоговых и до 169 цифровых портов ввода-вывода.

Имея такой широкий набор периферийных устройств и средства аналогового ввода-вывода СнК SmartFusion способна заменить сразу несколько отдельных микросхем и значительно уменьшить размер печатной платы, а также улучшить частотные характеристики устройства.

Семейство SmartFusion представлено тремя устройствами (табл. 1.4).

Таблица 1.4. Микросхемы семейства SmartFusion

ПЛИС SmartFusion	**A2F060**	**A2F200**	**A2F500**
Программируемая логическая матрица			
Системных вентилей	60,000	200,000	500,000
Логических ячеек	1,536	4,608	11,520
Блоков ОЗУ объемом 4,608 бит	8	8	24
Микропроцессорная субсистема (МСС)			
FLASH ПЗУ, кбайт	64	256	512
СОЗУ кбайт	16	64	64
10/100 Ethernet MAC	Нет	Да	Да
Контроллер внешней памяти	26-бит адрес, 16-бит данные	26-бит адрес, 16-бит данные	26-бит адрес, 16-бит данные
PLL	1	1	2
32 КГц интегрированный генератор	1	1	1
100 МГц интегрированный RC генератор	1	1	1
Кварцевый генератор (1.5 МГц - 20 МГц)	1	1	1
Аналоговый блок			
АЦп	1	2	3

ЦАП	1	2	3
Аналоговых блоков (SCBs)	1	4	5
Компараторов	2	8	10
Каналов измерения тока	1	4	5
Каналов измерения температуры	1	4	5
Биполярных каналов измерения напряжения	2	8	10

SmartFusion — первый шаг, сделанный Microsemi в сторону создания эффективных систем-на-кристалле. Накопленный опыт был использован при создании следующего поколения — SmartFusion2.

1.4. Система-на-кристалле SmartFusion2 и ПЛИС IGLOO2

Новое четвертое поколение ПЛИС и СнК Microsemi воплотились в микросхемах семейств SmartFusion2 и IGLOO2, системе-на-кристалле и ПЛИС, удовлетворяющих самым высоким требованиям потребляемой мощности, надежности и безопасности. Семейства ведут своё начало от ПЛИС и СнК предыдущего поколения, как видно из их названий, SmartFusion и IGLOO, и в полной мере используют весь накопленный корпорацией опыт.

В микросхемах нового поколения пересмотрена архитектура программируемой матрицы, добавлены математические (DSP) блоки, блоки SERDES (Serializer/Deserializer), при этом производительность выросла в 3 раза, а ёмкость увеличилась в 5 раз.

По сути, SmartFusion2 и IGLOO2 представляют собой одну и ту же микросхему, отличием является отсутствие в IGLOO2 микроконтроллера и его подсистемы (микроконтроллерная подсистема, MSS). Вместо этого в IGLOO2 присутствует высокопроизводительная подсистема памяти (HPMS), которая частично дублирует функции MSS (рис. 1.10).

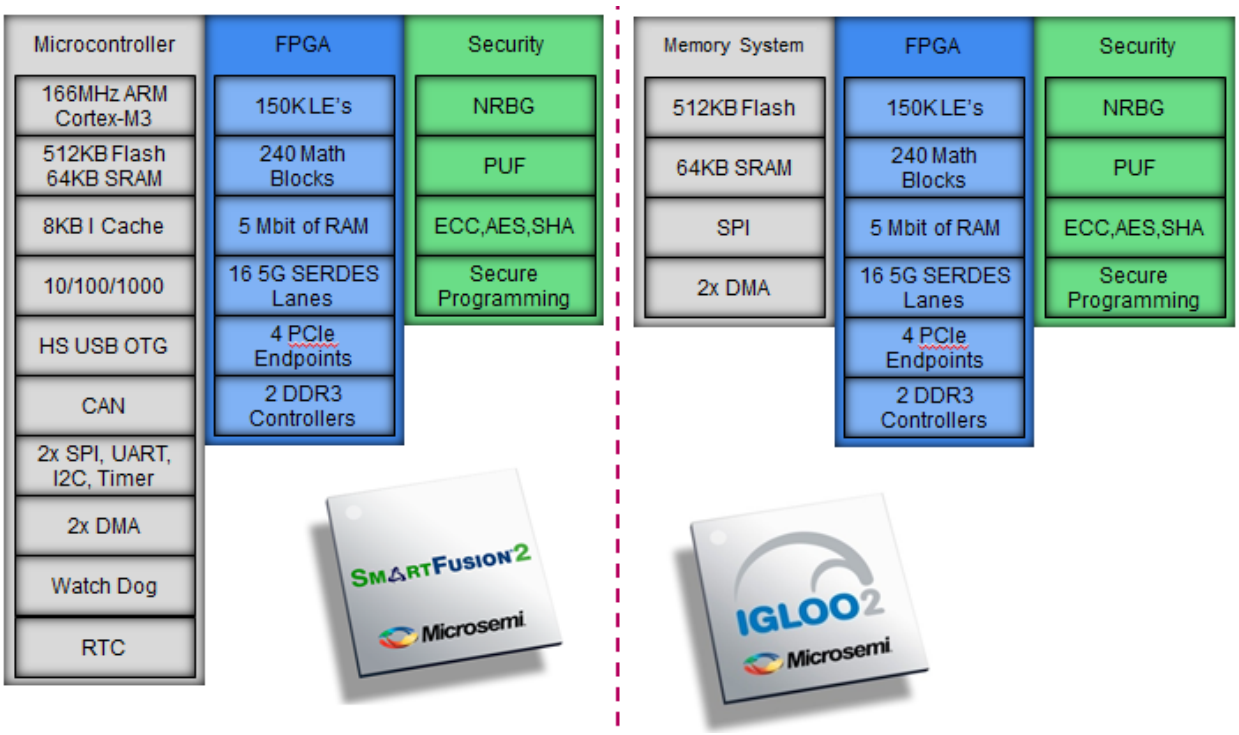


Рис. 1.10. Сравнение возможностей SmartFusion2 и IGLOO2

Как наглядно показывает рисунок 1.10, возможности этих микросхем чрезвычайно схожи и, можно сказать, что IGLOO2 — усечённая версия SmartFusion2. Если учесть, что они фактически совместимы по выводам, то получается одна замечательная деталь. Предположим, что для некоторого проекта необходима ПЛИС и выбор пал на IGLOO2. Разработчики выпустили продукт и задумались над новой версией устройства. И через некоторое время пришли к выводу, что неплохо бы иметь «на борту» микроконтроллер. В этом случае решение очевидно — просто заменить IGLOO2 на соответствующую ей SmartFusion2, при этом не требуется заново разводить печатную плату, при которой, возможно, возникли бы ошибки.

Структура SmartFusion2 представлена рис. 1.11. В состав микросхемы входят:

- микроконтроллерная подсистема с процессором ARM-Cortex M3;
- матрица ПЛИС;
- контроллер DDR;
- блоки SERDES 5 гбит/с;
- интегрированные источники тактового сигнала и ФАПЧ;
- системный контроллер.

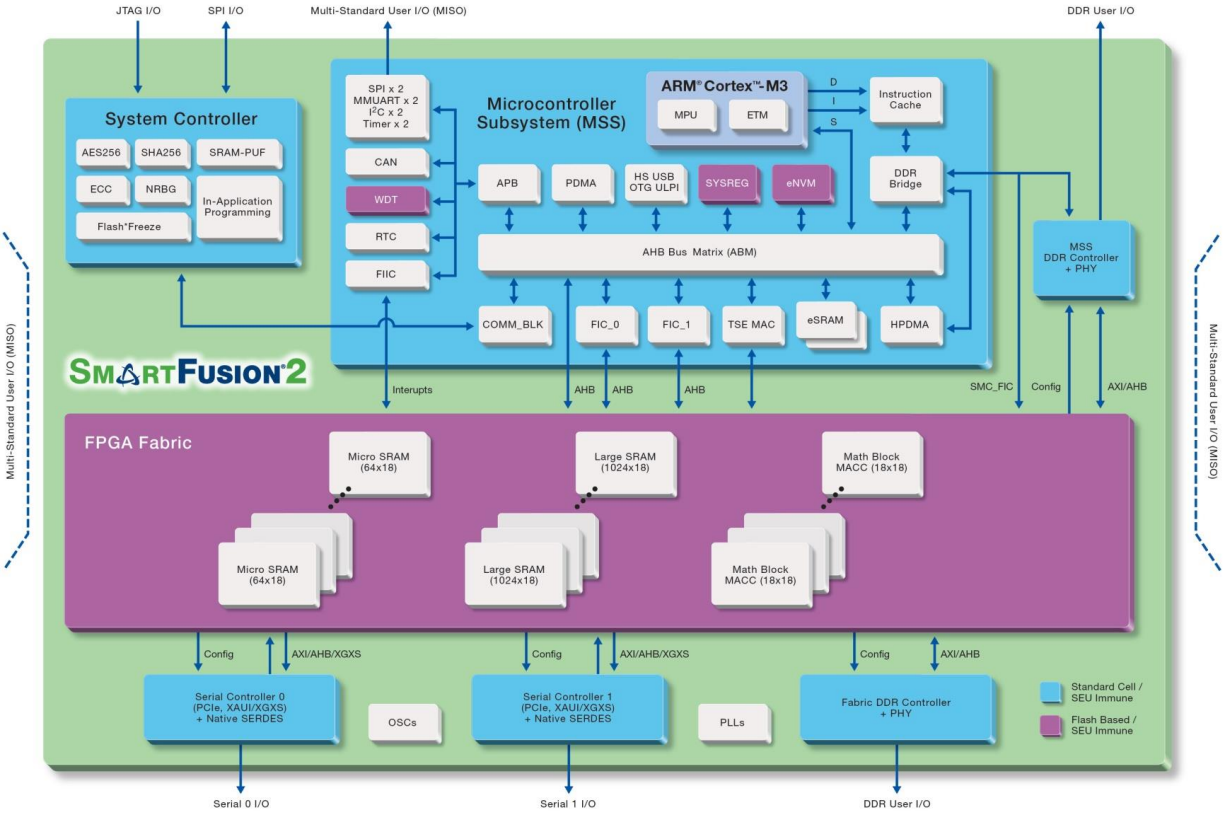


Рис. 1.11. Структура SmartFusion2

Состав микроконтроллерной подсистемы:

- 166 МГц ARM Cortex-M3
  - 1.25 DMIPS/МГц
  - 8 КВ кэш инструкций
  - Модуль защиты памяти (MPU)
  - Обмен в подсистеме по матричной шине AHB, до 10 ведущих и 7 ведомых
- Периферийные устройства
  - Triple Speed Ethernet (TSE) 10/100/1000 Мбит/с MAC
  - USB 2.0 High Speed On-The-Go (OTG) контроллер
  - CAN контроллер, 2.0B совместим, соответствует ISO11898-1
  - По два: SPI, I2C, Multi-Mode UARTs (MMUART)
  - Сторожевой таймер (Watchdog)
  - Один 64-бит (или два 32-бит) таймер общего назначения
  - Таймер реального времени (RTC)

Внушительный список, способный удовлетворить требования многих проектов. Все периферийные устройства подключены к матричной шине AHB и имеют свой адрес в адресном пространстве шины и процессора. Доступ к ним из пользовательской программы осуществляется с помощью драйверов и функций API.

Структура IGLOO2 показана на рисунке 1.12. Все затрагиваемые далее устройства и интерфейсы относятся как к SmartFusion2, так и к IGLOO2.

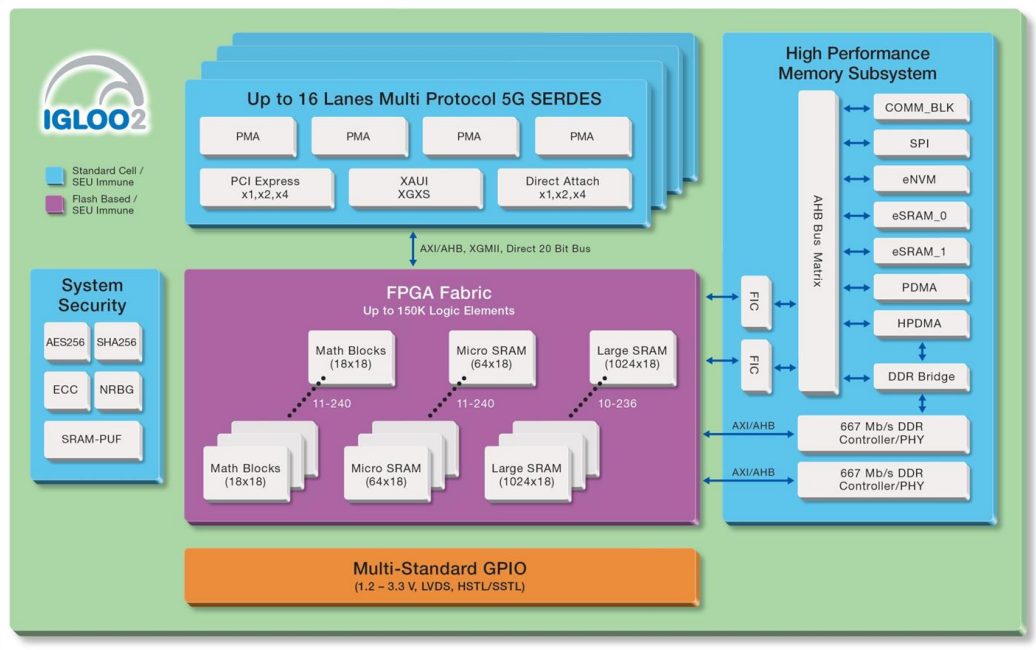


Рис. 1.12. Структура IGLOO2

Матрица ПЛИС SmartFusion2 и IGLOO2 берёт за основу логический элемент на LUT4 и D-триггере (рис. 1.13).

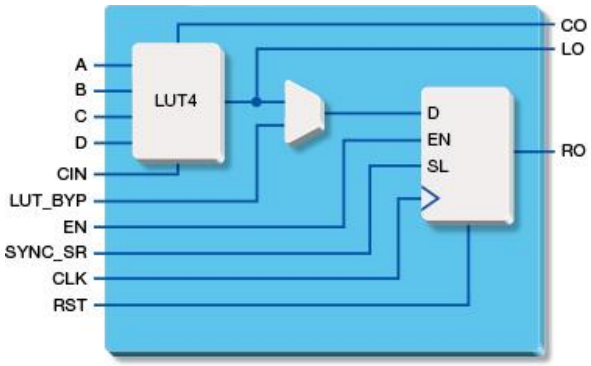


Рис. 1.13. Логический элемент матрицы ПЛИС

Также впервые матрица ПЛИС включает в себя математические DSP блоки для аппаратного ускорения математических вычислений. Структура DSP-блока представлена на рис. 1.14. Возможности блока:

- до 240 математических (DSP) блоков;
- умножение 18 x 18 знаковое, 17 x 17 беззнаковое;
- Выходы каскадирования (44-бит CDIN и CDOUT) для составления больших умножителей/сумматоров/аккумуляторов без дополнительной логики;
- Обратная связь для построения фильтров;
- Пример операции: (A x B) + C или (A x B) + D или (A x B) + C + D.

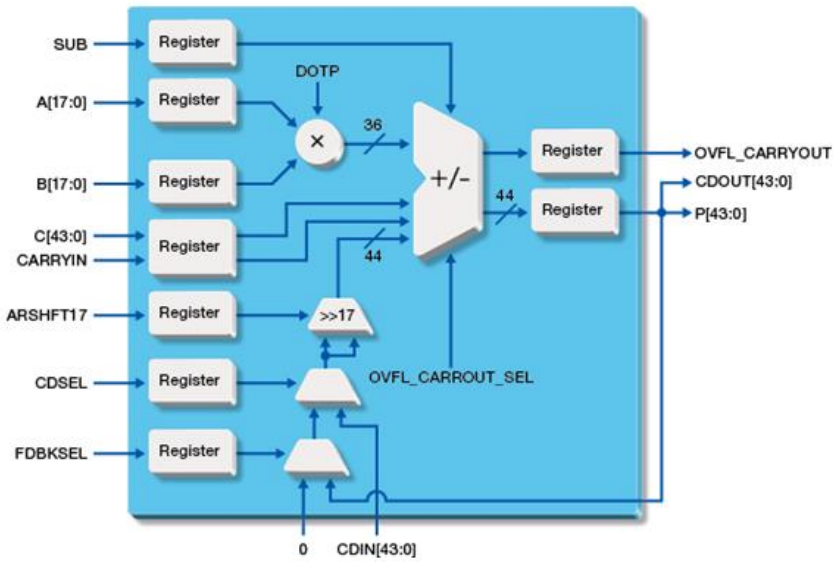


Рис. 1.14. Математический DSP блок

В матрицу включены блоки статического ОЗУ двух типов:

- Large SRAM;
- Micro SRAM.

Первые, соответственно, удобны для построения больших массивов памяти, а вторые пригодятся при реализации FIFO и небольших просмотровых таблиц.

Микросхемы SmartFusion2 и IGLOO2 включают блоки SERDES (рис. 1.15) с аппаратной реализацией контроллера PCIe Gen1/Gen2 x1, x2, x4 и возможностью реализации собственных высокоскоростных последовательных интерфейсов посредством EPCS (extended physical coding sublayer).

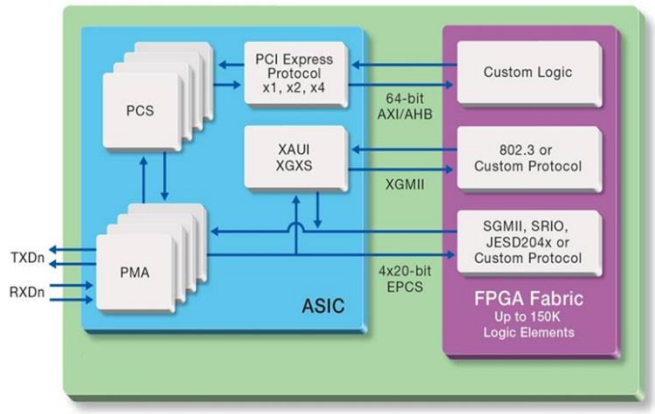


Рис. 1.15. Блоки SERDES в SmartFusion2 и IGLOO2

Hard IP PCIe освобождает разработчика от необходимости реализовывать нижние уровни протокола в матрице ПЛИС, необходимо только описать функции прикладного уровня и подключить получившийся блок к шине AHB или AXI. Этот блок инициирует транзакции по шине AHB или AXI, а всю работу по формированию и обработке пакетов PCIe решает аппаратный модуль.

Для наиболее эффективного взаимодействия матрицы ПЛИС с микроконтроллерной подсистемой или подсистемой памяти разработаны специальные интерфейсы FIC (рис. 1.16). Устройство в матрице ПЛИС может быть либо ведущим, либо ведомым на шине AHB и подключается к соответствующему порту FIC. Взаимодействие происходит в соответствии с адресным пространством матричной шины AHB внутри MSS. Таким образом, можно, например, включить в систему третий UART и обращаться к нему из прикладной программы Cortex-M3.

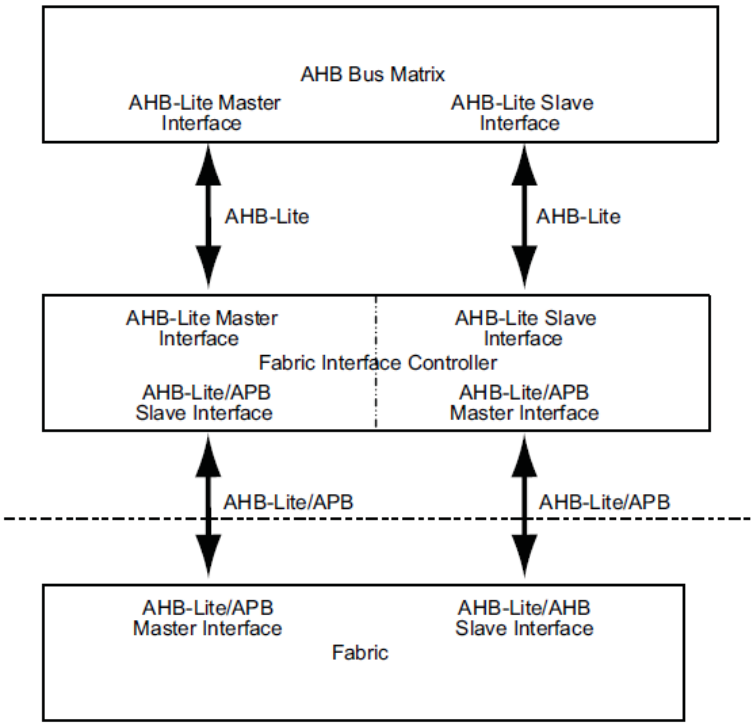


Рис. 1.16. Интерфейсы FIC

2. Состав и установка пакета САПР Libero SoC

2.1. Комплектация пакета САПР Libero SoC

Интегрированная среда разработки Libero SoC — мощная полнофункциональная платформа для создания, отладки и верификации, синтеза и физической имплементации проектов, а так же генерации файлов прошивки под все семейства Flash ПЛИС корпорации Microsemi SoC. Средствами Libero SoC корпорация Microsemi поддерживает полный цикл разработки и отладки систем на основе Flash-based ПЛИС и “систем-на-кристалле”. Менеджер проектов обеспечивает гибкое и эффективное управление файлами проекта и процессом разработки. Графический редактор SmartDesign позволяет быстро и легко создавать как простые, так и большие комплексные “системы-на-кристалле”.

Libero включает в себя, так же, инструменты для анализа и оптимизации потребления для малопотребляющих семейств ПЛИС IGLOO Series and ProASIC3 Series и комбинированных аналого-цифровых ПЛИС SmartFusion и Fusion.

В состав среды Libero SoC входят следующие программные продукты:

- System Builder и SmartDesign для быстрого старта;
- Synposys Synplify Pro для синтеза;
- Mentor Graphics Model*Sim* для симуляции;
- Средства оптимизации трассировки по потреблению и производительности;
- SmartPower для анализа потребления;
- SmartTime для временного анализа;
- Identify для внутрисхемной отладки;
- FlashPro для загрузки конфигурации;
- SoftConsole для разработки встроенного ПО.

Существует возможность использования сторонних (third-party) средств разработки, подключив их к среде Libero и, по необходимости, загрузив с сайта Microsemi необходимые для работы библиотеки. Основной неотъемлемой частью Libero являются средства трассировки, оптимизированной по потреблению и производительности, а также средства временного анализа и анализа потребления.

2.2. Получение инсталлятора и установка пакета Libero SoC

Замечание: все действия, описанные ниже выполнялись в ОС Windows 7 Professional 64-bit Service Pack 1, при этом использован дистрибутив Libero SoC версии 11.5. Если на каком-либо шаге у вас возникает проблема, свяжитесь со службой технической поддержки ЗАО «Актел.ру» support@actel.ru.

Для получения программы-установщика Libero SoC необходимо зайти на официальный сайт Microsemi. Файлы для Windows и Linux доступны по адресу: http://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc#downloads.

Для выполнения всех дальнейших действий требуется регистрация в портале The Microsemi SoC Customer Portal, для чего система откроет страницу с предложением ввести логин и пароль либо зарегистрироваться (рис. 2.1).

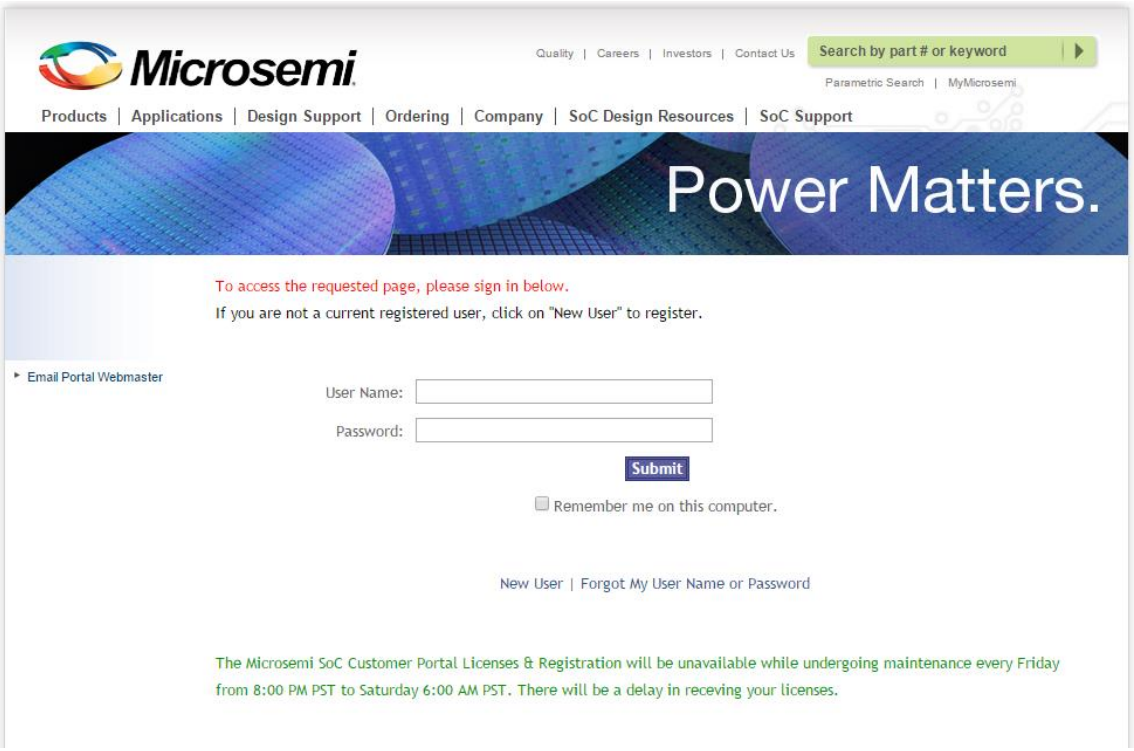


Рис. 2.1. Страница входа в Microsemi SoC Customer Portal

В случае выбора регистрации нового пользователя на следующей странице предлагается ввести свой электронный адрес, после чего система предоставит к заполнению регистрационную форму (рис. 2.2).

Рис. 2.2. Регистрационная форма участника Microsemi SoC Customer Portal

После регистрации снова будет выведена страница входа в портал. Введите указанные при регистрации имя и пароль, нажмите Submit, и система переведёт вас в личный кабинет. Теперь ещё раз перейдём по ссылке http://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc#downloads. Теперь система переместит пользователя на нужную страницу.

Существует два варианта установочного файла: маленький файл, установка с использованием Интернет и большой файл, автономная установка. Выберем автономную установку для универсальности, так как не у всех есть постоянный доступ к Интернет (рис. 2.3).

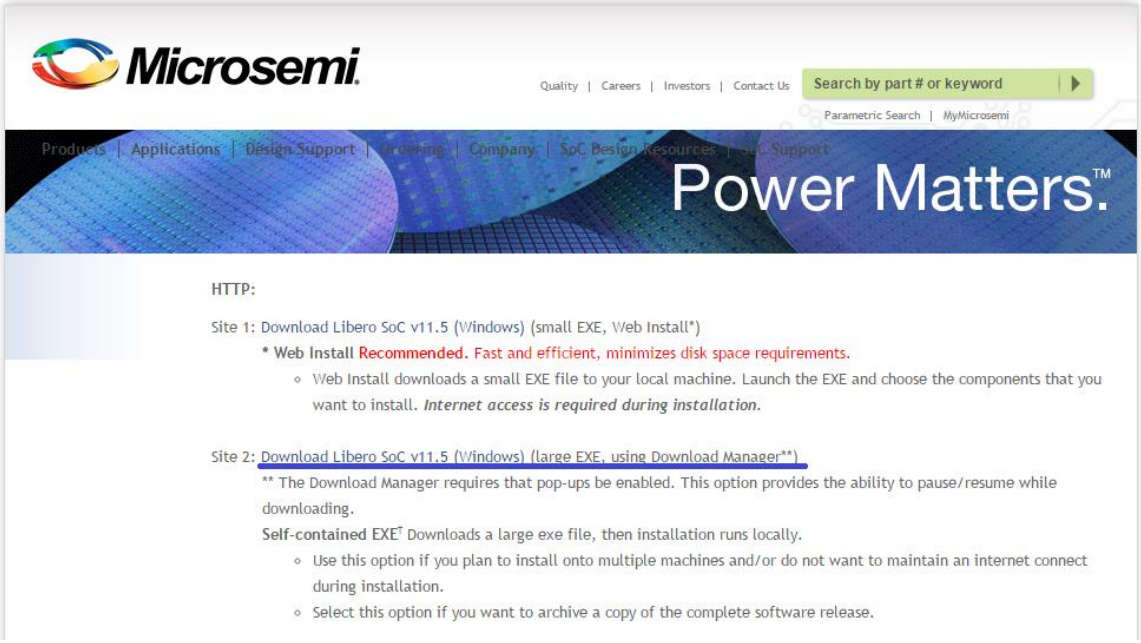


Рис. 2.3. Скачивание программы-установщика

По нажатию соответствующей ссылки откроется новое окно, в котором описаны системные требования и дана ссылка на скачивание (рис. 2.4). Нажимаем download the file without the Microsemi Download Manager. По завершении скачивания файл окажется в папке, которая указана в браузере как каталог для сохранения загруженных файлов. На рисунке 15 окно показано в браузере Google Chrome, где скачанный файл отображается на нижней панели.

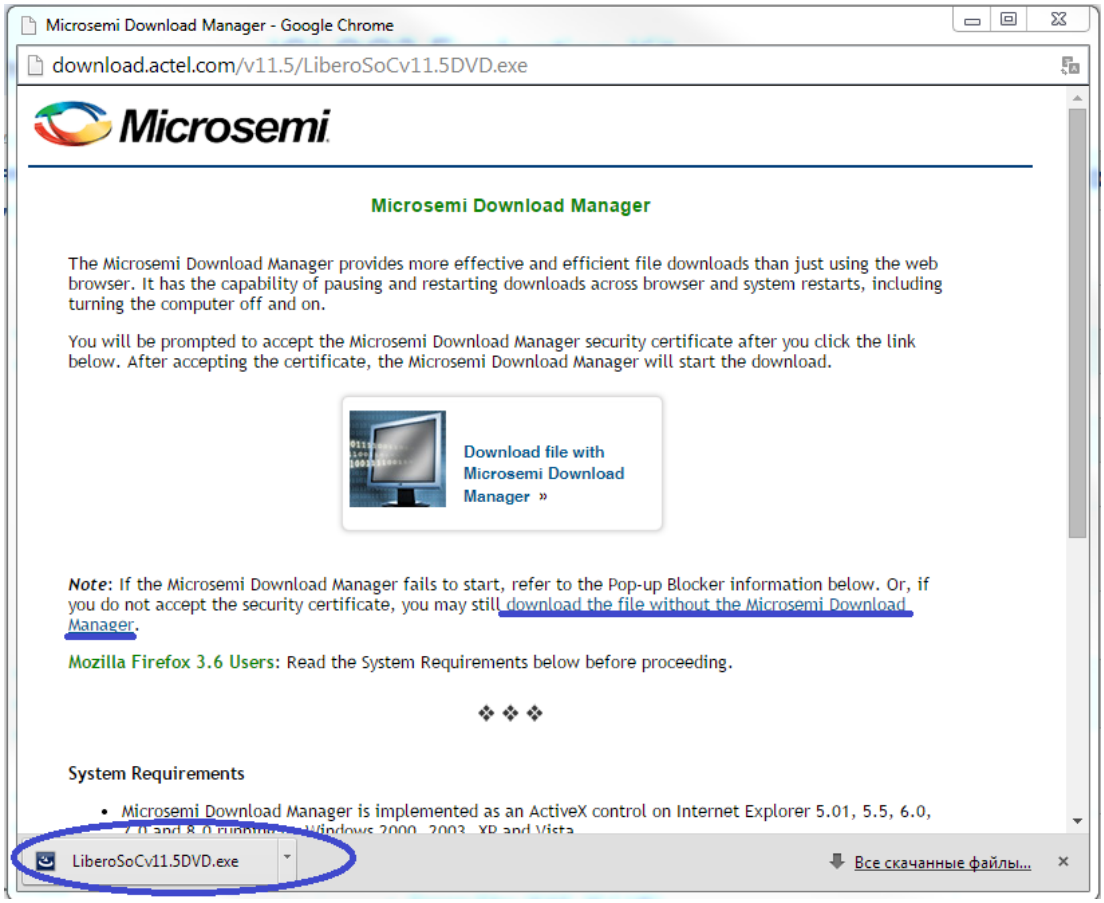


Рис. 2.4. Окно скачивания Libero SoC

Убедившись, что пользователь, от лица которого выполнен вход в Windows, обладает правами администратора, открываем установщик (рис. 2.5).

Замечание: перед установкой желательно закрыть все приложения, антивирус и удалить предыдущие версии Libero.

Предлагается выбрать из двух вариантов: Install Libero SoC, что означает полную установку пакета, включая Synopsys Synplify Pro ME и Mentor Graphics ModelSim ME и Install Libero SoC SA (Standalone), что означает установку *только* среды Libero. Второй вариант рекомендуется использовать только опытным разработчикам, которые предпочитают использовать иные инструменты для логического синтеза и моделирования, так что в рамках данного руководства выбираем вариант по умолчанию.

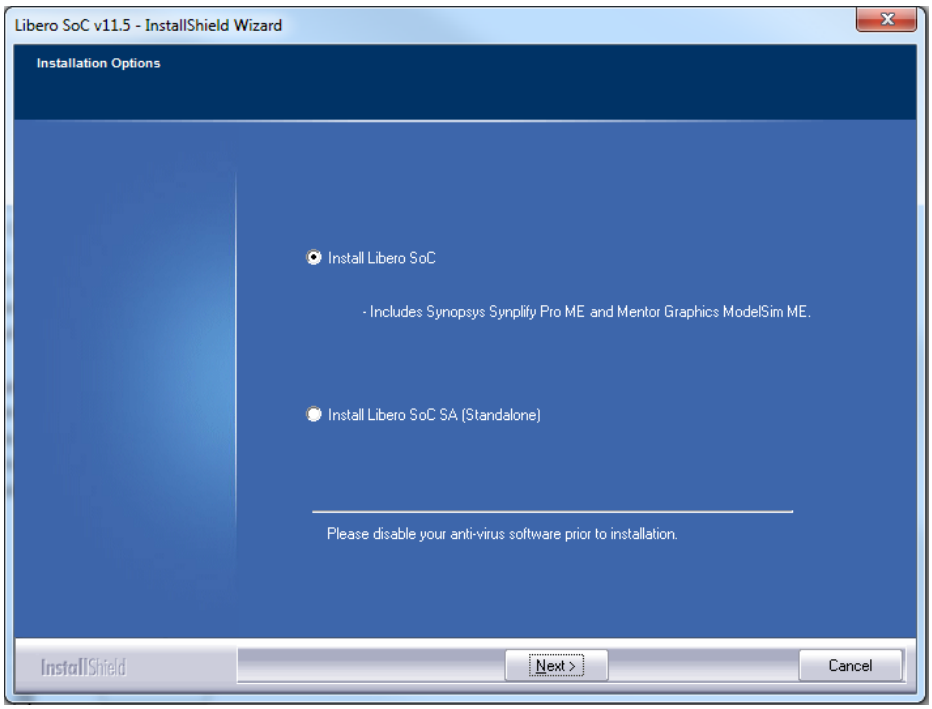


Рис. 2.5. Первое окно установщика

Нажмём Next, в следующем окне нам предлагается прочитать и согласиться с лицензионным соглашением (рис. 2.6). Активируем радиокнопку Yes и нажмём далее.

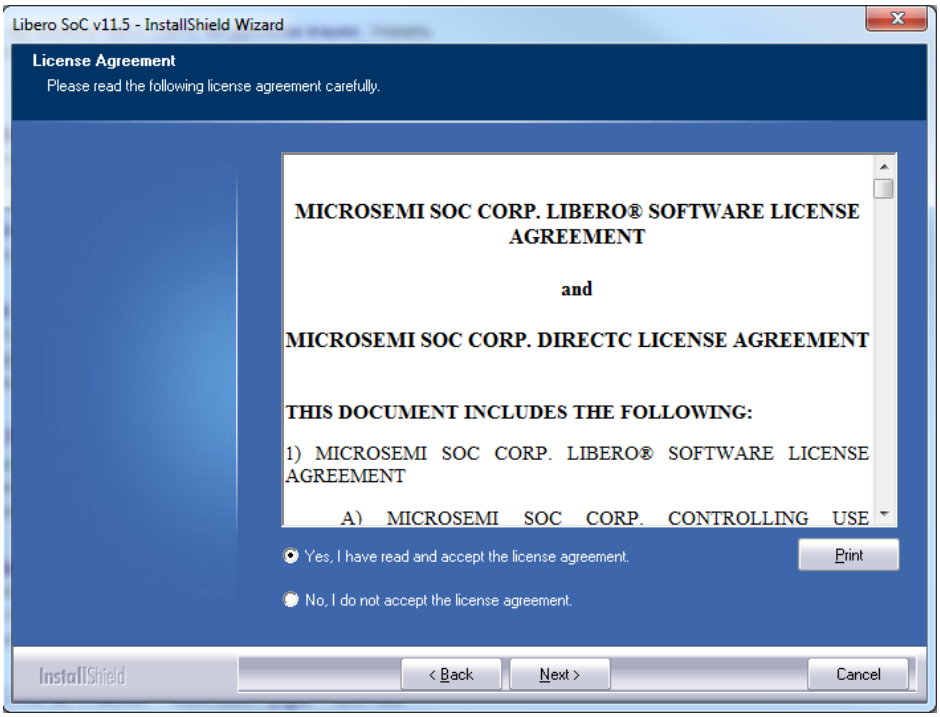


Рис. 2.6. Лицензионное соглашение

В следующем окне предлагается выбрать путь для установки (рис. 2.7). Укажем требуемый путь и пойдем далее, нажав кнопку Next.

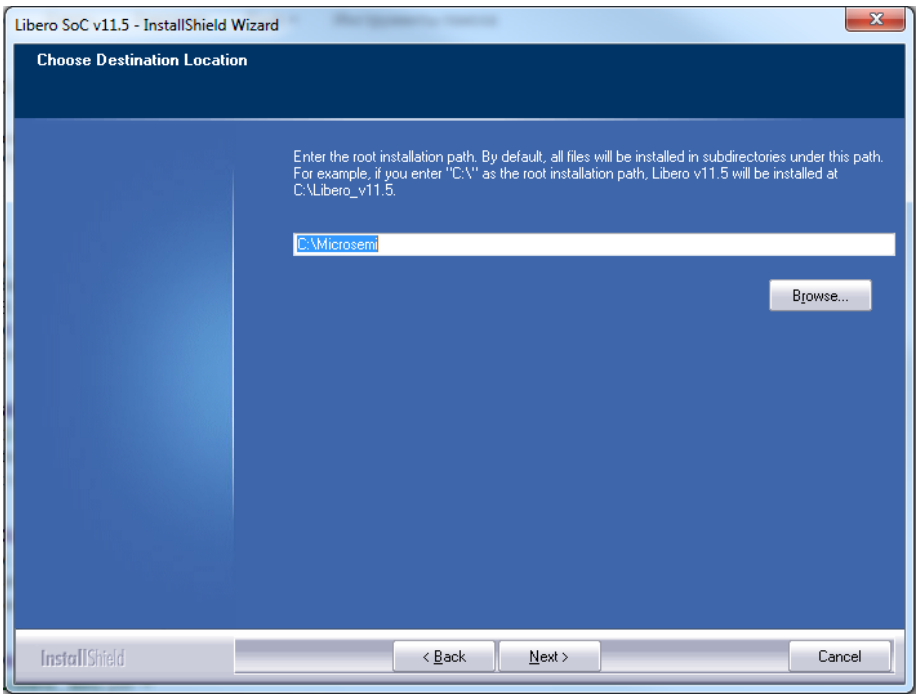


Рис. 2.7. Указание пути установки

В следующем окне нас спросят о том, какие возможности и инструменты нужны нам для работы. По умолчанию выбраны Synplify Pro ME, Identify ME и Modelsim ME. В том случае, если вы собираетесь работать с софт-процессорами, либо процессором ARM-Cortex M3 в составе SmartFusion2, вам необходимо выбрать также SoftConsole. Synphony Model Compilier адресован разработчикам систем цифровой обработки сигналов, он позволяет разрабатывать системы на высоком уровне абстракции в среде MATLAB Simulink и транслировать их в готовое HDL-описание. Если вы относитесь к этой категории, необходимо отметить этот инструмент. И последнее, на этом этапе можно выбрать библиотеки для симуляции в Modelsim, которые необходимо установить. Если вы планируете работать только с каким-то конкретным семейством, нет нужды устанавливать все библиотеки. Итак, отметим всё, что необходимо и пойдём далее (рис. 2.8).

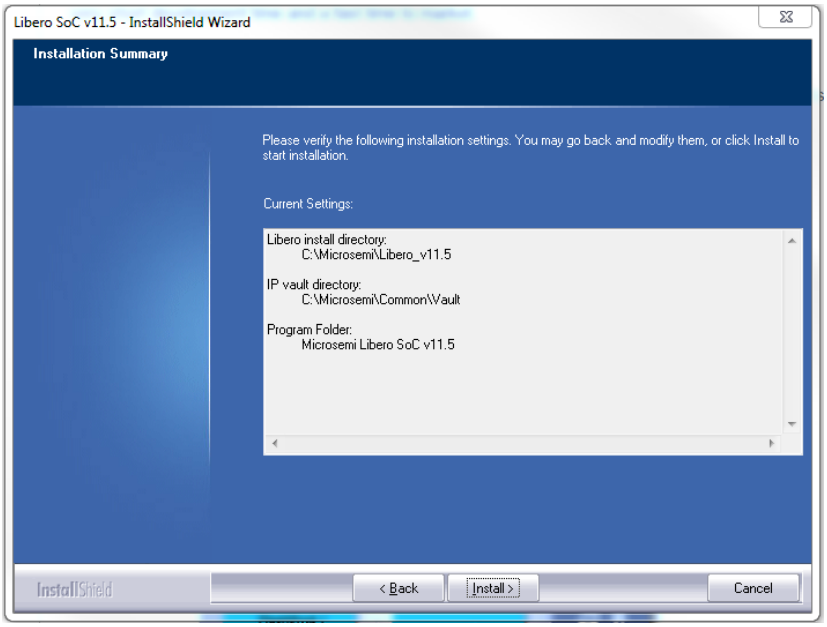


Рис. 2.8. Финальное окно установщика

Мы на итоговой странице, если всё выбрано правильно, нажмем кнопку Install и запустим процесс установки. После проверки соответствия системы требованиям программы запустится диалоговое окно установки драйверов программаторов. Нажмём далее и удостоверимся, что всё прошло успешно (рис.2.9).

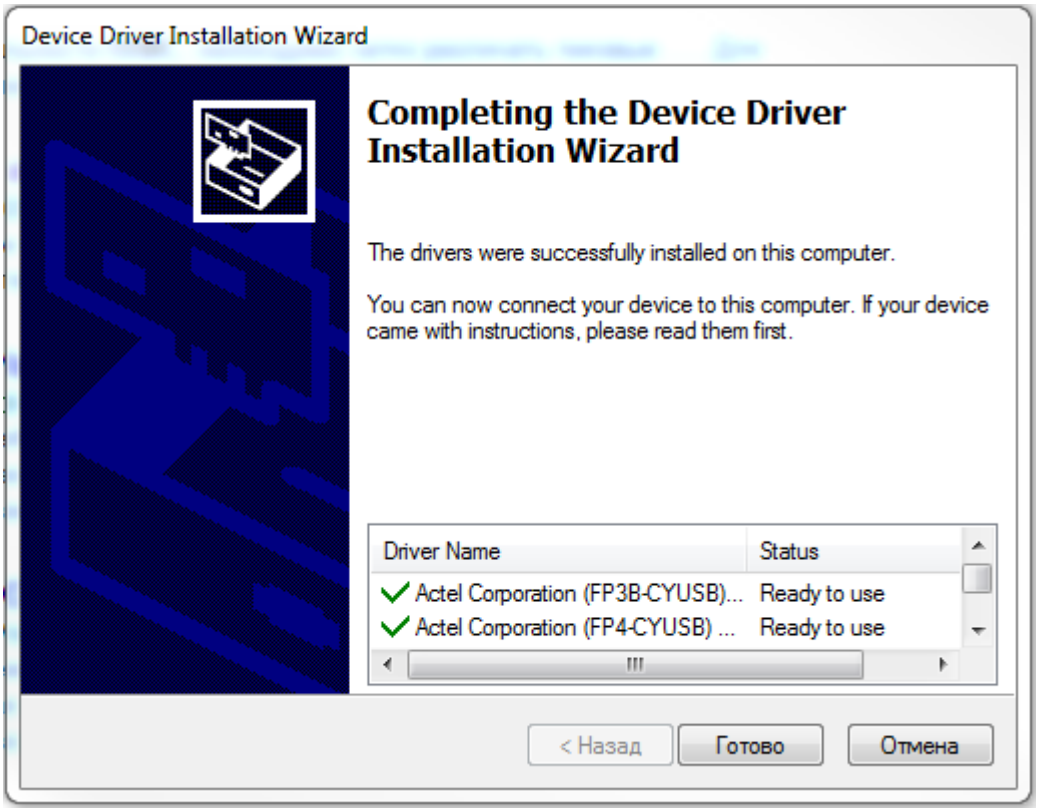


Рис. 2.9. Установка драйверов программаторов

После этого запускается основной процесс установки. Процесс займет достаточно длительное время, в зависимости от производительности вашего компьютера. В конце установки вам предложат подключить программатор FlashPro, если он имеется (рис. 2.10).

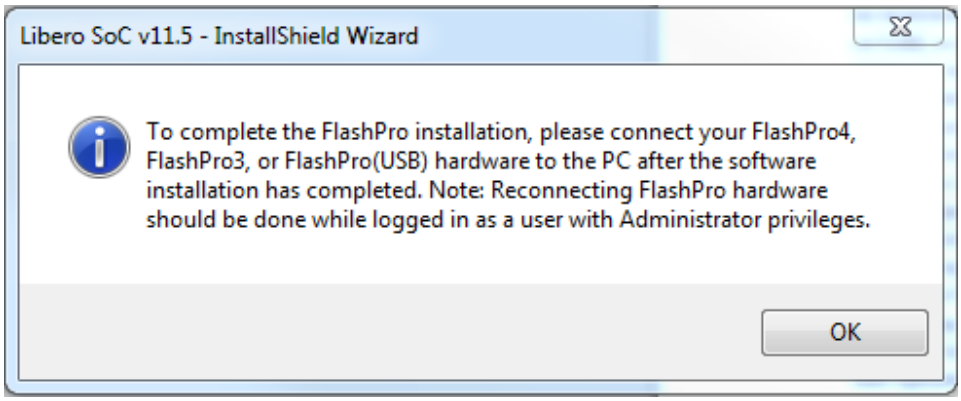


Рис. 2.10. Диалоговое окно с предложением подключить программатор FlashPro

После нажатия кнопки ОК, инсталлятор выведет окно с предложением ввести или получить файл лицензии (рис. 2.11).

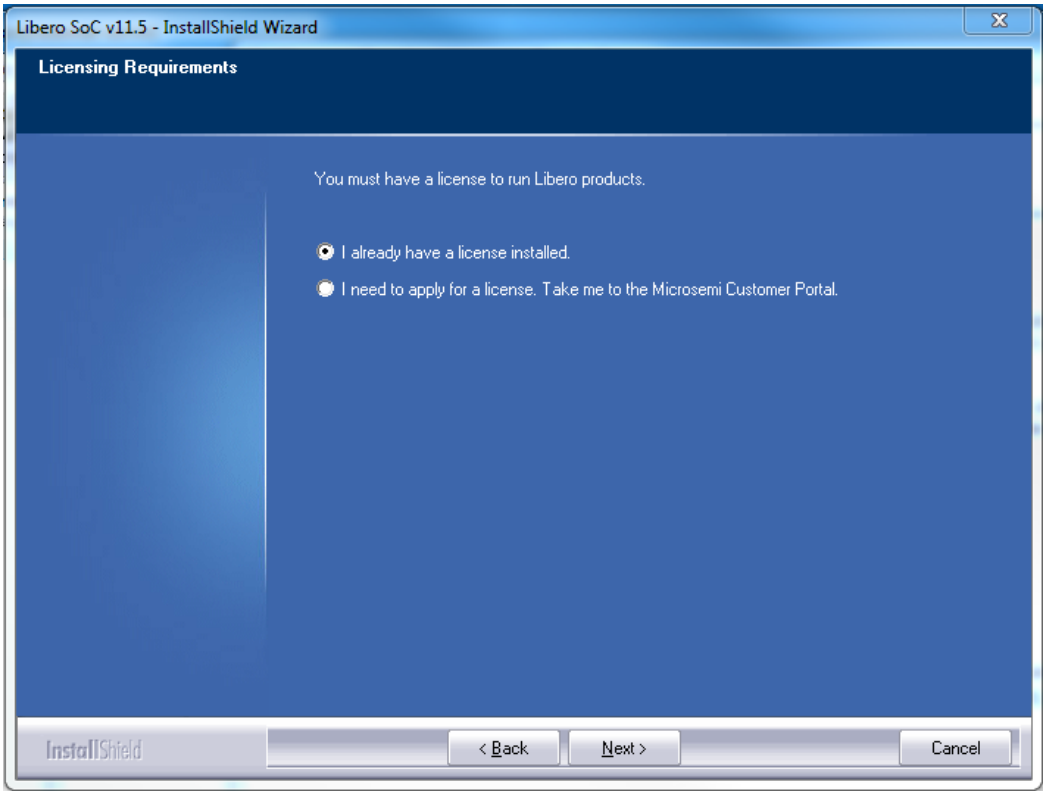


Рис. 2.11. Окно с предложением ввести или получить файл лицензии

Как получить и установить бесплатную лицензию Gold будет рассказано в следующем разделе, оставим настройки по умолчанию и нажмём Next. Откроется финальное окно инсталлятора (рис. 2.12).

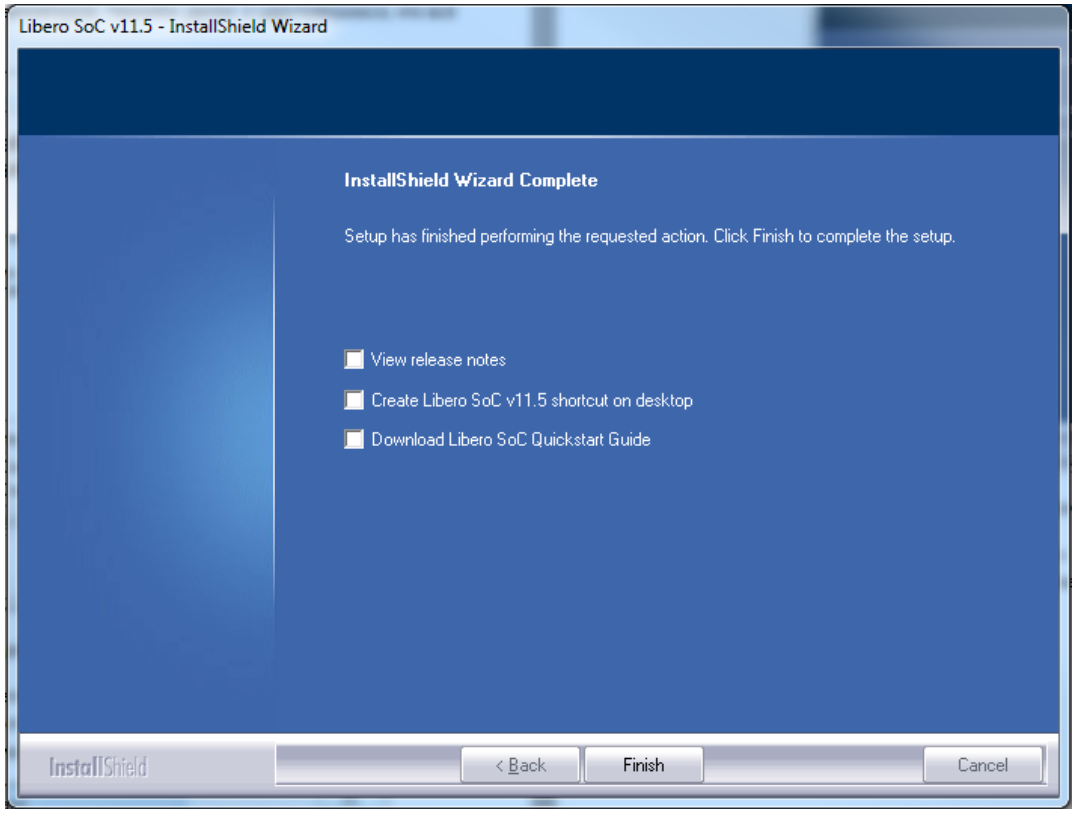


Рис. 2.12. Финальное окно инсталлятора

После нажатия кнопки Finish, установка будет закончена, и мы сможем увидеть группу программ Microsemi Libero SoC в меню Пуск (рис. 2.13).

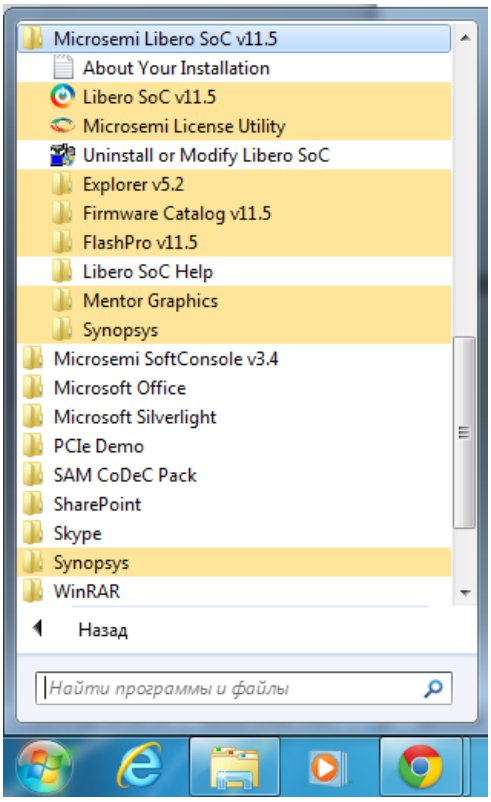


Рис. 2.13. Группа программ Microsemi Libero SoC в меню Пуск

После того, как Libero SoC успешно установлена, необходимо получить лицензию на её использование. Рассмотрим этот процесс на примере лицензии Gold, которой достаточно для старта и ведения большинства проектов.

2.3. Получение файла лицензии

Для получения файла лицензии Gold необходимо перейти по следующей ссылке: http://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc#licensing. На странице предлагается подать заявку на получение бесплатной лицензии (рис. 2.14).

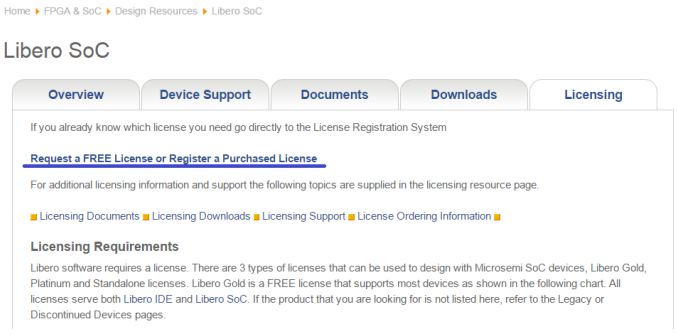


Рис. 2.14. Страница получения лицензии Libero SoC

Перейдём по ссылке “Request a FREE or Register a Purchased License”. Попадём в личный кабинет (рис. 2.15)

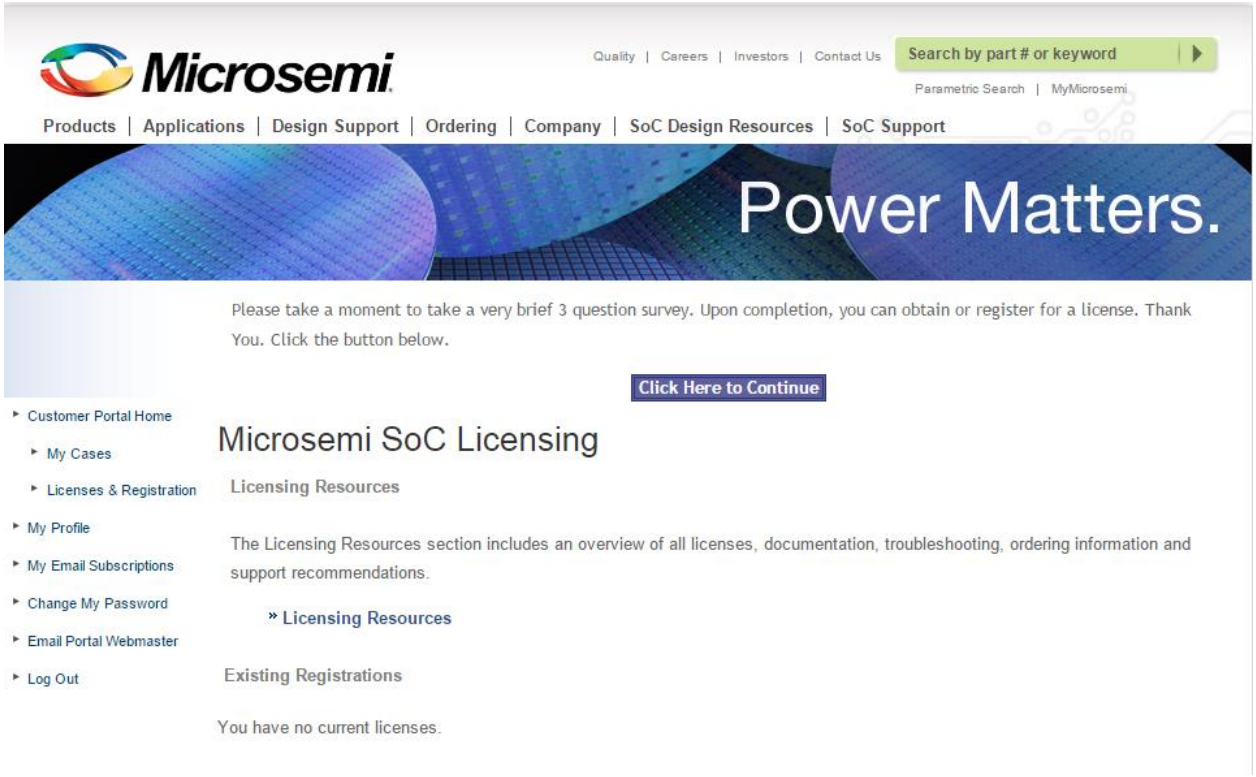


Рис. 2.15. Личный кабинет в портале

Для отправки запроса на получение лицензии следует нажать Click Here to Continue и ответить на несколько вопросов. После этого система отобразит страницу регистрации, где можно запросить бесплатную лицензию Gold или зарегистрировать платную Platinum. Нажмём Request Free License для продолжения (рис. 2.16).

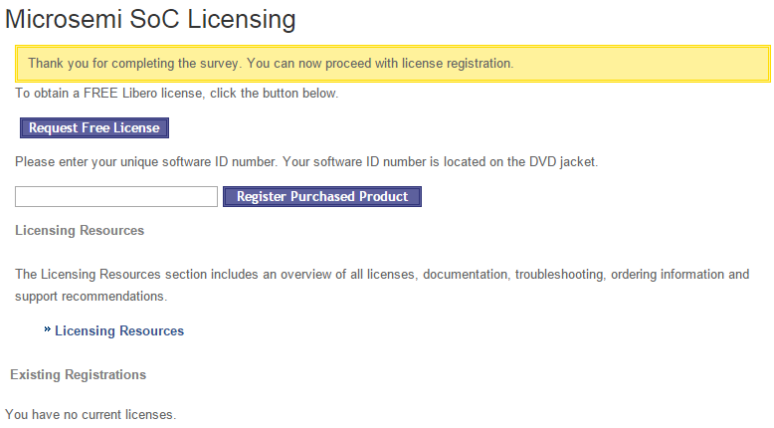


Рис. 2.16. Страница получения лицензий

На следующей странице будут предложены следующие варианты:

- Libero Gold Node Locked for Windows
- Libero Gold Floating License for Windows or Linux Server
- Libero Gold Floating License for Solaris Server

- Synopsys Synphony Model Compiler ME. Requires MATLAB/Simulink from Mathworks.

В случае установки под Windows, выбираем первый вариант. Он означает лицензию, привязанную к физическому носителю, жёсткому диску, поэтому система потребует ввести ваш DiskID (рис. 2.17).

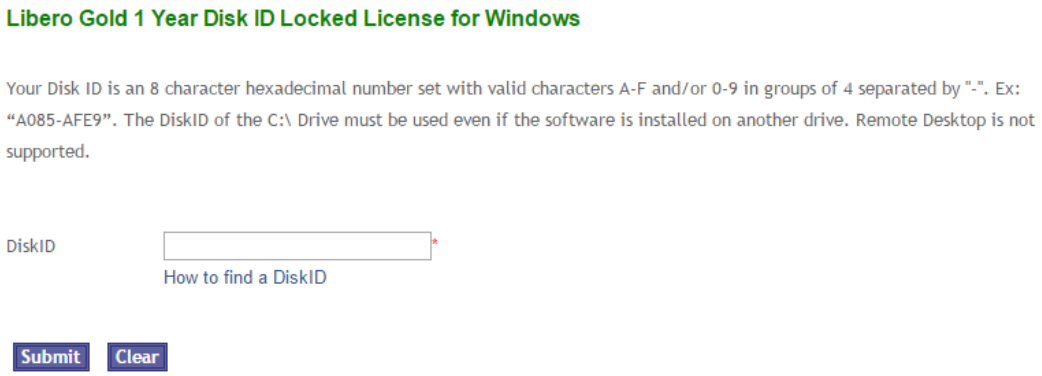


Рис. 2.17. Запрос DiskID пользователя

Для того, чтобы узнать свой DiskID требуется открыть командную строку Windows (cmd) и набрать в ней команду `vol C:`. Далее следует скопировать полученное число формата XXXX-XXXX в строку запроса в браузере. После нажатия Submit система выдаст персональный Software ID и уведомит, что лицензия будет отправлена на почту в течение 30 минут.

К пришедшему письму будет прикреплен файл с расширением *.dat. Скачав файл, открываем приложение Microsemi License Utility в группе программ Microsemi Libero SoC в меню Пуск (рис. 2.18).

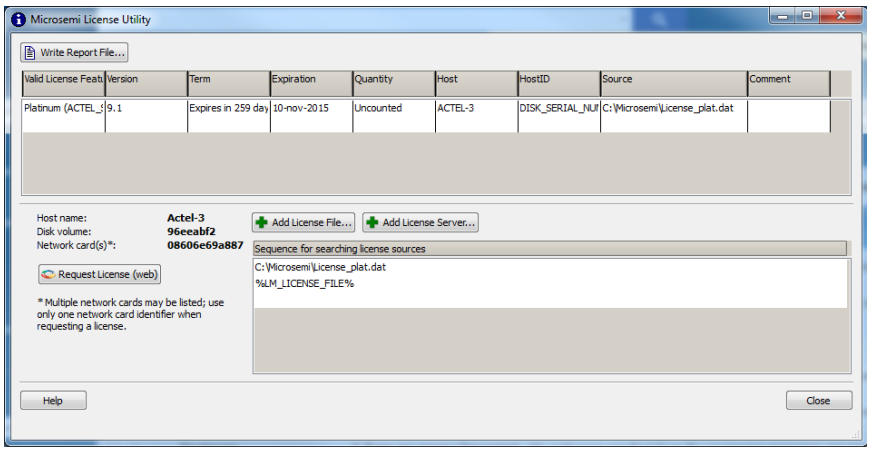


Рис. 2.18. Приложение Microsemi License Utiilty

Осталось только прикрепить полученный файл лицензии, для этого нажимаем Add License File и указываем путь к файлу. Необходимо удостовериться, что соответствующая запись появилась в окне утилиты как на рисунке 2.18.

3. Проектирование для ПЛИС, изготовленных по Flash-технологии. Среда Libero SoC

3.1.Общие характеристики и интерфейс среды Libero SoC

После установки лицензии можно начинать работу с Libero SoC, для этого запустим Libero SoC из группы программ в меню Пуск и рассмотрим интерфейс приложения (рис. 3.1).

Рис. 3.1. Главное окно Libero SoC

Интерфейс Libero лаконичен и подходит для быстрого старта. Меню и панель быстрого доступа обеспечивают оперативный доступ ко всем органам управления проектом, файлами и интерфейсом приложения. На приветственной странице в центре окна приводится краткое описание среды и некоторых семейств Microsemi. Для демонстрации Libero в рабочем режиме рассмотрим интерфейс создания нового проекта.

Для создания нового проекта можно нажать New… в левой части экрана (секция Projects) или выбрать Project -> New Project. На экране появится диалоговое окно создания нового проекта (рис. 3.2). В левой части экрана показан процесс создания нового проекта:

- Детали проекта (имя, расположение, описание, язык);
- Выбор ПЛИС;
- Настройки ПЛИС;
- Выбор шаблона проекта;
- Добавление исходных HDL-текстов;
- Добавление файлов ограничений.

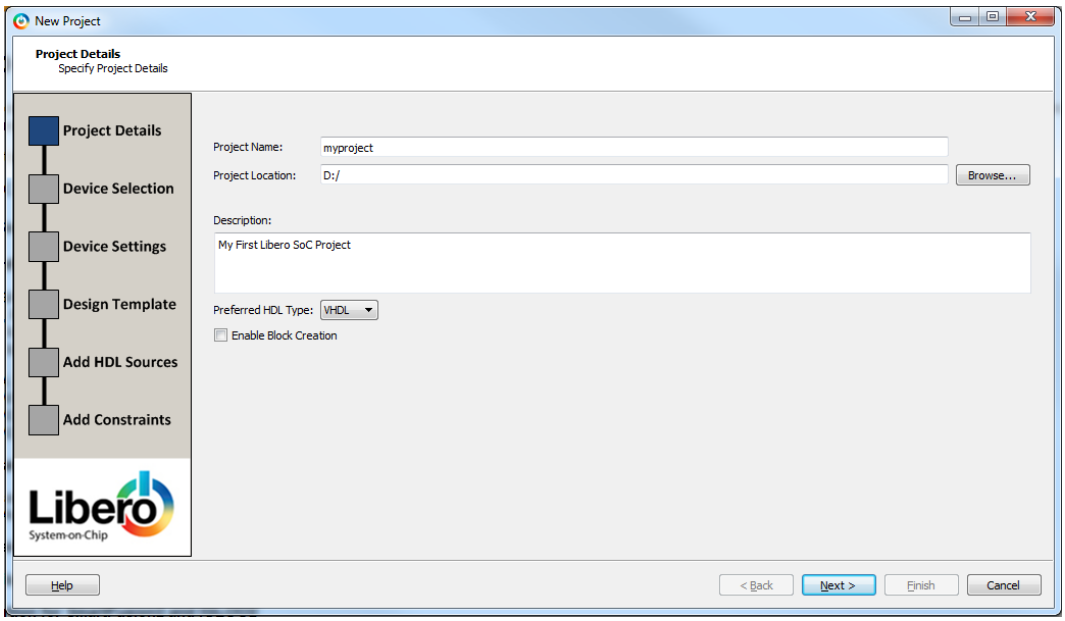


Рис. 3.2. Диалоговое окно создания нового проекта

На первом этапе вводится информация об имени и расположении проекта и предпочтительный язык HDL — VHDL или Verilog. Опционально можно ввести описание проекта. Заполним поля, как показано на рисунке 3.2 и нажмём Next.

Следующим шагом разработчику предлагается выбрать используемый кристалл (рис. 3.3). При выборе можно использовать фильтр по различным параметрам и характеристикам микросхемы, от семейства до температурного диапазона, а также строку поиска. В качестве примера в данном случае выбран первый пункт в появившемся списке — A3P015-1QN68.

Замечание: при выпуске новых версий Libero интерфейс программы и отдельных компонент могут меняться. Все изменения документируются в соответствующем Release Notes.

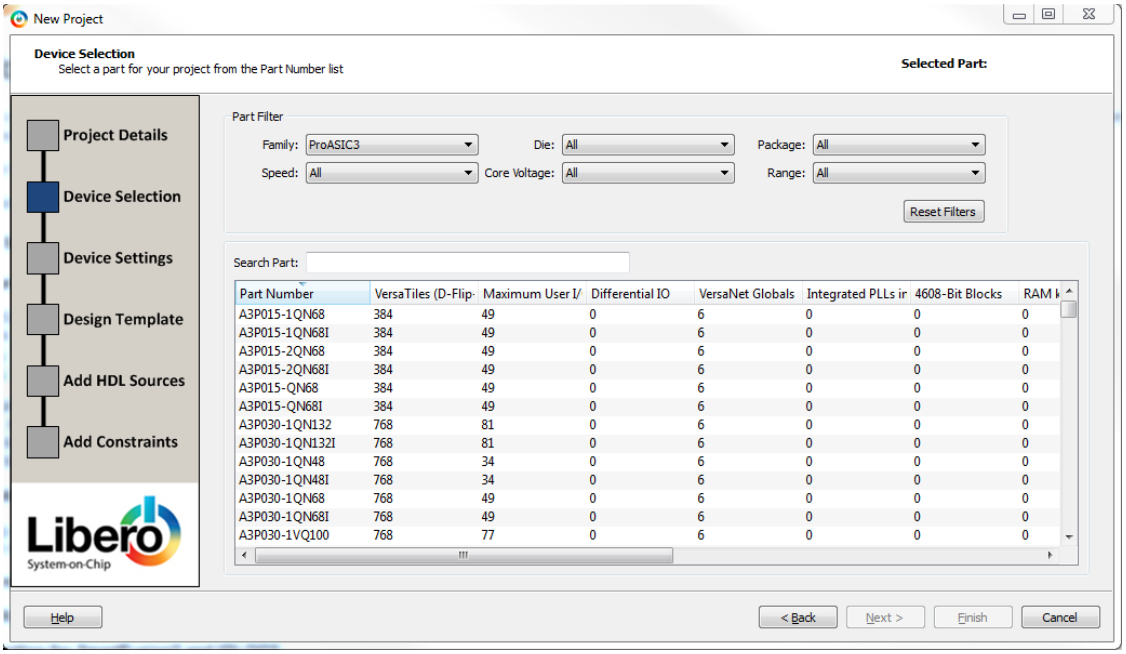


Рис. 3.3. Выбор ПЛИС при создании проекта

Следующий шаг содержит всего один пункт — выбор стандарта ввода-вывода, по умолчанию это LVTTL. Выбранный стандарт будет использован по умолчанию для всех внешних выводов микросхемы.

После нажатия Next, пользователь попадает на страницу выбора шаблона проекта. Шаблоны поддерживаются только для проектов IGLOO2/SmartFusion2 и предполагают настройку подсистем MSS и HPMS. Подробнее о настройке этих подсистем и проектировании для IGLOO2 и SmartFusion2 будет рассказан в разделе 3.4.

Следующим этапом предлагается импортировать в проект существующие HDL-файлы (рис. 3.4). Система предлагает два варианта:

- Import File — скопировать файл в каталог проекта и прикрепить его к проекту;
- Link File — прикрепить существующий файл к проекту.

Заключительный этап позволяет импортировать в проект файлы ограничений. Выбор производится аналогично HDL-файлам. Заметим, что данные действия можно выполнить в уже созданном проекте в среде Libero.

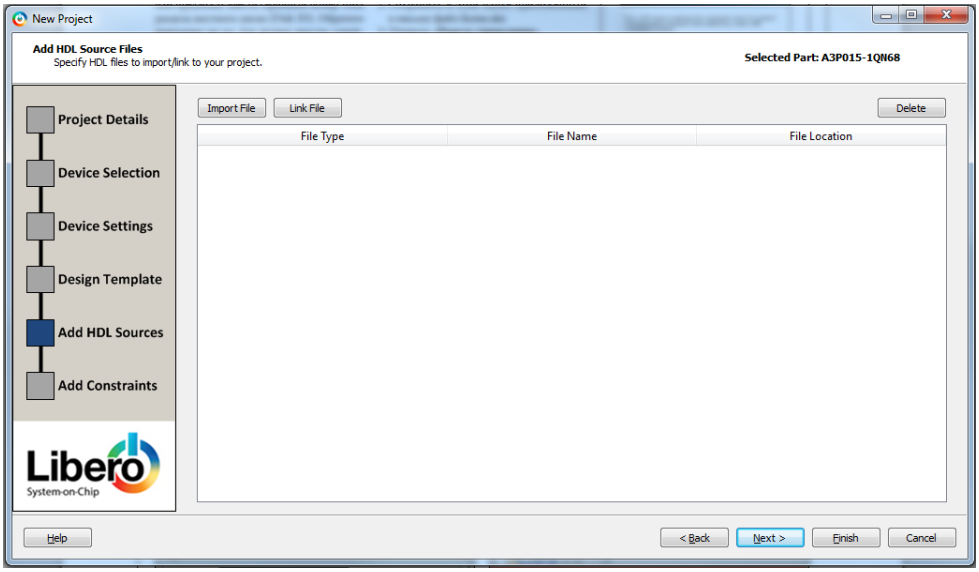


Рис. 3.4. Импорт существующих файлов в проект

После нажатия Finish завершается создание проекта. Перед пользователем появляется главное окно Libero в рабочем состоянии (рис. 3.5). Окно содержит несколько основных областей.

- Область 1 — органы управления процессом разработки, каталогом IP, файлами;
- Область 2 — рабочая область для создания описаний графическим и текстовым способом, а также для отображения отчётов;
- Область 3 — сообщения и лог действий.

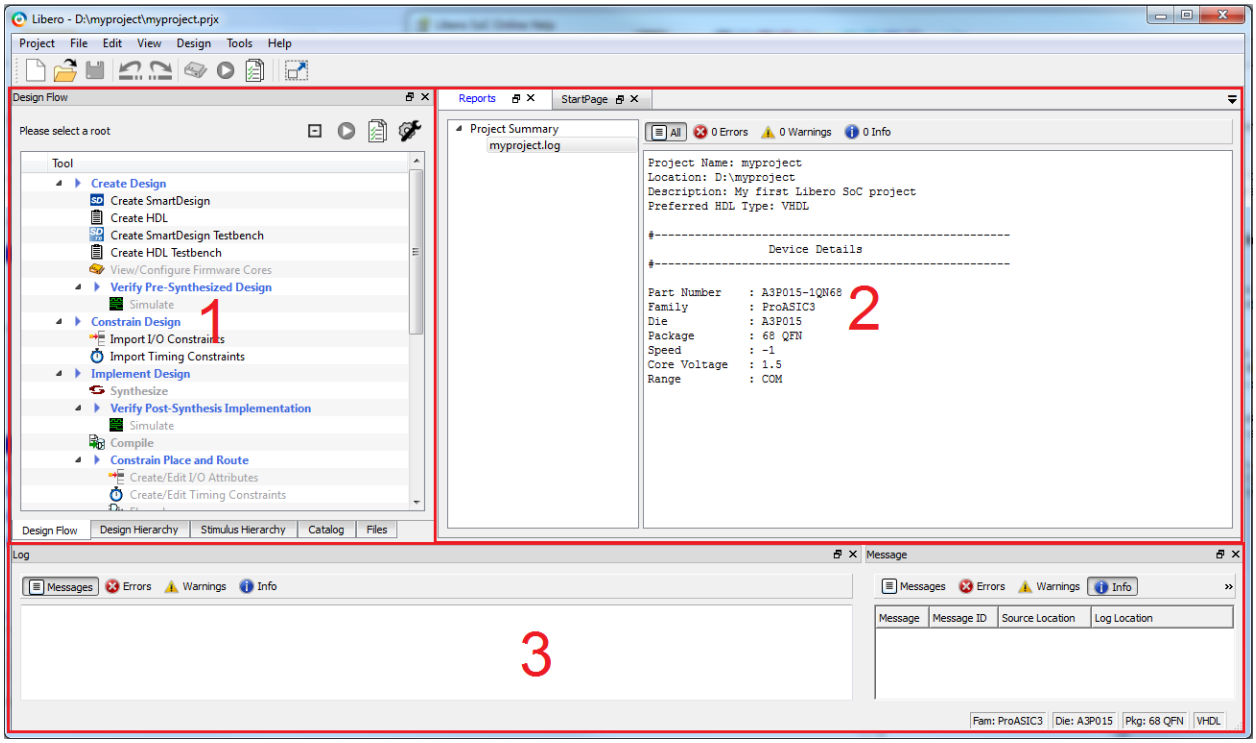


Рис. 3.5. Главное окно Libero SoC в рабочем состоянии

Также окно содержит привычные органы управления и контроля — меню, панель быстрого доступа и панель статуса. Любая область может быть сконфигурирована по желанию пользователя.

После создания проекта пользователь видит в области 2 первый, и пока единственный отчёт — о создании проекта. Сведения в нём отражают информацию, введённую при создании проекта. Также эту информацию можно наблюдать в нижней части окна на панели статуса. Разработчик всегда может вернуться к базовым настройкам проекта, открыв соответствующий инструмент File -> Project Settings. В данном окне содержатся как настройки, которые вводились при создании проекта, так и некоторые другие пункты, которые будут рассмотрены позже.

3.2. Этапы проектирования в среде Libero SoC

В области 1 по умолчанию открыта вкладка Design Flow, которая демонстрирует разработчику все шаги процесса проектирования устройства, а также даёт возможность запускать их все последовательно или запустить какой-либо отдельный процесс. Проектирование в Libero включает следующие этапы:

1. Создание описания устройства несколькими доступными способами.
2. Задание ограничений.
3. Логический синтез.
4. Компиляция.
5. Размещение и трассировка.

6. Проверка на соответствие требованиям потребления и производительности.
7. Разработка встроенного ПО (опционально).
8. Генерация конфигурационного файла.
9. Конфигурирование (программирование) ПЛИС.
10. Внутрисхемная отладка.

Конечно, это итерационный и циклический процесс с множеством переходов к предыдущим шагам. Отражение данного процесса в Libero (рис. 3.6) представляет собой более развернутую версию представленного алгоритма.

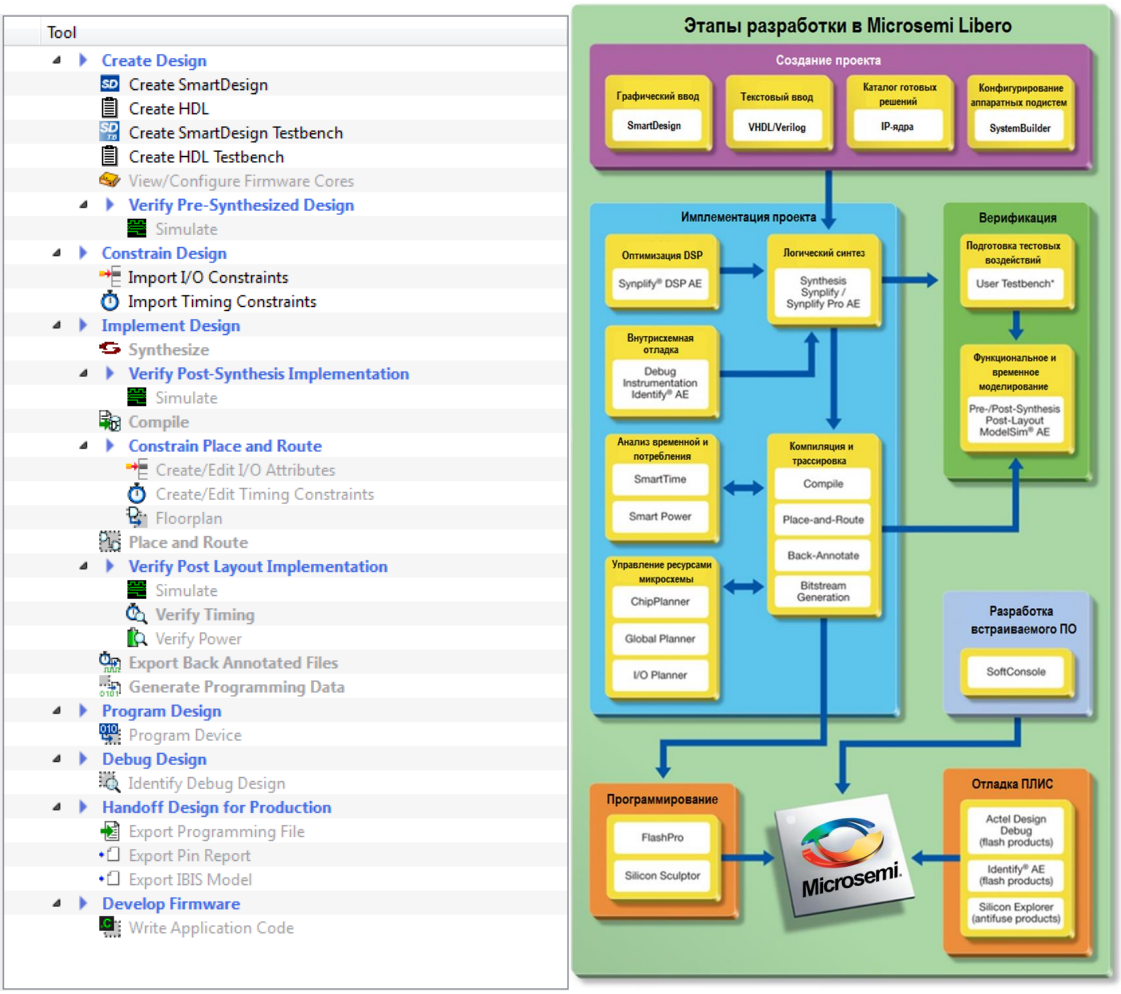


Рис. 3.6. Процесс проектирования (Design Flow) в Libero SoC

Первый этап проектирования представлен секцией Create Design. Здесь можно создать HDL-описания отдельных узлов устройства и графическое описание с помощью инструмента SmartDesgin. В эту секцию также включено создание тестовых воздействий (тестбенч) для симуляции. О создании описания устройства подробнее будет рассказано в разделе 3.3.

Задание временных и физических ограничений (2) может происходить как до исполнения остальных пунктов процесса путём импорта соответствующих файлов (Import I/O Constraints, Import Timing Constraints), так и во время этого процесса с использованием графических сред (Create/Edit I/O Attributes, Create/Edit Timing Constraints).

Логический синтез (3) представляет собой процесс генерации списка соединений (netlist) логических вентилей. В Design Flow Libero логическому синтезу может предшествовать симуляция. Кроме того выполнить симуляцию можно после логического синтеза проекта.

Компиляция (Compile, 4) и размещение и трассировка (Place and Route, 5) представляют собой проверку и оптимизацию списка соединений, трансляцию его в низкоуровневое описание с использованием примитивов конкретного кристалла, и размещение и соединение полученных элементов. Для экспорта файлов симуляции служит пункт Export Back Annotated Files.

После выполнения этих этапов разработчик получает возможность также провести симуляцию полученной модели, учитывающей временные характеристики дизайна в конкретной ПЛИС, а также произвести статический временной анализ и анализ по потреблению (6). Это соответственно пункты Verify Timing и Verify Power.

В случае использования в проекте софт-процессора или процессора ARM Cortex-M3 в составе SmartFusion2, разработчик имеет возможность разработать для него встроенное ПО, используя встроенную среду SoftConsole (Write Application Code, 7).

Генерация конфигурационного файла (Generate Programming Data, 8) и прошивка ПЛИС (Program Device, 9) являются заключительными этапами разработки и начальных этапов отладки, после чего можно привлечь инструменты внутрисхемной отладки (Debug Design, 10) для тестирования проекта на натурном образце и в составе устройства.

Для запуска всего процесса служит кнопка . При этом процесс стартует с последнего незавершенного этапа и доходит до конечного (Generate Programming Data, генерация файла прошивки). Также, двойным щелчком на каждом из пунктов можно запустить его индивидуально, однако если не были выполнены предыдущие, то сначала будут запущены они.

3.3. Способы описания устройств в Libero SoC

3.3.1. Схемный редактор SmartDesign

Схемный редактор SmartDesign является удобным способом создания описаний устройств на уровне модулей. Модулями могут быть:

- Символы HDL-описаний пользователя;
- Подсистемы HPMS и MSS (только для SmartFusion2/IGLOO2);
- IP-ядра из каталога;
- Макросы.

Для создания нового компонента SmartDesign достаточно сделать двойной щелчок по пункту Create SmartDesign во вкладке DesignFlow. Появится диалоговое окно ввода имени компонента (рис. 3.7).

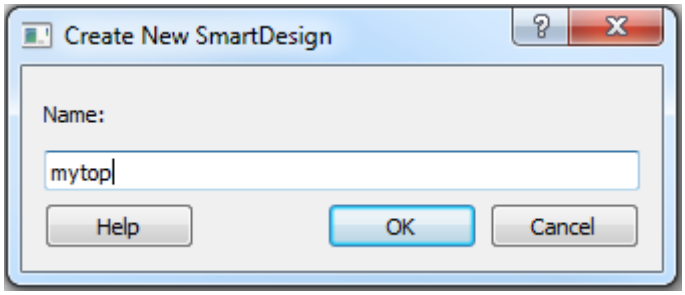


Рис. 3.7. Диалоговое окно ввода имени SmartDesign

После ввода имени появляется рабочее поле SmartDesign — canvas (рис. 3.8), на котором можно помещать модули и монтировать нужную схему соединений.

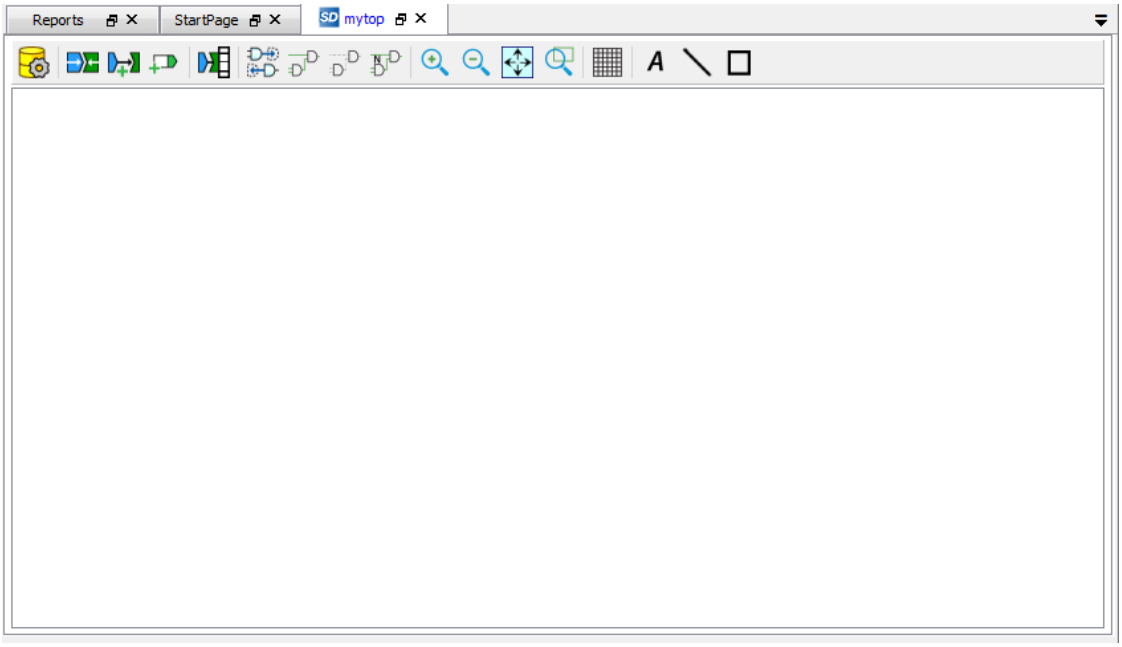


Рис. 3.8. Рабочее поле SmartDesign (canvas)

Основные органы управления рабочим полем сведены в таблицу 3.1 (доступны также в выпадающем меню по правому щелчку мыши в области рабочего поля).

Таблица 3.1. Основные органы управления

	Описание	Назначение
	Generate Component	Сгенерировать HDL-описание компонента. Необходимо для запуска синтеза и следующих шагов маршрута проектирования.
	Auto Connect	Установить все возможные соединения по умолчанию (например, интерфейсы AMBA)
	Connection mode	Вход в режим соединения. В этом режим меняется внешний вид курсора и появляется возможность соединять порты щелчком мыши.

	Add port	Добавить входной/выходной порт компонента
	Quick Connect	Открыть диалоговое окно QuickConnect
	Auto Arrange Instances	Упорядочить элементы в рабочем поле
	Route All Nets	Заново провести линии связи между элементами
	Hide Nets	Скрыть линии связи между элементами
	Zoom In	Масштаб (аналогично Zoom Out)
	Zoom To Fit	Масштабировать так, чтобы все элементы были видны на экране
	Enable Background Grid	Включить сетку

Когда в рабочем поле присутствуют элементы (блоки), можно манипулировать ими — перемещать по полю, подключать к другим блокам, а также получать доступ к их настройкам и описанию. Двойной щелчок по любому из блоков открывает доступ к его исходному тексту (HDL-описание, подробнее будет рассмотрено в разделе 3.3.2) или настройкам (IP-ядро, подробнее будет рассмотрено в разделе 3.3.3). По щелчку правой кнопкой мыши на портах ввода-вывода блоков пользователь получает доступ к меню управления портами (рис. 3.9).

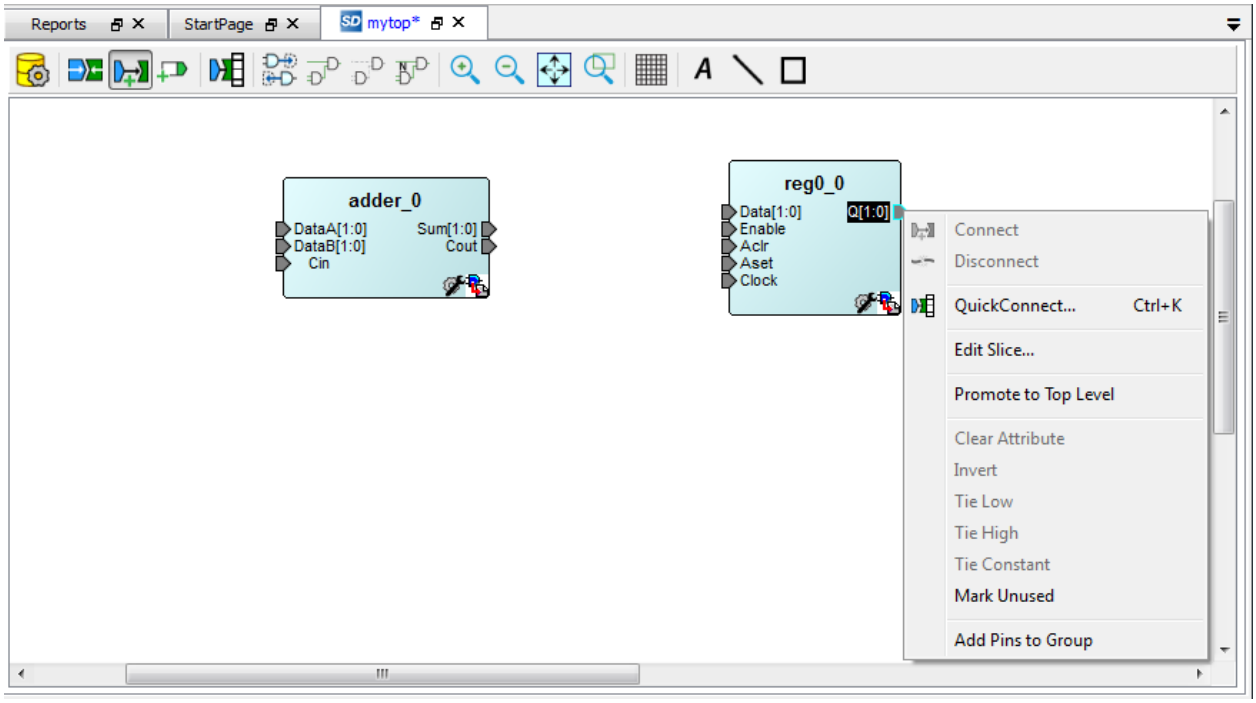


Рис. 3.9. Рабочее поле с блоками и меню управления портами ввода-вывода

Команды меню приведены в таблице 3.2.

Таблица 3.2. Команды меню управления портами

Команда	Направление порта	Назначение
Connect	Вх/вых	В случае двух или более выбранных портов — соединить порты
Disconnect	Вх/вых	В случае двух или более выбранных портов — разъединить порты
Quick Connect	Вх/вых	Открыть диалоговое окно QuickConnect
Edit Slice	Вх/вых	В случае шины — разбить на подгруппы
Promote to Top Level	Вх/вых	Добавить в поле внешний порт компонента SmartDesign и соединить с выбранным выводом блока
Clear Attribute	Вх/вых	Снять предыдущие соединения/назначения
Invert	Вых	Инвертировать порт
Tie Low/Tie High/Tie Constant	Вх	Подтянуть к GND/VCC/константе в случае шины
Mark Unused	Вых	Отметить, как неиспользуемый
Add Pins to Group	Вх/вых	Создать группу выводов и добавить туда выбранные

Используя описанные инструменты достаточно просто соединить необходимые компоненты (рис. 3.10).

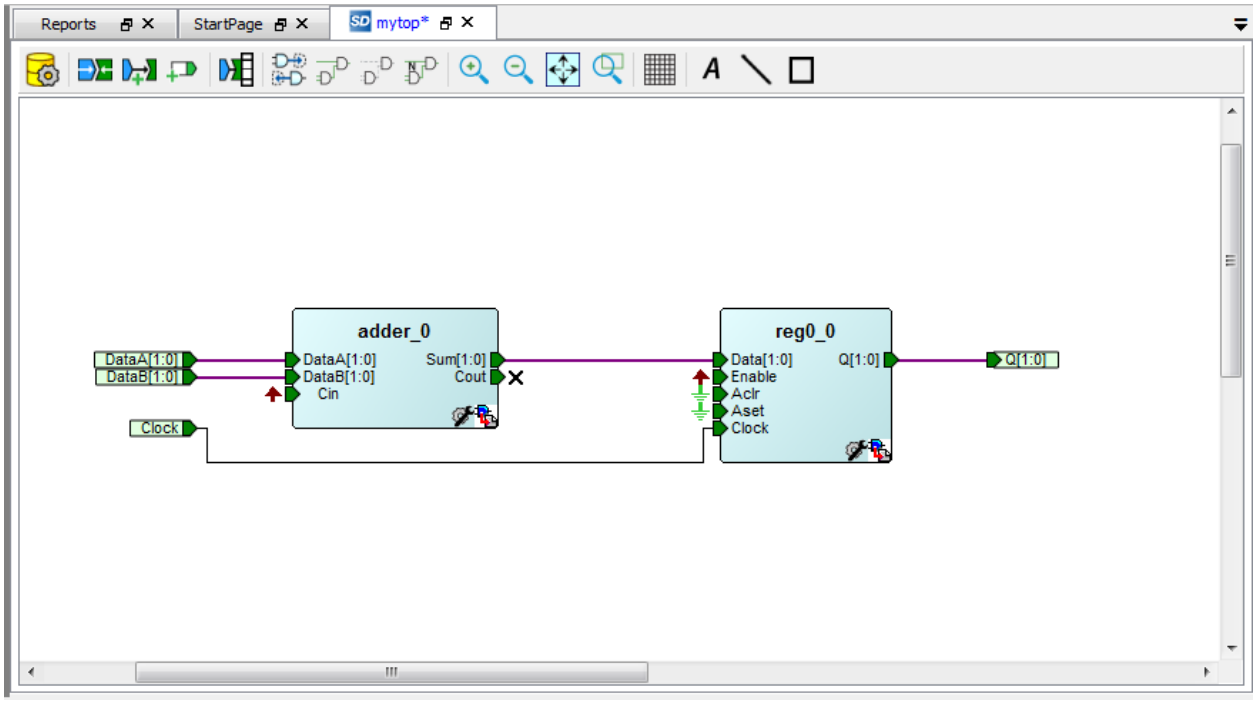


Рис. 3.10. Рабочее поле с соединенными блоками

Рассмотрим более подробно инструмент QuickConnect (рис. 3.11). В левой части выбирается один из блоков в рабочем поле и выделяется один или более из его выводов. После этого выделяется один или более выводов в правой части, где расположены все выводы, сгруппированные по блокам в рабочем поле. Щелкнув правой кнопкой мыши по выделению в левой или правой области, пользователь открывает меню с пунктами из таблицы 3.2. Для соединения выбранных выводов необходимо нажать Connect.

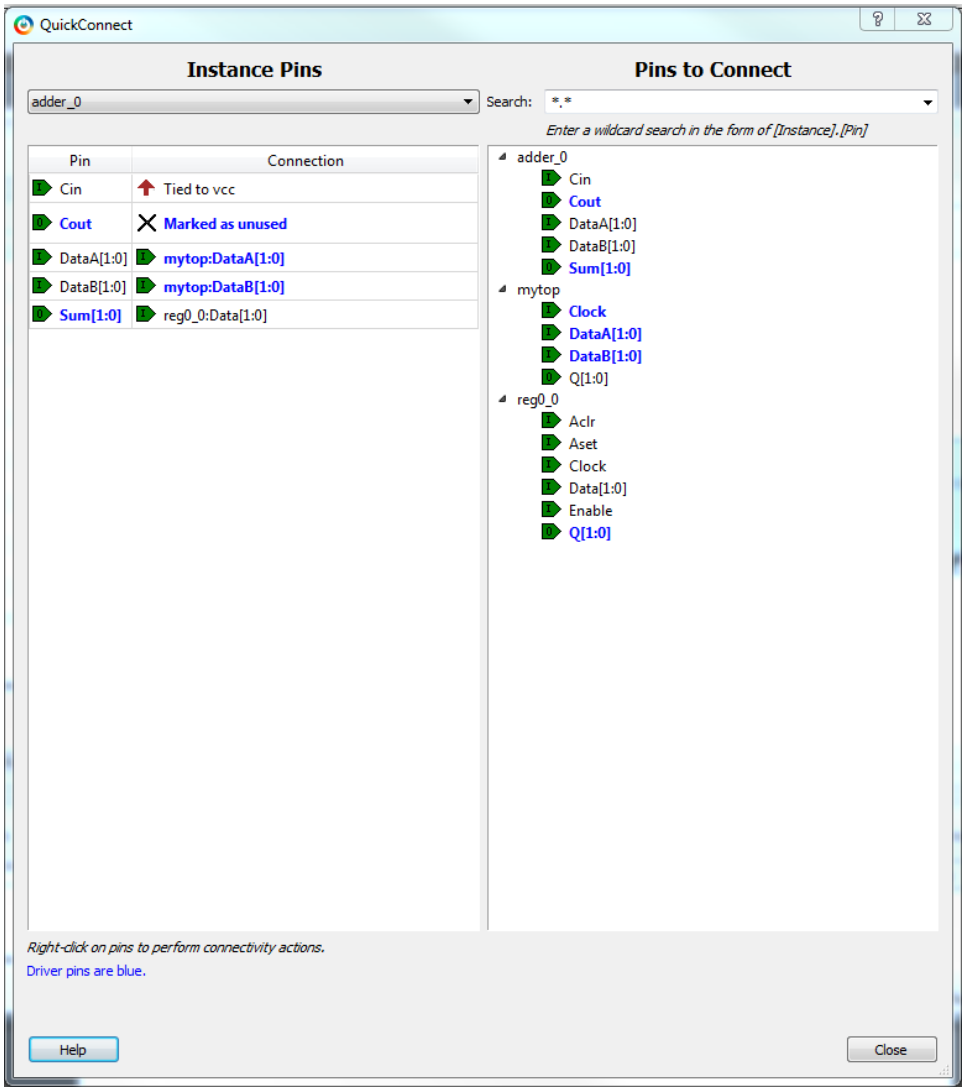


Рис. 3.11. Инструмент QuickConnect

Также можно выполнять все действия, описанные в таблице 3.2.

После окончания редактирования компонента SmartDesign его нужно сгенерировать нажатием кнопки . После этого будет сгенерировано его описание и он займёт своё место в Design Hierarchy (рис. 3.12).

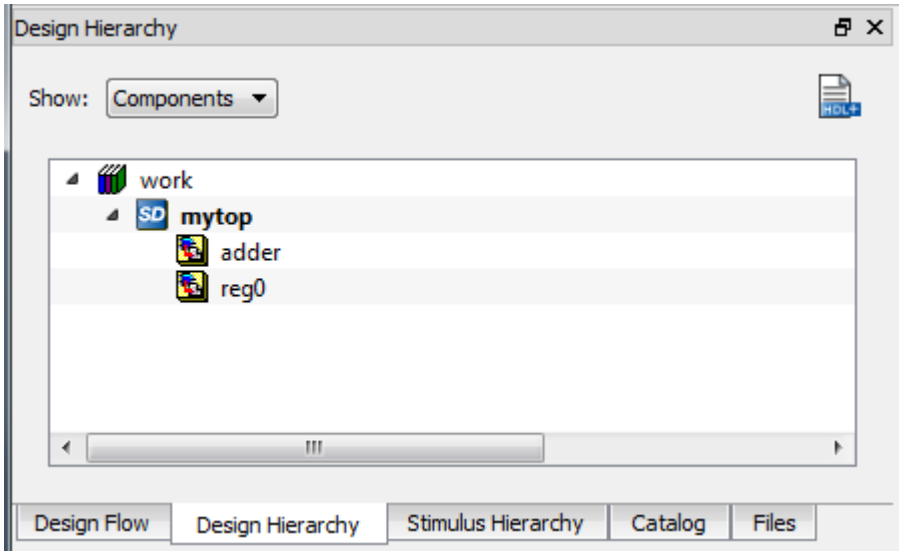


Рис. 3.12. Иерархия проекта после генерации компонента SmartDesign

При этом появятся соответствующие сообщения в областях Log и Message (рис. 3.13)

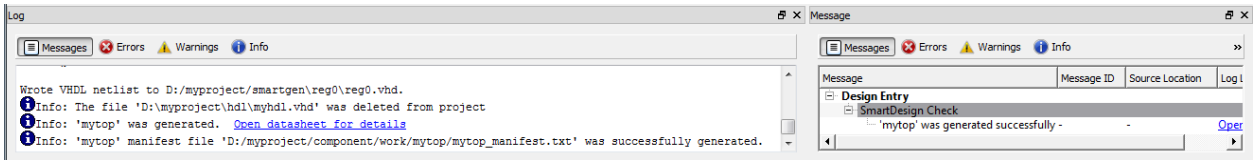


Рис. 3.13. Сообщения об успешной генерации компонента

В случае появления ошибок сообщение даст ссылку на соответствующий отчет с указанием причины отказа. Зачастую это неподключенные порты или не сгенерированные внутренние компоненты. Такие компоненты отмечаются символом и их необходимо обновить, нажав на них правой кнопкой мыши и выбрав Update Instance(s) with Latest Component.

3.3.2. HDL-редактор

Libero SoC имеет встроенный редактор исходных текстов на языках VHDL/Verilog с подсветкой синтаксиса и средствами управления вводом. Для создания нового HDL-файла можно сделать двойной щелчок по пункту Create HDL подгруппы Create Design в Design Flow (область окна 1) (рис. 3.14).

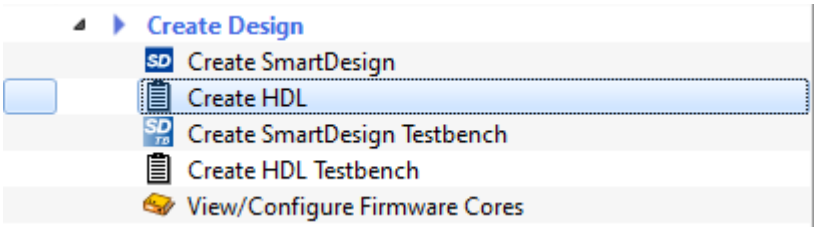


Рис. 3.14. Создание нового HDL-файла

После этого появится диалоговое окно создания нового файла (рис. 3.15).

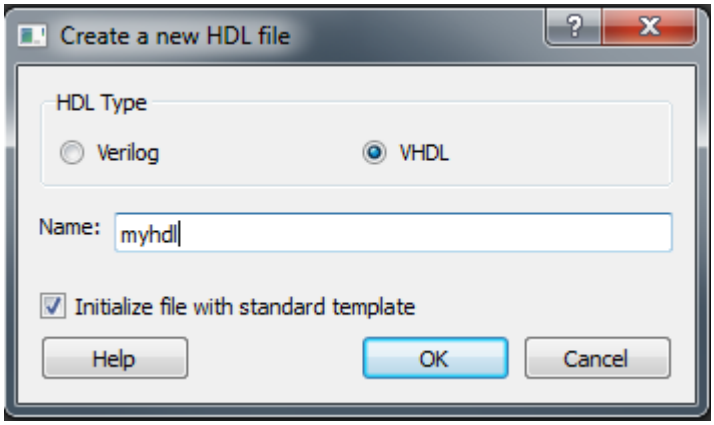


Рис. 3.15. Диалоговое окно создания нового HDL-файла

Предлагается выбор языка и ввод имени файла. Также имеется возможность снабдить описание стандартным шаблоном на выбранном языке (Initialize file with standard template), что ускорит процесс написания текста.

После нажатия ОК откроется редактор для написания текста (рис. 3.16).

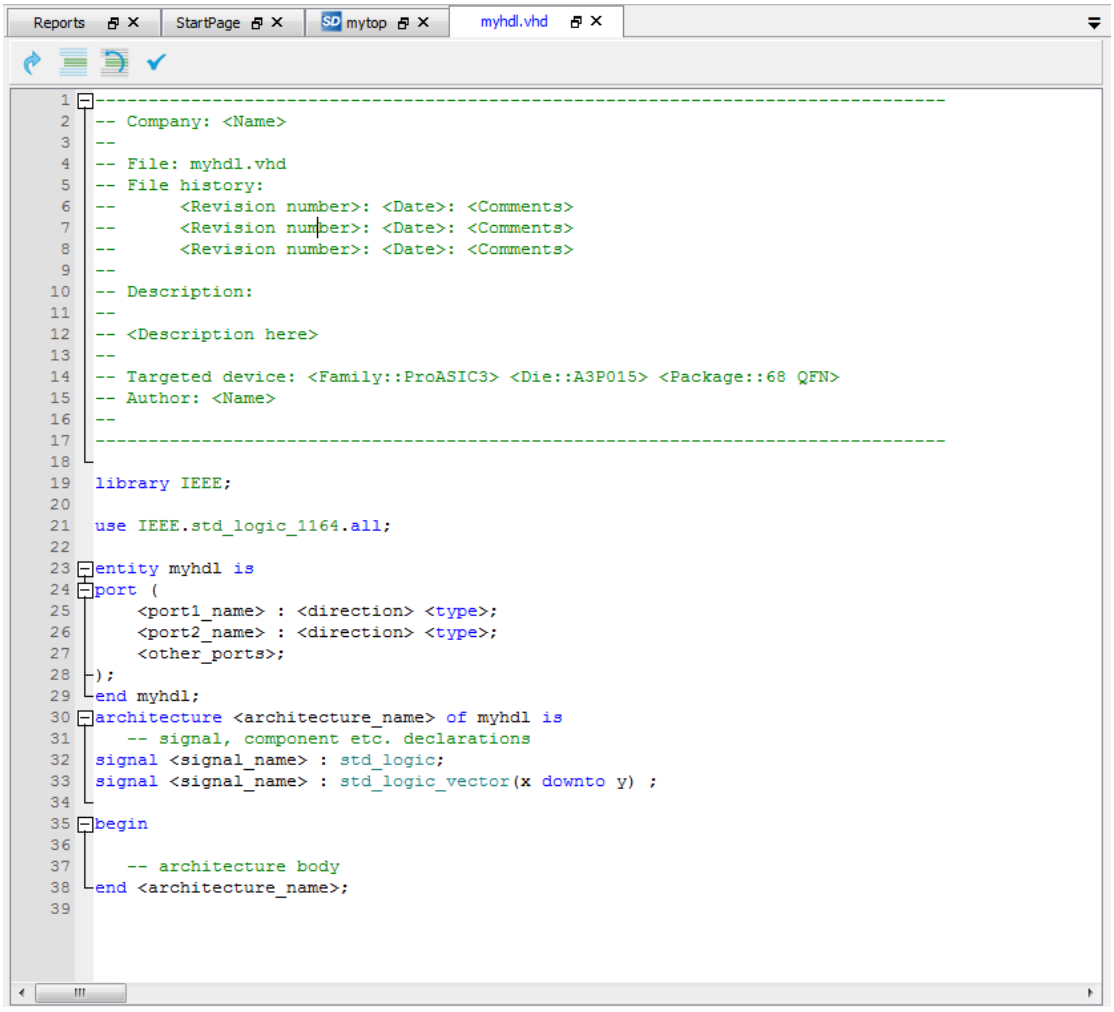


Рис. 3.16. HDL-редактор Libero SoC

В редакторе на рис. 3.16 уже содержится шаблон заголовка файла, подключена базовая библиотека и организованы стандартные объявления.

В процессе работы над описанием в редакторе разработчик может пользоваться инструментами, перечисленными в таблице 3.3.

Таблица 3.3. Инструменты HDL-редактора

	Описание	Назначение
	Go to line	Перейти к указанной в диалоговом окне строке текста
	Comment Selection	Закомментировать выделенное
	Uncomment Selection	Раскомментировать выделенное
	Check HDL file	Проверить описание на синтаксические ошибки

После каждого изменения файла следует проверять его на предмет синтаксических ошибок, нажав кнопку . В случае отсутствия ошибок среда известит об этом диалоговым окном (рис. 3.17).

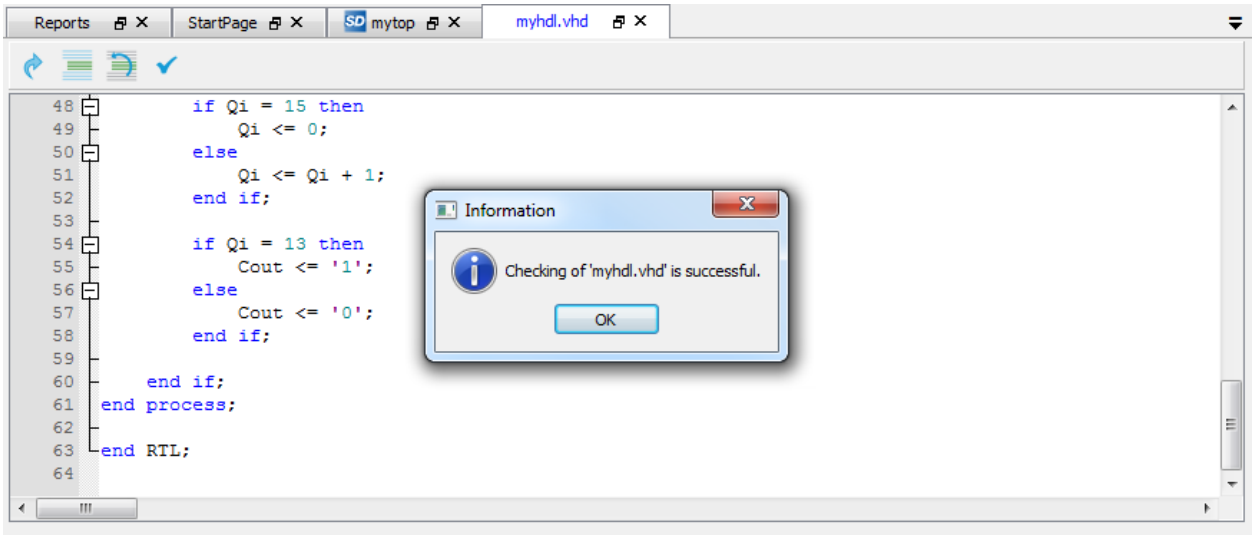


Рис. 3.17. Диалоговое окно при успешной проверке HDL-файла

В случае появления ошибок необходимо обратиться в поле сообщений, в котором будет сообщена причина и код ошибки.

3.3.3. Каталог IP-ядер

Со средой Libero SoC предоставляется мощный и полный каталог IP-компонентов, включающий наиболее популярные решения. Доступ к каталогу имеется в области главного окна 1 во вкладке Catalog (рис. 3.18).

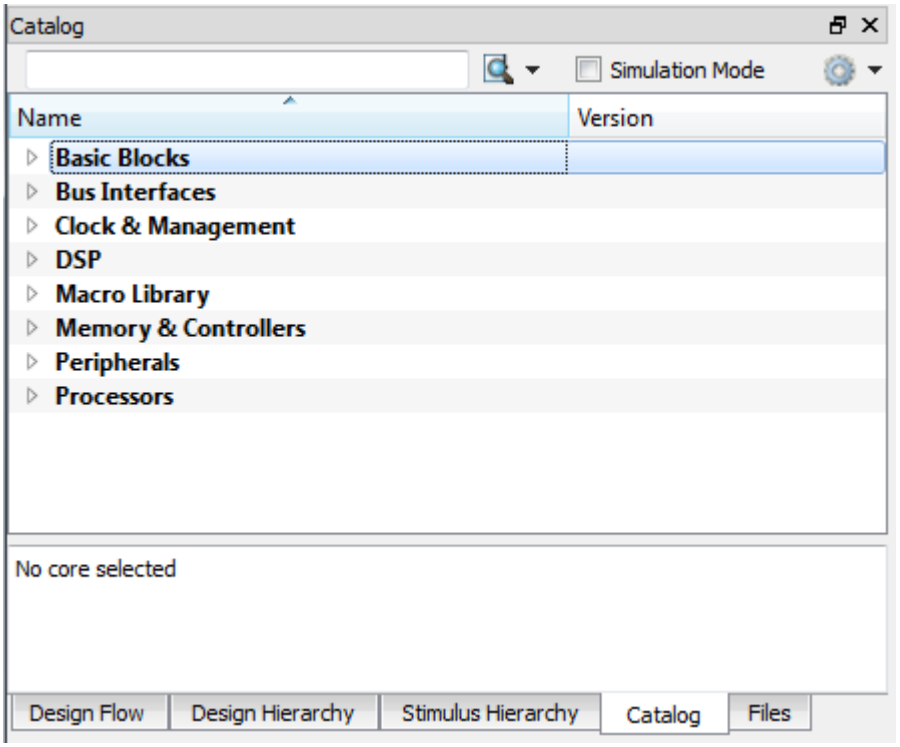


Рис. 3.18. Каталог IP-ядер

Состав каталога и доступные решения зависят от выбранного кристалла и его характеристик. В состав каталога входят следующие IP-ядра следующих категорий:

- Арифметические;
- Шины/интерфейсы;
- Тактирование и управление;
- Цифровая обработка сигналов;
- Ввод-вывод;
- Библиотека примитивов;
- Память;
- Периферийные модули;
- Процессоры.

Раскрыв определенную категорию, пользователь выбирает нужное ядро и перетаскивает его в рабочую область SmartDesign или инициирует его создание двойным щелчком. При этом открывается диалоговое окно конфигурирования выбранного ядра (рис. 3.19). Ядра также можно вставлять в виде компонентов в HDL-описания.

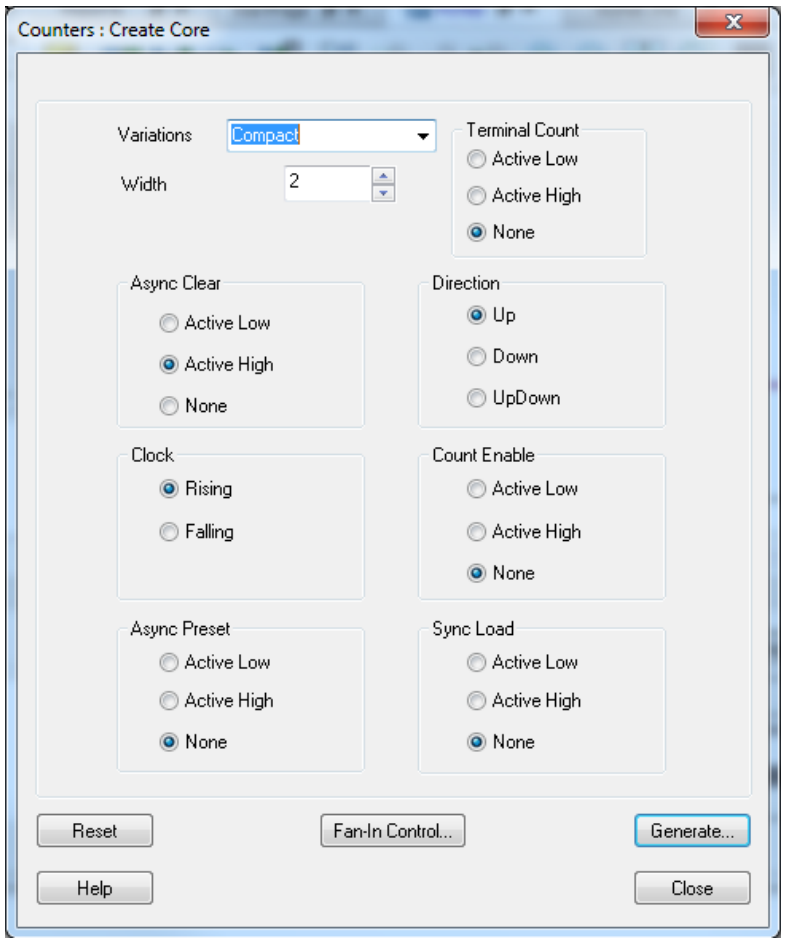


Рис. 3.19. Диалоговое окно конфигурирования счётчика

После настройки ядра и нажатия OK компонент появляется в Design Hierarchy и/или в рабочем поле SmartDesign (рис. 3.20).

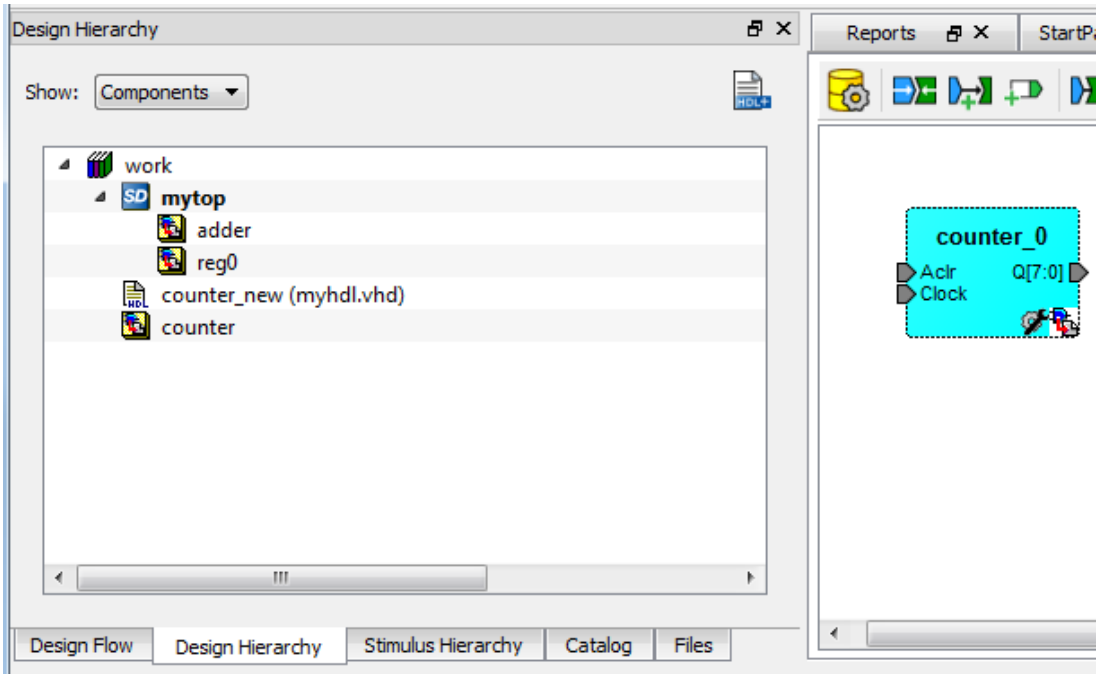


Рис. 3.20. Сгенерированный компонент-счётчик

Диалоговое окно конфигурирования IP-ядра можно вызвать в любой момент, щелкнув по нему два раза.

Каталог IP имеет режим симуляции (Simulation Mode), в этом режиме отображаются компоненты, используемые для функционального моделирования (рис. 3.21).

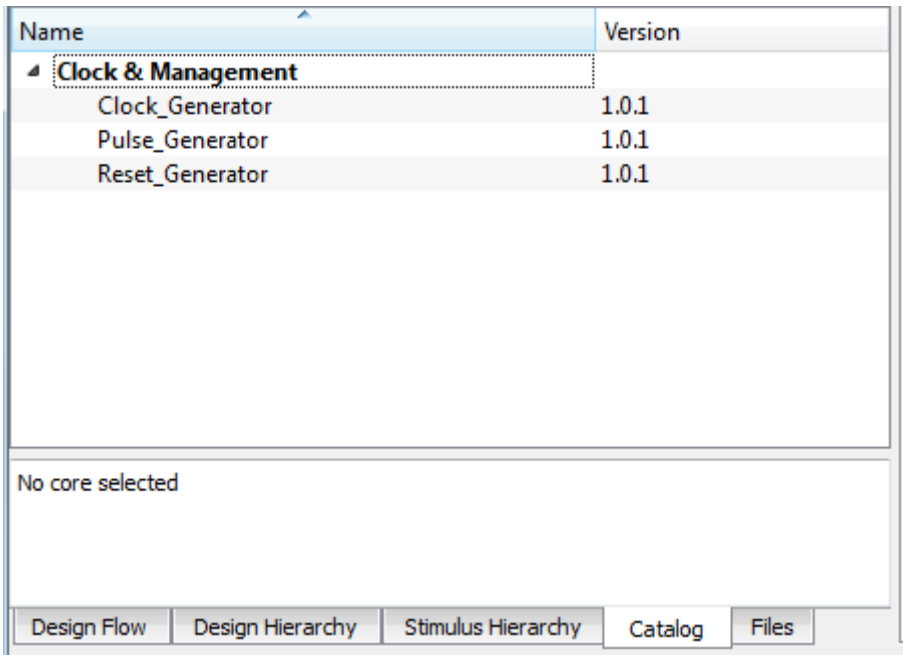


Рис. 3.21. Каталог IP режиме симуляции

В каталог можно импортировать IP-ядра сторонних разработчиков из файла специального формата. Для этого необходимо нажать и выбрать пункт Add Core to Vault. Это меню также содержит пункт обновления содержимого (Reload), а также вызов окна настроек каталога (Options) (рис. 3.22).

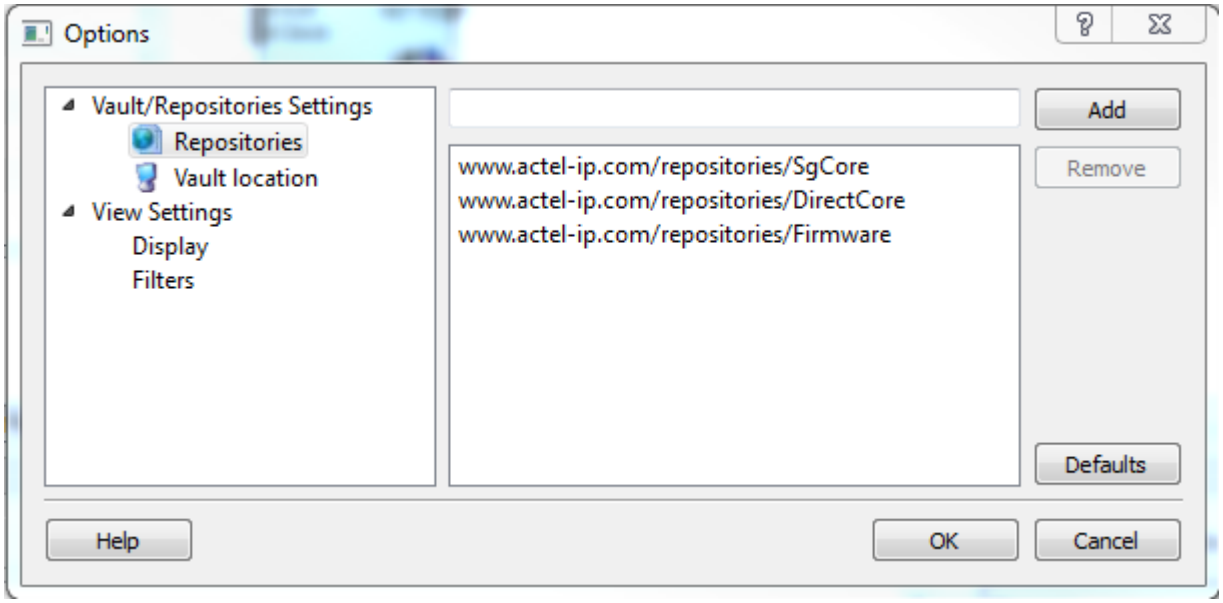


Рис. 3.22. Настройки каталога IP

В настройках можно выбрать адреса удаленных репозиториев обновления каталога (Repositories). Обновление происходит автоматически, среда извещает пользователя о необходимости замены устаревших ядер в проекте. Также можно указать расположение локального хранилища каталога (Vault Location). Также можно настроить внешний вид каталога.

3.4. Разработка устройств на ПЛИС SmartFusion2 и IGLOO2

3.4.1. Подсистемы MSS и HPMS

Разработка проектов для SmartFusion2 и IGLOO2 в Libero SoC имеет ряд особенностей в силу отличающейся архитектуры этих микросхем, а также наличия обособленных аппаратных подсистем Microcontroller Subsystem (MSS) и High-Performance Memory Subsystem (HPMS). Частично эти темы были затронуты в обзоре главы 2. Рассмотрим этот вопрос с позиций разработки в Libero.

Подсистема MSS (рис. 3.23) содержит процессор ARM Cortex-M3, в прикладной программе которого можно обращаться ко всем узлам подсистемы. Для написания встроенного программного обеспечения используется идущая в комплекте с Libero среда SoftConsole (см. главу 5). Все периферийные устройства подсистемы имеют уникальные диапазоны адресов в адресном пространстве матричной шины AHB, обратившись по которым разработчик получает доступ к регистрам периферийного устройства. Также, в адресное пространство внутренней шины включены два контроллера для взаимодействия с матрицей ПЛИС (Fabric Interface Controller, FIC).

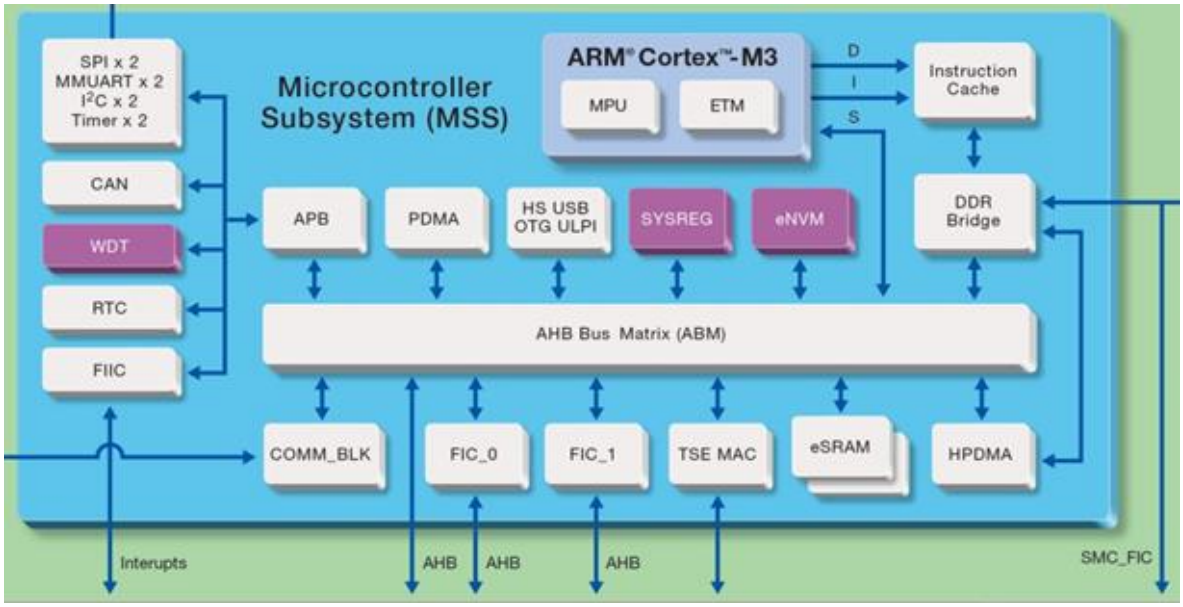


Рис. 3.23. Подсистема MSS

Контроллеры FIC могут быть настроены как master и slave и использовать интерфейсы APB и AHB. Всё это дает возможность гибко и эффективно распределить ресурсы между матрицей ПЛИС и аппаратной подсистемой.

Подсистема HPMS (рис. 3.24) отличается от MSS усеченным набором периферийных устройств, в том числе отсутствует процессор. Контроллеры FIC аналогичны MSS.

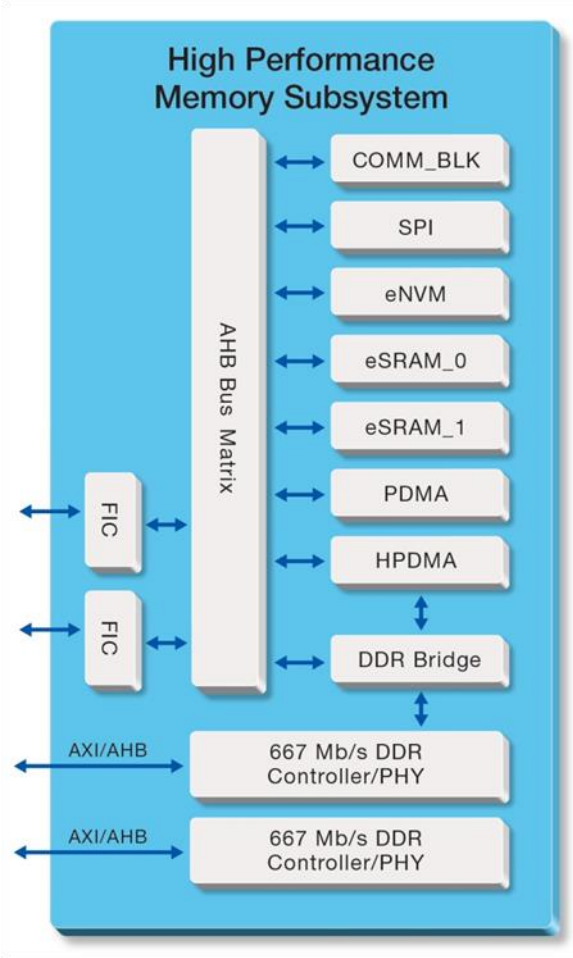


Рис. 3.24. Подсистема HPMS

Для использования MSS или HPMS в составе проекта, её нужно должным образом сконфигурировать. Одним из способов конфигурирования MSS и единственным способом конфигурирования HPMS является использование инструмента SystemBuilder.

3.4.2. Инструмент SystemBuilder

Встроенный в Libero SoC инструмент SystemBuilder является мощным и простым в использовании средством настройки сложных подсистем HPMS и MSS. Инструмент выполнен в формате мастера настройки и задаёт разработчику ряд простых вопросов о требуемой системе, её составе и свойствах. В конце работы инструмент автоматически генерирует соответствующий компонент, готовый к использованию в проекте.

Вызвать SystemBuilder можно двумя способами. Один из способов, когда проект уже создан, двойным щелчком по пункту System Builder в Design Flow. Также вызвать SystemBuilder можно сразу при создании проекта (рис. 3.25).

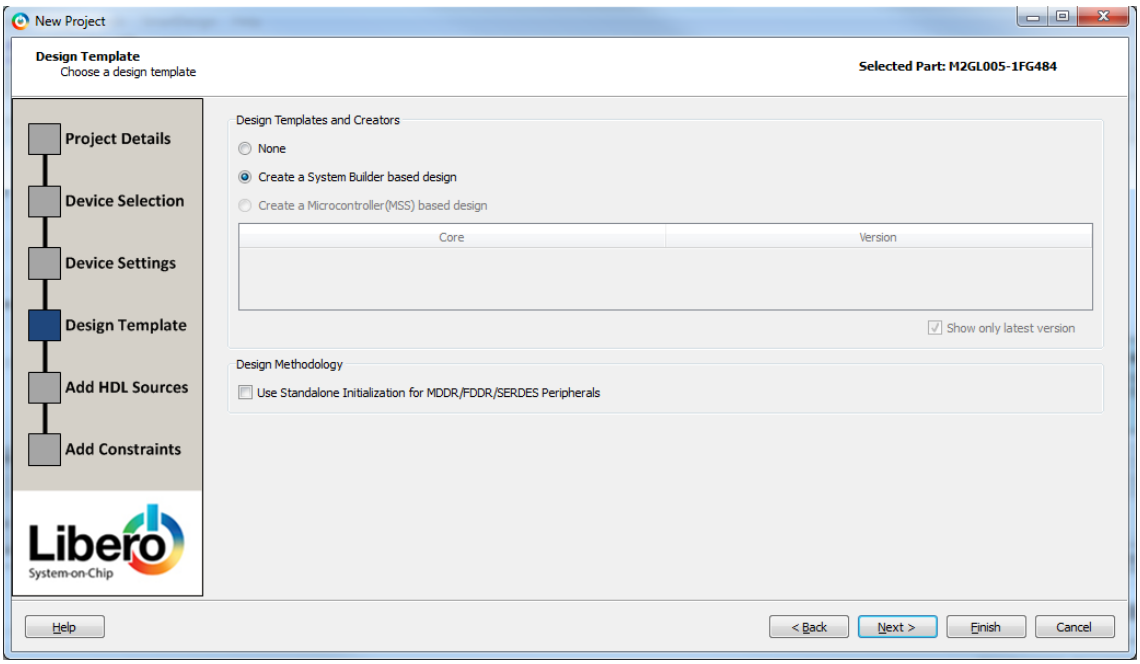


Рис. 3.25. Вызов SystemBuilder в процессе создания проекта

В любом случае появится диалоговое окно, в котором нужно ввести имя создаваемой подсистемы (рис. 3.26).

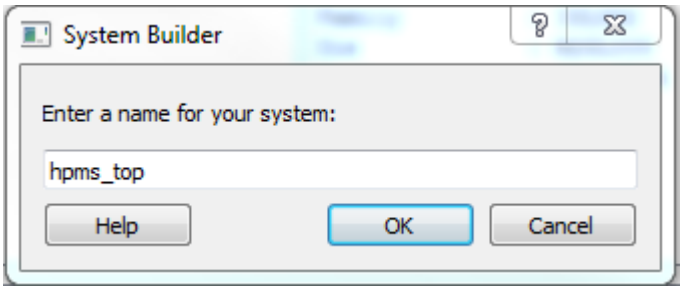


Рис. 3.26. Диалоговое окно ввода имени подсистемы

После нажатия ОК появляется главное окно SystemBuilder на первом шаге процесса конфигурирования. Первым делом предлагается выбрать необходимый состав и возможности подсистемы (рис. 3.27). В данном примере показано окно для микросхемы семейства IGLOO2.

Рисунок в правой части окна отражает структурную схему используемой ПЛИС. В ходе выбора необходимых составляющих, их символы будут менять цвет — синий означает активное состояние, и серый означает, что этот блок не будет включён в подсистему. То же справедливо и на следующих шагах.

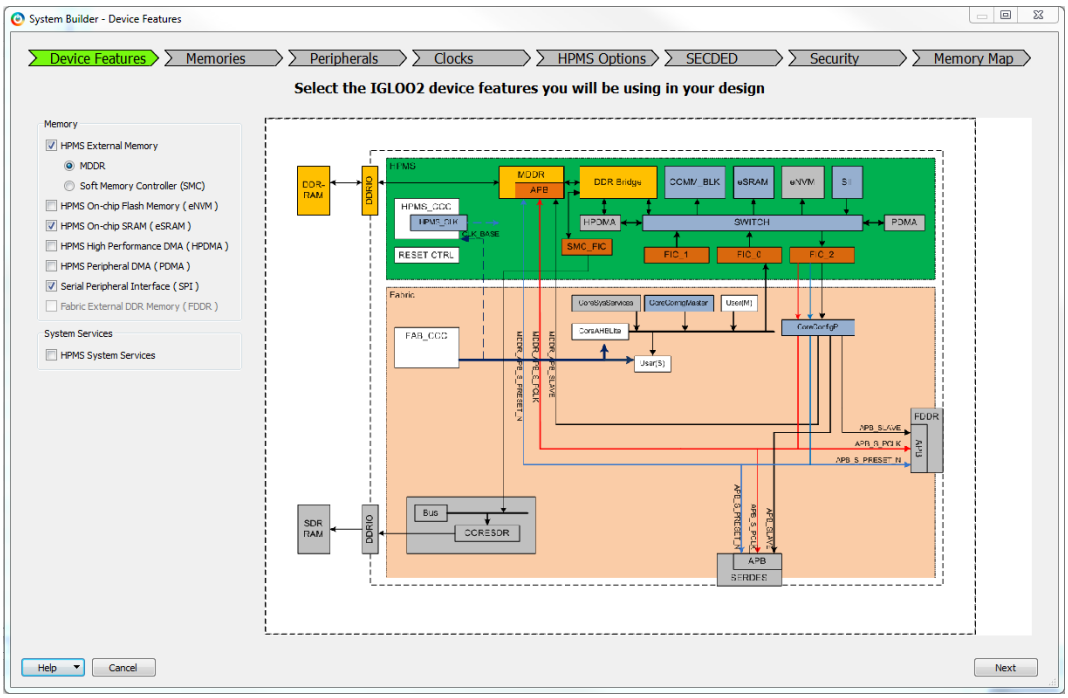


Рис. 3.27. Первый шаг SystemBuilder — состав подсистемы

Если в ходе выбора состава подсистемы были отмечены MDDR и/или FDDR и/или eNVM (Flash-память), то следующим шагом будет настройка соответствующих контроллеров и инициализации памяти (рис. 3.28).

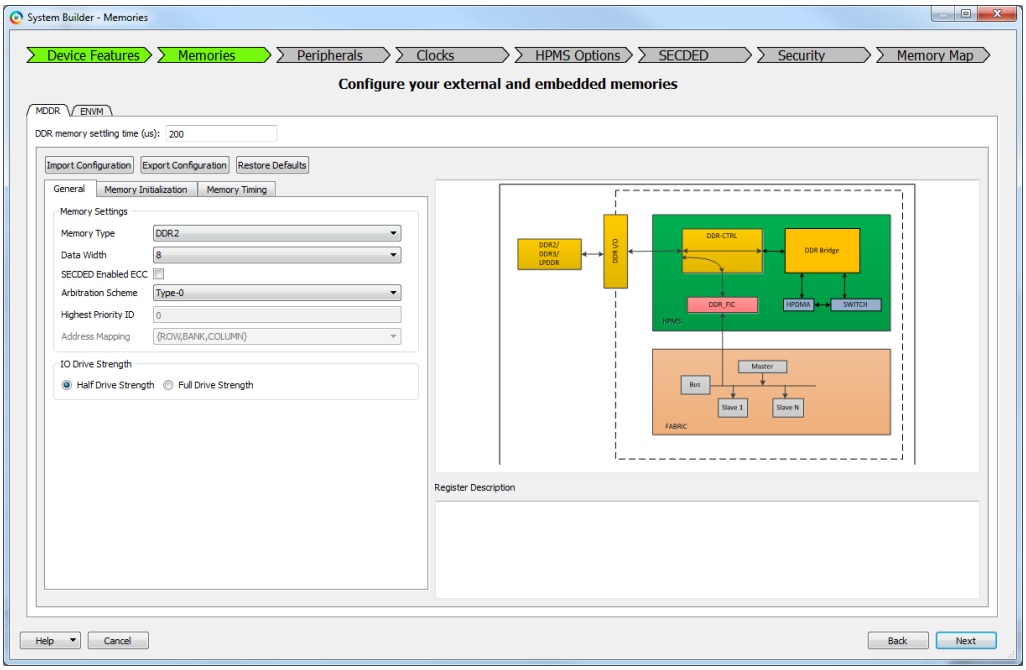


Рис. 3.28. Второй шаг (опционально) — настройка контроллеров и инициализации памяти

Следующим шагом разработчику предлагается добавить необходимые периферийные устройства, подключив их к шине через контроллеры FIC. Также можно добавить в FIC интерфейсные выводы APB/AHB master и slave (рис. 3.29). По умолчанию уже присутствуют интерфейсы, обеспечивающие работу составляющих, выбранных на шаге 1.

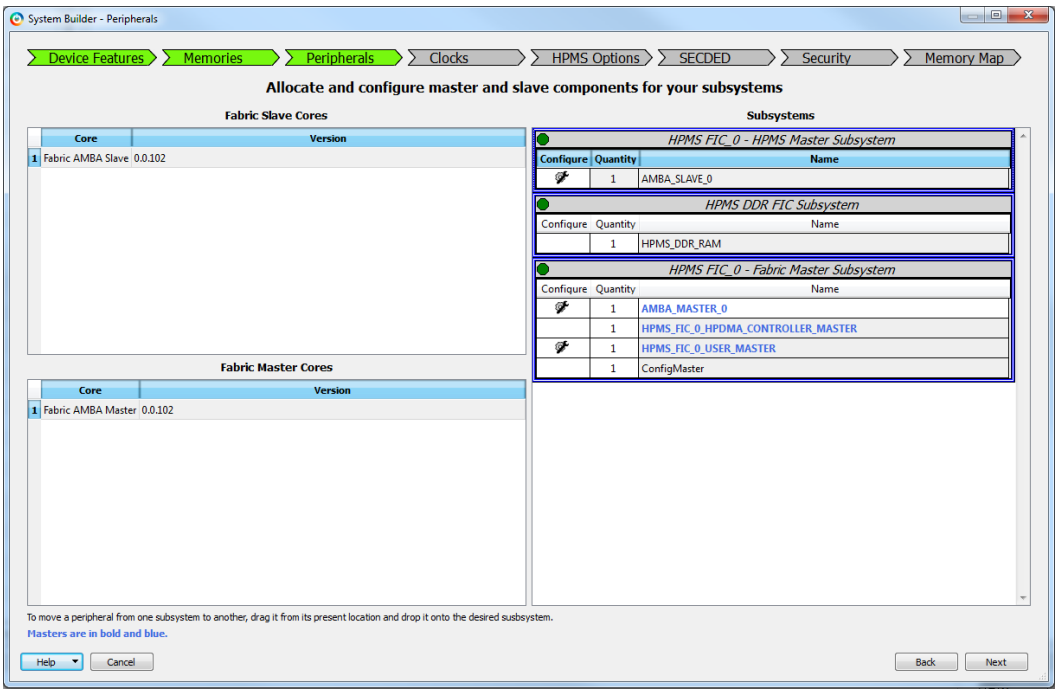


Рис. 3.29. Третий шаг — добавление периферийных устройств интерфейсов

Далее разработчик выбирает источники тактовых сигналов на кристалле (рис. 3.30). Среди возможных вариантов — встроенные источники тактового сигнала. На этой странице также можно настроить средства управления тактированием (CCC) и PLL в их составе, выбрать частоту тактового сигнала для HPMS, FIC и т.д.

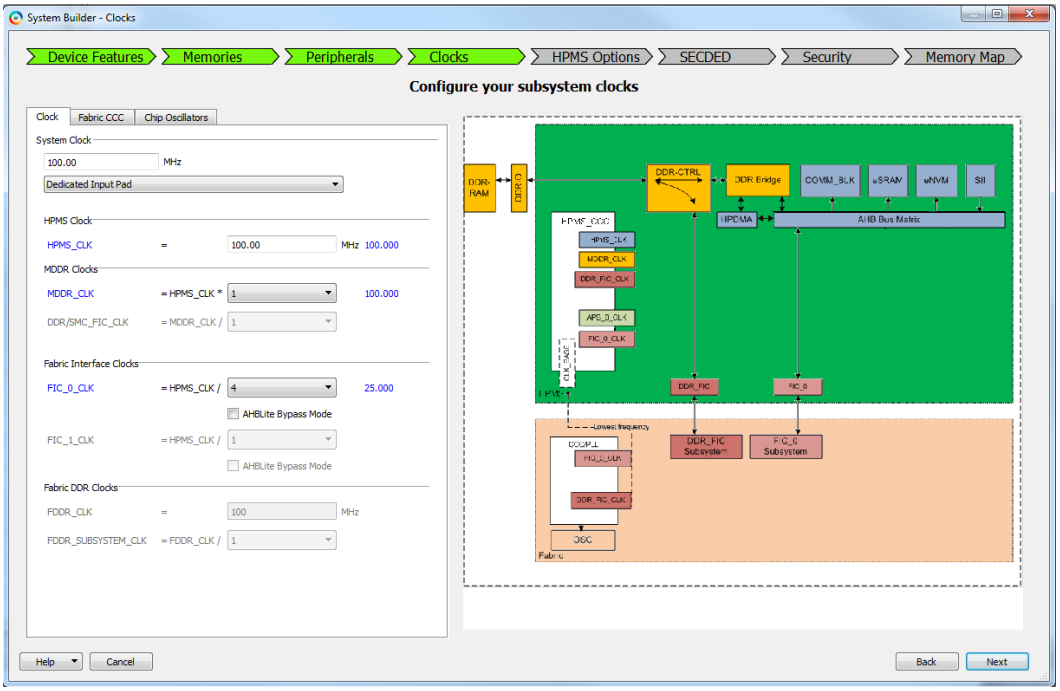


Рис. 3.30. Четвертый шаг — настройка тактовых сигналов и PLL

Следующим шагом разработчик может выставить настройки матричной шины AHB и моста DDR (рис. 3.31).

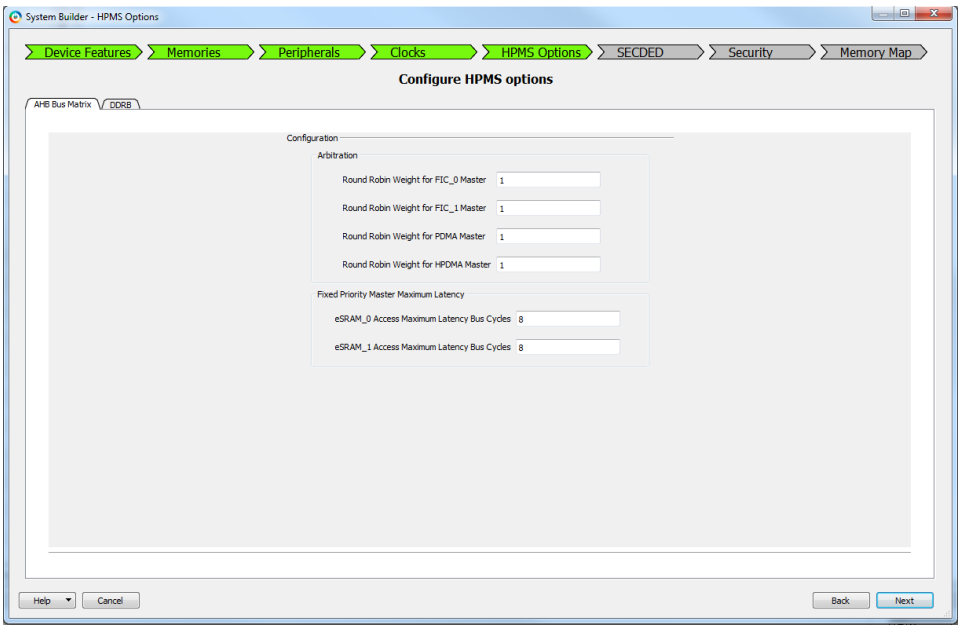


Рис. 3.31. Пятый шаг — настройки AHB Bus Matrix и DDR Bridge

Шестой и седьмой шаг — настройки кодов коррекции SECDED и опций безопасности. На финальной странице на экран выводится таблица распределения адресов памяти для создаваемой подсистемы (рис. 3.32).

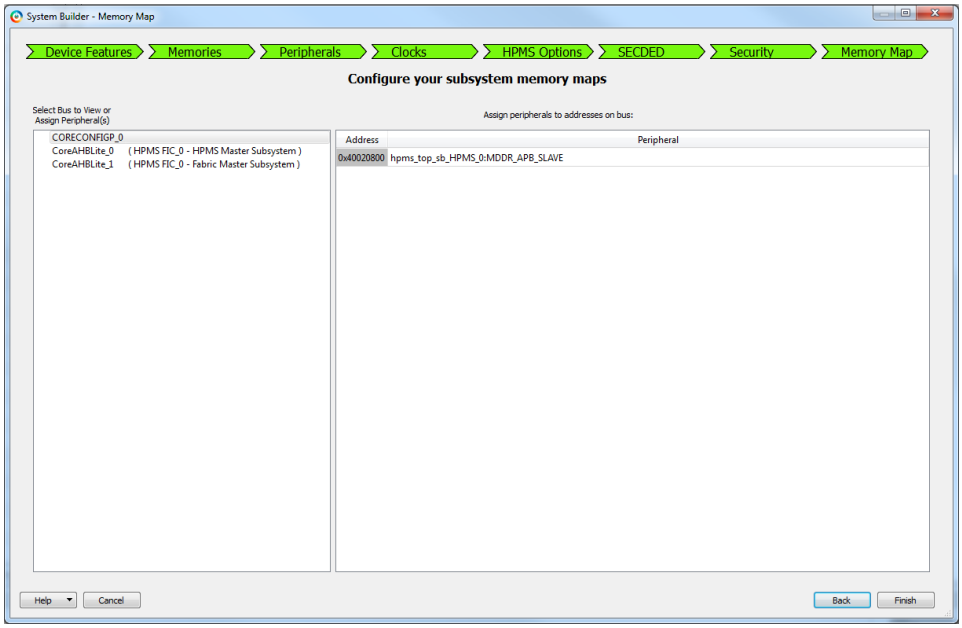


Рис. 3.32. Финальная страница — карта памяти создаваемой подсистемы

После нажатия Finish некоторое время Libero будет формировать требуемую подсистему, по окончании этого процесса в поле сообщений появится соответствующее оповещение, а в рабочем поле SmartDesign появится символ подсистемы с портами, к которым можно присоединять блоки в области матрицы ПЛИС (рис. 3.33).

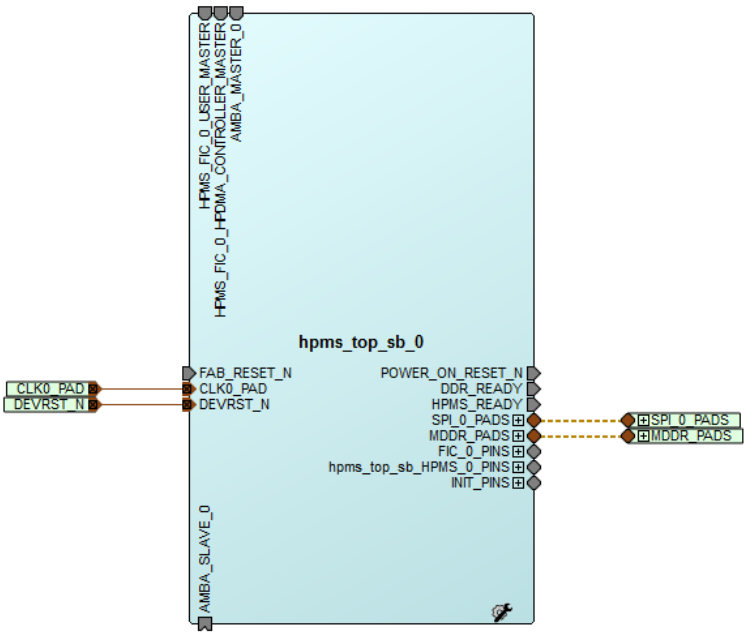


Рис. 3.33. Символ подсистемы HPMS, сконфигурированной пользователем

Полученный блок включает в себя ряд сгенерированных компонентов и IP-ядер. Некоторые порты уже имеют закрепленные назначения на выводах микросхемы, это аппаратный сброс, порты встроенных в HPMS аппаратных периферийных устройств.

3.4.3. Инструмент Configure MSS

Настройку подсистемы MSS SmartFusion2 можно проводить как в SystemBuilder, таким же образом, как описано в разделе 3.4.2, так и с помощью специального инструмента Configure MSS. Этот инструмент можно выбрать как при создании проекта, так и в Design Flow, секция Create Design. После открытия на экране появится диалоговое окно с выбором версии ядра подсистемы, списком документации и предложением ввести имя подсистемы (рис. 3.34).

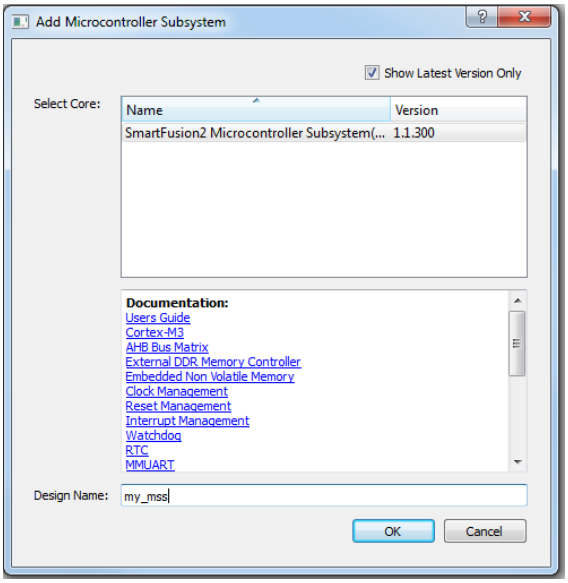


Рис. 3.34. Диалоговое окно создания новой MSS

В рабочем поле SmartDesign при этом сразу окажется символ подсистемы с настройками по умолчанию (рис. 3.35).

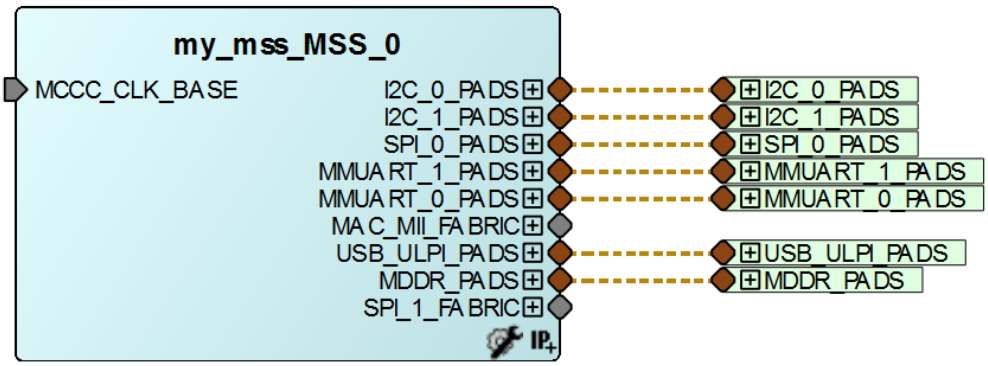


Рис. 3.35. Символ MSS с настройками по умолчанию

Двойным щелчком пользователь открывает вкладку настройки подсистемы (рис. 3.36).

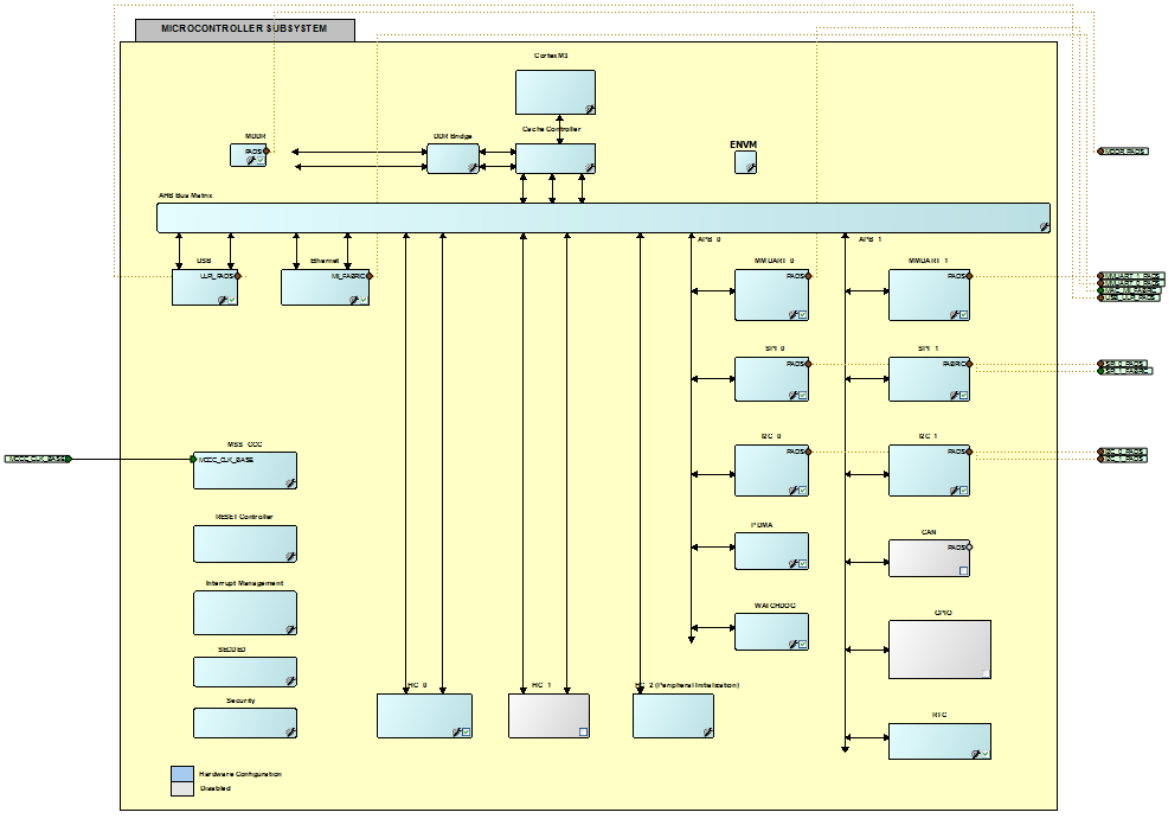


Рис. 3.36. Настройка подсистемы MSS

В поле данной вкладке содержится интерактивная структурная схема подсистемы MSS. В правом нижнем углу некоторых блоков имеется флажок, с помощью которого можно включать или исключать элементы из системы, синий цвет — включено, красный — исключено. Исключить можно не все элементы. Двойным щелчком на любом узле подсистемы пользователь вызывает настройку соответствующего узла (рис. 3.37).

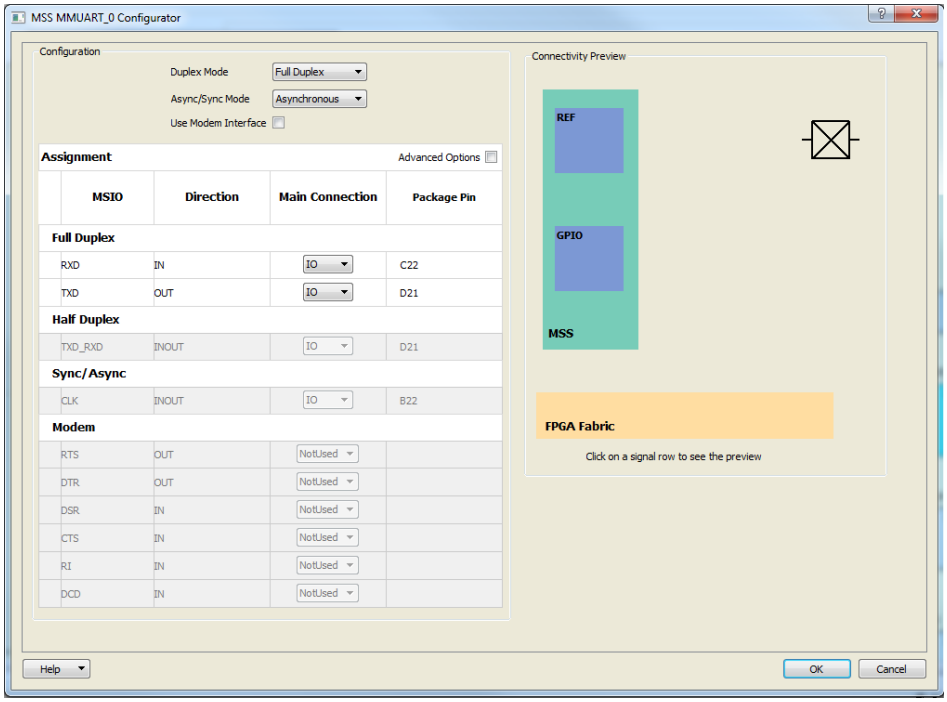


Рис. 3.37. Настройка периферийного узла UART в MSS

После выбора требуемых периферийных устройств необходимо нажать для сохранения изменений, при этом символ MSS в SmartDesign будет показан с восклицательным знаком . Обновив его, разработчик получает новый символ, соответствующий его подсистеме (рис. 3.38).

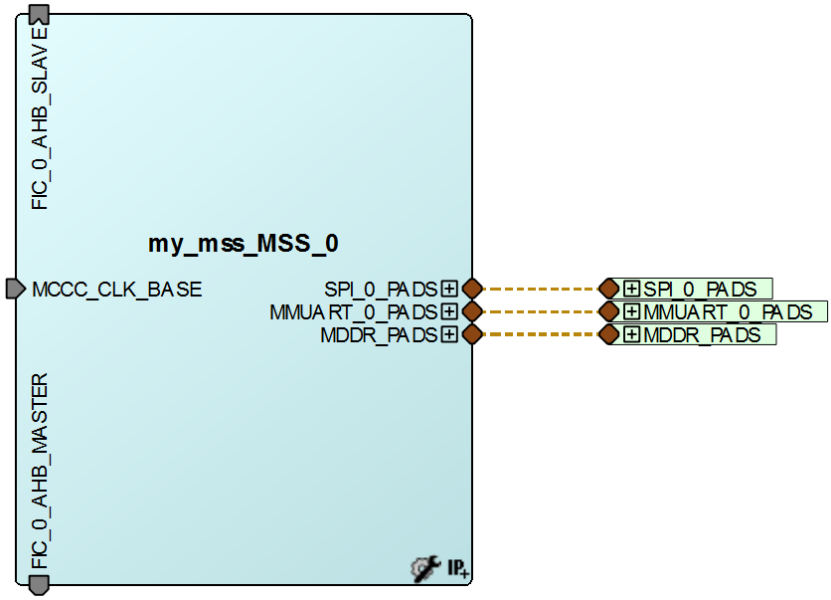


Рис. 3.38. Обновлённый символ MSS

Также, как и с символом, полученным с помощью SystemBuilder, далее разработчик подключает его порты к собственным блоком, формируя требуемый дизайн.

3.5. Синтез проектов в среде Libero SoC (Synopsys Sinplify)

Синтезатор Synopsys Synplify Pro ME, интегрированный в Libero SoC, служит для проведения логического синтеза и первичной оптимизации проектов для всех ПЛИС Microsemi. Как и остальные встроенные инструменты, запустить Synplify Pro ME можно из Design Flow, пункт Synthesize (рис. 3.39).

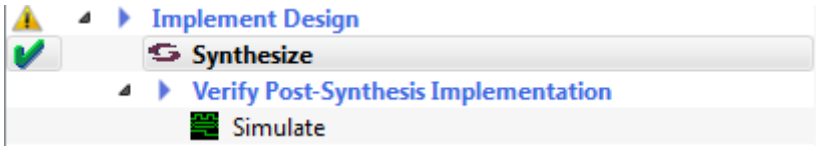


Рис. 3.39. Запуск Synplify из Design Flow

Данный этап выполняется в общем процессе Design Flow, формируя на выходе список соединений (Netlist) логических вентилей из описания поведенческой модели схемы на языке описания аппаратуры. Зачастую выполнение этапа автоматизировано и не требует от разработчика работы непосредственно со средой Synplify. При этом имеется возможность запуска

оболочки синтезатора и работы в ней. Для этого необходимо выбрать соответствующий пункт Open Interactively из выпадающего меню (рис. 3.40).

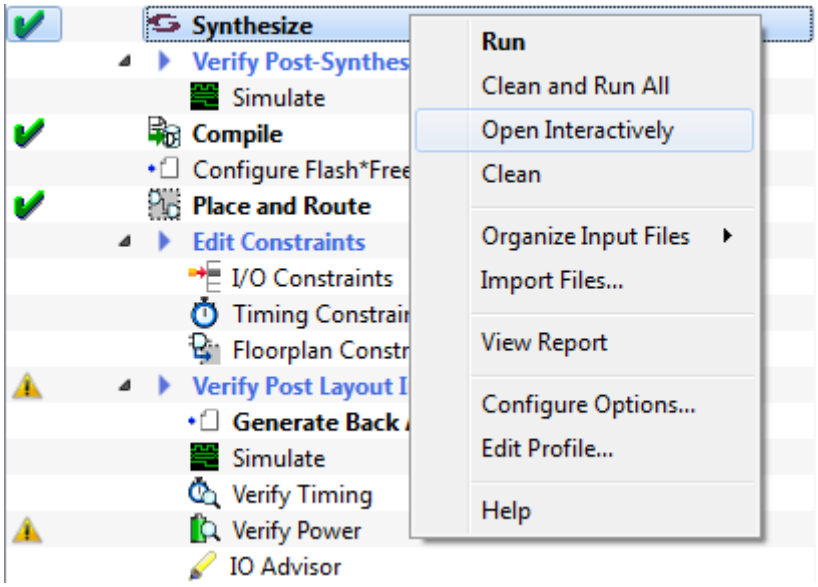


Рис. 3.40. Запуск Synplify в интерактивном режиме

После нажатия откроется окно Sinplify (рис. 3.41).

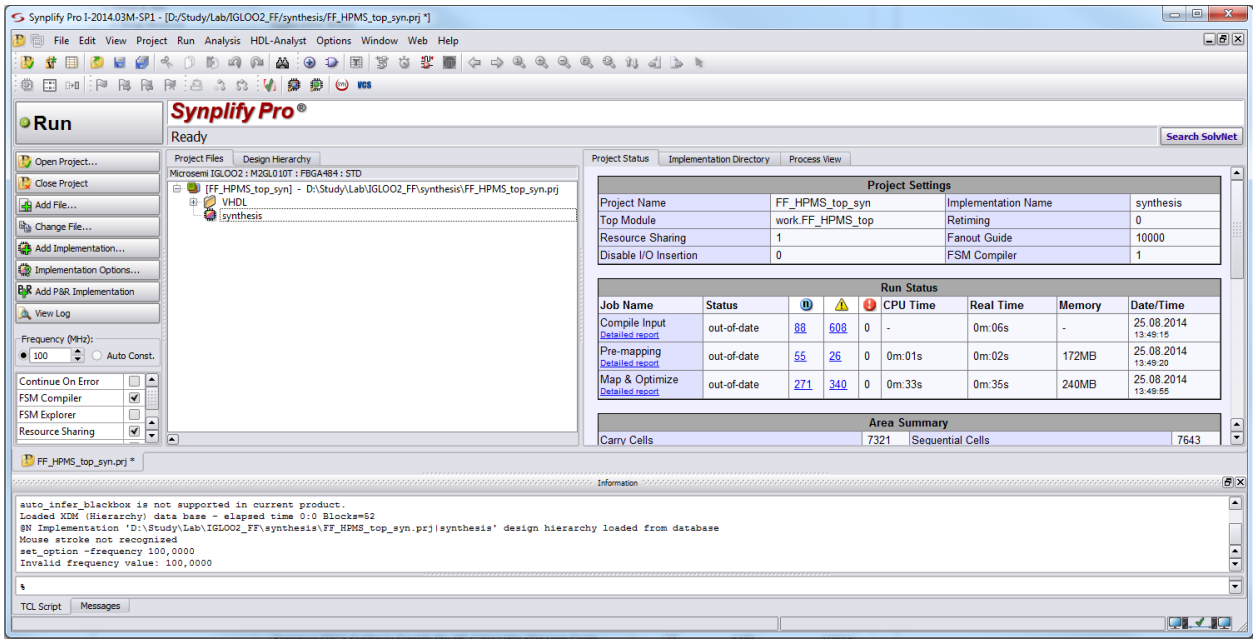


Рис. 3.41. Окно инструмента Synopsys Synplify

Подробнее о работе в оболочке синтезатора можно узнать по ссылке: http://www.microsemi.com/products/fpga-soc/design-resources/design-software/synplify-pro-me.

Для успешного и эффективного проведения синтеза необходимо соблюсти ряд требований. В первую очередь — определить файл временных ограничений для синтезатора (рис. 3.42).

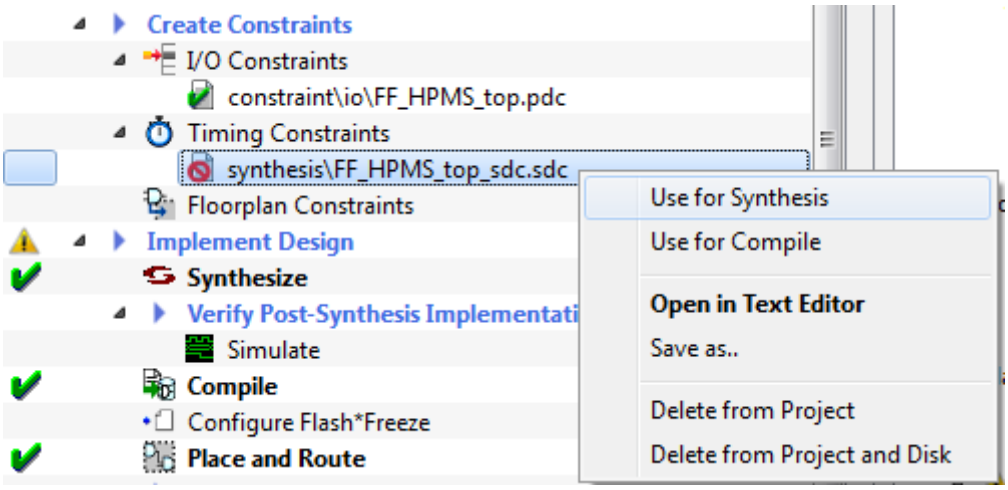


Рис. 3.42. Пометка файла временных ограничений, как используемого при синтезе

Импортировать файлы временных ограничений можно также из выпадающего меню (Import Files… на рис. 3.40).

Во-вторых, можно организовать входные файлы синтезатора и организовать их порядок (Organize Input Files на рис. 3.40). Пункт содержит два подпункта: Organize Source Files и Organize Constraints Files. Процесс настройки для обоих пунктов идентичен. В открывающемся при нажатии на одном из пунктов окне уже выставлены настройки по умолчанию (рис. 3.43).

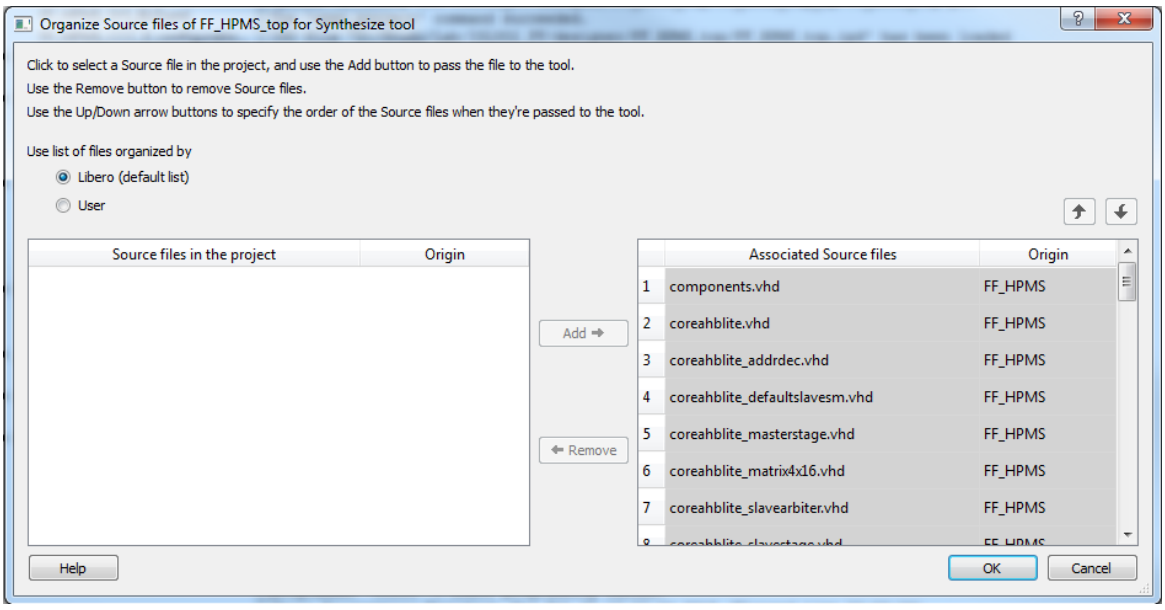


Рис. 3.43. Настройка входных файлов для Synplify

Активируя радиокнопку User разработчик получает возможность исключить из синтеза любой файл и включить нужные (кнопки Remove и Add), а также менять порядок файлов (рис. 3.44).

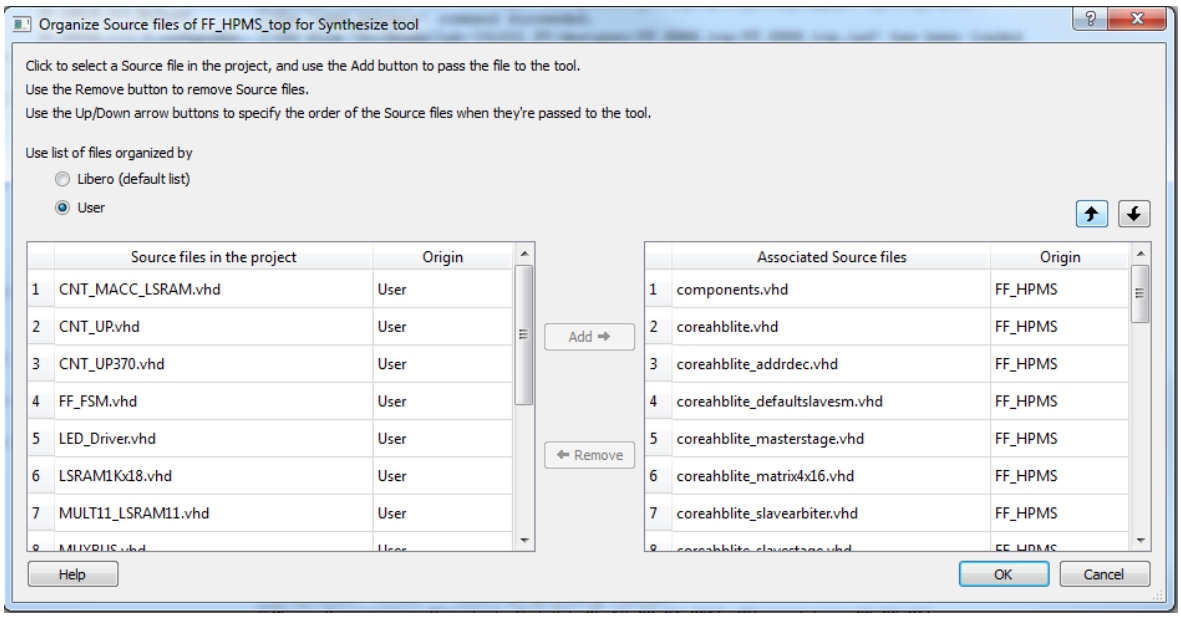


Рис. 3.44. Изменение состава файлов для синтеза и порядка их следования

После того, как все настройки произведены, и все файлы подключены, логический синтез можно выполнить отдельно двойным щелчком по пункту Synthesize. После того, как процесс закончится разработчик может просмотреть отчёт о работе инструмента (рис. 3.45).

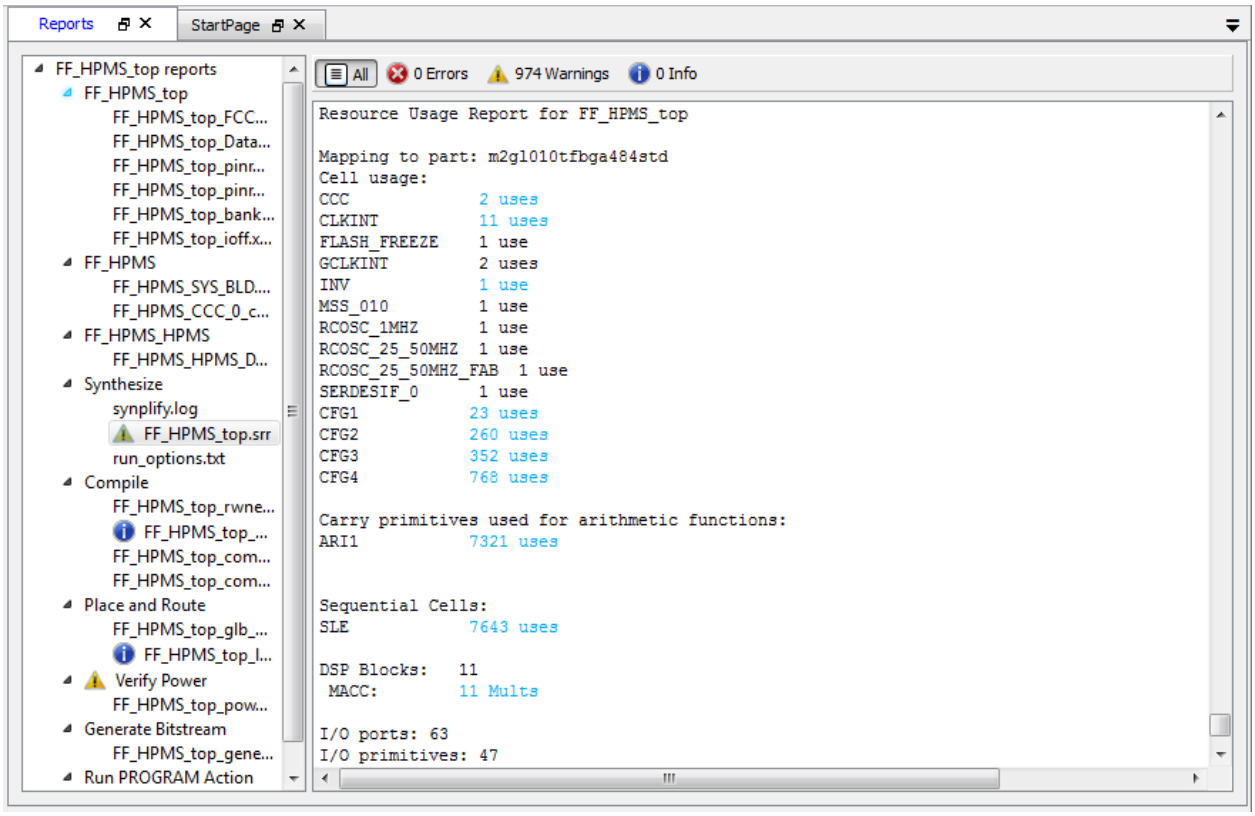


Рис. 3.45. Отчёт о работе Synplify

В отчёте будут содержаться все предупреждения и возможные ошибки, а также количество затрачиваемых ресурсов, пример которого показан на рис. 3.45.

3.6.Моделирование устройств в среде Libero SoC (Modelsim)

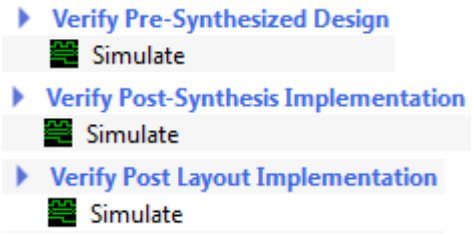
Симулятор Mentor Graphics ModelSim ME — мощная и гибкая система для функционального и временного моделирования, которая позволяет убедиться в работоспособности проекта на разных стадиях проектирования, описать тесты и задать необходимые тестовые воздействия и увидеть временные диаграммы функционирования устройства. Моделирование в Libero с использованием Modelsim делится на три этапа:

- поведенческое (pre-synthesis);
- структурное (post-synthesis);
- временное, динамическое (post-layout, back-annotated).

Поведенческая симуляция происходит на уровне исходного HDL-кода, данный этап позволяет предварительно проверить правильность логики работы устройства. Структурная симуляция может быть запущена после логического синтеза и происходит на уровне вентилей, при этом учитываются задержки на вентилях. И, наконец, временное моделирование запускается после размещения и разводки связей, на выходе которых формируется модель, максимально близкая к реальным условиям функционирования на выбранном кристалле.

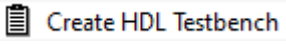
Настоятельно рекомендуется не игнорировать ни одну из этих стадий и тщательно подходить к вопросу тестирования и верификации вашей системы, в противном случае можно потерять много времени и средств на поиск ошибки на более поздних стадиях проектирования.

На вкладке Design Flow рабочей области 2 главного окна Libero за перечисленные этапы отвечают следующие пункты:



Двойной щелчок по любому из этих пунктов приведет к открытию Modelsim (по умолчанию в интерактивном режиме). Для старта любого этапа симуляции необходимо выполнить некоторую подготовительную работу.

Естественно, в первую очередь подготовить нужный проект и убедиться, что ошибок в HDL и SmartDesign файлах нет. После того, как проект готов к моделированию, необходимо создать специальный файл, описывающий тестовое окружение для модуля, который нужно проверить, а также набор входных воздействий для проведения тестирования. Такой файл называется testbench, тестбенч. Обычно тестбенч описывается с помощью того же языка описания аппаратуры, что и остальной проект. Отличие заключается в том, что данный файл имеет особый формат и не предназначается для синтеза. Независимо от языка тестбенч обычно несколько отличается от обычного HDL файла, в нём используется особые ключевые слова и он подразумевает последовательное исполнение инструкций.

Libero позволяет создать тестбенч двумя способами: описание на HDL (

) и создание тестового окружения с использованием уже описанного выше

инструмента SmartDesign (Create SmartDesign Testbench). Рассмотрим второй вариант чуть подробнее.

Двойной щелчок мыши инициирует уже описанную процедуру создания нового файла SmartDesign. Необходимо ввести имя нового компонента и нажать OK.

Замечание: существует правило именования тестбенчей, которое упрощает понимание структуры проекта, следует называть его по имени модуля, тестирование которого он осуществляет с постфиксом _tb, например, top_tb.

Также, в разделе 3.3.3 был описан режим Simulation каталога IP-ядер. С помощью этих инструментов, привычным способом создаётся простое тестовое окружение для верхнего уровня системы (рис. 3.46).

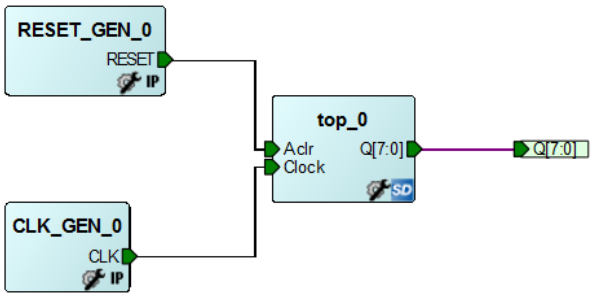


Рис. 3.46. Тестбенч, созданный с помощью SmartDesign

При создании тестбенча его структуру можно наблюдать так же, как и структуру проекта во вкладке Stimulus Hierarchy рабочей области 2 главного окна Libero (рис. 3.47).

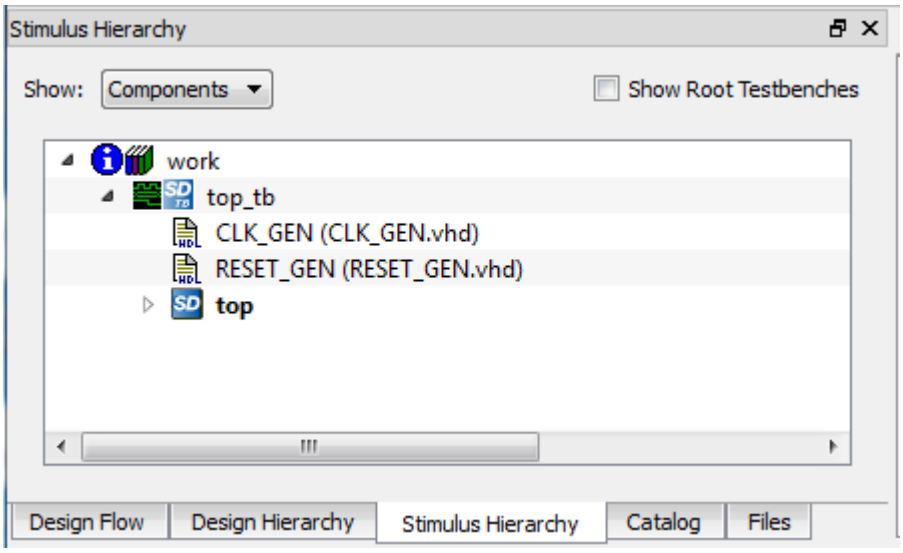


Рис. 3.47. Структура тестового окружения для модуля top

Структура SmartDesign автоматически транслируется в HDL-файл с таким же названием, который поступает на вход симуляции. Доступ к нему открыт через выпадающее меню по щелчку правой кнопки мыши на верхнем уровне иерархии (Open HDL File, рис. 3.48).

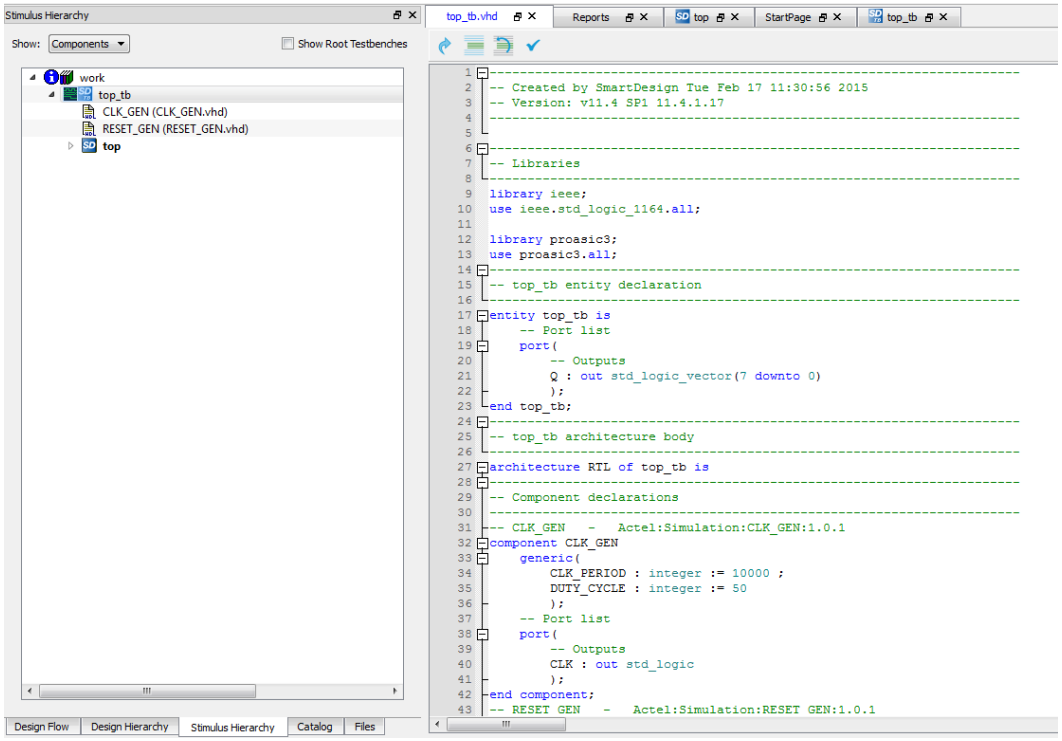
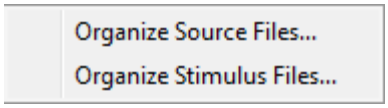


Рис. 3.48. Текстовое представление тестбенча

Стоит потратить время на изучении структуры этого файла, принципов построения тестового окружения для симуляции и используемых ключевых слов, иначе невозможно провести грамотное и результативное моделирование.

Теперь обратим внимание на настройки симуляции. Правый щелчок мыши по пунктам Simulate в Design Flow вызывает выпадающее меню, в котором присутствуют пункты, схожие с таковыми для синтеза (см. рис. 3.40). Остановимся на пункте Organize Input Files, которые делится на две подкоманды:



При нажатии открывается окно подобное описанному в предыдущем разделе (см. рис. 3.43,3.44). Organize Source Files отвечает за организацию и порядок исходных файлов проекта, которые компилируются *отдельно и независимо* внутри среды Modelsim. Иногда имеет смысл расставить файлы в порядке, отличном от установок по умолчанию. Organize Stimulus Files отвечает за состав и порядок файлов тестового окружения (все эти файлы отображаются в Stimulus Hierarchy). Здесь также можно расставить входные файлы по приоритету в случае возникновения каких-либо специфических проблем.

Часть настроек симуляции производится в окне Project Settings, вызываемом из меню Project. В этом окне в частности можно задать настройки, относящиеся к автоматизации симуляции. Симуляция инициализируется и может выполняться с помощью специального файла с расширением *.do. Этот файл содержит код на языке Tcl и выполняется интерпретатором, встроенным в Modelsim. В первую очередь это файл run.do, содержащий инициализирующий код, подключающий нужные библиотеки, компилирующий исходные файлы и подготавливающий временную диагарамму. В окне Project Settings, пункт Simulation Options -> DO File по умолчанию стоит галочка Use automatic DO file (рис. 3.49), что означает, что среда автоматически сгенерирует инициализирующий скрипт, чего в некоторых случаях достаточно.

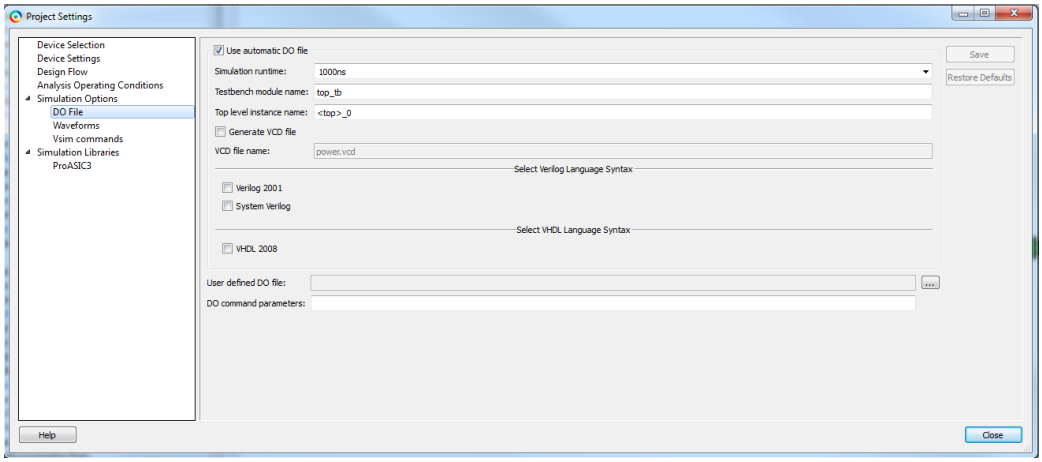


Рис. 3.49. Настройки, связанные с инициализирующим DO файлом

Альтернативой этому является создание собственного инициализирующего DO файла (указывается в User defined DO file), что поощряется для более опытных разработчиков.

В следующем пункте (Waveforms) можно настроить вывод сигналов на временную диаграмму по умолчанию или подключить пользовательский DO file – wave.do, в котором описаны дополнительные действия по инициализации временной диаграммы и описан процесс автоматической симуляции.

После создания части проекта, подлежащей симуляции и тестового окружения, а также по необходимости произведя все вышеописанные настройки, пользователь может запустить симуляцию в Modelsim. Для этого нужно выбрать требуемый этап симуляции и сделать двойной щелчок на пункте Simulate (либо щёлкнуть правой кнопкой мыши и выбрать Open Interactively). Если ошибок не допущено, откроется главное окно Modelsim (рис. 3.50).

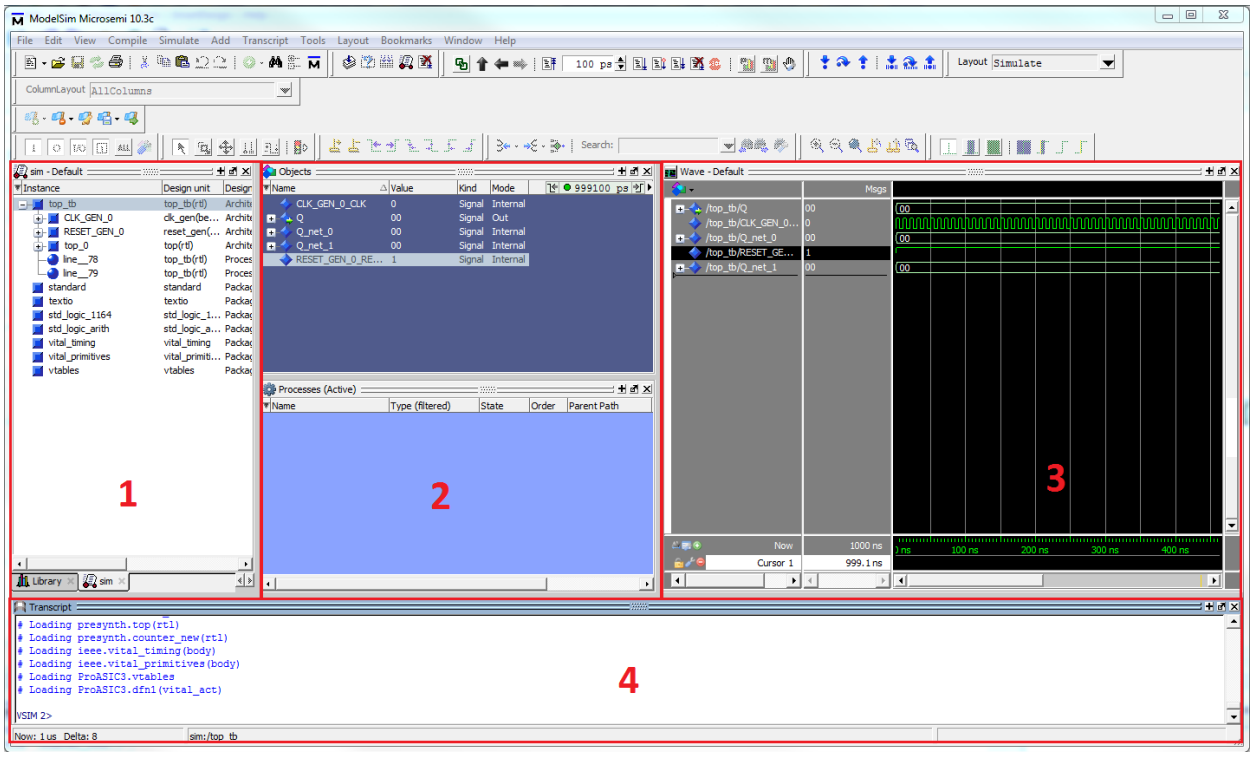


Рис. 3.50. Главное окно Modelsim

Окно делится на несколько рабочих областей:

- Область 1 – иерархия симуляции и используемые библиотеки;
- Область 2 – верхняя часть содержит объекты (сигналы, переменные) выбранного слева блока, нижняя часть показывает процессы выбранного блока;
- Область 3 – временные диаграммы, здесь составляется список сигналов для наблюдения и воздействия, сюда же выводятся результаты моделирования
- Область 4 – командная строка для работы с командами Tcl и вывод сообщений лога событий.

По умолчанию в область 3 выведена часть сигналов, относящаяся к верхнему уровню иерархии симуляции. Имена сигналов здесь имеют иерархическую структуру, то есть представляют собой полный путь к сигналу по иерархии симуляции. Раскрывая детальную структуру можно добавлять для наблюдения или воздействия любые из существующих объектов способом drag-and-drop (рис. 3.51).

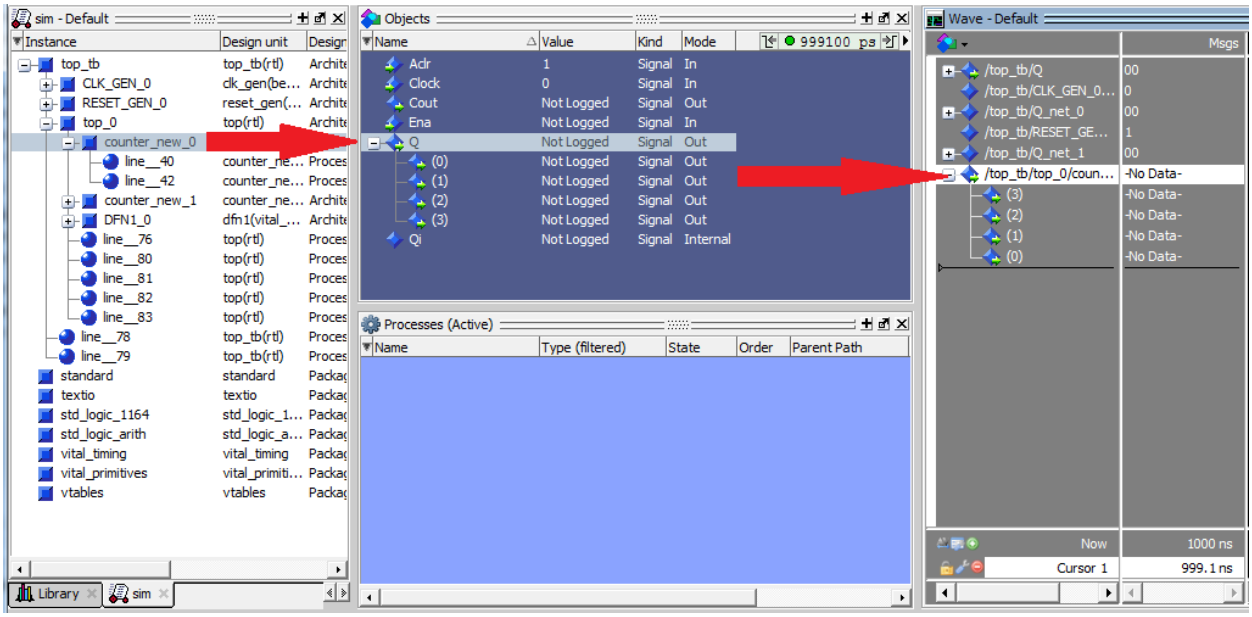


Рис. 3.51. Добавление произвольных объектов на временную диаграмму

Все выполняемые с помощью графического интерфейса действия имеют свой эквивалент в виде Tcl команды и отображаются в области 4 (рис. 3.52).

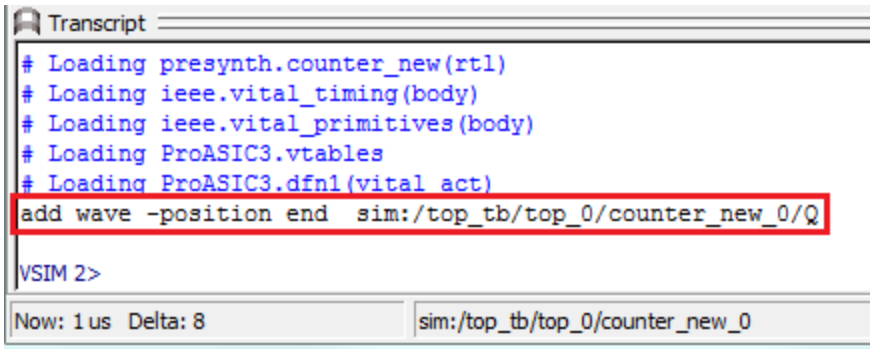
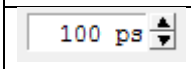
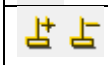
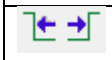
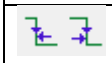
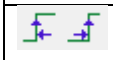


Рис. 3.52. Отображение совершенного действия в области командной строки

Эти команды могут быть описаны в пользовательском wave.do файле, либо введены в командной строке последовательно, в любом случае их выполнит интерпретатор.

Панель быстрого доступа в главном окне Modelsim имеет как привычные кнопки Открыть, Сохранить, Печать и т.д., так и важные органы управления процессом симуляции. Основные средства управления перечислены в таблице 3.4.

Таблица 3.4. Основные средства управления симуляцией в Modelsim

	Описание	Назначение
	Перезапуск симуляции	Возвращение пространства симуляции к исходному состоянию (выполнение инициализации)
	Задание времени симуляции	Поле ввода времени симуляции
	Запуск	Запуск симуляции на указанное время
	Продолжение	Продолжение прерванной симуляции
	Запуск без ограничения по времени	Запуск «бесконечной» симуляции
	Прервать	Прервать текущий процесс
	Остановить	Остановить симуляцию
	Добавить/удалить курсор	Добавить на временную диаграмму интерактивный курсор
	Найти предыдущее/следующее изменение сигнала	Выставить курсор на временную метку изменения сигнала относительно текущего местоположения
	Найти предыдущий/следующий спад сигнала	Выставить курсор на временную метку спада сигнала относительно текущего местоположения
	Найти предыдущий/следующий фронт сигнала	Выставить курсор на временную метку фронта сигнала относительно текущего местоположения
	Масштабирование временной диаграммы	Изменение масштаба временной диаграммы

После запуска симуляции временная диаграмма отобразит её ход или результат после завершения, в данном примере — работу счётчика (рис. 3.53).

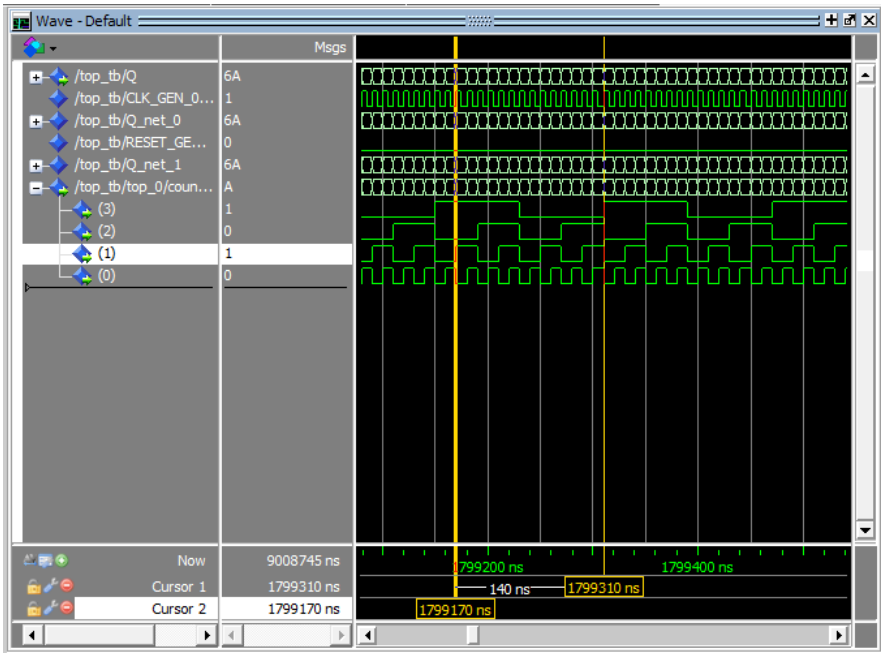


Рис. 3.53. Результат выполнения симуляции

На рисунке 3.53 видны добавленные курсоры, которые очень удобно использовать для анализа полученной диаграммы. Они показывают временную метку, на которой стоят, а также разницу во времени с предыдущим и следующим курсором. С помощью описанных в табл. 3.4 инструментов можно выставлять курсоры ровно по определенному перепаду сигнала.

Свернутые объекты на диаграмме со значком + означают группу сигналов или шину. Её можно разворачивать для детального анализа, а также просматривать её значение в разных форматах (например, в шестнадцатеричном, рис. 3.54).

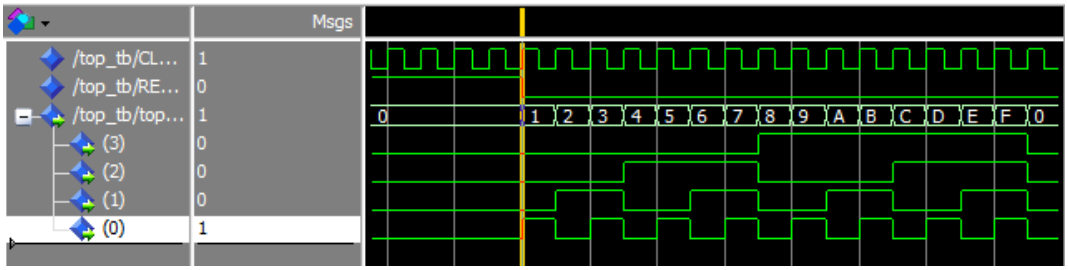


Рис. 3.54. Просмотр значения шины в шестнадцатеричном формате

Для смены формата можно выбрать из выпадающего меню по правому щелчку мыши на объекте Radix и выбрать формат из списка (рис. 3.55).

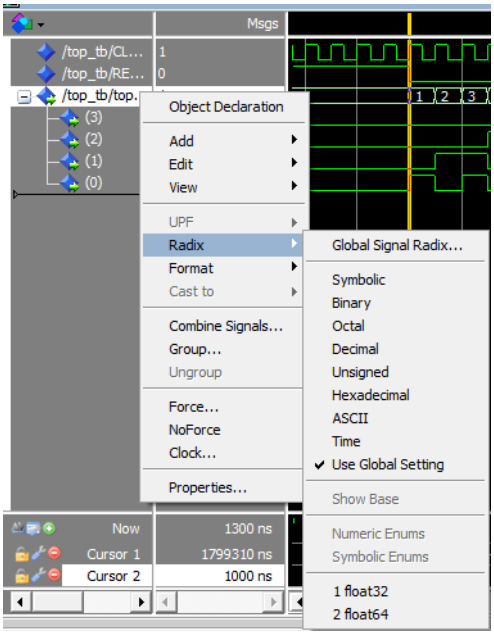


Рис. 3.55. Выбор формата представления значения сигнала

Также имеется возможность представления значение в аналоговом виде, что очень удобно при решении задач обработки сигналов (рис. 3.56).

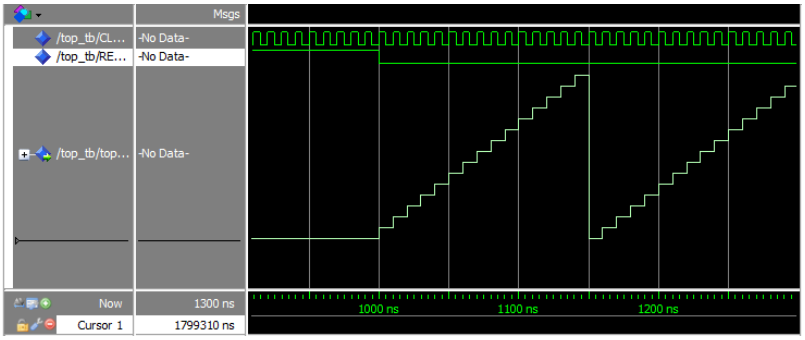


Рис. 3.56. Аналоговое представление сигнала счётчика

Для любого сигнала можно задать значение прямо на временной диаграмме, для этого нужно выбрать из выпадающего меню по правому щелчку мыши на объекте Force и задать значение в появившемся окне (рис. 3.57).

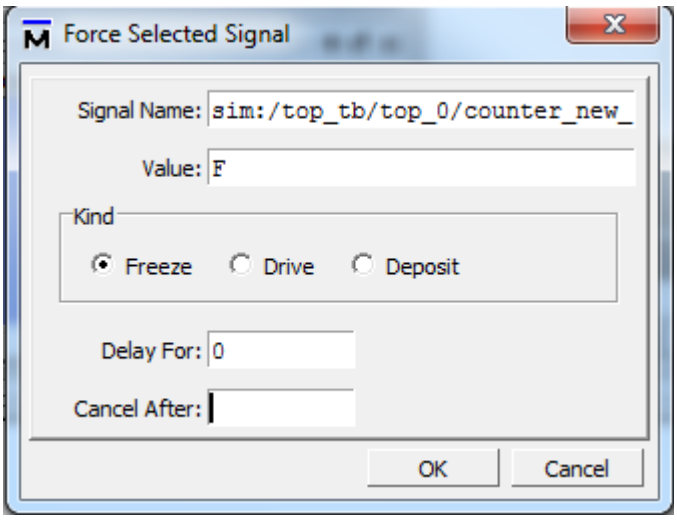
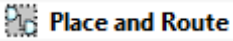


Рис. 3.57. Окно задания воздействия

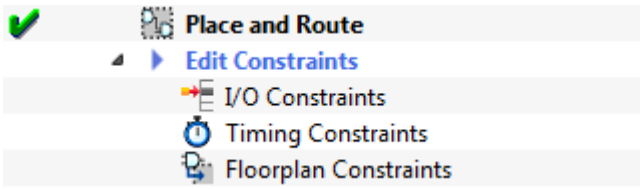
Однако, рекомендуется тщательно продумывать процесс симуляции и основную часть воздействий задавать в используемом тестбенче. Прибегать к вышеописанному методу стоит при необходимости что-либо быстро проверить или исправить мелкий недочёт.

3.7.Размещение и трассировка проекта

После того, как предварительная симуляция, синтез и компиляция проведены, начинается этап размещения и трассировки (Place and Route), в ходе которого специальный инструмент Libero, основываясь на полученном списке соединений (Netlist), составит план размещения логических элементов на выбранном кристалле, а также топологию связей между ними. Для выполнения этого этапа можно сделать двойной щелчок по 

.

Прежде чем проводить проект через этот этап необходимо выполнить ряд условий. Во-первых, задать ограничения для разводки и статического временного анализа. Для выполнения этих операций в Libero предусмотрен ряд инструментов:



Это соответственно графические среды для задания физических ограничений и назначений I/O, задания временных ограничений для разводки и ограничения по размещению логики на кристалле. Для запуска каждого инструмента необходимо произвести на нём двойной щелчок.

Первым пунктом идёт назначение внешних входов/выходов проекта выводам микросхемы. Для этого служит инструмент I/O Editor (рис. 3.58)

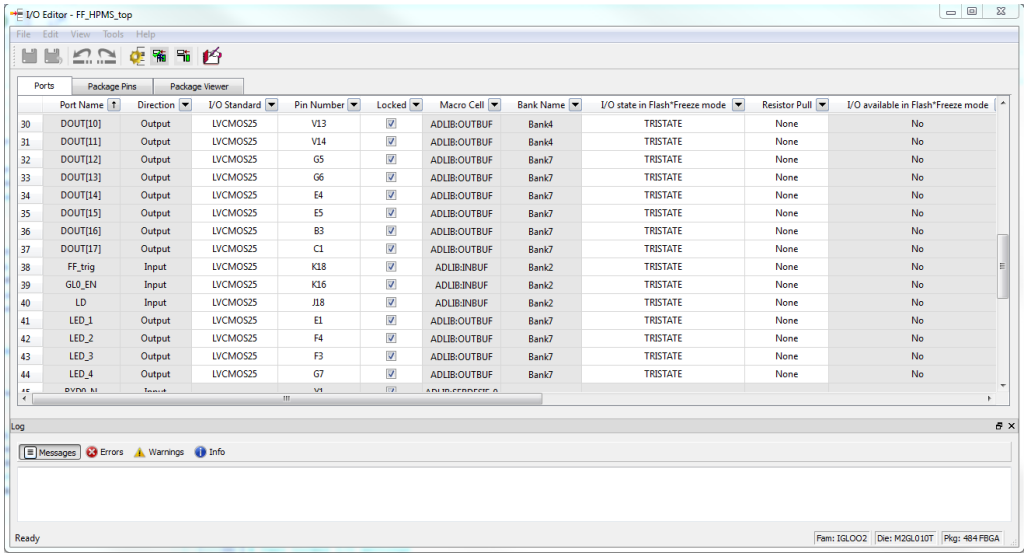


Рис. 3.58. Назначение портов ввода/вывода

Инструмент автоматически отображает все входные и выходные порты дизайна и даёт возможность выбрать номер вывода (Pin Number), которому каждый порт назначается. Также, разработчик выбирает стандарт ввода/вывода, подтяжку резистором, выходной ток и нагрузку, триггер Шмитта по входу и т.п.

Для задания временных ограничений для разводки и временного анализа (будет рассмотрено в след. разделе) имеется инструмент Constraints Editor, являющийся частью системы SmartTime (рис. 3.59).

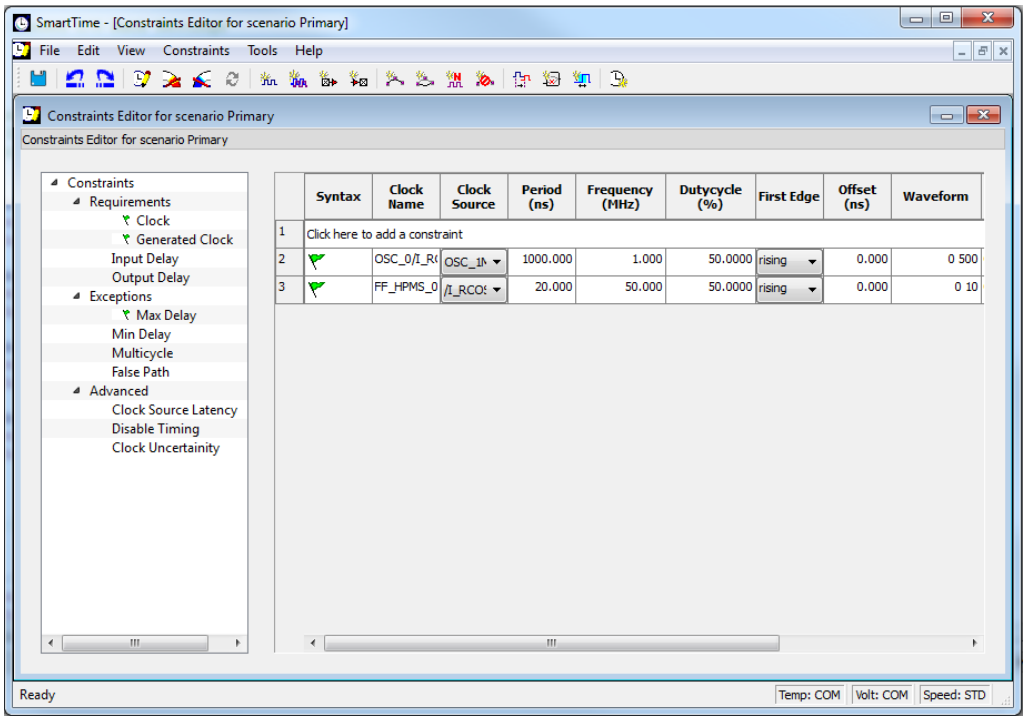


Рис. 3.59. Constratins Editor в SmartTime

Constraints Editor позволяет задавать привычные временные ограничения с помощью удобного графического интерфейса. Также, имеется возможность экспорта временных

ограничений в формате SDC, что позволяет просматривать текстовый вариант и использовать их же для синтеза.

Наконец, инструмент ChipPlanner служит для анализа и задания ограничений на физическое размещение логики на кристалле (рис. 3.60).

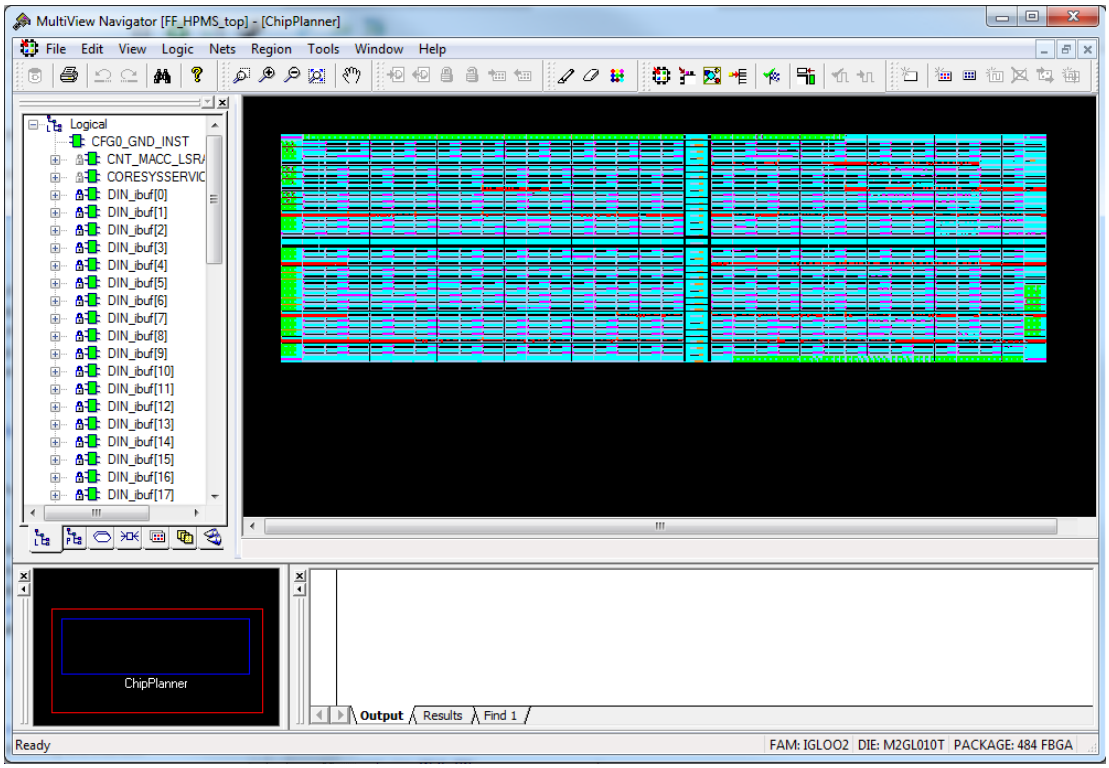
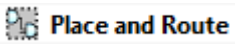


Рис. 3.60. Инструмент ChipPlanner

Также необходимо задать параметры размещения и трассировки, для этого необходимо сделать правый щелчок мыши по 

и выбрать Configure Options (рис. 3.61).

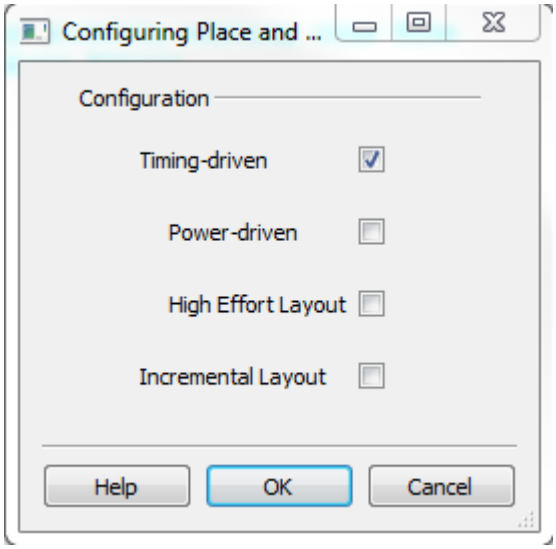


Рис. 3.61. Конфигурация размещения и трассировки

Предлагается выбрать следующие опции:

- Timing-driven – оптимизация по времени;
- Power-driven – оптимизация по потреблению (только для SF2 и IGLOO2);
- High-Effort Layout – разводка в несколько проходов(только для SF2 и IGLOO2);
- Incremental Layout – начинать новую итерацию с результата прошлого запуска.

Для всех ПЛИС в Libero есть возможность зафиксировать текущее состояние размещения элементов и связей на кристалле. Это особенно актуально для сложных проектов с большим количеством отдельных блоков, например, один из блоков с высокими требованиями к временным характеристикам можно разработать отдельно и зафиксировать его, тем самым обеспечив соблюдение его временных характеристик. Для семейств SmartFusion2 и IGLOO2 для фиксации разводки служит т.н. Block Flow. Он позволяет создать требуемый блок в отдельном проекте и экспортировать его вместе с данными о разводке. Для этого нужно, во-первых, открыть Project -> Project Settings и в пункте Design Flow отметить галочкой Enable Block Creation. В этом режиме в Design Flow (рабочая область 1 главного окна Libero) появится пункт Publish Block. Двойной щелчок по нему приводит к открытию диалогового окна (рис. 3.62).

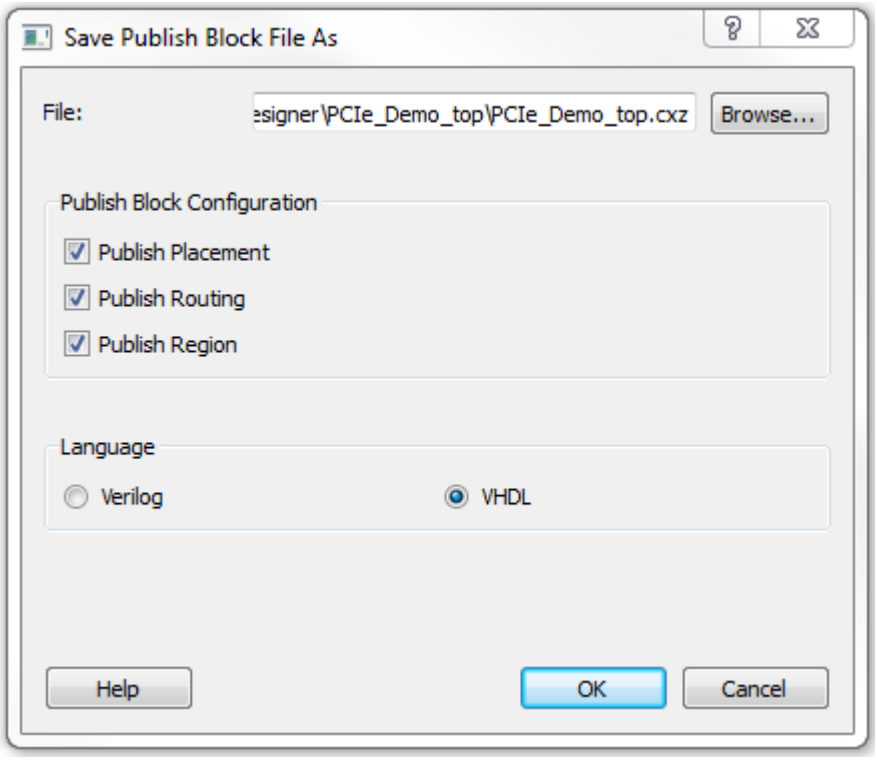


Рис. 3.62. Диалоговое окно публикации (экспорта) блока

Предлагается выбрать, какие характеристики зафиксировать:

- Publish Placement – расположение элементов на кристалле;
- Publish Routing – связи между элементами;
- Publish Region – информацию о заданных пользователям областях кристалла.

Для импорта блока в целевой проект нужно выполнить команду File -> Import -> Blocks.

3.8. Статический временной анализ и временное моделирование

Для финальной оценки временных характеристик разрабатываемого устройства в Libero предусмотрено два специальных инструмента:

- Временной анализатор в SmartTime;
- Временное моделирование в Modelsim.

Первым этапом оценки временных характеристик является статический временной анализ, который в Libero проводится с помощью инструмента SmartTime. Инструмент определяет пути между конечными точками дизайна (порты ввода-вывода, триггеры) и рассчитывает задержки на этих путях, связанные со временем обработки в комбинационных схемах и временем распространения по линиям связи.

В разделе 3.6 были перечислены три этапа симуляции устройства. Первые два служат для предварительной оценки корректности работы логики и функциональной верификации. Последний этап — временное моделирование (post-layout simulation) – предназначен для проверки работы устройства с учётом всех особенностей целевого кристалла, при этом в результате такого моделирования разработчик получает временную диаграмму, максимально соответствующую поведению дизайна внутри кристалла.

Пренебрегать каким-либо из этих двух этапов чревато некорректно работой устройства, поэтому выполнение их настоятельно рекомендуется.

Как открыть инструмент SmartTime было рассказано выше (см. разд. 3.7), попасть в окно временного анализатора можно нажав для анализа максимальных задержек на путях и для анализа минимальных задержек (либо пункты Max Delay Analysis и Min Delay Analysis в меню Tools). При открытии окно содержит отчёт о проведении статического временного анализа (рис. 3.63).

Замечание: перед описанными ниже действиями проверьте, заданы ли временные ограничения (как минимум тактовые частоты) и их корректность.

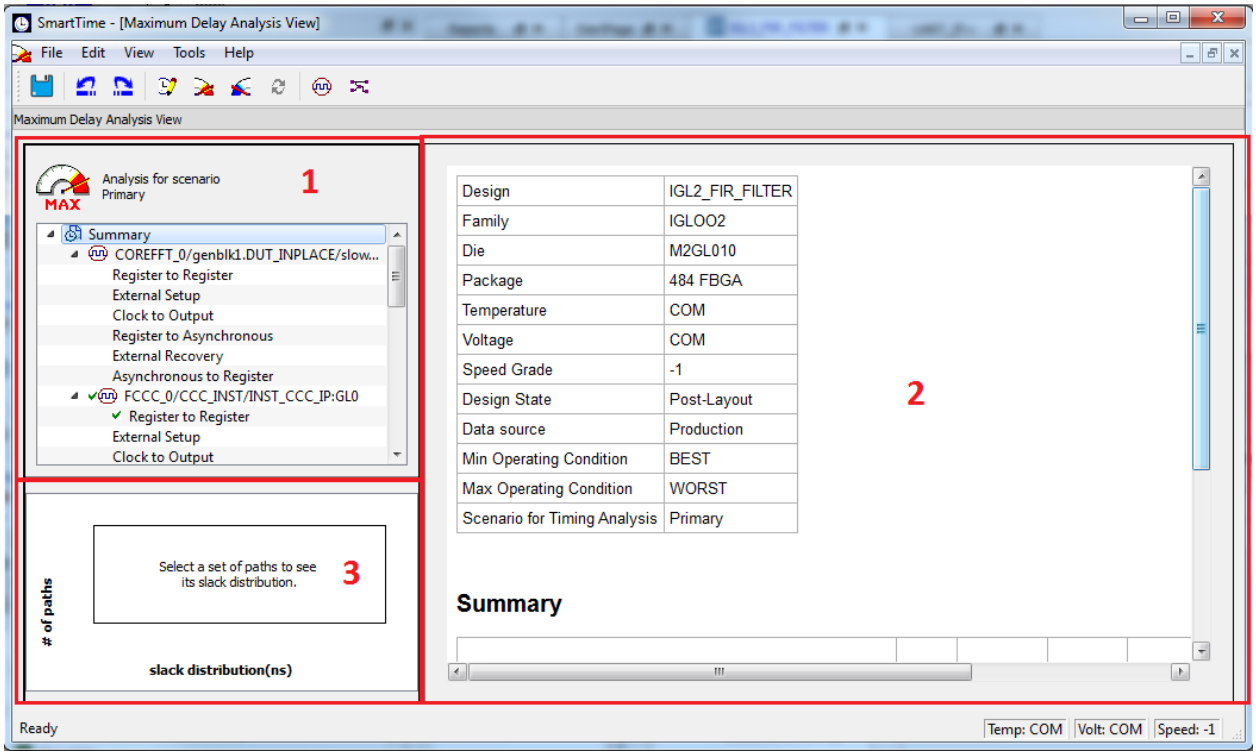


Рис. 3.63. Окно SmartTime Maximum Delay Analysis

Окно делится на области:

1) – отчёты о путях, сгруппированные по тактовым сигналам, путям pin-to-pin и пользовательским группам;

2) область вывода информации о выбранных в 1 группах и подгруппах;

3) гистограмма распределения запаса времени (slack) по путям, отображенным в 2.

Значки «зеленая галочка» в области 1 означают, что данные пути удовлетворяют требованиям, напротив «красный крестик» означает, что в данной подгруппе есть проблемные пути, которые не удовлетворяют заданным разработчиком временным ограничениям.

Когда пользователь выделяет отдельный тактовый сигнал в области 1, по нему появляется отдельный отчёт, содержащий желаемую частоту в данном домене, максимальную рассчитанную частоту. Если выделить подгруппу внутри тактового сигнала, появляются отчеты о всех путях домена в табличном виде (рис. 3.64).

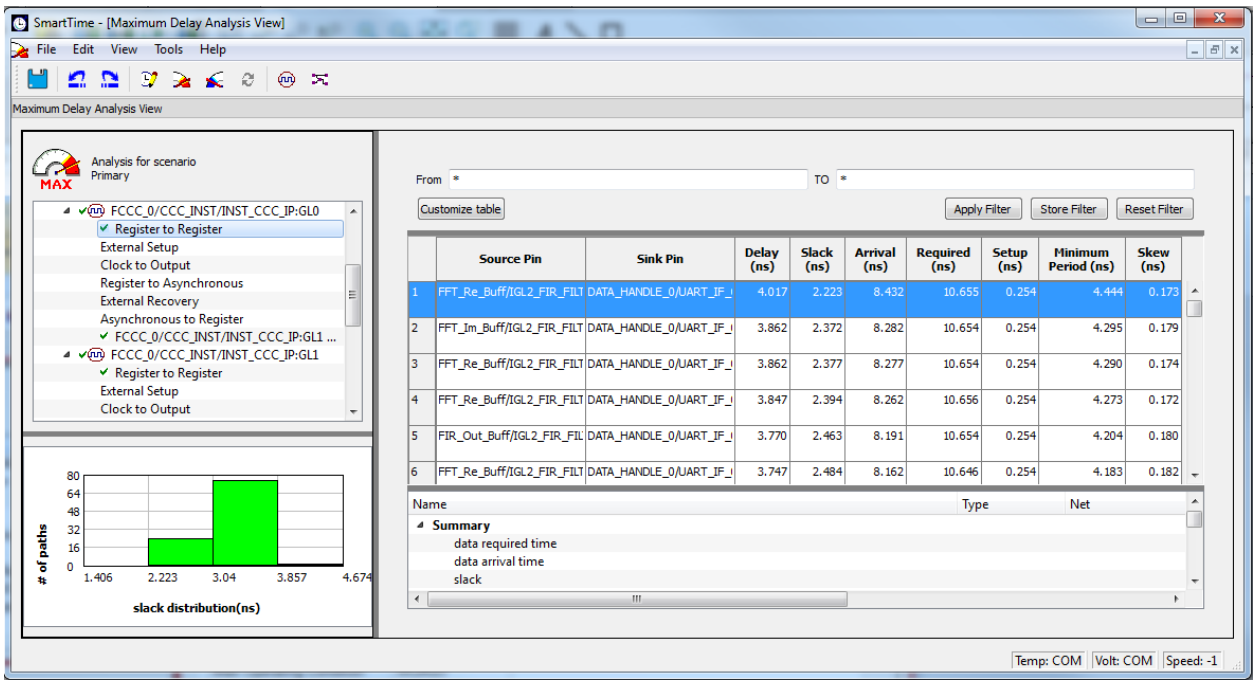


Рис. 3.64. Информация о подгруппе в тактовом домене

При этом в области 3 выводится гистограмма распределения slack. Зеленая колонка означает положительный slack и пути, удовлетворяющие временным характеристикам. В случае, если какие-либо пути слишком длинные для заданной частоты, колонка будет красной, как и числа в таблице (рис. 3.65).

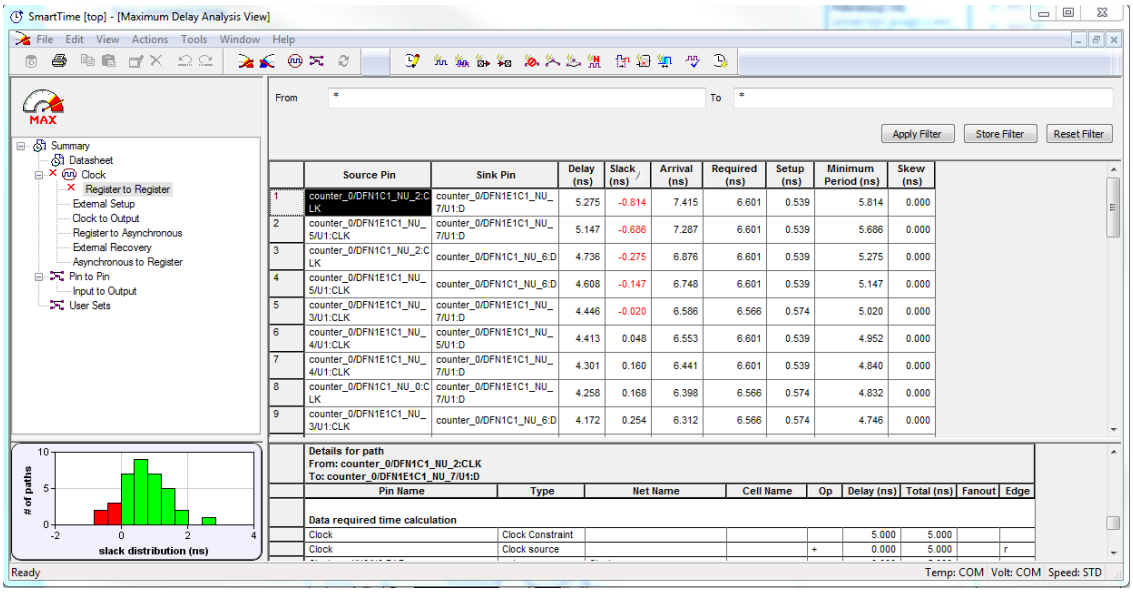


Рис. 3.65. Пример отчета с путями, не соответствующими временным ограничениям

Это означает, что заданная во временных ограничениях частота слишком высокая для указанных путей и необходимо выполнить оптимизацию (например, с использованием подхода, называемого конвейеризацией).

Разработчик может просмотреть детальную информацию по конкретному пути, сделав двойной щелчок по соответствующей строке таблицы (рис. 3.66).

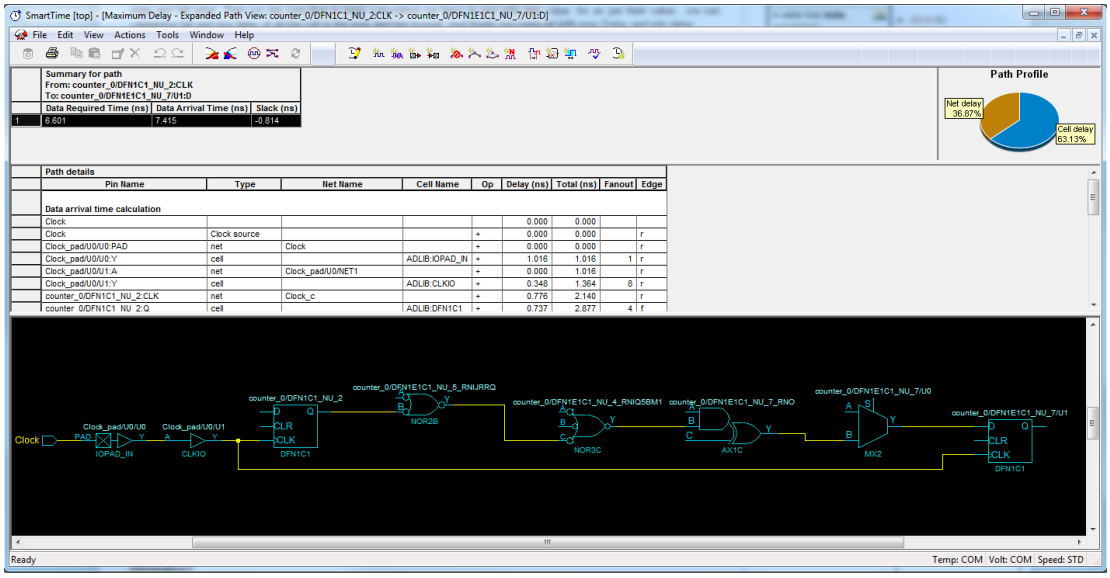


Рис. 3.66. Детальная информация по пути

Разработчику доступны основные характеристики пути:

- Data Required Time – время, в которое данные должны поступить на вход приёмника;
- Data Arrival Time – время, в которое данные фактически поступают на вход приемника;
- Slack – временной запас (чем больше, тем на еще большей частоте может функционировать путь).

В таблице имеются подробные указания о расчете, как требуемого времени, так и времени фактического поступления данных, указаны вентили, через которые проходят данные. В нижней области путь представлен графически, слева располагается конечная точка источник, справа – конечная точка приёмник данных.

Замечание: дополнительное удобство при анализе предоставляет возможность синхронного просмотра путей в SmartTime и ChipPlanner: выделяя элемент в графической области SmartTime (см. рис.3.66), он автоматически выделяется в ChipPlanner (см. рис. 3.60 разд. 3.7). таким образом, можно не только рассмотреть путь, но и увидеть где конкретно на кристалле он пролегает.

Наконец, круговая диаграмма в правом верхнем углу окна показывает соотношение задержек на вентилях (cell delay) и задержек распространения по линиям связи (propagation delay).

Если временной анализатор дал положительный результат по требованиям разработчика, можно перейти к этапу временного моделирования в Modelsim. Для начала работы в этом режиме необходимо файлы обратного аннотирования (Back-Annotated), которые содержат специальный файл в формате *.sdf (Standard Delay Format) и список соединений после размещения и трассировки, позволяющие провести более точное моделирование с позиции

временных характеристик. Для этого в Design Flow нужно сделать двойной щелчок на пункте Generate Back Annotated Files. Файлы обратной аннотации будут отображены в списке окна Organize simulation files (см. разд. 3.6) для Simulate в группе Verify Post Layout Implementation (рис. 3.67).

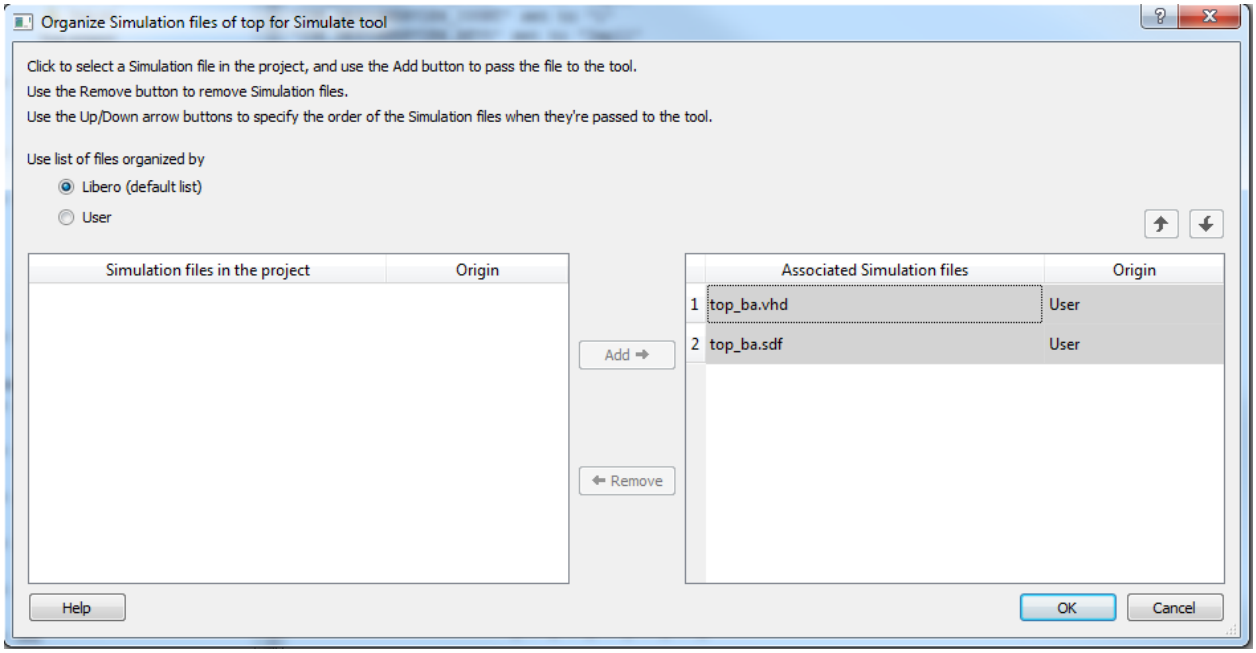


Рис. 3.67. Список файлов для временного моделирования

Далее запуск Modelsim и процесс моделирования не сильно отличается от описанного в разд. 3.6. Основными особенностями временного моделирования является наиболее низкоуровневая структура в иерархии симуляции, так как в качестве входного файла выступает список соединений в терминах логических ресурсов кристалла (рис. 3.68).

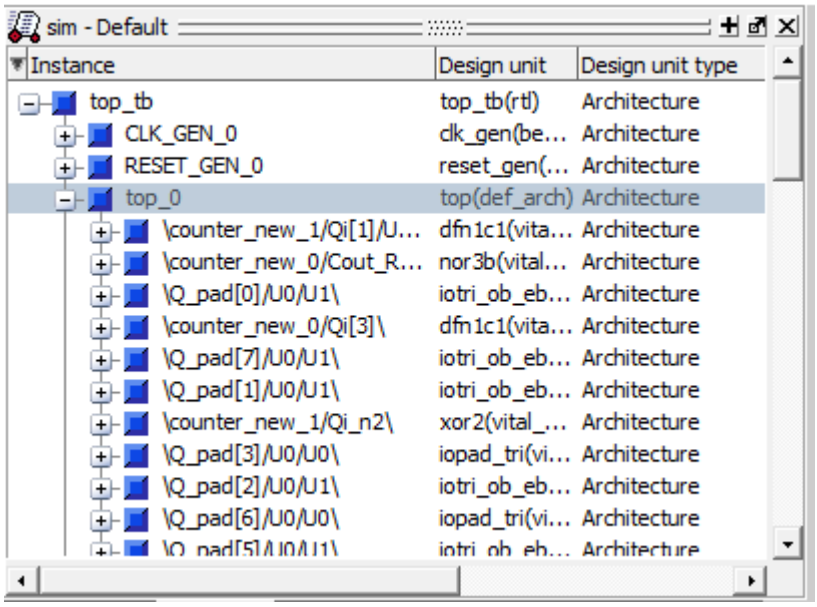


Рис. 3.68. Иерархия симуляции при временном моделировании

На временной диаграмме, в отличие от случая, описанного в разд. 3.6, теперь можно увидеть задержки в распространении данных относительно тактового сигнала (рис. 3.69).

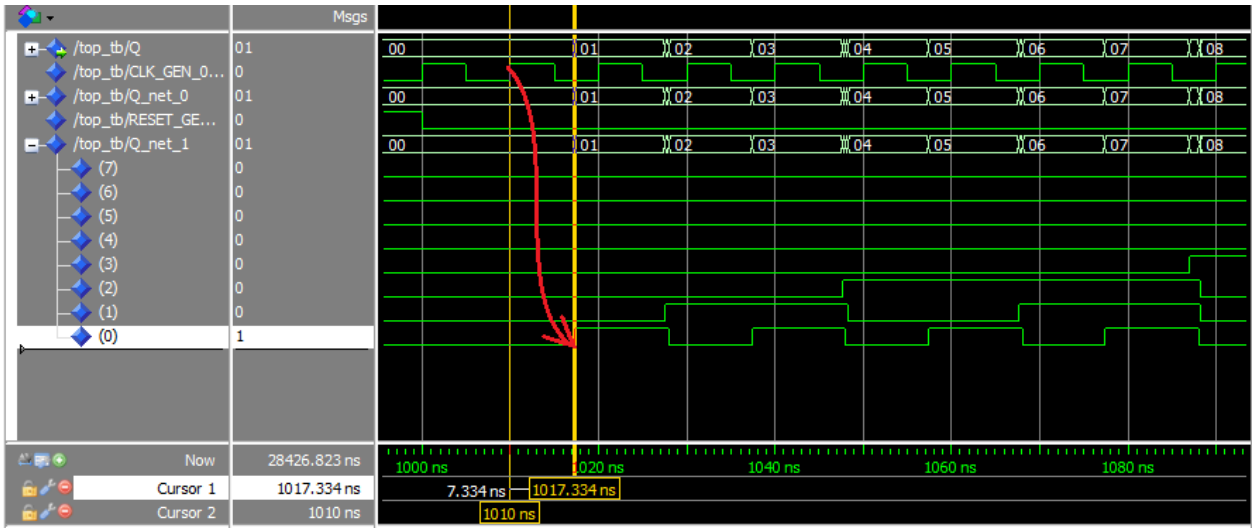


Рис. 3.69. Временные задержки при временном моделировании в Modelsim

Данный этап — последний в процессе отладки проекта путём моделирования. Но следует заметить, что весь процесс разработки носит итерационный характер, поэтому проходить через все этапы придётся по несколько раз.

3.9.Программирование ПЛИС с помощью программатора FlashPro

После завершения всех этапов разработки и возникновения необходимости провести отладку на аппаратной платформе, разработчик генерирует конфигурационный файл для программирования ПЛИС. Для этого в Design Flow служат следующие пункты (для ProASIC3, IGLOO и т.п. и SmartFusion2/IGLOO2 соответственно): Generate Programming Data и Generate Bitstream. По двойному щелчку начнется формирование специального файла программирования. По окончании этого процесса и появления отчёта о генерации файла в области сообщений и лога событий у разработчика появляется возможность загрузить полученную конфигурацию в устройство.

Для программирования всех Flash-ПЛИС используется программатор FlashPro (на момент написания последняя версия – FlashPro5). Программатор имеет стандартный 10-контакный разъём с выводами интерфейса JTAG с одной стороны и интерфейс USB для подключения к ПК с другой стороны (рис. 3.70).

Рис. 3.70. Программатор FlashPro5

Перед соединением программатора с разъемом на печатной плате следует убедиться, что питания платы отключено и не подаётся на ПЛИС. При подключении программатора к ПК он определится в системе и загорится светодиод POWER. Если светодиод не загорелся, то программатор не определился в системе и надо подождать. После успешного подключения программатора нужно подать питание ПЛИС, после чего сделать двойной щелчок по Program Device или Run PROGRAM Action. Запустится процесс программирования, который может занять до 5-7 минут, в зависимости от ёмкости микросхемы.

Замечание: категорически не рекомендуется прерывать процесс программирования, физически обрывать связь с программатором во время программирования, последствия могут быть повреждение микросхемы ПЛИС и программатора вплоть до полного выхода из строя.

По окончании процесса разработчику будет предоставлен отчет (рис. 3.71).

```
FlashPro
Version: 11.4.0.112
Release: v11.4

Software Version: 11.4.0.112
Driver : 3.0.0 build 1
programmer '73590' : FlashPro4
Opened
'E:/Libero_11.4_updates/Livelink_downloads_validation/IGL2_FIR_FILTER_DF/DesignFiles/IGL2_FIR_F\
ILTER/designer/IGL2_FIR_FILTER/IGL2_FIR_FILTER_fp/IGL2_FIR_FILTER.pro'
The 'open_project' command succeeded.
IPD file
'E:/Libero_11.4_updates/Livelink_downloads_validation/IGL2_FIR_FILTER_DF/DesignFiles/IGL2_FIR_F\
ILTER/designer/IGL2_FIR_FILTER/IGL2_FIR_FILTER.ipd' has been loaded successfully.
DESIGN : IGL2_FIR_FILTER;  CHECKSUM : 9553;  PDB_VERSION : 1.8
The 'set_programming_file' command succeeded.
Warning: Programming is already enabled for device 'M2GL010'.
The 'enable_device' command succeeded.
The 'set_programming_action' command succeeded.
programmer '73590' : Scan Chain...
Warning: programmer '73590' : Vpump has been selected on programmer AND an externally provided
         Vpump has also been detected. Using externally provided Vpump voltage source.
programmer '73590' : Check Chain...
programmer '73590' : Scan and Check Chain PASSED.
programmer '73590' : device 'M2GL010' : Executing action PROGRAM
programmer '73590' : device 'M2GL010' : EXPORT ISC_ENABLE_RESULT[32] = 00776444
programmer '73590' : device 'M2GL010' : EXPORT CRCERR: [1] = 0
programmer '73590' : device 'M2GL010' : EXPORT EDCERR: [1] = 0
programmer '73590' : device 'M2GL010' : TEMPGRADE: ROOM
programmer '73590' : device 'M2GL010' : EXPORT VPPRANGE: [3] = 2
programmer '73590' : device 'M2GL010' : VPPRANGE: HIGH
programmer '73590' : device 'M2GL010' : EXPORT TEMP: [8] = 64
programmer '73590' : device 'M2GL010' : EXPORT VPP: [8] = 77
programmer '73590' : device 'M2GL010' : Programming FPGA Array...
programmer '73590' : device 'M2GL010' :
================================================================================
programmer '73590' : device 'M2GL010' : EXPORT DSN[128] = 03e8aa010000cd5a0023000a00080021
programmer '73590' : device 'M2GL010' :
================================================================================
programmer '73590' : device 'M2GL010' : Finished: Thu Jul 31 14:09:04 2014 (Elapsed time
00:00:30)
programmer '73590' : device 'M2GL010' : Executing action PROGRAM PASSED.
programmer '73590' : Chain programming PASSED.
Chain Programming Finished: Thu Jul 31 14:09:04 2014 (Elapsed time 00:00:30)
```

Рис. 3.71. Пример отчёта о программировании ПЛИС

В случае удачного программирования светодиод PASS/FAIL на корпусе программатора загорится зеленым цветом. Если во время программирования возникает ошибка, светодиод будет гореть красным цветом и соответствующие сведения будут приведены в отчёте и логе. Наиболее частые причины возникновения ошибок:

- Отсутствие контакта с программатором;
- Неправильный файл программирования (например, для другой микросхемы);
- Нарушение целостности сигналов JTAG на плате;
- Неправильная конфигурация джамперов на отладочной плате;
- Неверный секретный ключ (если микросхема была прошита с защитным ключом);
- Неисправность микросхемы, неоригинальная микросхема.

В случае возникновения ошибки следует проанализировать отчёт, извлечь из него сообщение об ошибке и код ошибки, обратиться к разделу Troubleshooting документа FlashPro User Guide (http://www.microsemi.com/products/fpga-soc/design-resources/programming/flashpro#documents), в котором представлена таблица с кодами ошибок и разъяснением возможных причин ошибки. Следует тщательно исследовать свою систему на предмет возможных неисправностей. В случае невозможности устранения — обратиться в службу техническое поддержки ЗАО «Актел.ру».

Программатор FlashPro также служит отладочным средством при внутрисхемной отладке ПЛИС (будет рассмотрено в разделе 3.10) и при разработке и отладке встроенного программного обеспечения для микропроцессора ARM Cortex-M3, встроенного в SmartFusion2, в среде SoftConsole.

3.10. Внутрисхемная отладка проектов для ПЛИС в Libero SoC

3.10.1. Инструмент SmartDebug

Внутрисхемная отладка программируемых устройств представляет собой переход от виртуальной модели и симуляции к тестированию проекта в составе целевого оборудования, при этом в отличие от традиционных натурных экспериментов, позволяющих отслеживать сигналы только на внешних выводах микросхемы, внутрисхемная отладка предполагает мониторинг состояния внутренних элементов микросхемы.

Для проведения отладки в составе системы в состав Libero SoC включены два инструмента:

- Microsemi SmartDebug;
- Synopsys Identify ME.

SmartDebug — новый инструмент корпорации Microsemi, представленный в версии 11.0, и доступный эксклюзивно для микросхем SmartFusion2 и IGLOO2. Инструмент использует встроенные в эти микросхемы аппаратные средства отладки, позволяющие в реальном времени получить доступ к любому логическому элементу и отслеживать его состояние в реальном времени.

Для запуска отладчика SmartDebug необходимо убедиться, что все предыдущие шаги маршрута проектирования выполнены успешно и ПЛИС содержит нужную конфигурацию. Для работы SmartDebug необходимо, чтобы программатор FlashPro был соединён с ПЛИС, как при программировании (см. разд. 3.9). Для запуска SmartDebug нужно сделать двойной щелчок на SmartDebug Design в Design Flow. Откроется диалоговое окно (рис. 3.72).

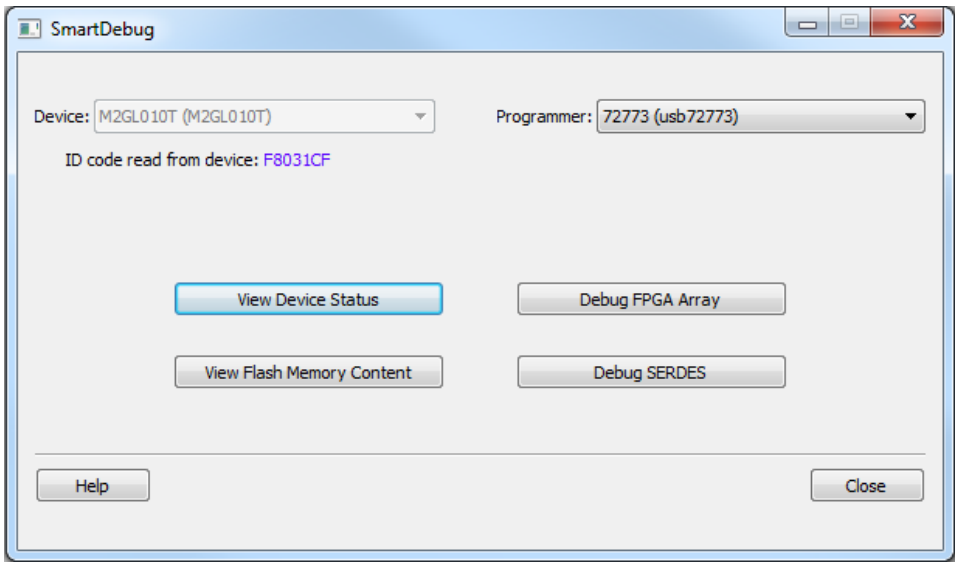


Рис. 3.72. Диалоговое окно SmartDebug

В верхней части окна расположена информация об используемом кристалле, его ID код, а также информация о программаторе.

Четыре кнопки в центре окна отвечают за вызов того или иного инструмента:

- View Device Status – просмотреть информацию о микросхеме;
- Debug FPGA Array – отладка матрицы ПЛИС;
- View Flash Memory Content – просмотр содержимого встроенной Flash памяти;
- Debug SERDES – отладка высокоскоростных интерфейсов.

Подробнее рассмотрим процесс отладки матрицы ПЛИС. После нажатия соответствующей кнопки откроется новое окно (рис. 3.73).

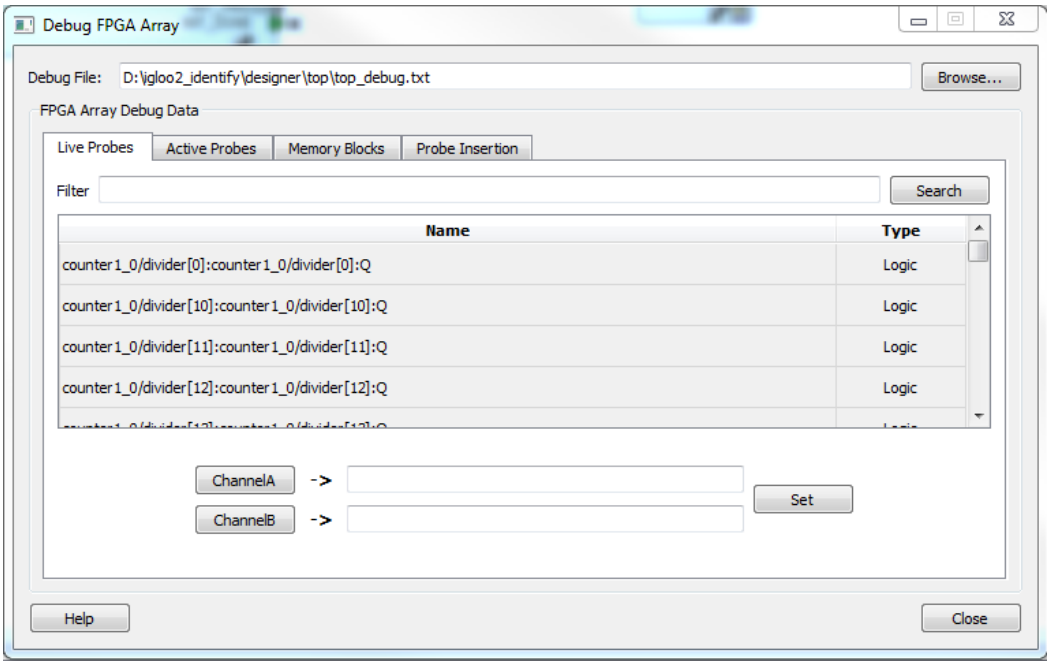


Рис. 3.73. Окно отладки матрицы ПЛИС

Окно содержит четыре вкладки:

- Live Probes – позволяет вывести внутренние сигналы на два выделенных вывода микросхемы для просмотра осциллографом;
- Active Probes – позволяет просматривать состояния внутренней логики и задавать входные воздействия в реальном времени;
- Memory Blocks – позволяет просматривать состояние внутренней памяти и записывать значения в ячейки;
- Probe Insertion – позволяет вывести внутренний сигнал на любой вывод микросхемы.

На вкладке Liver Probes разработчик назначает любой внутренний сигнал на один из выделенных каналов (конфигурацию каналов см. в документации на микросхему) путём выделения нужного сигнала в таблице и нажатия кнопки с именем нужного канала (см. рис. 3.73).

На вкладке Active Probes разработчик может выбрать любой из внутренних элементов дизайна с помощью диалогового окна, вызываемого кнопкой Select Active Probes (рис. 3.74).

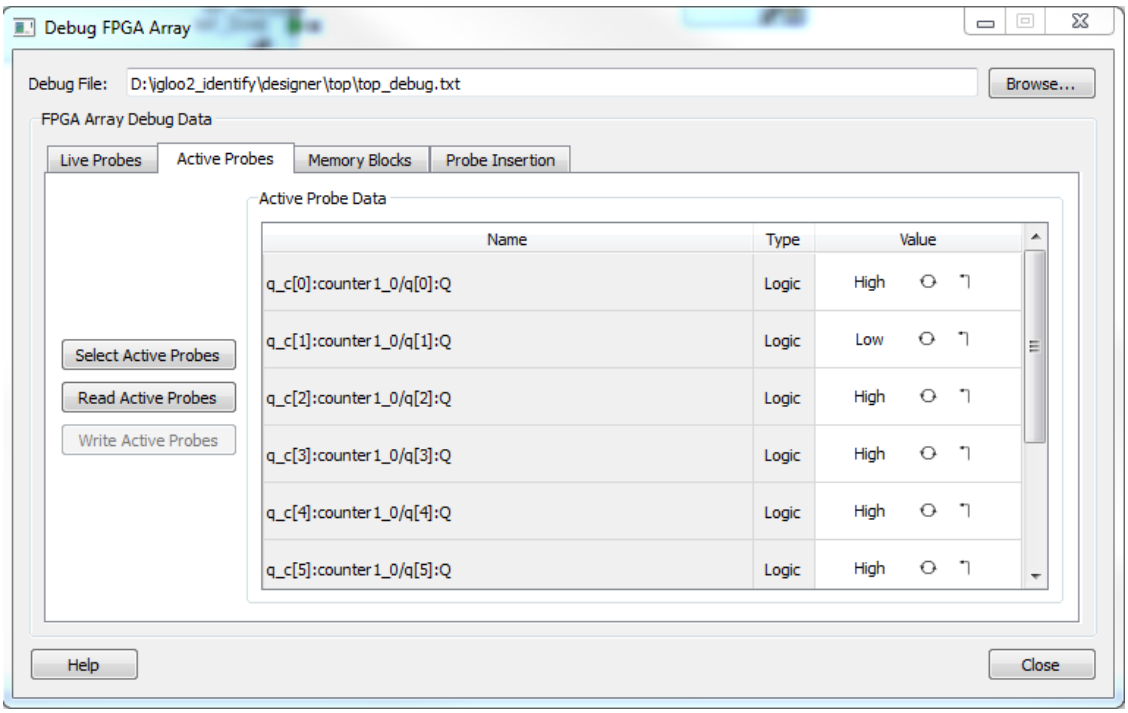


Рис. 3.74. Вкладка Active Probes в SmartDebug

В диалоговом окне Select Active Probes Points (рис. 3.75) содержатся две таблицы, в правой выводятся доступные сигналы, в левой – выбранные сигналы, выбор производится выделением требуемого сигнала и нажатием кнопки Add. Можно добавить все доступные сигналы нажатием кнопки Add All. Аналогично сигналы удаляются из списка активных кнопками Remove и Remove All.

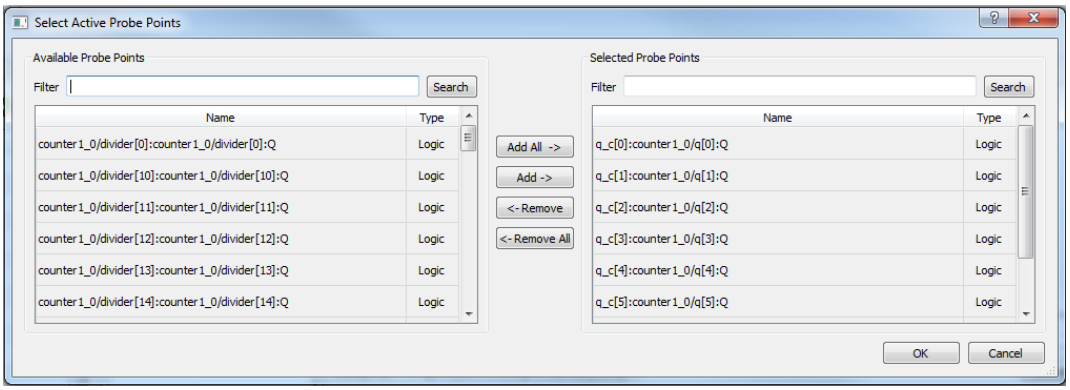


Рис. 3.75. Диалоговое окно добавления Active Probes

По нажатию ОК разработчик возвращается к основному окну. Чтение всех сигналов и обновление таблицы происходит по нажатию кнопки Read Active Probes (см. рис. 3.74). Для изменения значения сигнала разработчик может нажать на символ напротив требуемого сигнала, по нажатию происходит переключение между значениями 0 и 1, при этом происходит изменение цвета текста на красный. Для отмены служит символ . Все произведенные назначения вступают в силу по нажатию кнопки Write Active Probes (рис. 3.76).

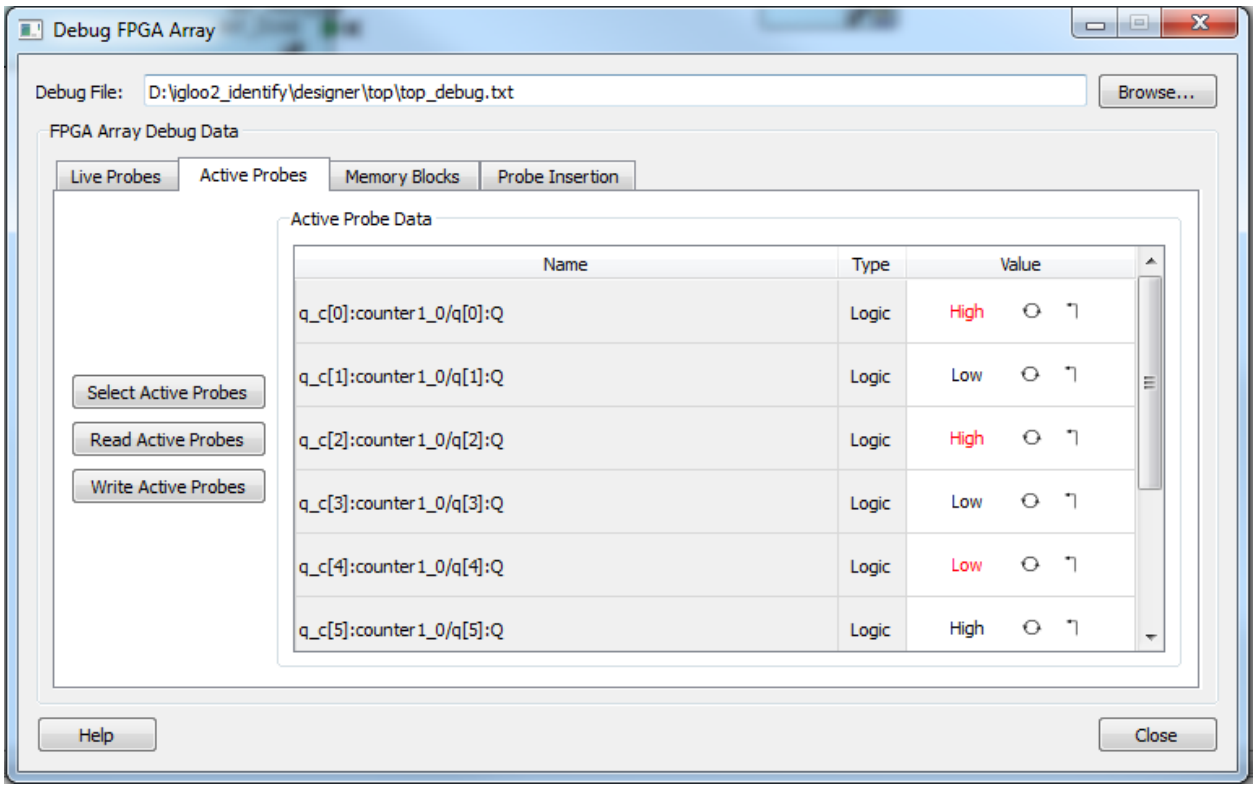


Рис. 3.76. Изменение состояния внутренних сигналов ПЛИС

На вкладке Memory Blocks разработчик может просмотреть и изменить состояние внутреннего блока памяти (рис.3.77).

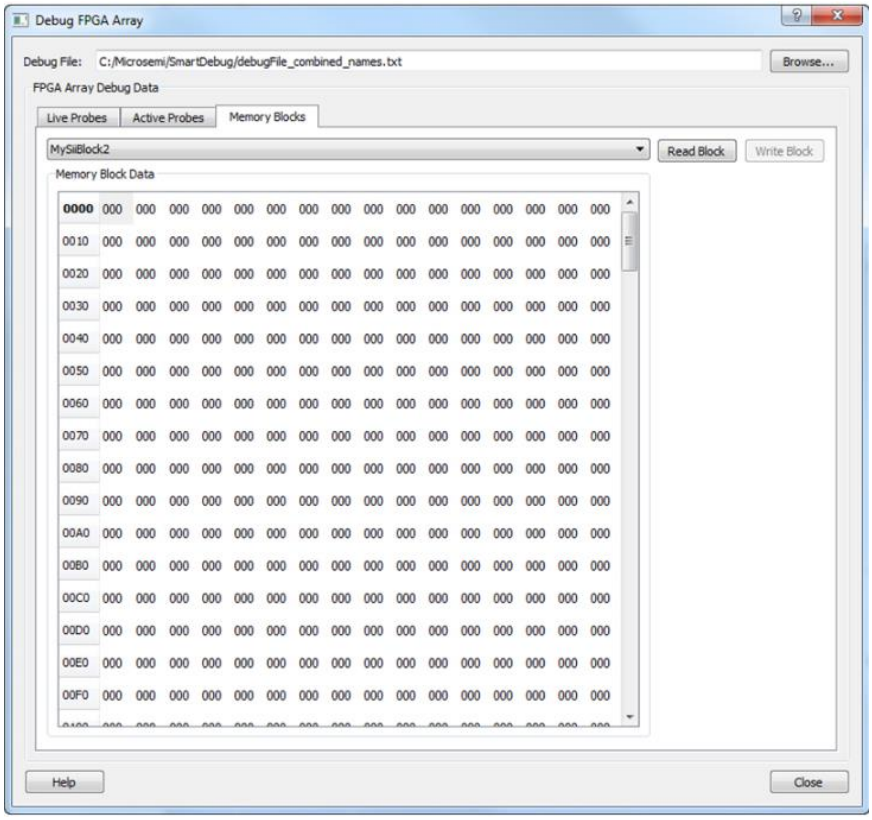


Рис. 3.77. Вкладка Memory Blocks в SmartDebug

По нажатию кнопки Read Block считывается содержание блока памяти. Значение в ячейках таблицы можно изменять (рис. 3.78).

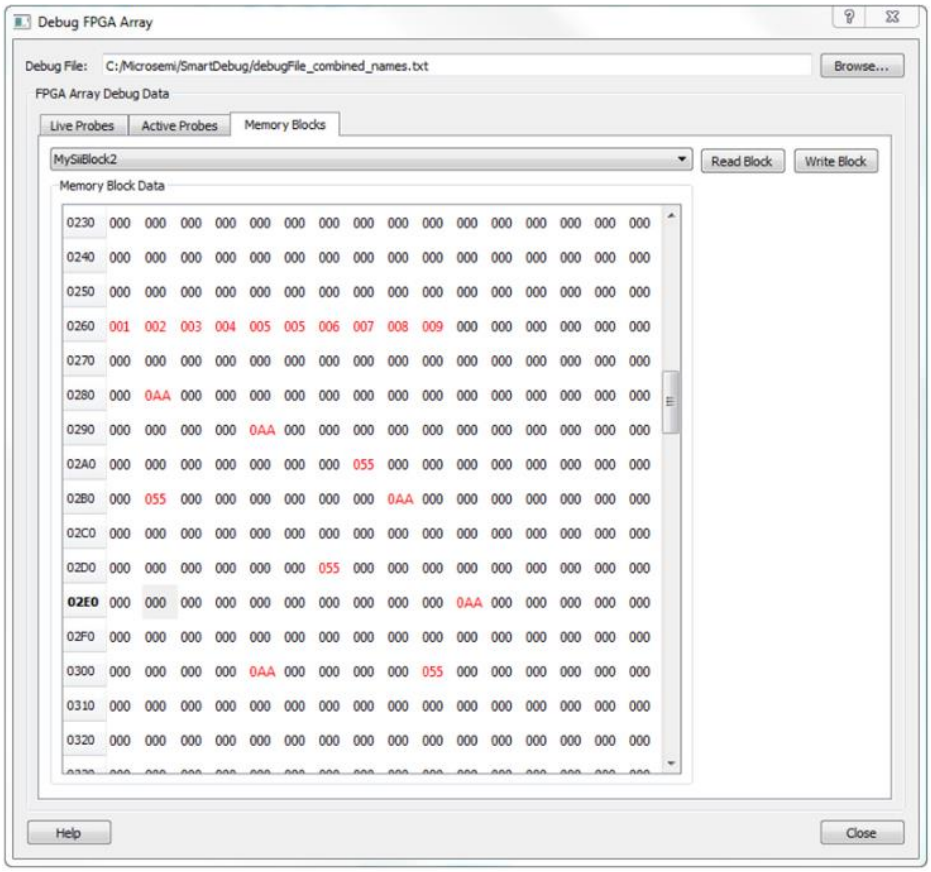


Рис. 3.78. Изменение состояния ячеек блока памяти

Все изменения будут внесены в блок памяти по нажатию кнопки Write Block.

Наконец, на вкладке Probe Insertion можно вывести состояния определенных сигналов на внешние выводы микросхемы (рис. 3.79).

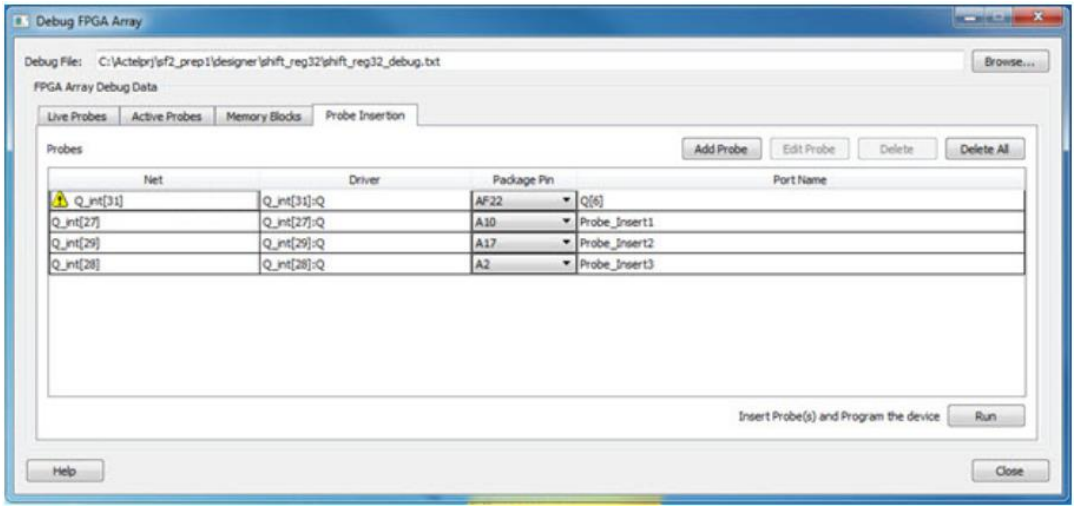


Рис. 3.79. Вкладка Probe Insertion в SmartDebug

Вывод состояния сигналов в таком случае возможен только путем перепрограммирования устройства, для этого нужно нажать кнопку Run. Сигналам можно назначать любые выводы (в том числе уже назначенные, при этом перед названием сигнала в таблице появится значок предупреждения).

3.10.2. Инструмент Identify

Identify ME — универсальный инструмент отладки ПЛИС, включенный в состав Libero, позволяет быстро отследить и устранить функциональные ошибки в проекте с помощью наблюдения состояний внутренних сигналов в реальном времени прямо в HDL-коде. Для связи с ПЛИС, как и в предыдущем случае, используется программатор FlashPro.

Перед началом работы разработчик должен убедиться, что Identify установлен и присутствует в Project -> Tool Profiles (рис. 3.80).

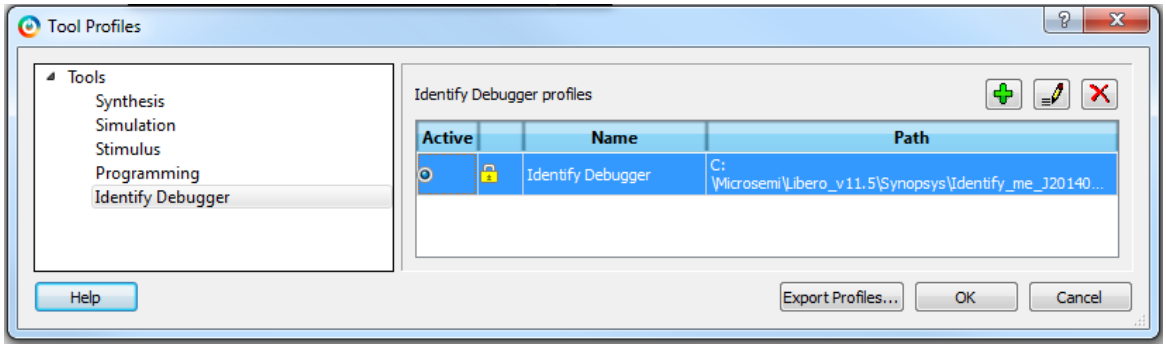


Рис. 3.80. Указание активности профиля и местоположение Identify

Работа с Identify проходит в двух подсистемах:

- Identify Instrumentor;
- Identify Debugger.

Типичный маршрут проектирования при этом изменяется на этапе синтеза (рис. 3.81). Для работы собственно самого отладчика (Debugger) необходимо синтезировать для него специальный список соединений с помощью Instrumentor, доступ к которому производится из Synopsis Synplify (см. разд. 3.5).

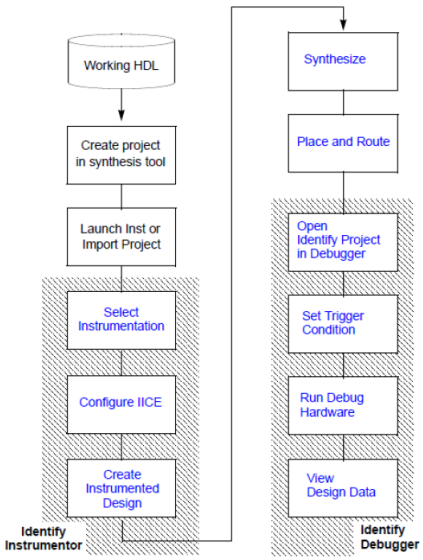


Рис. 3.81. Маршрут проектирования с этапами отладки в Identify

Для доступа к Instrumentor нужно открыть Synplify в интерактивном режиме, для чего выбираем из выпадающего меню пункт Open Interactively (рис. 3.82).

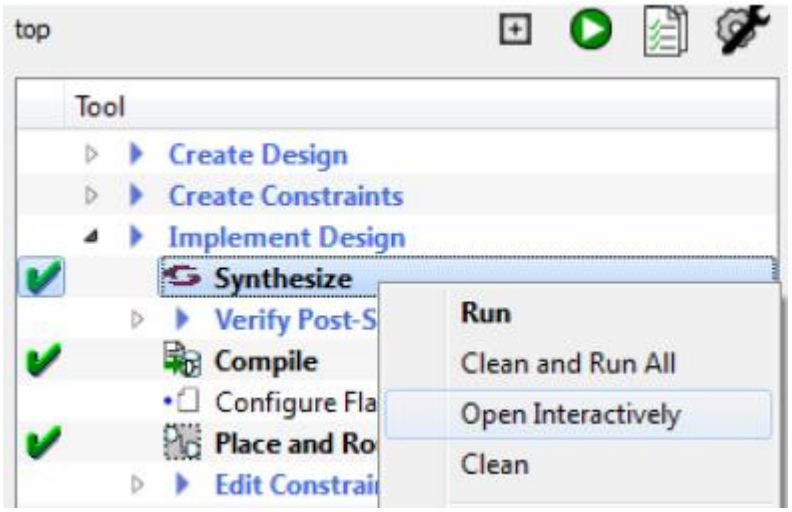


Рис. 3.82. Открытие Synplify в интерактивном режиме

Откроется главное окно Synplify, содержащее информацию о текущем проекте. Для того чтобы открыть Identify Instrumentor нужно нажать правой кнопкой мыши на “synthesis” в иерархии проекта и выбрать пункт Launch Identify Instrumentor (рис. 3.83).

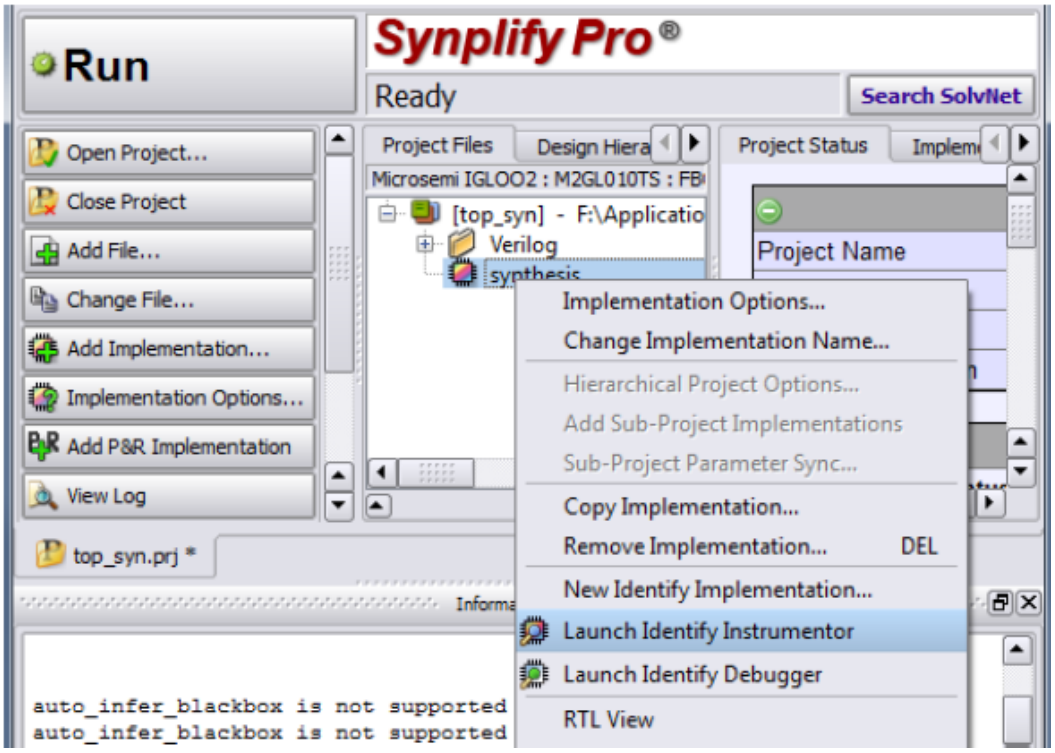


Рис. 3.83. Запуск Identify Instrumentor

В появившихся окнах с вопросами о желании создать новую имплементацию Identify и сохранить проект нужно нажать ОК. После этого откроется окно Identify Instrumentor (рис. 114). Иерархия исходного проекта можно видеть в левой части окна, в нём можно открывать все блоки проекта, исходный код которых отображается в правой части окна.

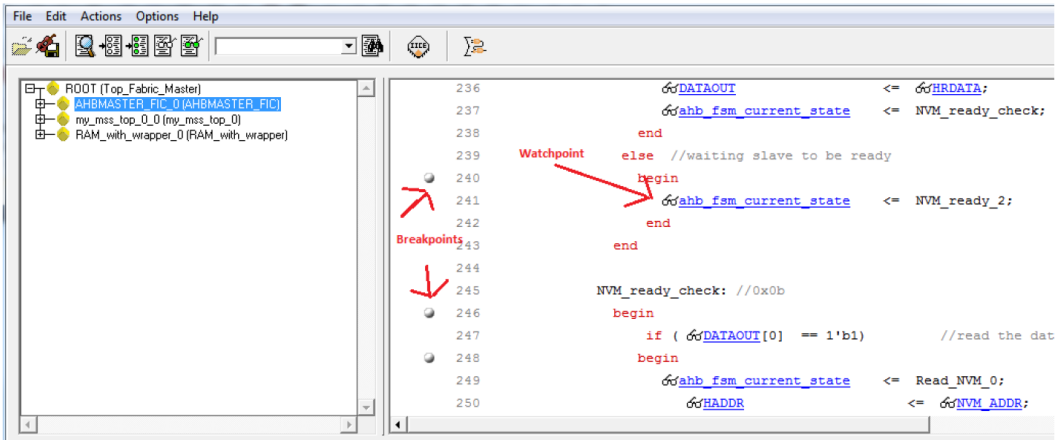


Рис. 3.84. Рабочая область Identify Instrumentor

В правой части окна можно расставлять точки наблюдения (Watchpoint) и метки останова (Breakpoint). Точки наблюдения отмечаются символом , а точки останова — символом (в зависимости от активности и функциональности цвет значков будет меняться). Если сделать щелчок мыши по значку «очки», разработчику будет предложено выбрать несколько возможных ролей сигнала в процессе отладки (рис. 3.85).

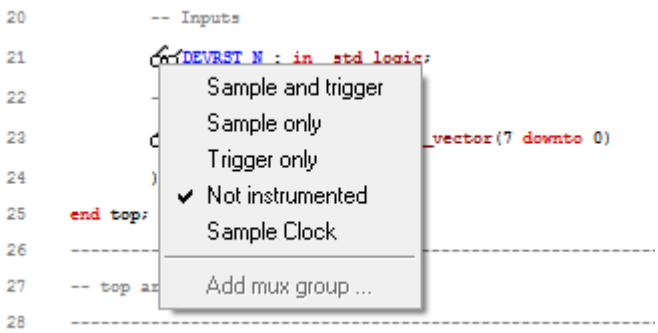


Рис. 3.85. Настройка роли сигнала в процессе отладки

Перечислены следующие варианты:

- Захват и триггер (Sample and trigger);
- Только захват (Sample only);
- Только триггер (Trigger only);
- Не задействован (Not instrumented);
- Сигнал является тактом захвата (Sample clock).

Для удобного поиска возможных реперных точек предлагается использовать кнопки на панели быстрого доступа (рис. 116), которые позволяют отобразить все возможные точки типа watchpoint и breakpoint, а также все расставленные точки.

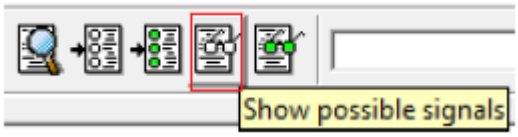


Рис. 3.86. Инструменты для работы с контрольными точками

Таким образом, легко найти и расставить все реперные точки в дизайне с помощью поиска (рис. 3.87).

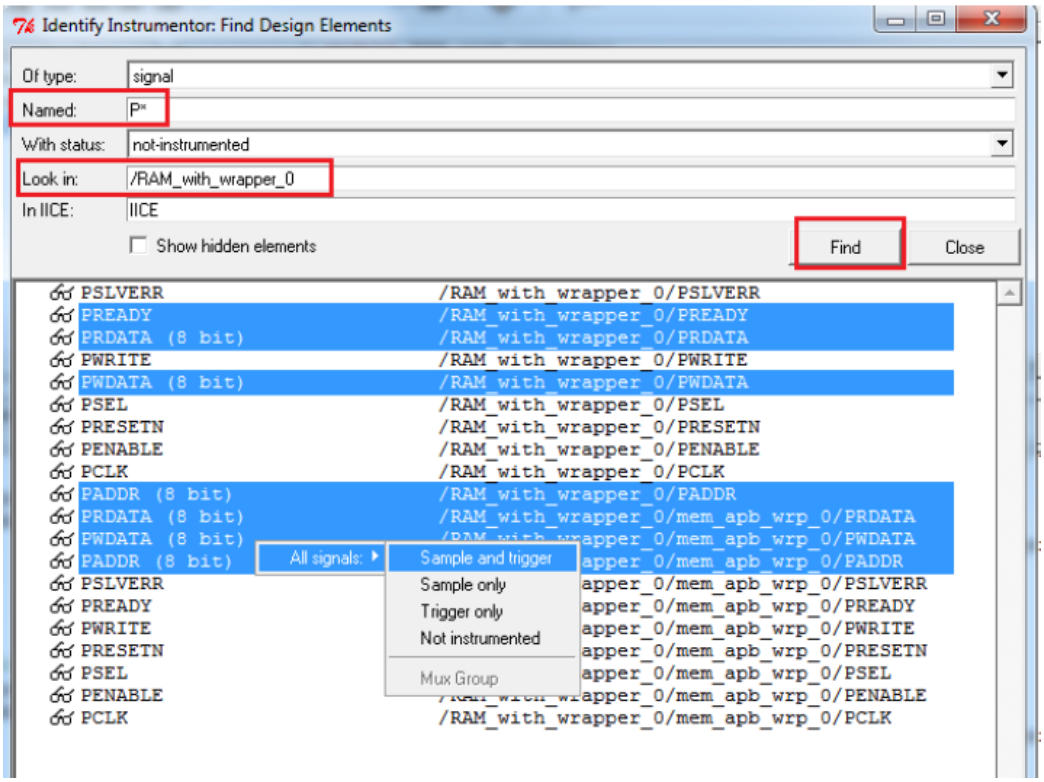


Рис. 3.87. Диалоговое окно поиска контрольных точек

После внесения всех настроек следует сохранить модифицированный дизайн для формирования списка соединений (рис. 3.88).

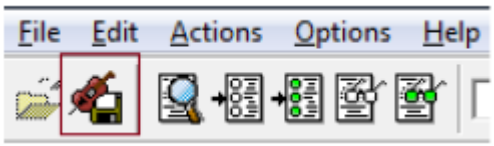


Рис. 118. Сохранение дизайна для отладки

После этого в иерархии проекта Synplify появится новая имплементация со значком лупы — та самая, которая будет задействована при отладке (рис. 3.89).

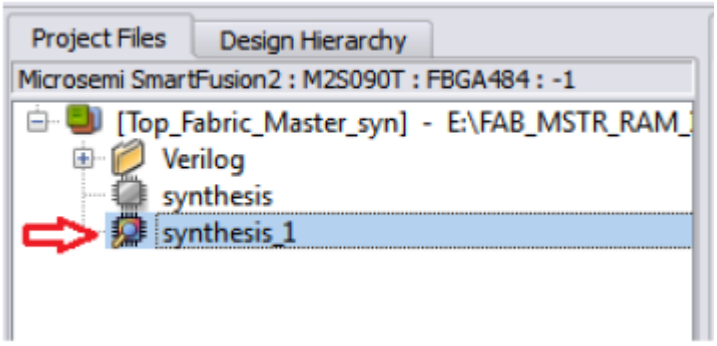


Рис. 3.89. Имплементация дизайна для отладки

Возвращаясь в Libero нужно скорректировать список исходных файлов для компиляции (Compile) так же, как в разд. 3.5 (рис. 3.90). Для компиляции следует использовать список связей только что созданной имплементации (в данном примере synthesis_1).

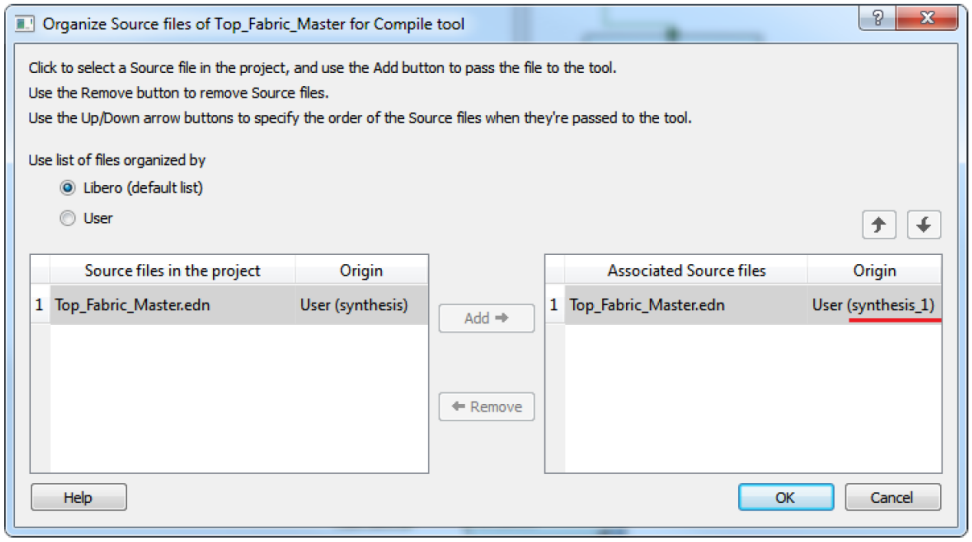


Рис. 3.90. Настройка исходных файлов для компиляции

После произведенных изменений нужно провести полный цикл Design Flow и запрограммировать устройство с помощью полученного файла прошивки. На этом этапе разработчик приступает собственно к отладке дизайна с помощью Identify Debugger. Для запуска инструмента нужно сделать двойной щелчок по Identify Debug Design (рис. 3.91).

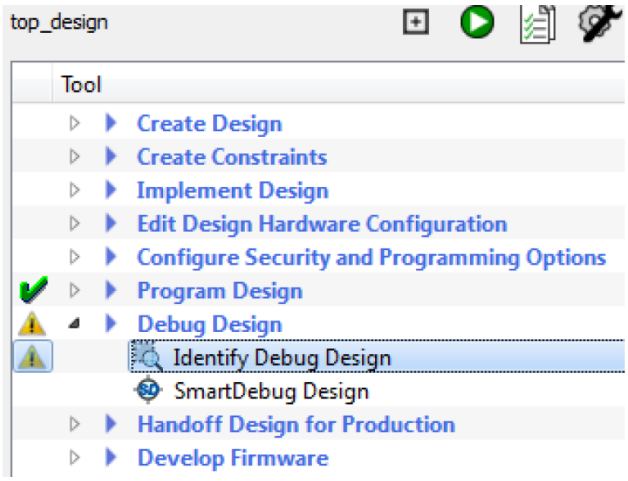


Рис. 3.91. Запуск Identify Debugger

Откроется окно отладчика, такое же, как на рис. 3.84. Содержание окна и контрольные точки будут соответствовать заданным на этапе подготовки. Далее необходимо задать активное состояние триггера, по которому данные будут отображаться пользователю. Для этого нужно выбрать иконку на панели быстрого доступа (рис. 3.92).

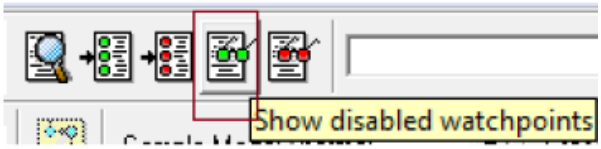


Рис. 3.92. Отображение отключенных точек наблюдения

Затем выбрать сигнал, который нужно назначить триггером, щелкнуть по нему правой кнопкой и выбрать пункт Set trigger expressions (рис. 3.93).

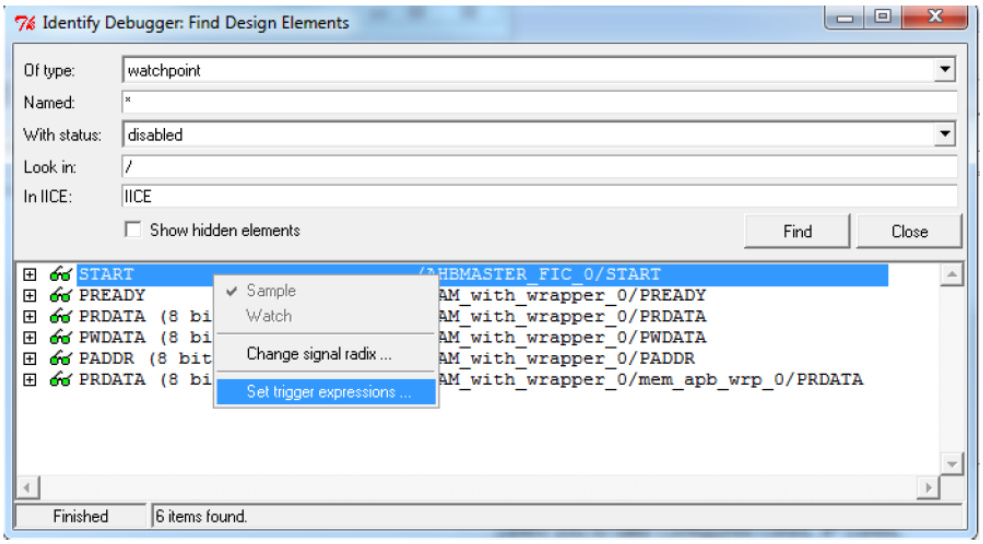


Рис. 3.93. Открытие окна задания триггера

В открывшемся окне нужно задать значение, при котором триггер срабатывает (рис. 3.94). Можно задать одно значение, по которому триггер срабатывает, либо задать два значения, переход между которыми будет условием срабатывания. Триггером лучше всего назначить внешний по отношению к ПЛИС переключатель, который удобно переводить из одного положения в другое быстро.

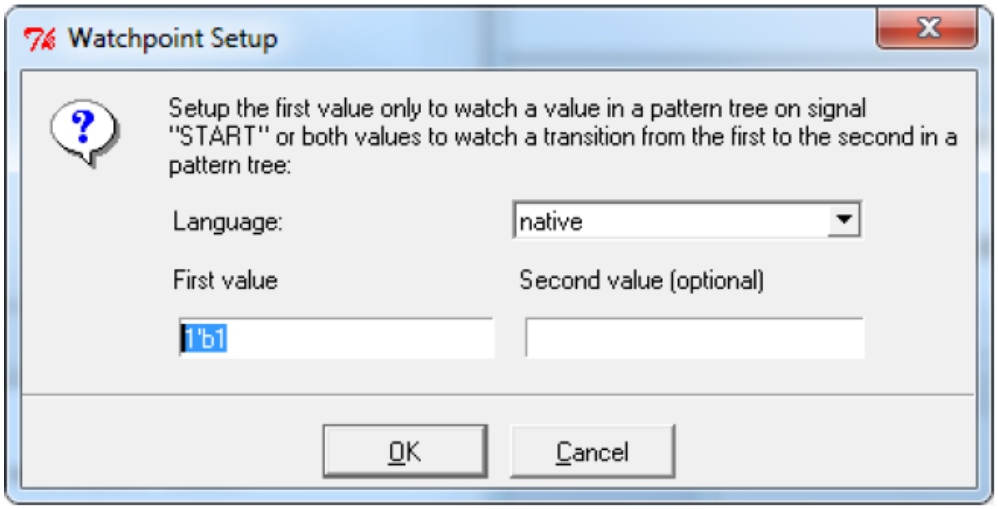


Рис. 3.94. Задание значения триггера

Стоит удостовериться, что система обнаружила программатор, для этого нужно зайти в Communication settings (рис. 3.95).

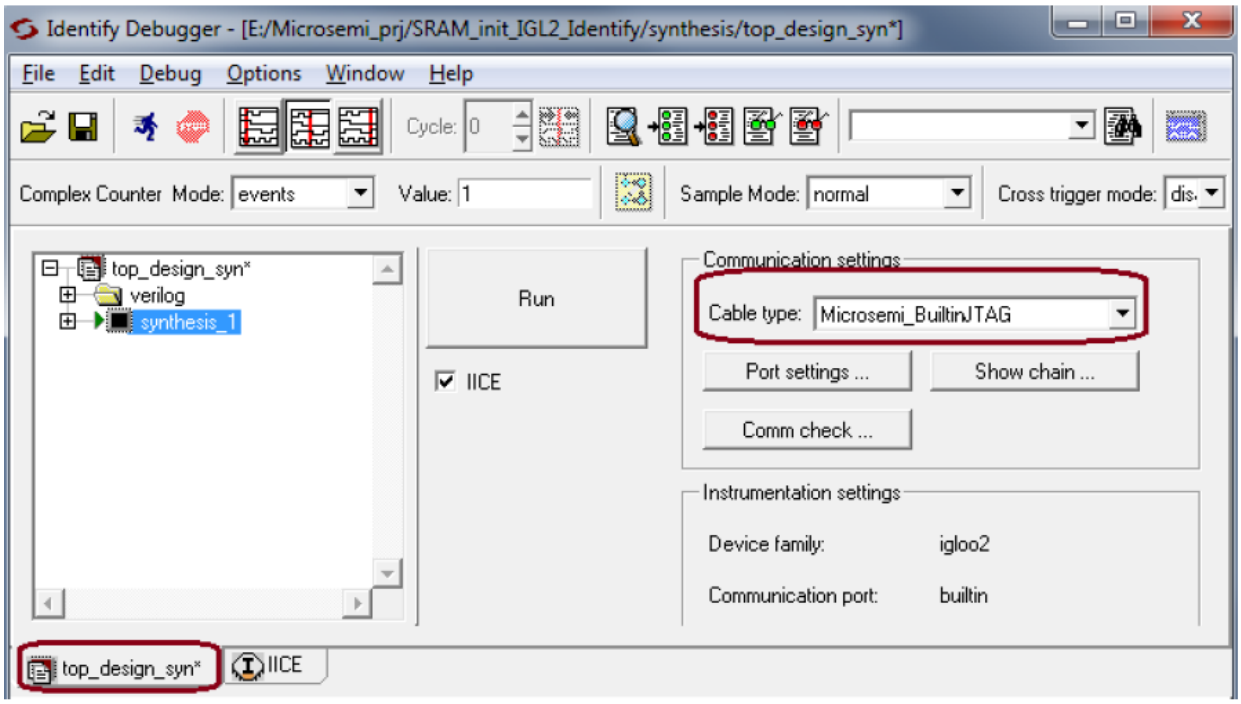


Рис. 3.95. Настройка связи с микросхемой

Теперь разработчик может запустить процесс отладки, нажав кнопку Run:

В области сообщений будет запись об ожидании срабатывания триггера (рис. 3.96).

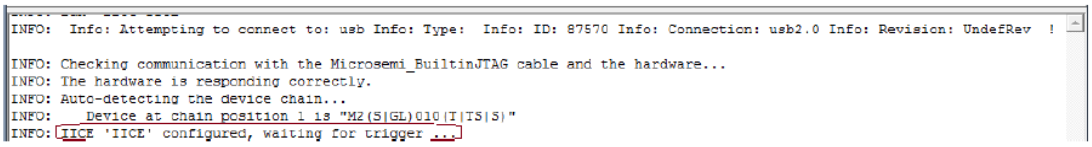


Рис. 3.96. Ожидание срабатывания триггера

После срабатывания триггера (изменения положения переключателя, связанного с триггером) разработчик может изучить временные диаграммы сэмплируемых синалов:

После нажатия отобразится окно временных диаграмм (рис.3.97).

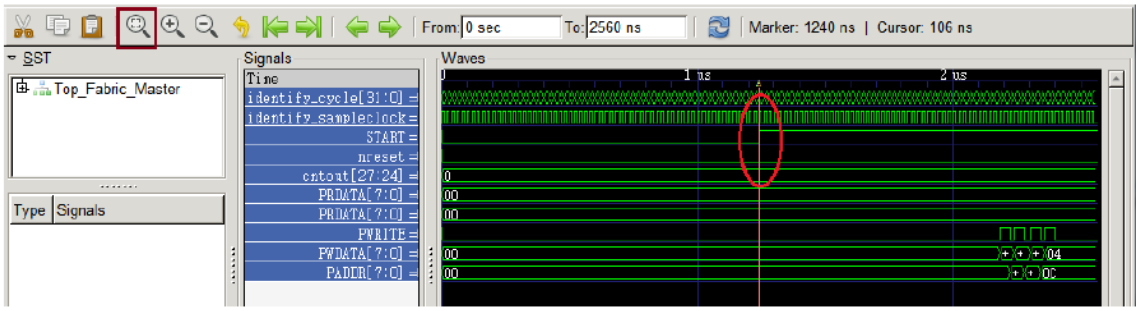


Рис. 3.97. Отображение временных диаграмм

На диаграмме имеется два дополнительных сигнала: identify_cycle отображает отсчёт до и после срабатывания триггера, а identify_sampleclock — тактовый сигнал сэмплирования.

4. Среда SoftConsole

4.1.Настройки взаимодействия, место в Design Flow

Среда автоматизированного проектирования SoftConsole поставляется вместе с Libero и является мощной и гибкой средой разработки встроенного программного обеспечения для микроконтроллера. SoftConsole может использоваться совместно с hard IP ARM Cortex-M3 в составе SmartFusion2, либо для разработки ПО для софт-микроконтроллеров в специальных версиях ПЛИС ProASIC3 и IGLOO.

Принцип взаимодействия разработки прошивки ПЛИС и ПО для встроенного микроконтроллера демонстрирует рис. 4.1.

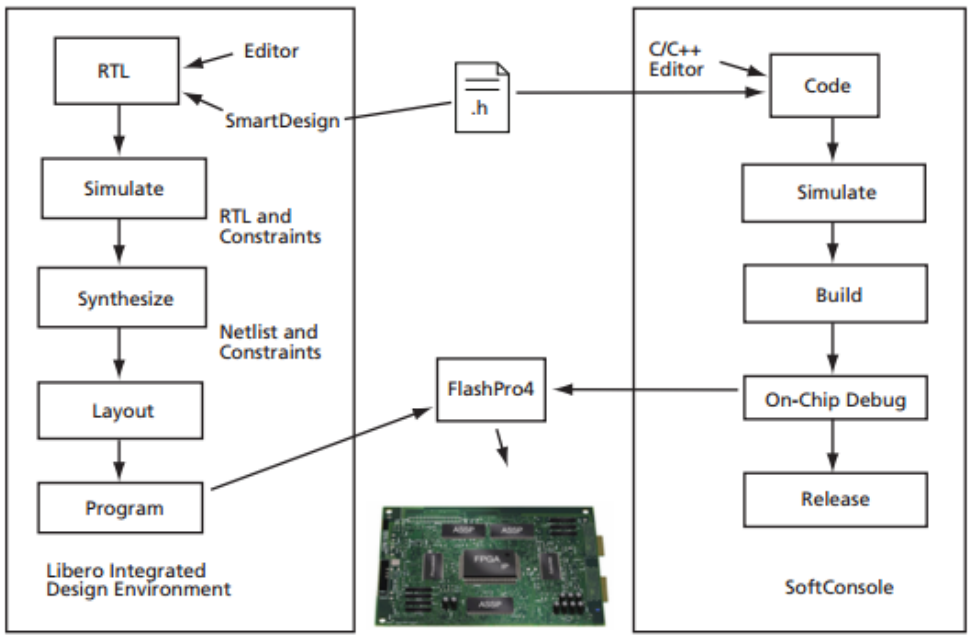


Рис. 4.1. Взаимодействие разработки прошивки ПЛИС и встроенного ПО

В качестве промежуточных уровней при взаимодействии с оборудованием используются уровень абстракции от оборудования (hardware abstraction layer, HAL), драйверы периферийных узлов, опционально — операционные системы и связующее ПО, на верхушке иерархии стоит пользовательская программа.

Для начала работы с SoftConsole нужно убедиться в доступности всех связующих средств, для этого необходимо открыть Firmware Catalog в группе программ Microsemi в меню Пуск (рис. 4.2).

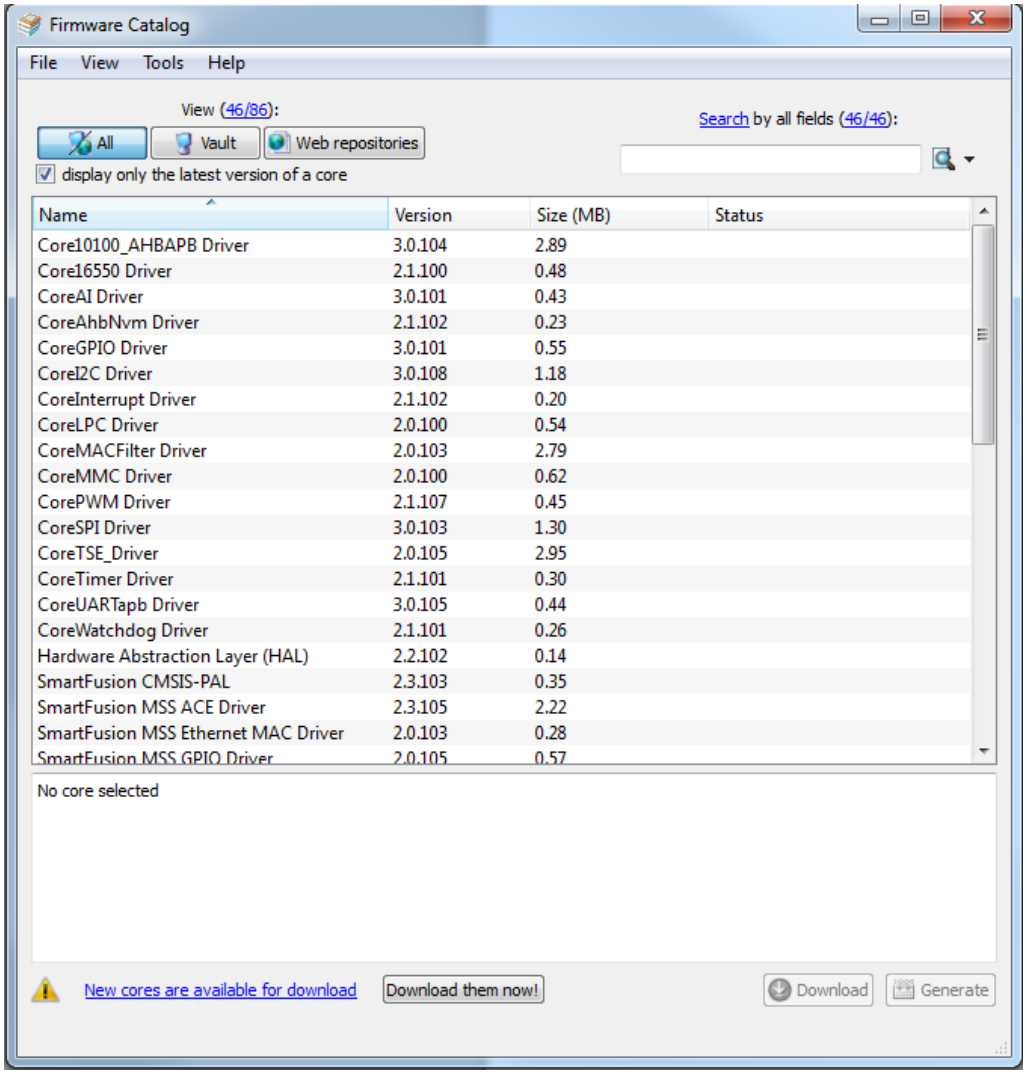


Рис. 4.2. Firmware Catalog

Каталог включает в себя средства обеспечения взаимодействия оборудования и пользовательской программы, драйверы периферийных устройств. Если в нижней части каталога отображается запрос на обновление (рис. 4.3), стоит обновить все компоненты до актуального состояния.

Рис. 4.3. Просьба обновить каталог

Доступ к написанию кода программы для микроконтроллера становится доступным на последней стадии Design Flow, когда система спроектирована, ПЛИС сконфигурирована программатором. Для взаимодействия со средой SoftConsole служат последние пункты (рис. 4.4).

Рис. 4.4. Пункты, отвечающие за написания встроенного ПО

Вызов Configure Firmware Cores открывает доступ к выбору драйверов, которые следует включить в проект, определяется это тем набором периферийных устройств, которые необходимы в проекте (рис. 4.5).

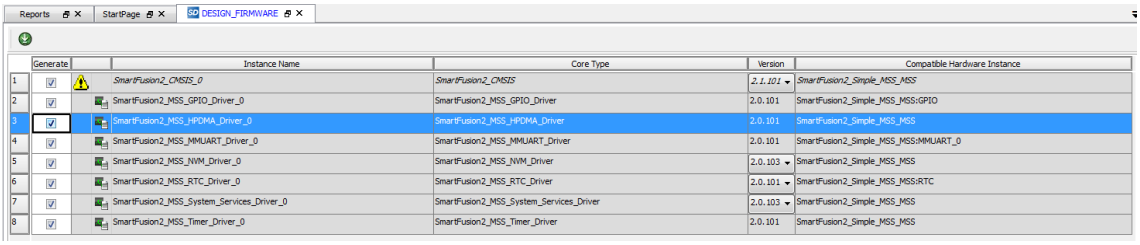


Рис. 4.5. Выбор необходимых драйверов периферийных узлов

Снятие галочки напротив драйвера означает его исключение, по умолчанию все драйверы включены. Также имеется возможность сгенерировать демонстрационный проект для каждого из периферийных узлов, для этого нужно нажать правой кнопкой мыши на имя драйвера и проследовать по меню Generate Sample Project -> Cortex-M3 -> SoftConsole -> тип проекта (рис. 4.6).

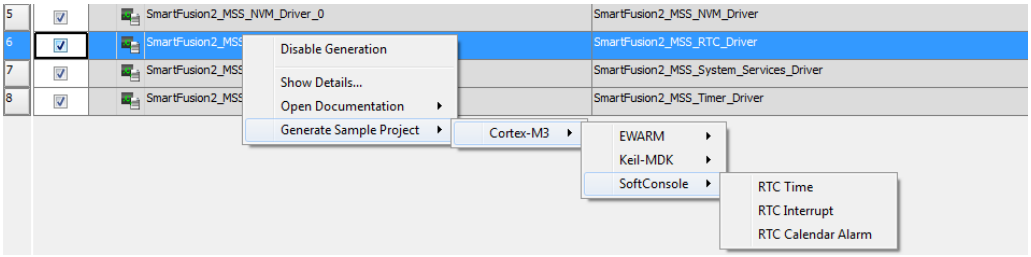


Рис. 4.6. Генерация тестового проекта

Рекомендуется начинать разработку новой программы именно с демонстрационного проекта, это значительно ускорит процесс проектирования ПО.

По окончании генерации система выдаст отчёт, папку с файлами проекта можно наблюдать во владке Files (рис. 4.7).

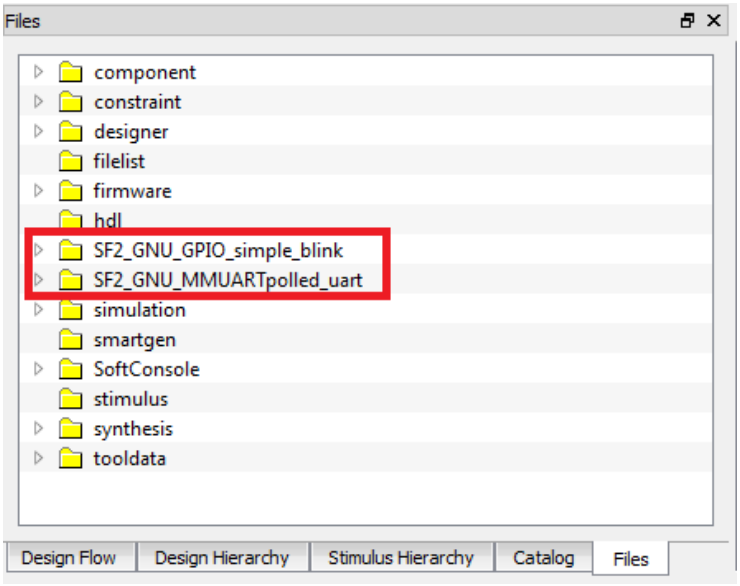


Рис. 4.7. Файлы тестовых проектов в корневой папке

Когда всё готово, разработчик генерирует проект, нажав Export Firmware, при этом появится диалоговое окно (рис. 4.8).

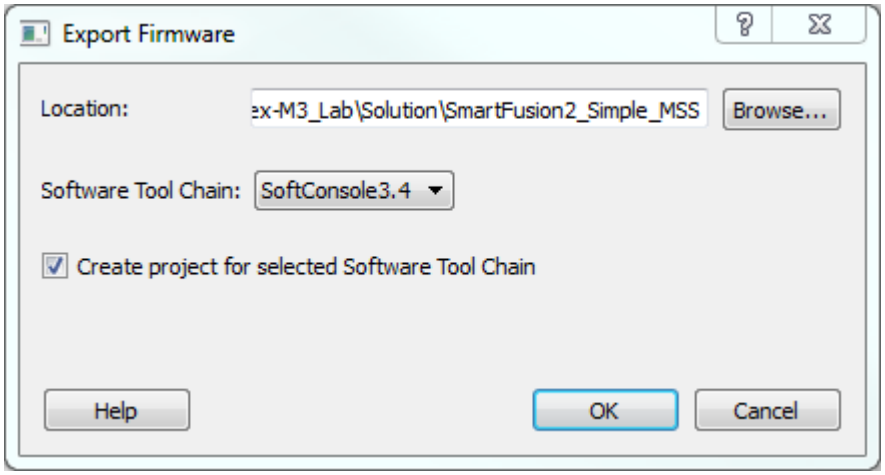


Рис. 4.8. Диалоговое окно генерации проекта

После нажатия ОК проект будет экспортирован, о чём пользователь будет уведомлен диалоговым окном и сообщением.

4.2. Интерфейс пользователя

Открыть SoftConsole можно из группы программ Microsemi в меню Пуск, перед открытием основного окна откроется диалоговое окно выбора проекта (Workspace), в котором следует выбрать папку со своим сгенерированным проектом (рис. 4.9).

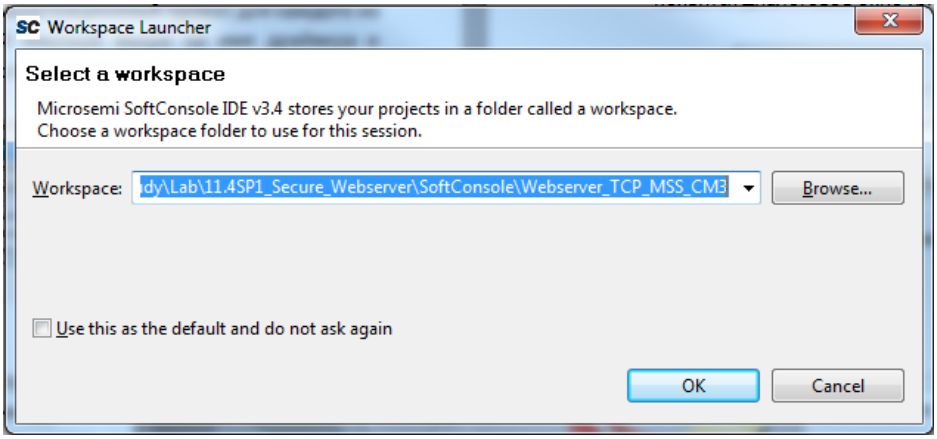


Рис. 4.9. Открытие проекта SoftConsole

После нажатия ОК откроется главное окно с выбранным проектом (рис. 4.10). В левой части окна располагается иерархия частей проекта, в центральной области редактор кода, в правой части структурное представление кода (символы) и в нижней — окно сообщений и консоль.

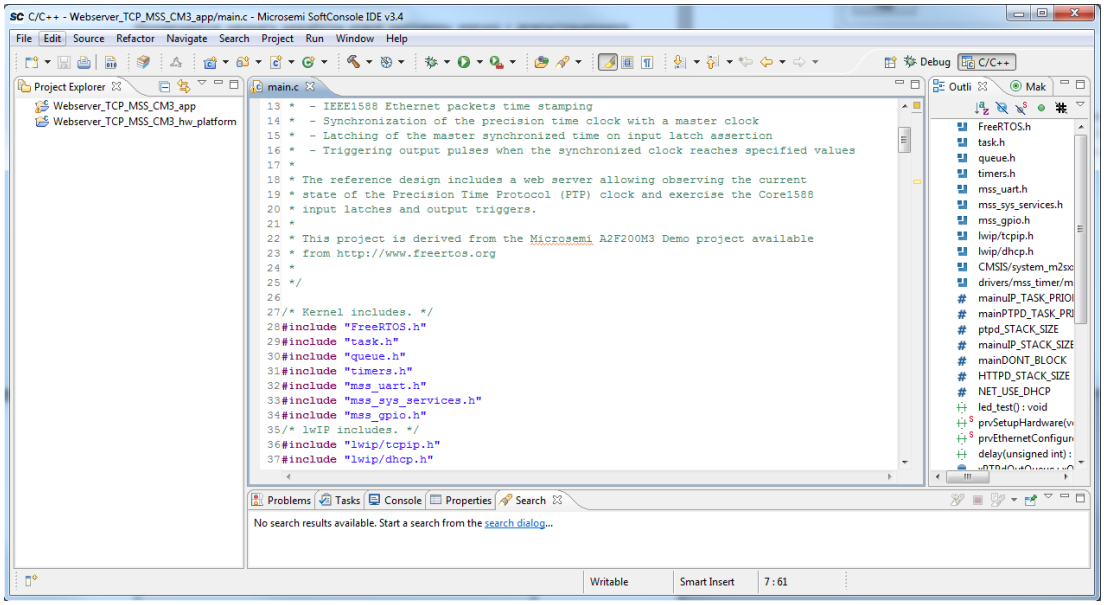


Рис. 4.10. Главное окно SoftConsole

Как можно заметить, проект содержит два компонента:

- *_app;
- *_hw_platform.

Первый отвечает за пользовательское наполнение приложения: пользовательский код, прикладные библиотеки, стороннее ПО, операционные системы и т.д. Второй компонент включает в себя библиотеки и средства взаимодействия с оборудованием, например заголовочные файлы конфигурации отдельных узлов и исходные файлы с функциями доступа к периферийным устройствам.

В общем случае самостоятельно править файлы hw_platform не нужно. Все действия по написанию программы разработчик выполняет в app. Основные действия по компиляции и сборке проекта выполняются из меню Project. Build All компилирует весь проект. Сделав правый щелчок по файлу разработчик может компилировать файлы в отдельности (рис. 4.11).

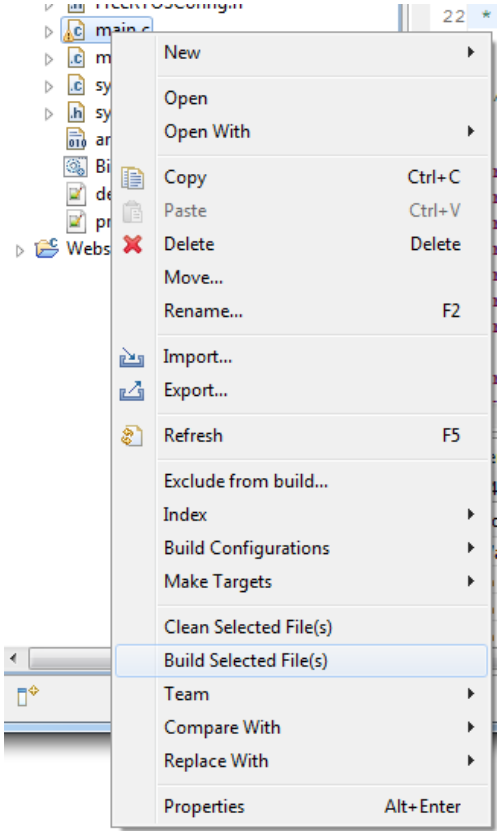


Рис. 4.11. Сборка отдельного файла и меню управления файлами

Предупреждения и ошибки компиляции отображаются в области лога (рис. 4.12).

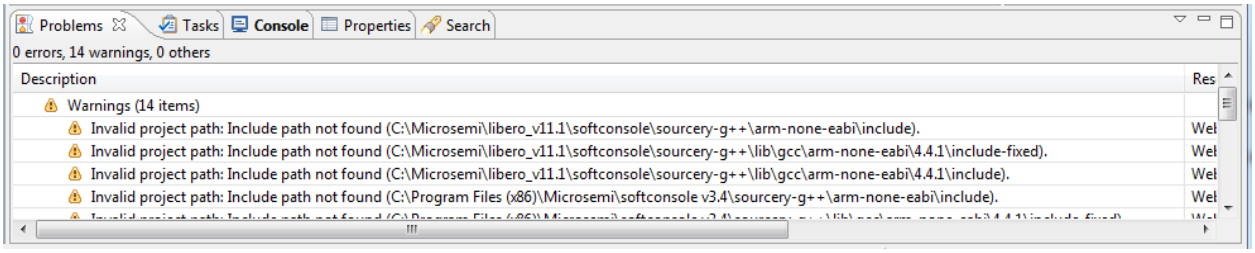


Рис. 4.12. Предупреждения и ошибки в области лога

Щелкнув правой кнопкой мыши по компоненту проекта и выбрав Пункт Propeties разработчик открывает окно с множеством специфических настроек проекта (рис. 4.13).

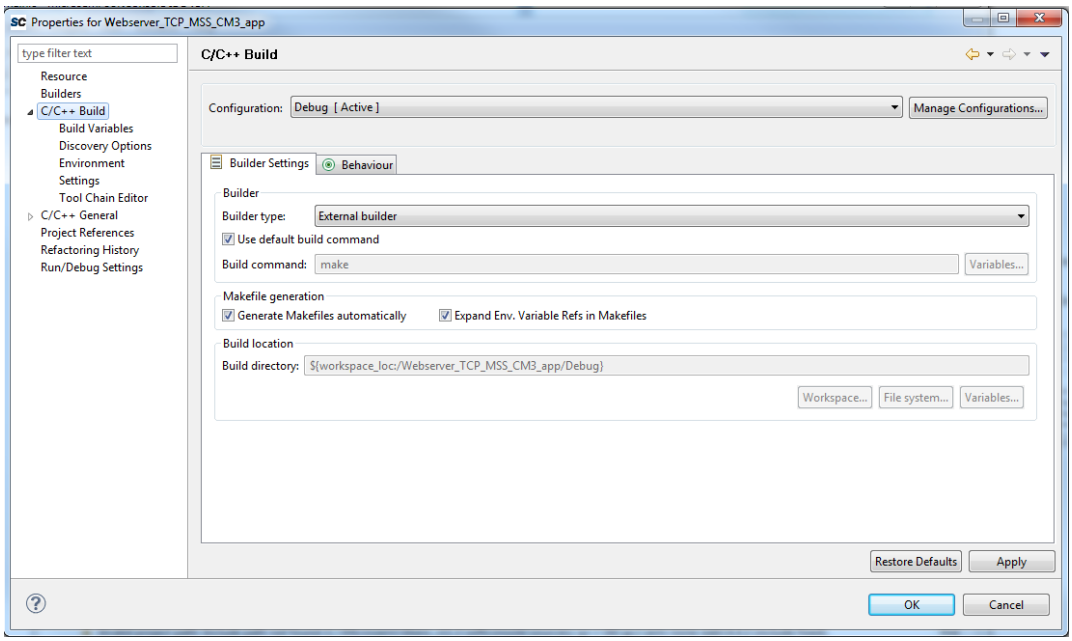


Рис. 4.13. Настройки проекта

В том числе в этом окне выбирается конфигурация сборки (см. пункт Configuration на рис. 4.13):

- Debug;
- Release.

Как следует из названий, первый вариант предназначается для итерационной отладки приложения, а второй собирает готовый вариант для прошивки в энергонезависимую память. Связанные с этим выбором настройки находятся в C/C++ Build (рис. 4.14).

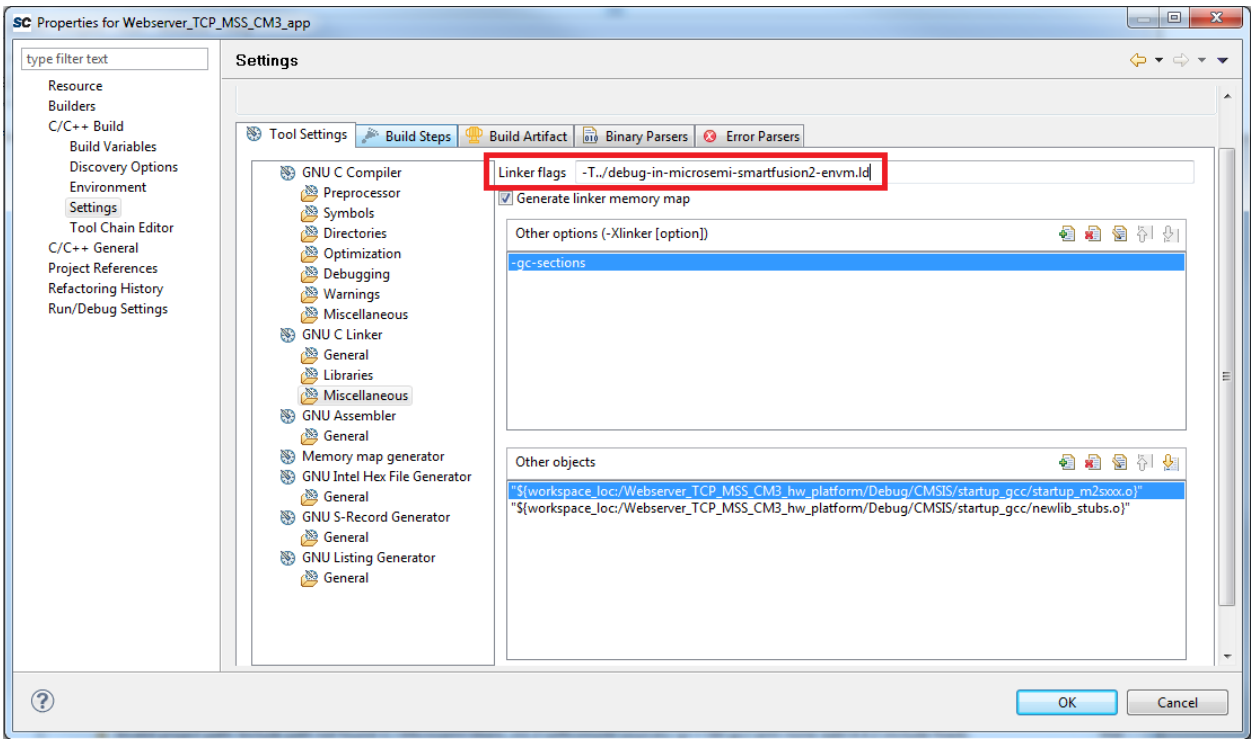


Рис. 4.14. Настройки флагов линкера

В данной строке пишется флаг, который указывает на память, в которой будет происходить отладка программы. debug-in-microsemi-smartfusion2-sram.ld указывает на отладку в памяти SRAM, что используется по умолчанию. Отладка во Flash-памяти используется, например, когда объёма СОЗУ не хватает, чтобы вместить программу, но крайне нежелательна, так как eNVM имеет ограниченное количество циклов перезаписи.

Все подробности по работе в SoftConsole можно найти по ссылке: http://www.microsemi.com/products/fpga-soc/design-resources/design-software/softconsole.

4.3. Процесс отладки

После проектирования и описания настроек проекта можно приступить к процессу отладки. Для этого нужно создать профиль отладки, для этого служит Run -> Debug Configurations (рис. 4.15).

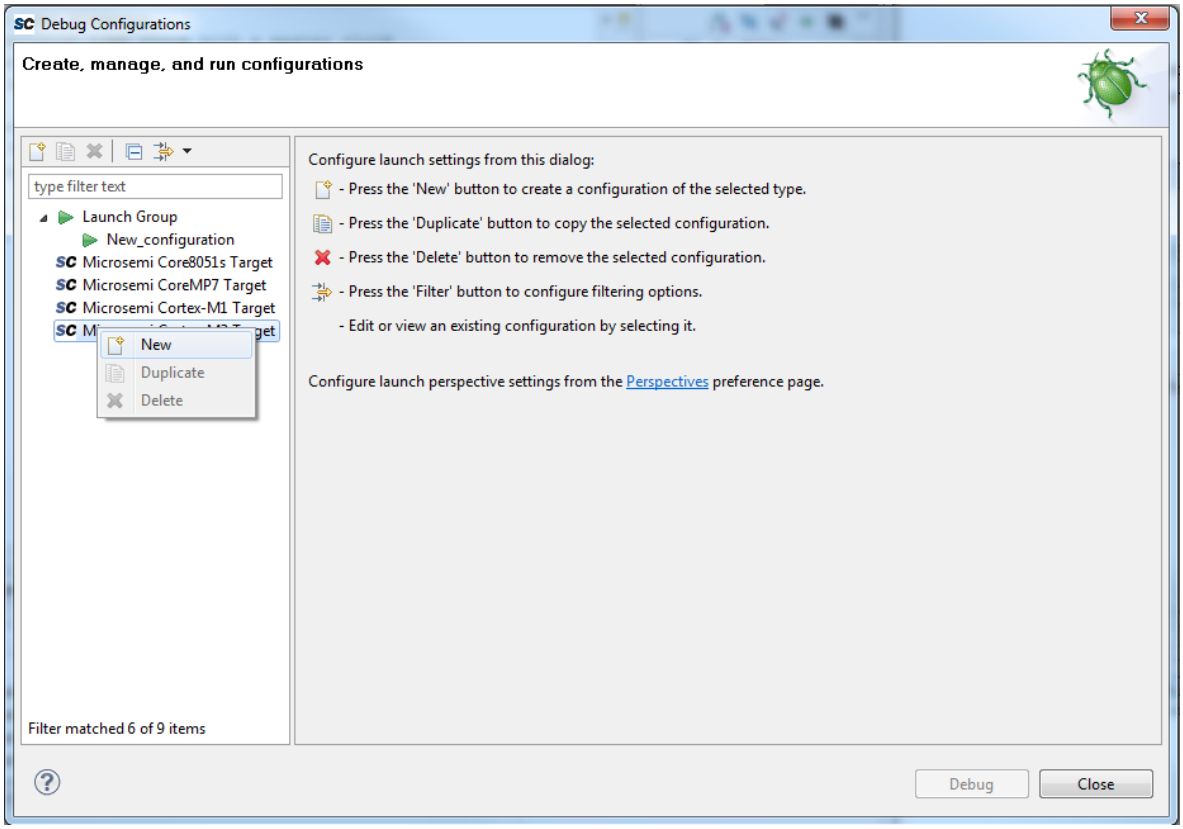


Рис. 4.15. Создание профиля отладки

В открывшемся окне нужно указать, какую программу предстоит отлаживать, для этого служит кнопка Search Project (рис. 4.16).

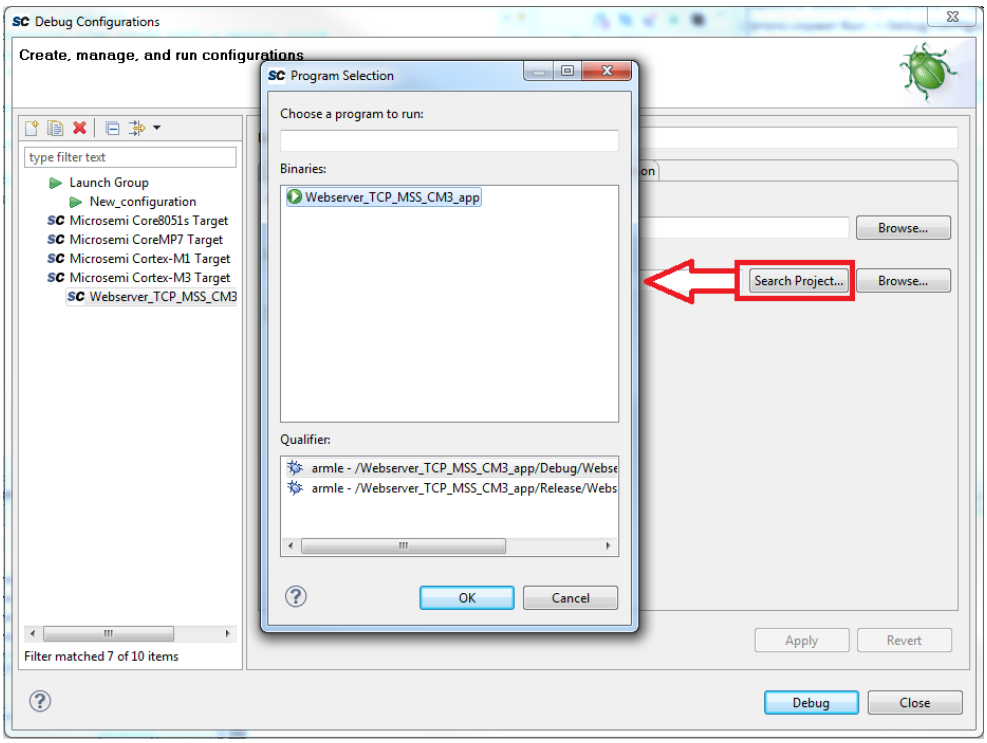


Рис. 4.16. Указание проекта для отладки

Для удобства можно добавить значок Debug в избранное, для этого на вкладке Common нужно отметить Debug (рис. 4.17).

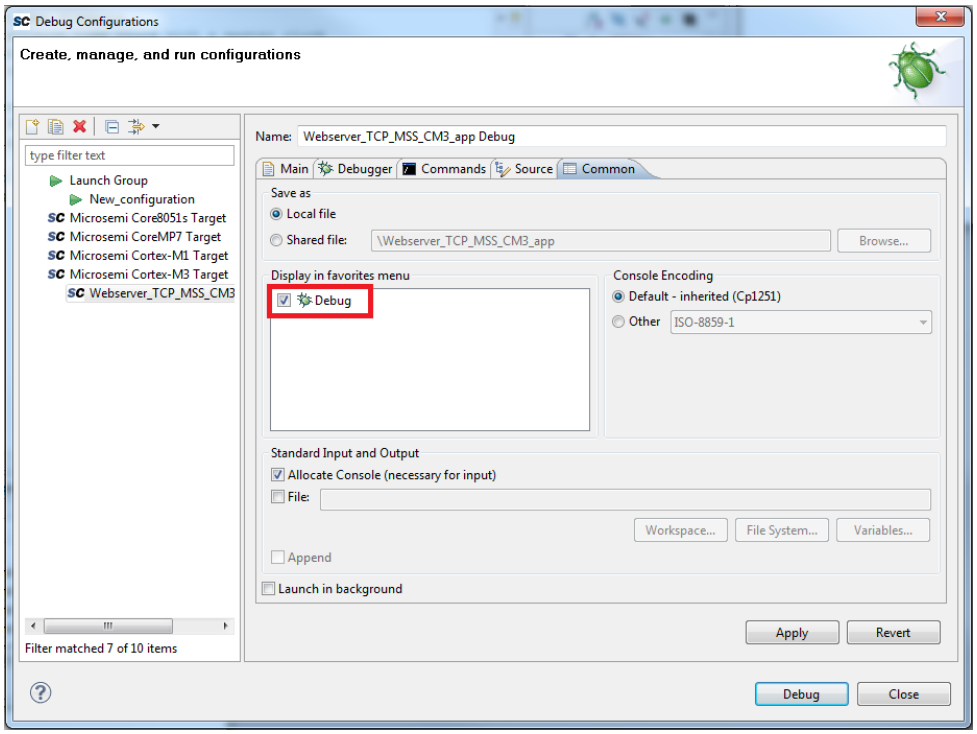


Рис. 4.17. Добавление Debug в избранное

Для сохранения изменений нужно нажать кнопку Apply. Удостоверившись, что на микросхему подано питание и кабель JTAG соединен с платой, можно начинать отладку, нажав кнопку Debug.

Появившееся окно предлагает перейти в перспективу отладки (поменять раскладку инструментов, обеспечив доступ к органам управления отладкой), следует нажать Yes и дождаться загрузки проекта в устройство (рис. 4.18). Переключение между перспективами разработки и отладки осуществляется переключателям в правом верхнем углу (Debug и C/C++)

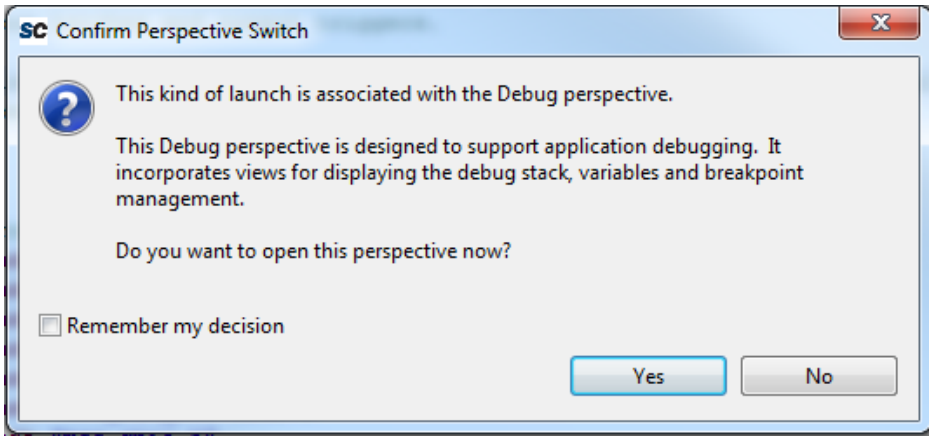


Рис. 4.18. Переключение в перспективу отладки

Замечание: в случае SoftConsole не стоит пугаться текста красного цвета в области сообщений — большинство из них не означают ошибку.

Как было обозначено выше, в перспективе отладки расположение инструментов поменяется, также появятся новые инструменты, которые отвечают за управление отладкой (рис. 4.19).

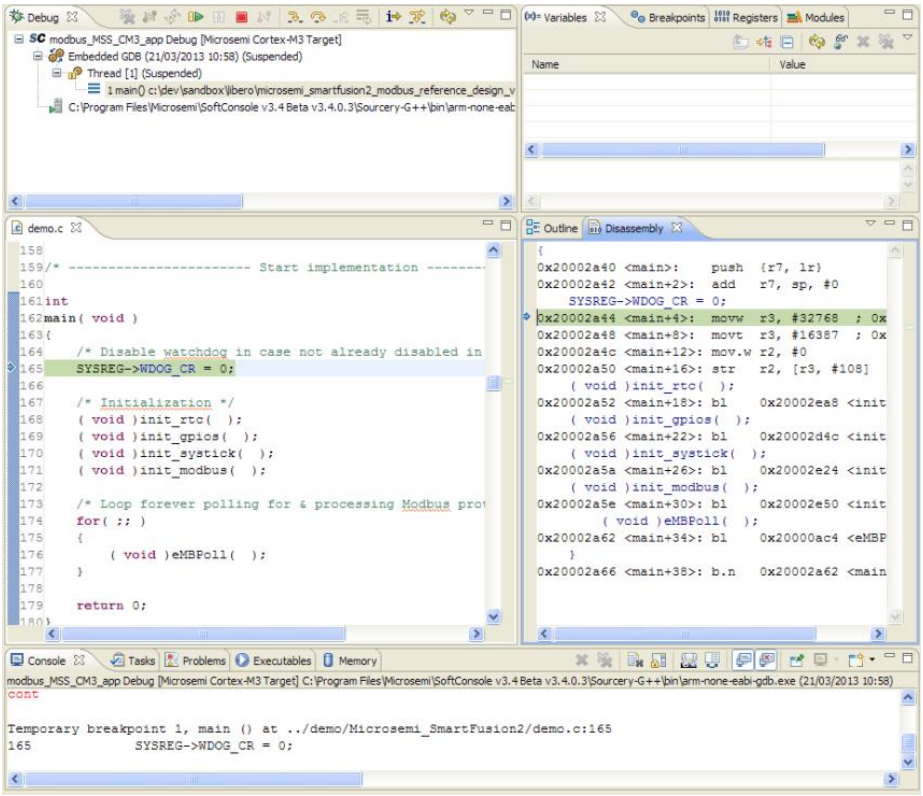


Рис. 4.19. Перспектива отладки

В левой верхней области показана иерархия проекта, конечными точками которой служат выполняемые потоки программы. В правой верхней области отображаются значение переменных, регистров процессора, контрольные точки и модули.

В центре располагается область кода, в котором отображаются контрольные точки (breakpoint), расставляя которые разработчик отмечает места кода, в которых нужно приостановить процесс выполнения. Есть возможность просмотра низкоуровневого кода на языке ассемблера (справа от кода на С). В той же области располагается информация о символах в исходном коде в отдельной вкладке (Outline).

Нижняя область, как и ранее, отображает ход выполнения процесса, сообщения и ошибки.

После инициализации и загрузки программы в память, процесс останавливается на первой контрольной точке (выставляется автоматически в начале функции main). Для запуска процесса служит кнопка . Процесс будет идти в соответствии с алгоритмом, заложенным в программу, останов будет иметь место, если счётчик команд достигнет очередной контрольной точки. Временная остановка выполнения программы производится с помощью кнопки Остановить процесс выполнения программы со сбросом текущего состояния можно, нажав кнопку .

Отладку можно проводить пошагово, для этого служит набор кнопок , которые выполняют соответственно шаг со входом внутрь функций, шаг без входа в функции и шаг с выходом из текущей функции.

Текущую сессию отладки можно остановить и перезапустить с помощью команды Terminate and Relaunch (рис. 4.20).

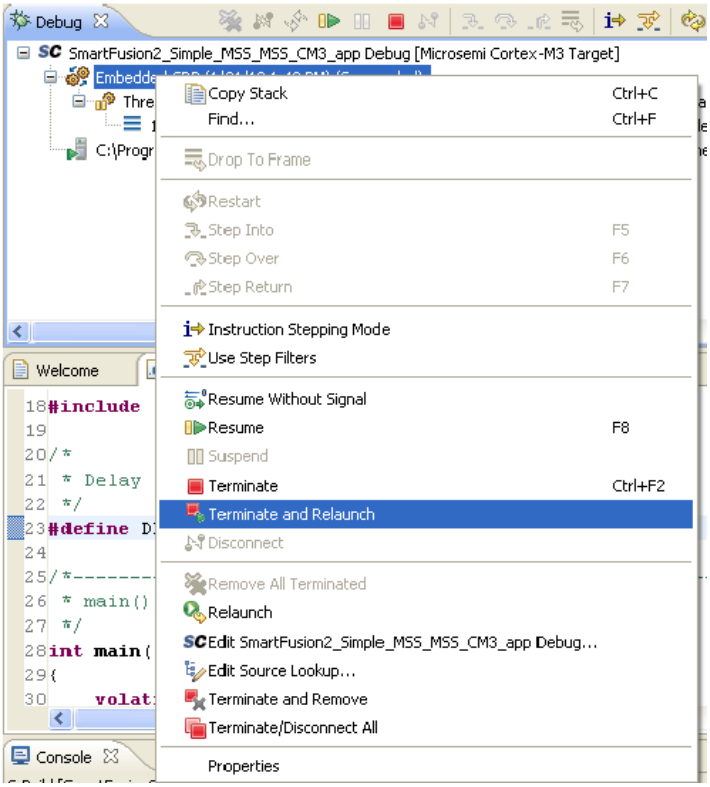


Рис. 4.20. Перезапуск сессии отладчика

При этом заново произойдет инициализация и загрузка программы в память. Командой Terminate and Remove пользователь может закончить и удалить текущую сессию отладки.

5. Примеры проектов в среде Libero SoC

Microsemi предоставляет широкий спектр демонстрационных проектов и обучающих материалов для каждого семейства микросхем ПЛИС и СнК. Данные материалы можно скачать на сайте http://www.microsemi.com/products/fpga-soc/soc-fpga в разделах, посвященных семействам и отладочным наборам на основе конкретных кристаллов. В текущей главе будут рассмотрен ряд руководств по созданию проектов для начинающих, будут описаны процессы создание наиболее популярных решений, в ходе прочтения разработчик будет создавать дизайн «с нуля» и вплоть до прошивки ПЛИС и проверке функциональности.

5.1. PCIe Control Plane Lab

В данном разделе будет рассмотрена возможность аппаратной реализации PCI®express в ПЛИС IGLOO2 и пример его использования для управления и мониторинга состояния отладочной платы IGLOO2 Evaluation Kit. Прочитав данный раздел, читатель сможет создать всю систему «с нуля», в качестве опорного материала на сайте корпорации предоставляется полностью реализованный проект для Libero последней версии.

После выполнения нижеописанных действий вы сможете:

- Создавать проект в Libero SoC;
- Создавать тестовые воздействия для симуляции;
- Проводить симуляцию дизайна;
- Генерировать файл для программирования;
- Запускать PCIe приложение.

Структурная схема проекта показана на рис. 5.1.

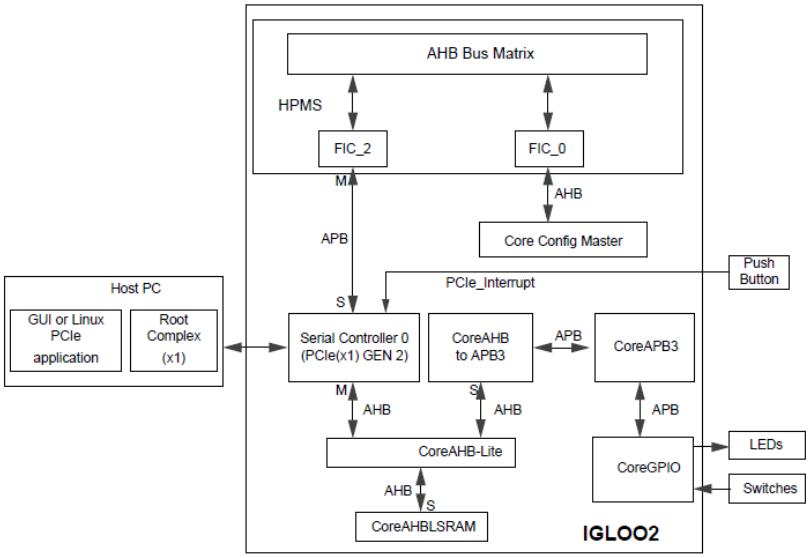


Рис. 5.1. Структурная схема проекта

Основным компонентом системы является блок сериализатора десериализатора (SERDES) и соответствующий IP-модуль SERDESIF, который позволяет настроить блоки SERDES для обеспечения требуемой функциональности. Блоки SERDES включают в себя полностью аппаратное исполнение PCIe, полностью соответствующее спецификациям 2.0 и 1.1. То есть, разработчику не требуется проектировать иерархию уровней PCIe, либо покупать это решение, необходимо реализовать только прикладной уровень. Такой подход значительно экономит ресурсы ПЛИС и увеличивает производительность системы.

В данном примере устройство вазимодействует с ПК по PCIe, получая команды, которые задаются пользователем в графической среде (GUI). SERDESIF_0 сконфигурирован как PCIe 2.0 x1 GEN2. С помощью шин AMBA SERDES связан с периферийными устройствами на кристалле, среди которых память SRAM (CoreAHBLSRAM), ядро ввода-вывода общего назначения (CoreGPIO). Последнее управляет состоянием светодиодов на плате и получает данные с группы переключателей. Кнопка на отладочной плате связана со сходом прерывания PCIe.

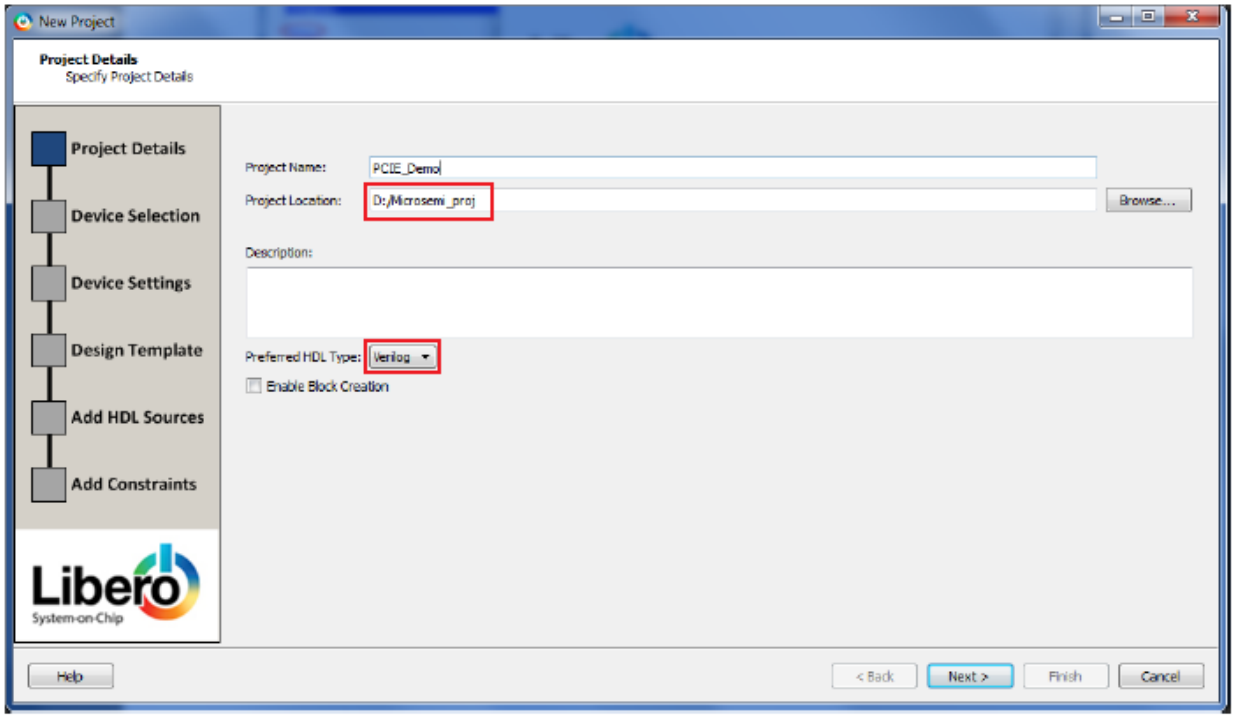
Шаг 1: Создание проекта Libero SoC

Откройте Libero SoC в группе программ Microsemi в меню Пуск. Начните создание нового проекта, щелкнув по New… (рис. 5.2).

Рис. 5.2. Менеджер проектов Libero SoC

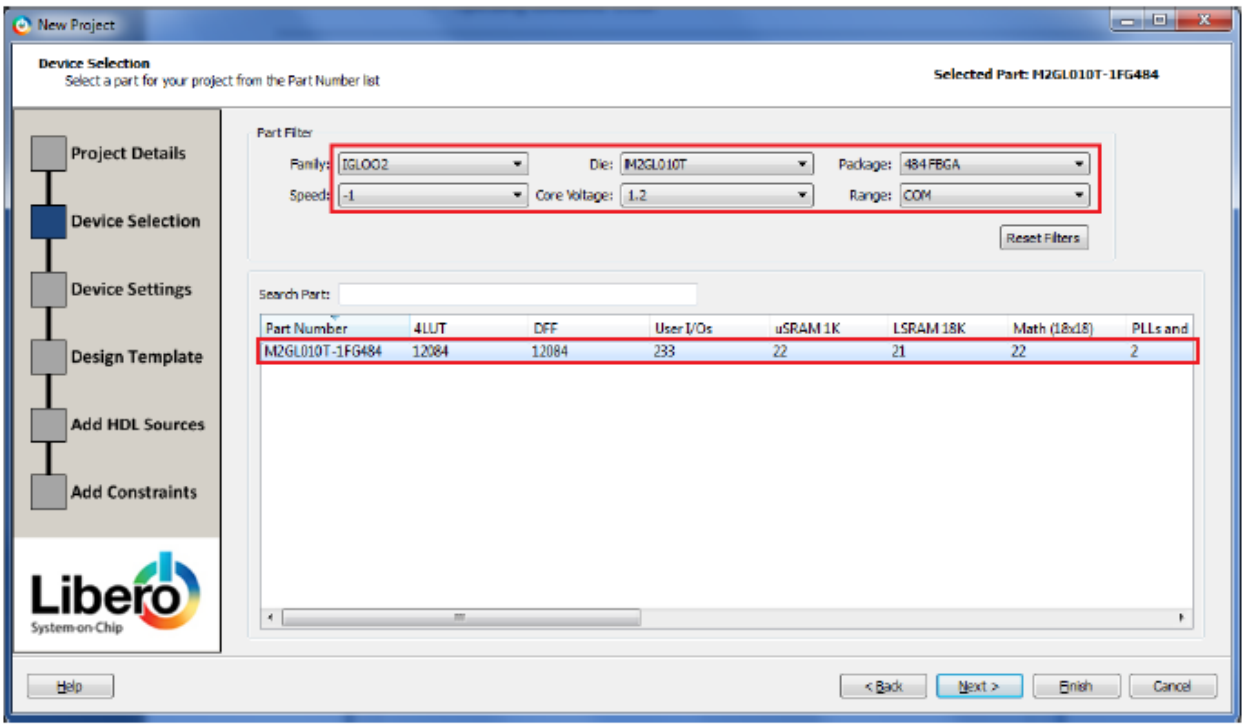
В открывшемся окне введите следующую информацию о проекте:

- Project Name: PCIE_Demo
- Project Location: Выберите расположение проекта (например D:/microsemi_prj)
- Preferred HDL Type: Verilog или VHDL

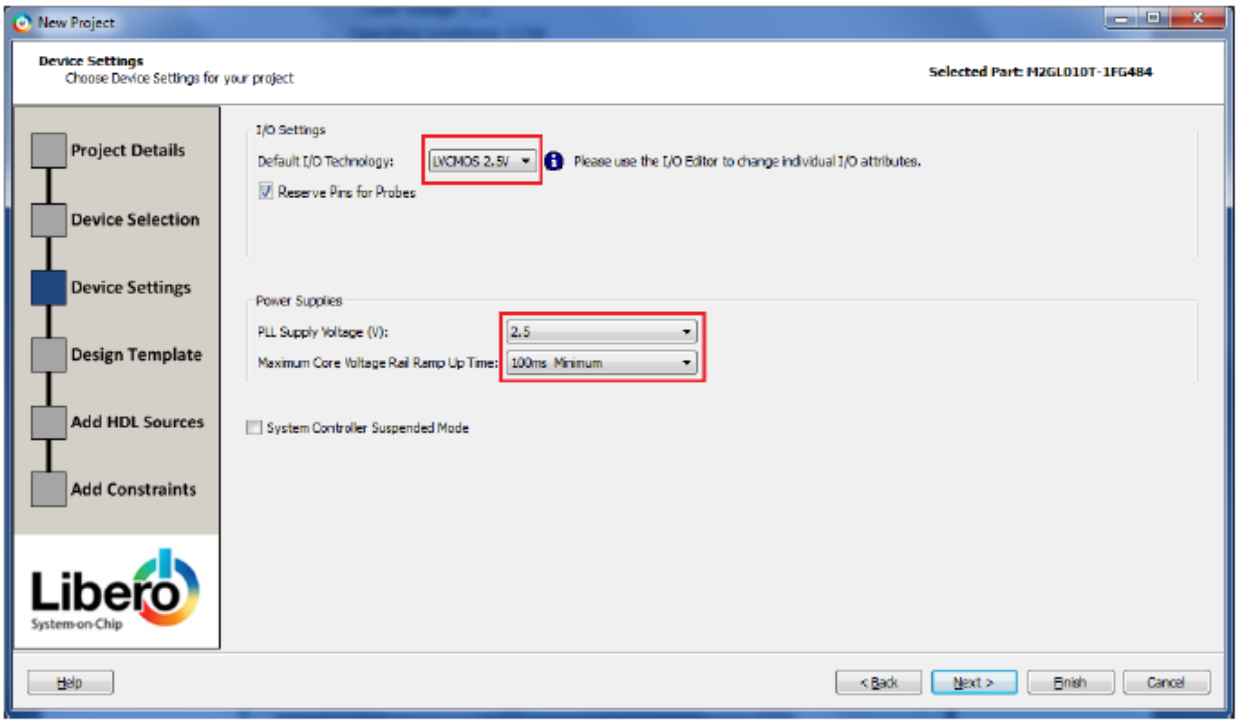


Нажмите Next. Введите следующие настройки:

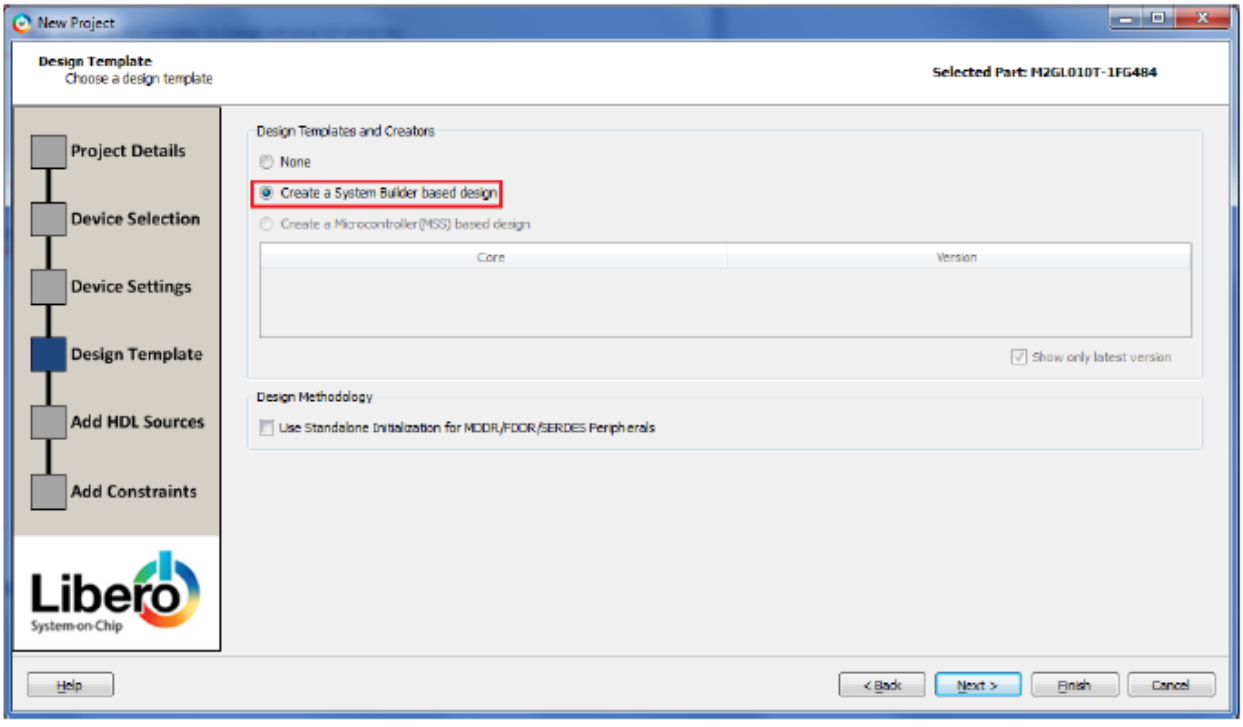
- Family: IGLOO2
- Die: M2GL010T
- Package: 484 FBGA
- Speed: -1
- Core Voltage: 1.2
- Range: COM



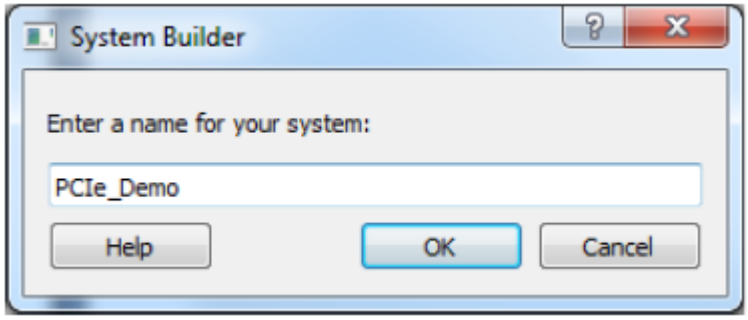
Нажмите Next. На следующем шаге введите настройки в соответствии с рисунком.



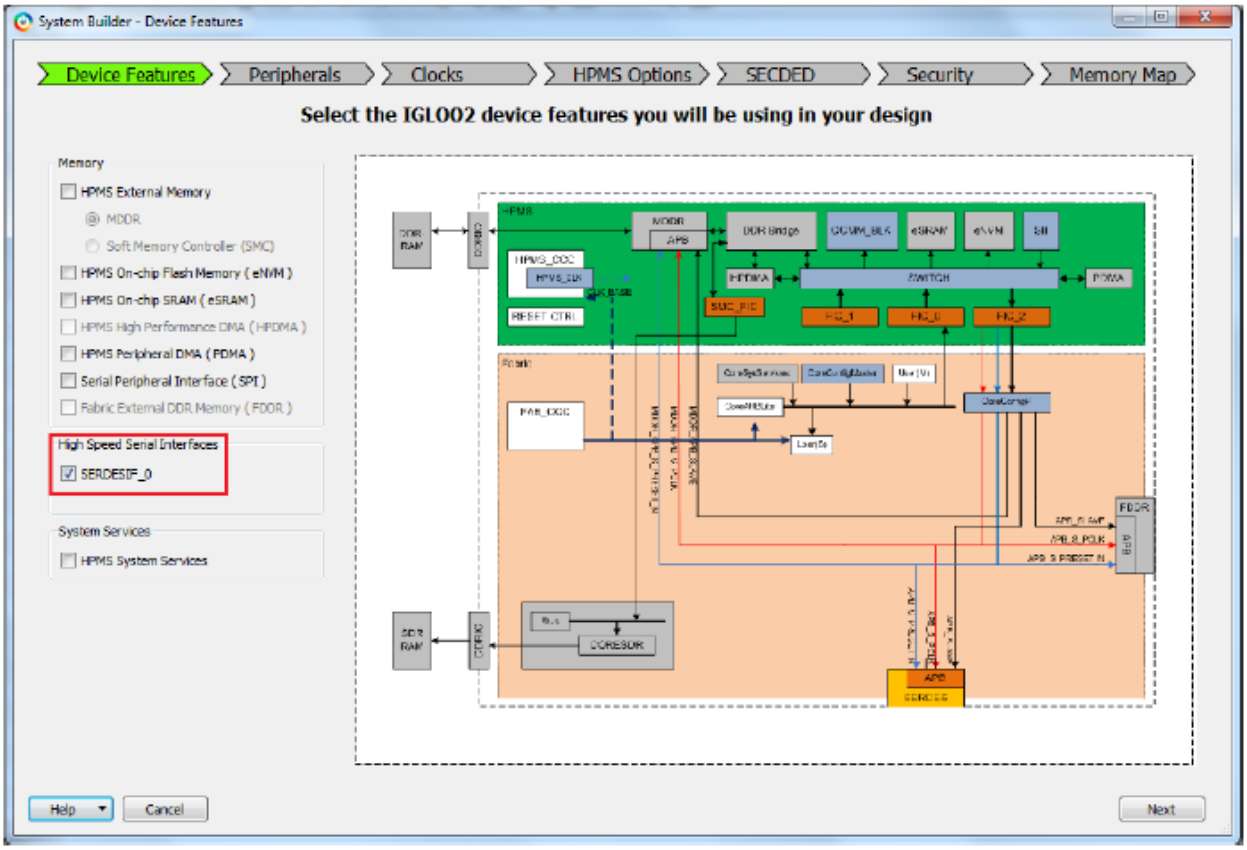
Нажмите Next. Отметьте Create a System Builder based design.



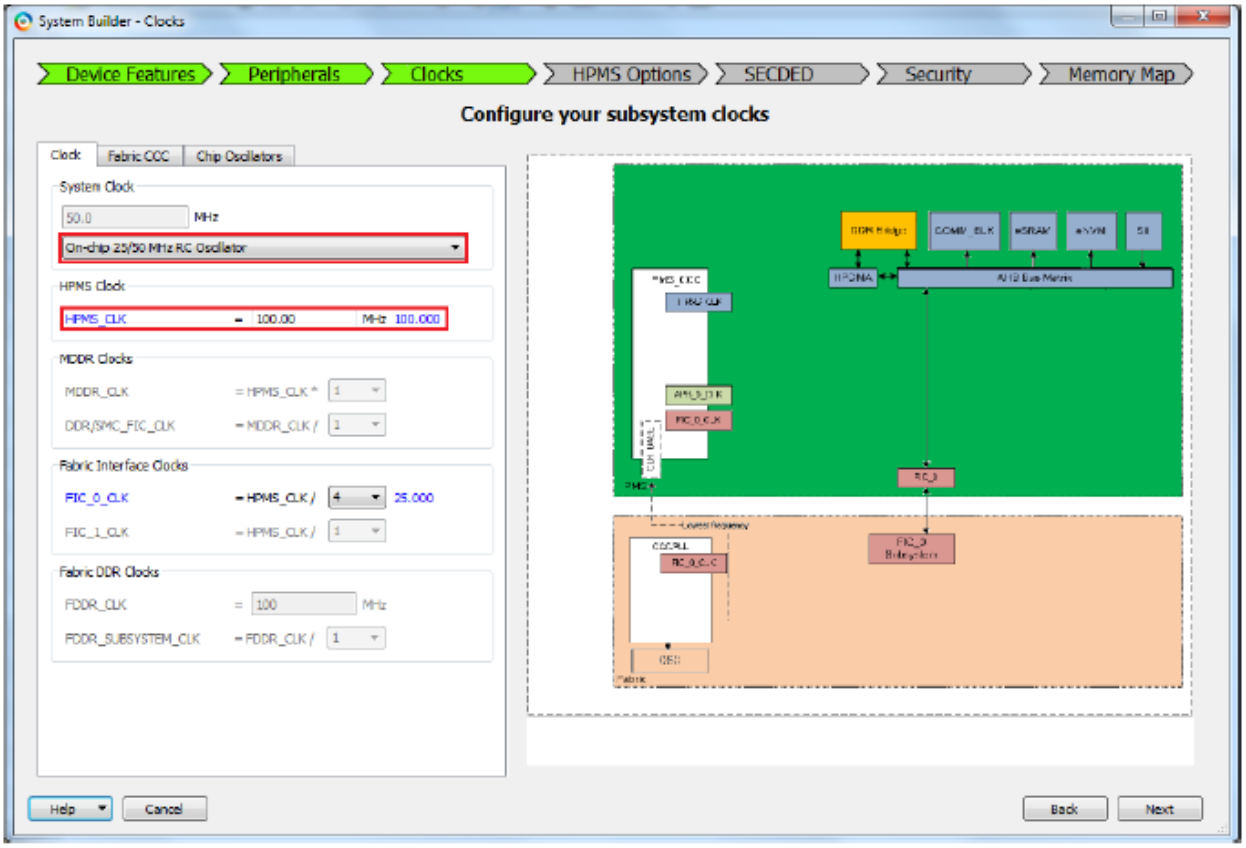
Далее нажимайте Next, пока не попадете на финальную страницу, затем нажмите Finish. В появившемся диалоговом окне введите имя PCIe_Demo.



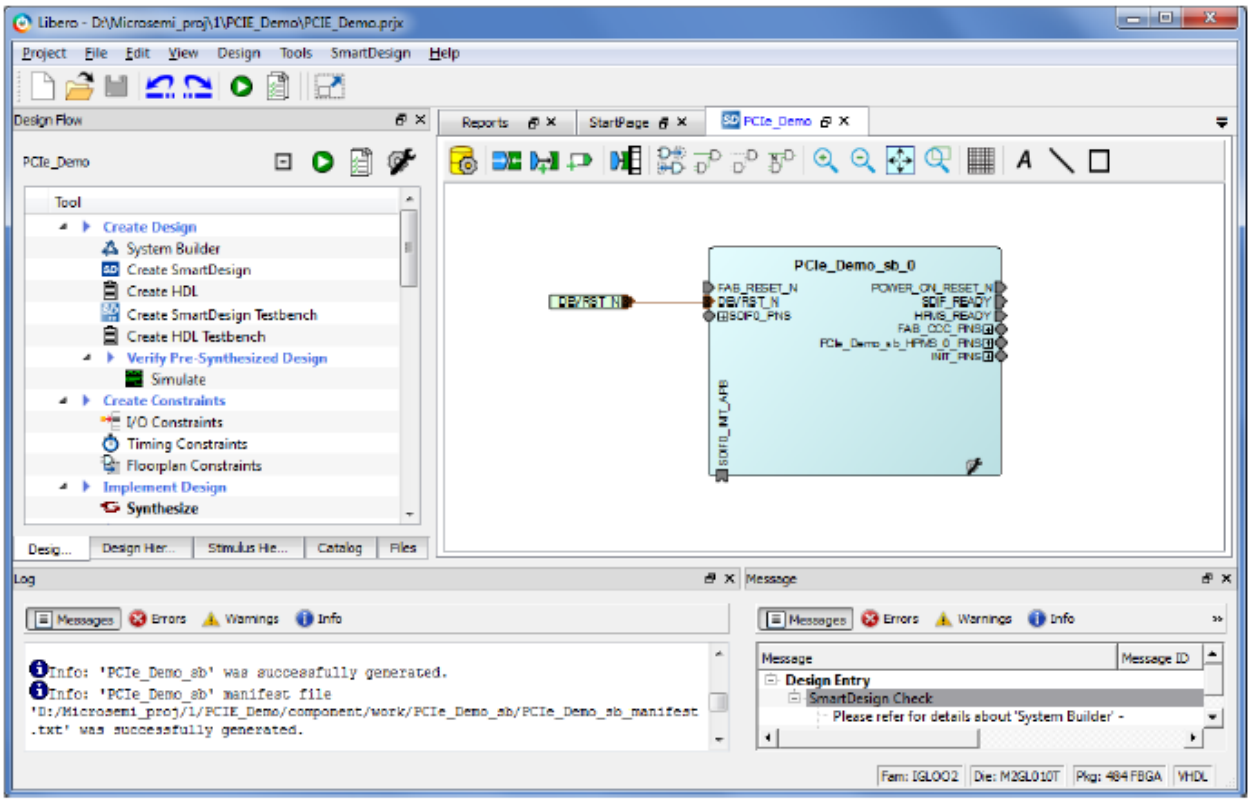
После нажатия ОК откроется инструмент SystemBuilder. На первом шаге отметьте SERDESIF_0, таким образом, инструмент подготовит необходимые ресурсы и настройки для использования SERDES.



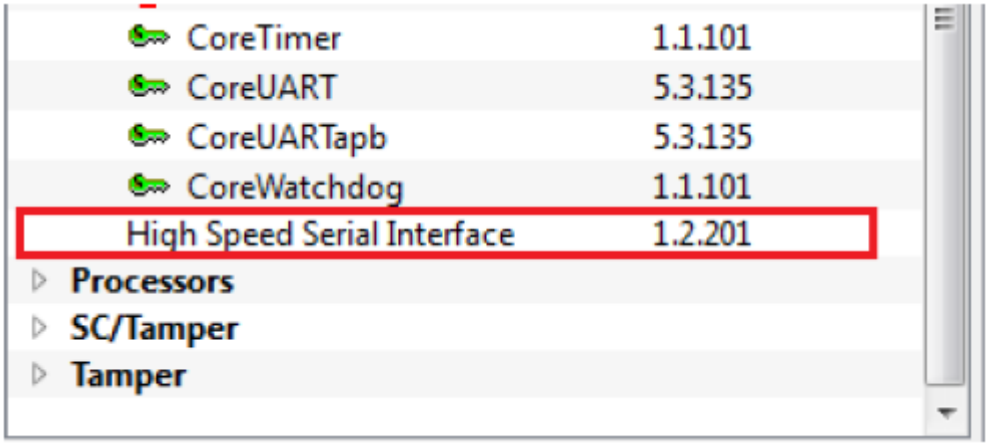
Щёлкните Next, на следующем шаге выберите из выпадающего списка On-chip 25/50 MHz RC Oscillator и задайте частоту HPMS_CLK 100 МГц.



Далее нажимайте Next и оставляйте настройки по умолчанию до появления финальной страницы с картой памяти устройства (Memory Map). Нажмите Finish. System Builder сгенерирует блок PCIe_Demo_sb_0, который будет помещён в рабочую область SmartDesign.

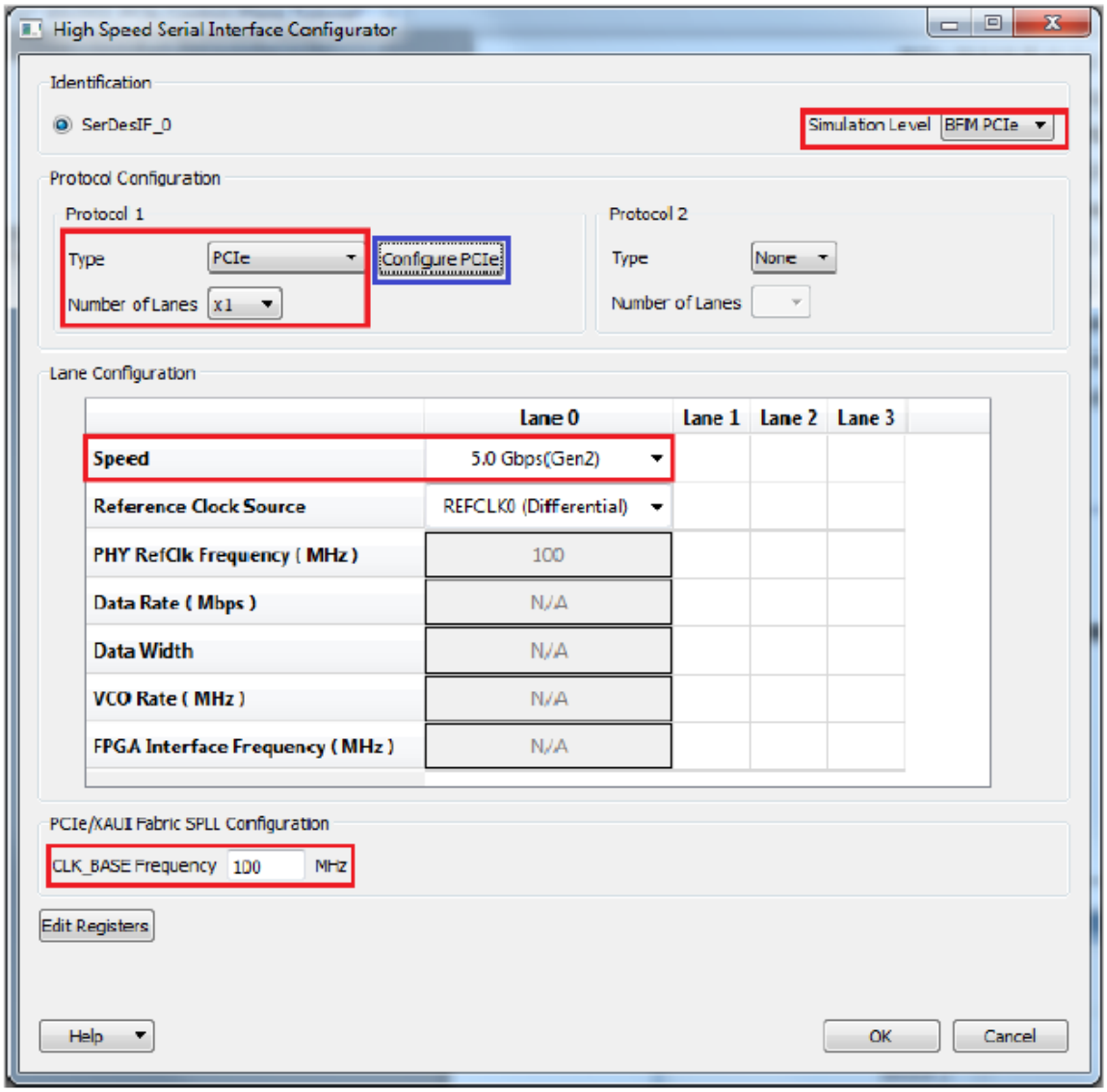


Далее откройте каталог IP и найдите пункт High Speed Serial Interface. Перетащите его в рабочую область SmartDesign, при этом откроется окно конфигурирования ядра.



Введите следующие настройки:

–Identification
- Simulation Level: BFM PCIe
–Protocol Configuration
- Protocol1: Type: PCIe
- Protocol1: Number of Lanes: x1
–Lane Configuration
- Speed: Lane0: 5.0 Gbps (Gen2)
–PCIe/XAUI Fabric SPLL Configuration
- CLK_BASE Frequency (MHz): 100



Нажмите кнопку Configure PCIe. Откроется окно конфигурации PCIe, куда нужно ввести следующие настройки:

–Identification Registers

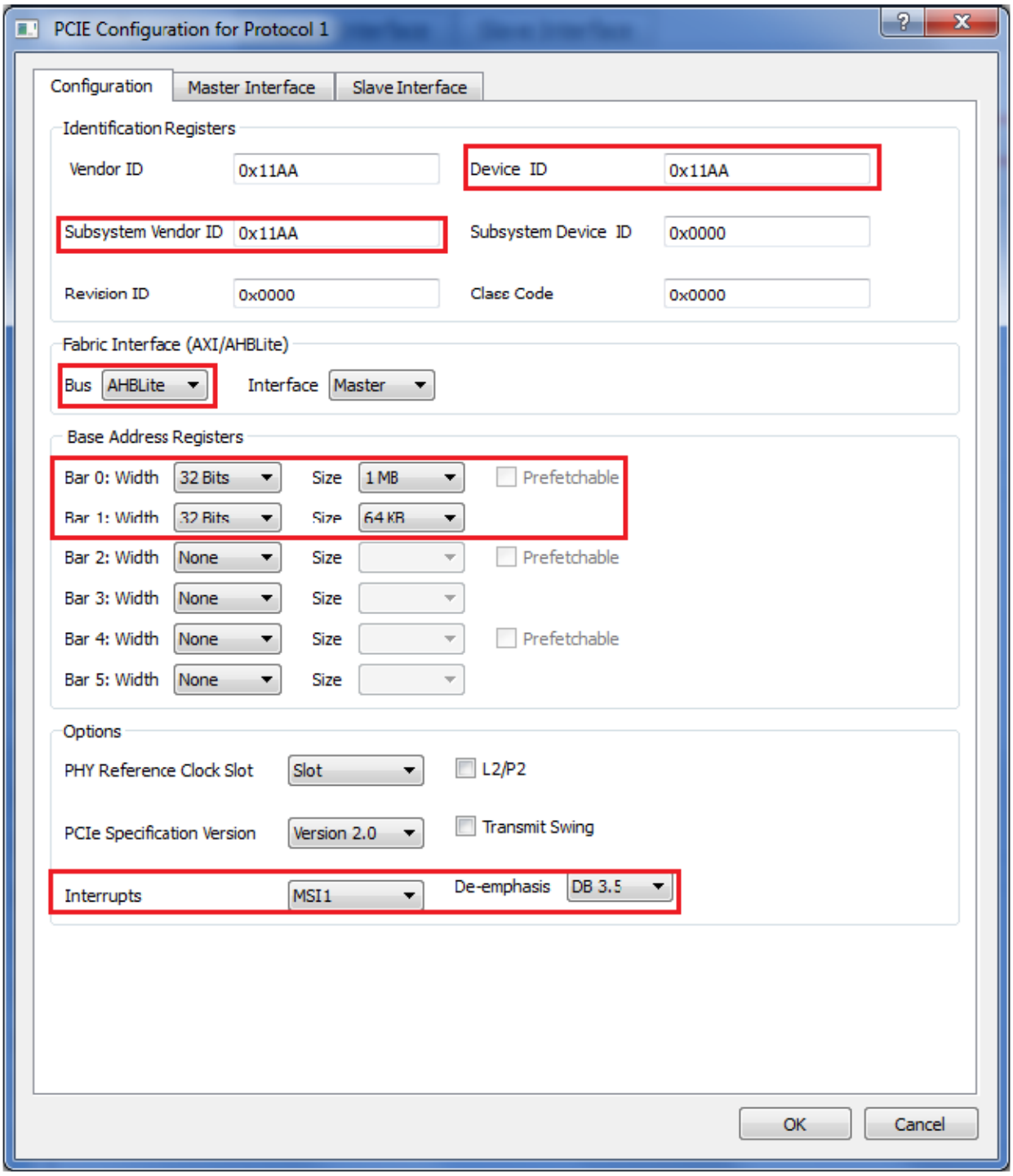
- Device ID: 0x11AA (Microsemi ID)
- Subsystem Vendor ID: 0x11AA (Microsemi ID)

–Fabric Interface (AXI/AHBLite)

- Bus: select as AHBLite from the drop-down list

–Base Address Registers

- Bar 0 Width: 32-bit, Size: 1 MB (to access CoreGPIO address space)
- Bar 1 Width: 32-bit, Size: 64 KB (to access CoreAHBLSRAM memory)



Перейдите во вкладку Master Interface. Произвдите следующие настройки:

–Отметьте Window 0 и введите:

Size: 1MB

PCIe BAR: Bar0

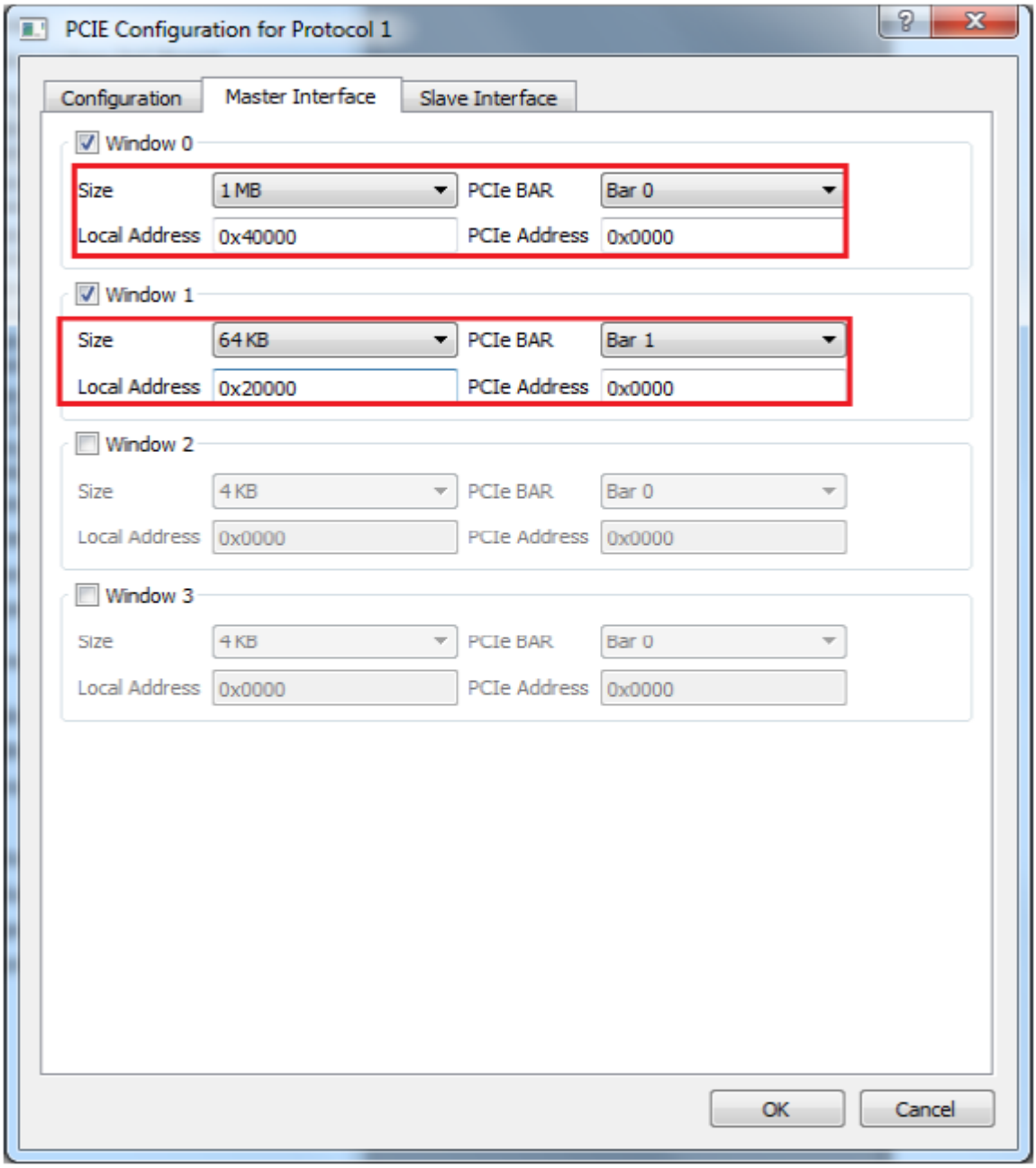
Local Address: 0x40000 (будет транслировано в адрес CoreGPIO 0x4000_0000)

– Отметьте Window 1 и введите:

Size: 64KB

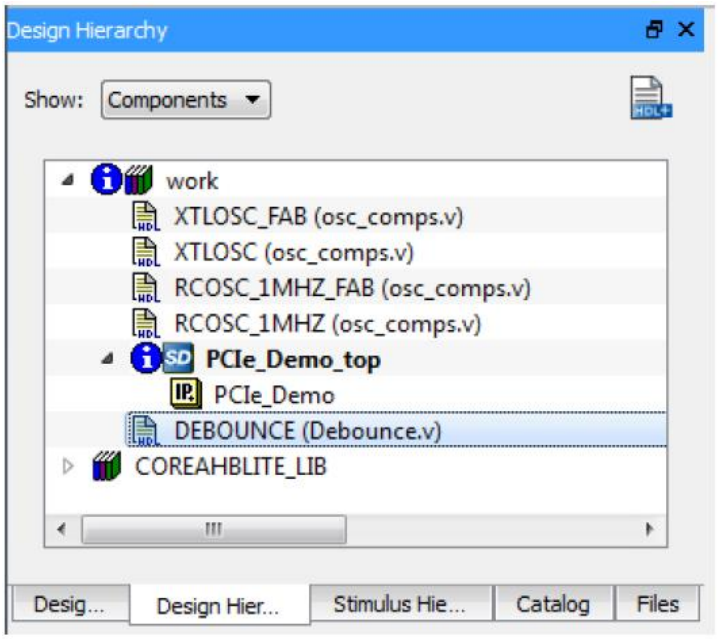
PCIe BAR: Bar1

Local Address: 0x20000 (будет транслировано в адрес CoreAHBLSRAM 0x2000_0000)

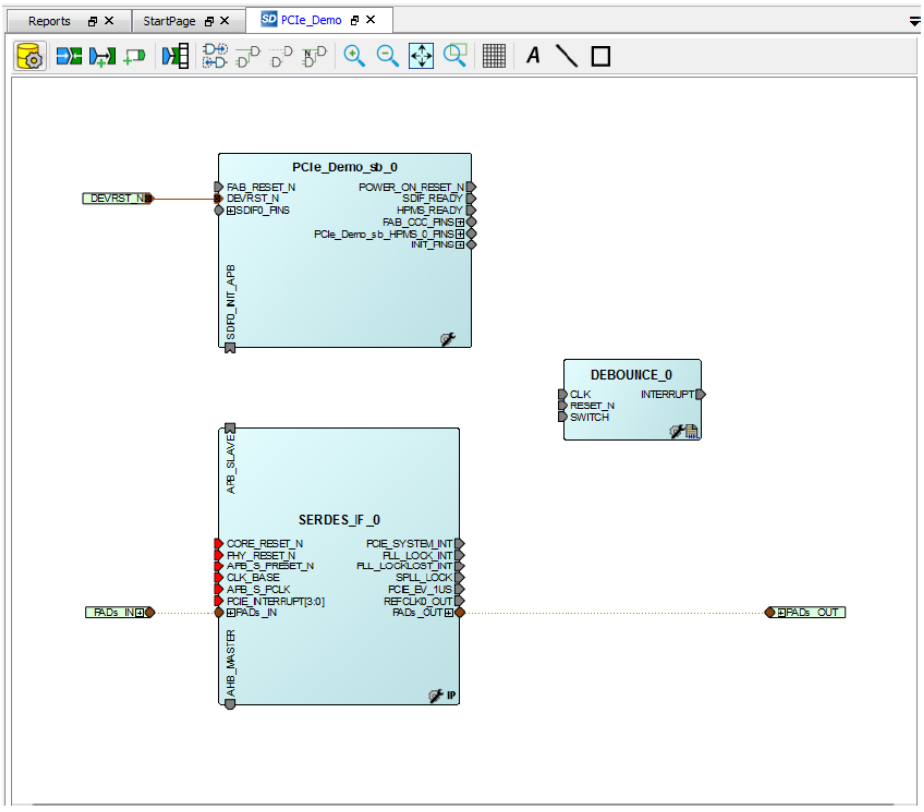


Нажмите OK. В главном окне настройки IP нажмите OK, процесс конфигурирования SERDES будет завершен. В рабочей области SmartDesign окажутся два блока — PICe_Demo_sb_0 (блок HPMS, настроенный с помощью SystemBuilder) и SERDES_IF_0 (блок SERDES, сконфигурированный, как PCIe).

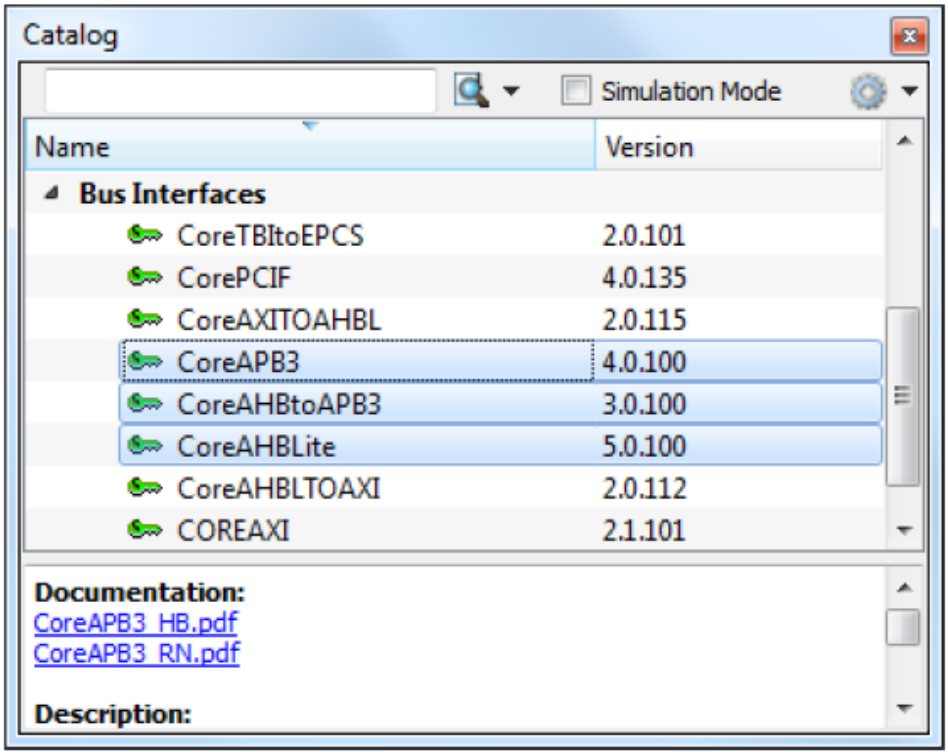
Данный проект использует кнопку SW4 отладочной платы для инициализации прерывания PCIe. Так как при нажатии тактовой кнопки имеет место эффект дребезга контактов, необходимо предотвратить его, для чего используется блок DEBOUNCE, описание которого находится в файлах Debounce.v и Debounce.vhd в папке Source files демонстрационного проекта (см. на www.microsemi.com). Для включения требуемого файла в проект нажмите File -> Import -> HDL Source Files и укажите путь к файлу. После этого можно увидеть компонент из файла в иерархии проекта.



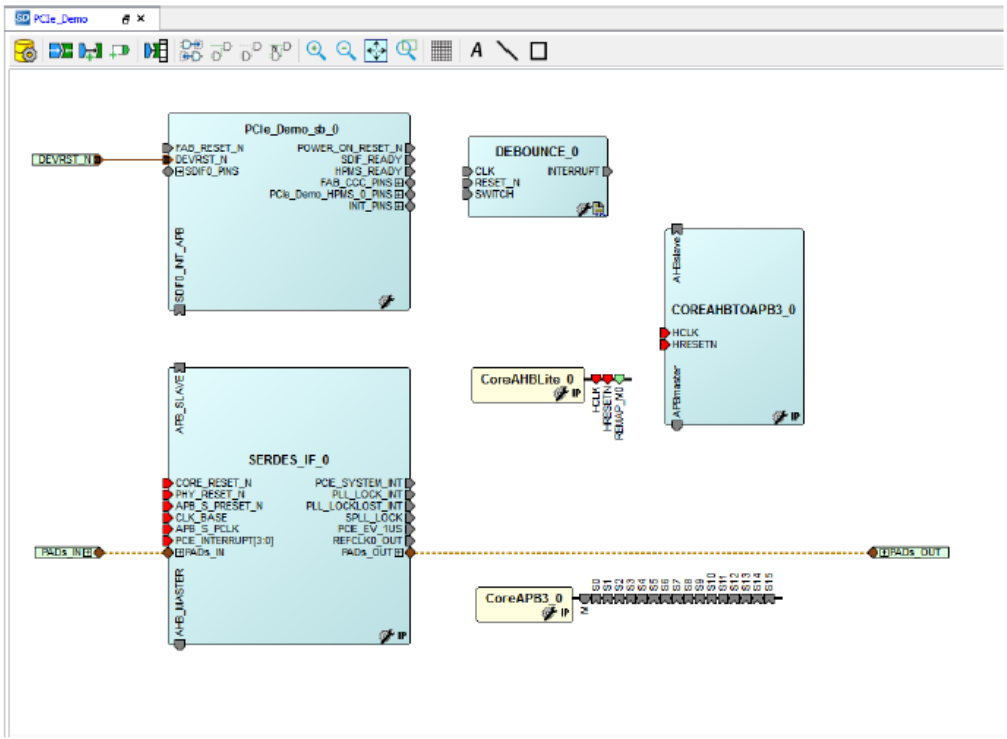
Пока компонент не принадлежит верхнему модулю проекта и просто является частью библиотеки work. Перетащите файл в рабочую область SmartDesign. Теперь рабочая область содержит три компонента.



Следующим шагом добавим в проект интерфейсы шин AMBA для обмена между компонентами дизайна. Для этого найдите соответствующие строки в Catalog -> Bus Interfaces. В проекте используются шины APB, AHB и мост между AHB и APB.



Перетащите эти три компонента в рабочую область для того, чтобы использовать соответствующие интерфейсы.

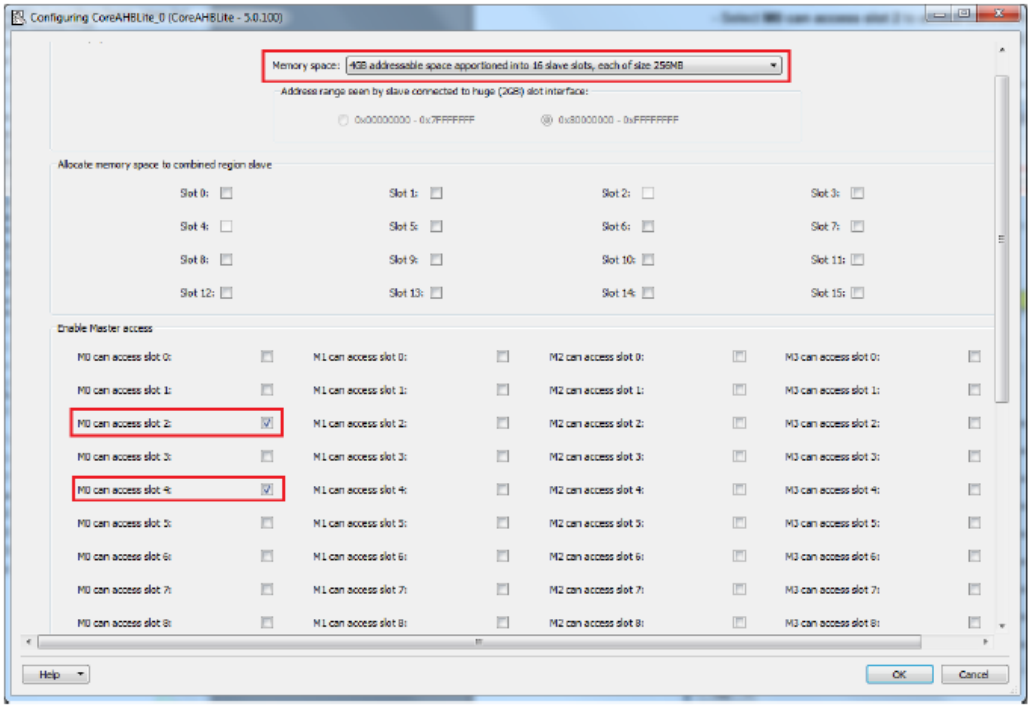


Для настройки каждого интерфейса по его символу можно сделать двойной щелчок мыши. Настройте CoreAHBLite_0 следующим образом:

- Memory Space: “4 GB addressable space apportioned into 16 slave slots, each of size 256 MB”.

- Установите флажок “M0 can access slot 2” (используется для доступа к CoreAHBLSRAM из PCIe).

- Установите флажок “M0 can access slot 4” (используется для доступа к CoreGPIO из PCIe).



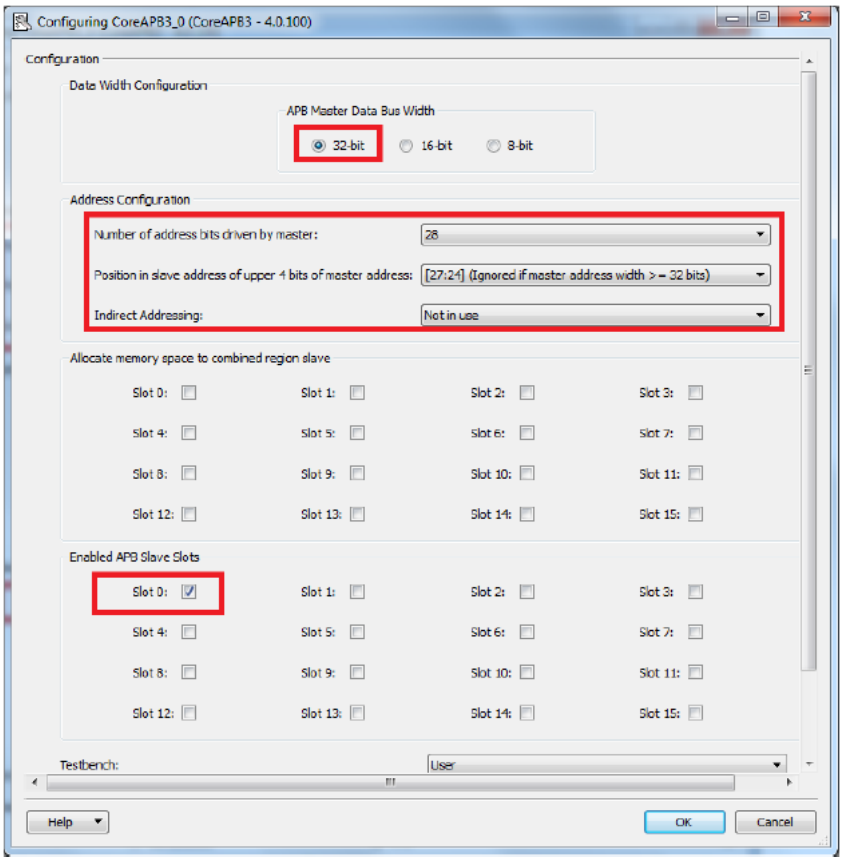
Настройте CoreAPB3_0 следующим образом:

–APB Master Data Bus Width: 32-bit.

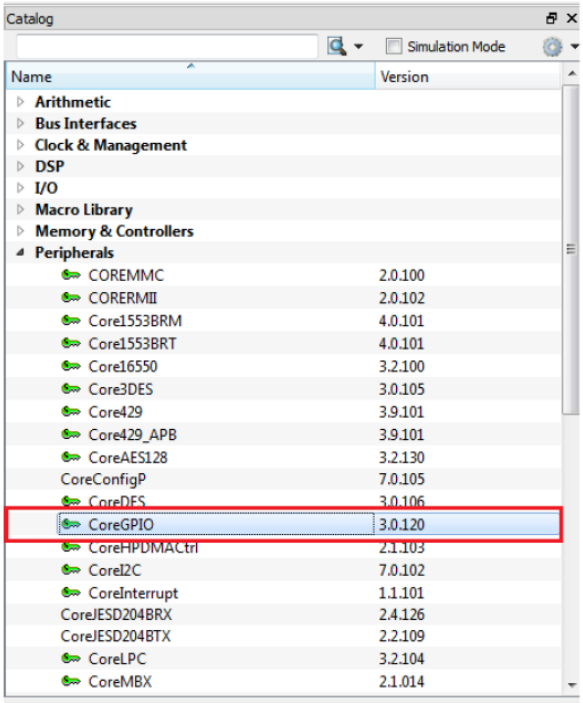
– Address Configuration

- Number of address bits driven by master: ‘28’.
- Position in slave address of upper 4 bits of master address: ‘[27:24](Ignored if master address width >=32 bits)’.

–Enabled APB Slave Slots: Slot 0.

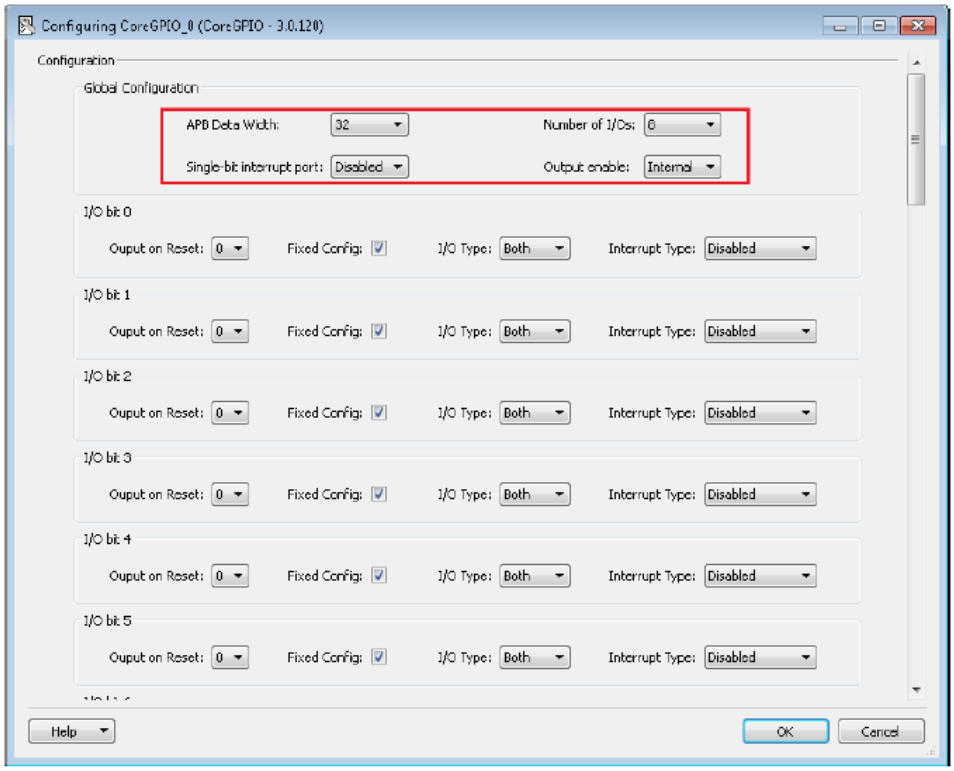


Далее добавим IP-ядро отвечающее за ввод-вывод общего назначения. Данное ядро будет управлять светодиодами и принимать информацию с переключателей на отладочной плате. Найдите строку CoreGPIO в Catalog -> Peripherals.

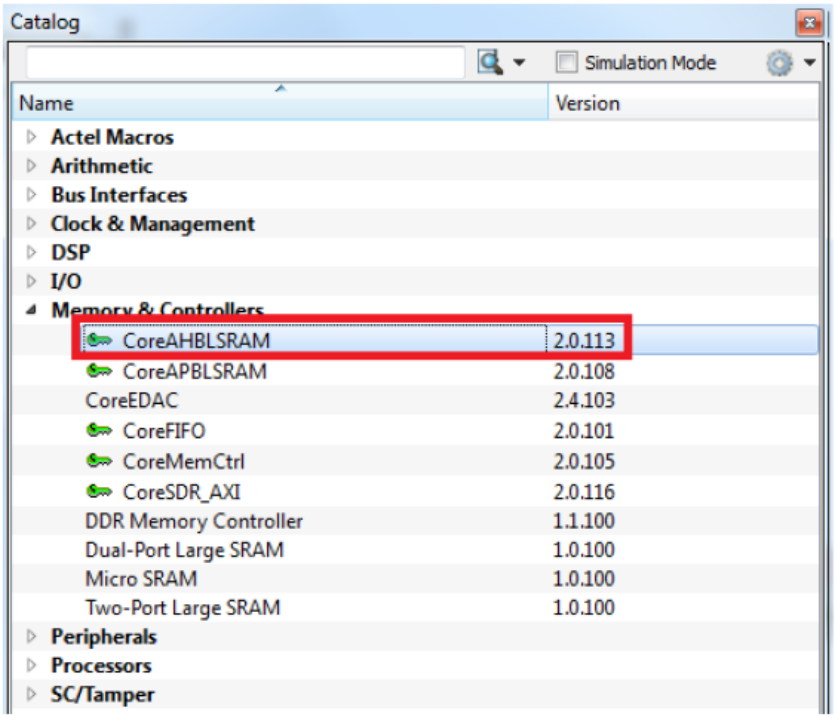


Перетащите компонент в рабочую область SmartDesign, при этом откроется окно конфигурирования IP-ядра. Произведите следующие настройки:

– APB Data Width: 32.
– Number of I/O’s: 8.
– Output: Internal
– Для всех I/O от 0 до 7 выберите I/O Type: Both.

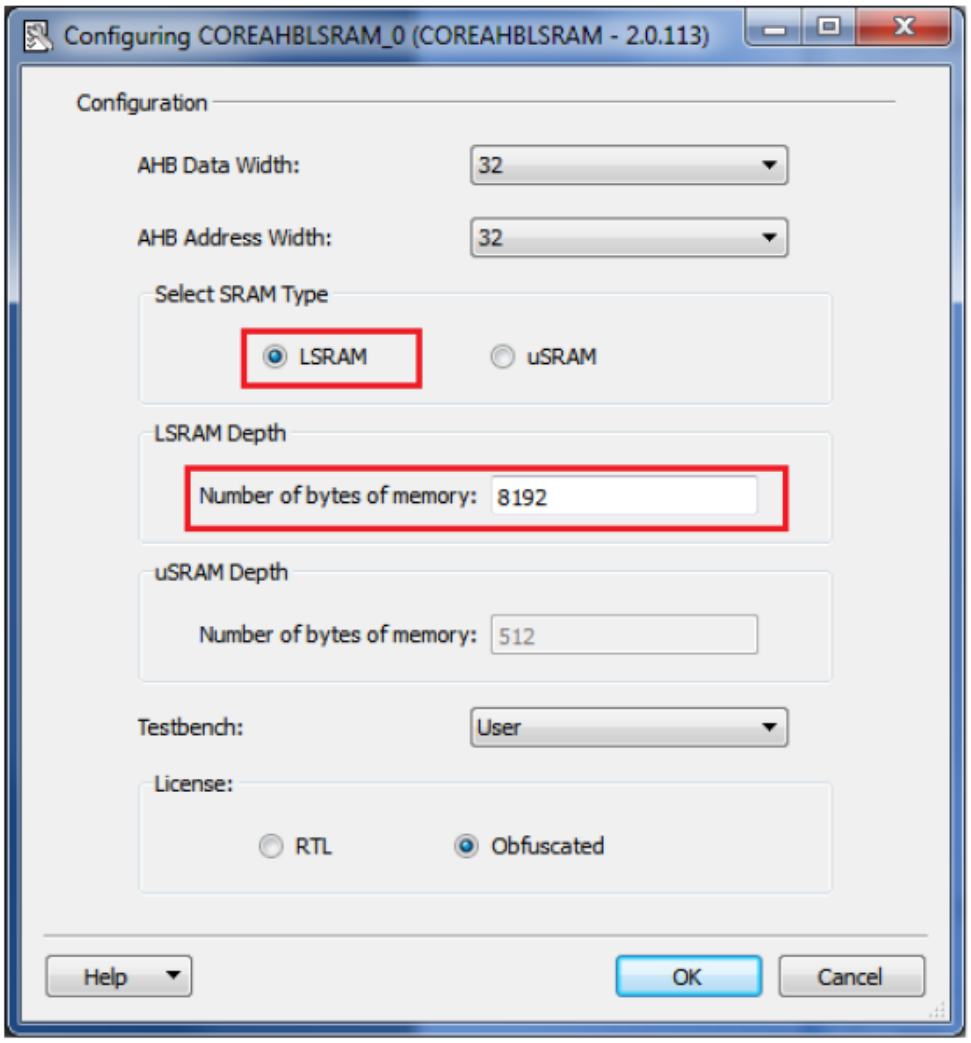


В проекте используется СОЗУ, которая будет доступна для чтения и записи через PCIe с ПК. Добавим этот компонент из Catalog -> Memory and Controllers.

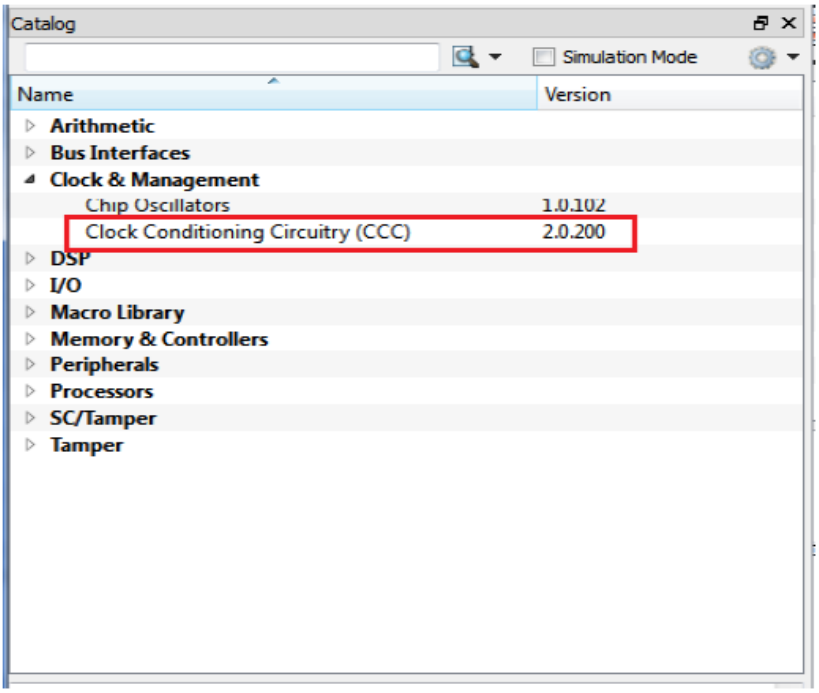


Перетащите компонент в рабочую область SmartDesign, при этом откроется окно конфигурирования памяти. Произведите следующие настройки:

– AHB Data Width и AHB Address: 32.
– Select SRAM Type: LSRAM.
– Number of bytes of memory: 8192.

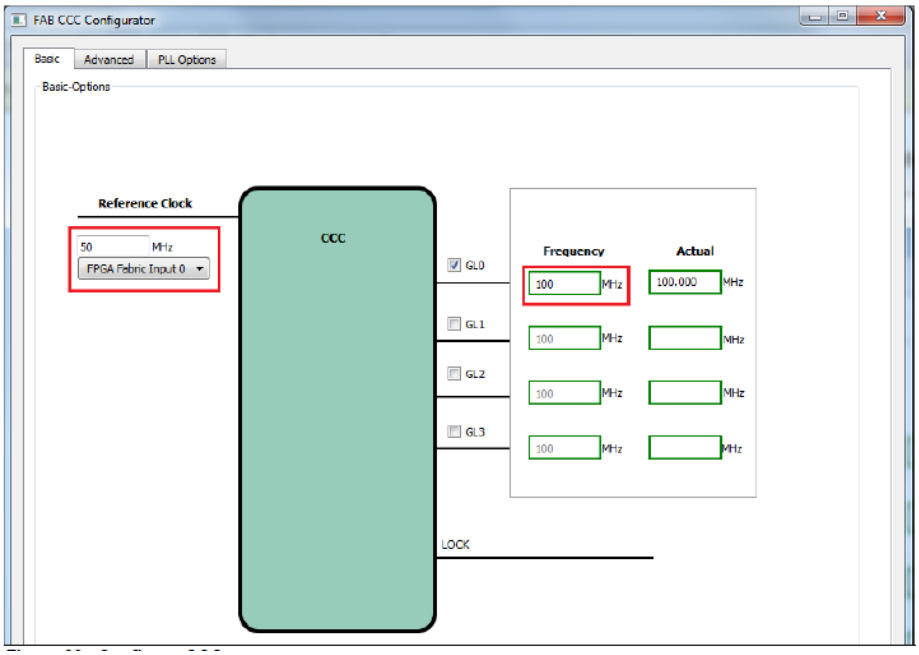


Для формирования тактовой частоты в проекте используются устройство распределения тактовых сигналов (CCC — Clock Conditioning Circuitry). Добавьте блок CCC из Catalog -> Clock and Management в рабочую область SmartDesign.

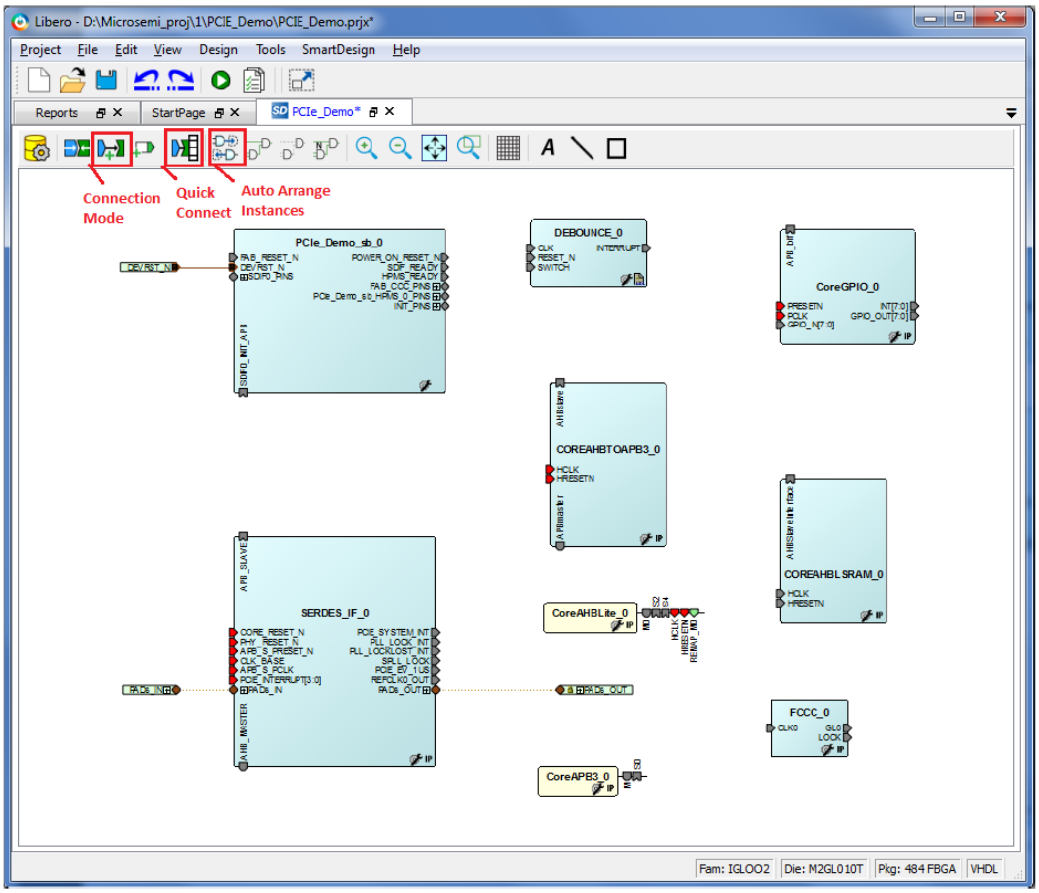


В появившемся окне введите следующие настройки:

–Reference Clock: 50 MH и FPGA Fabric Input 0.
– GL0 Frequency: 100 MHz.



Теперь на верхнем уровне проекта имеются все необходимые и должным образом сконфигурированные компоненты. Осталось распределить связи между портами компонентов. Для этого можно воспользоваться режимом Connection Mode, инструментом Quick Connect, также можно упорядочить символы компонентов с помощью кнопки Auto Arrange Instances.



Воспользовавшись одним из этих способов (см. разд. 3.3.1), соедините компоненты

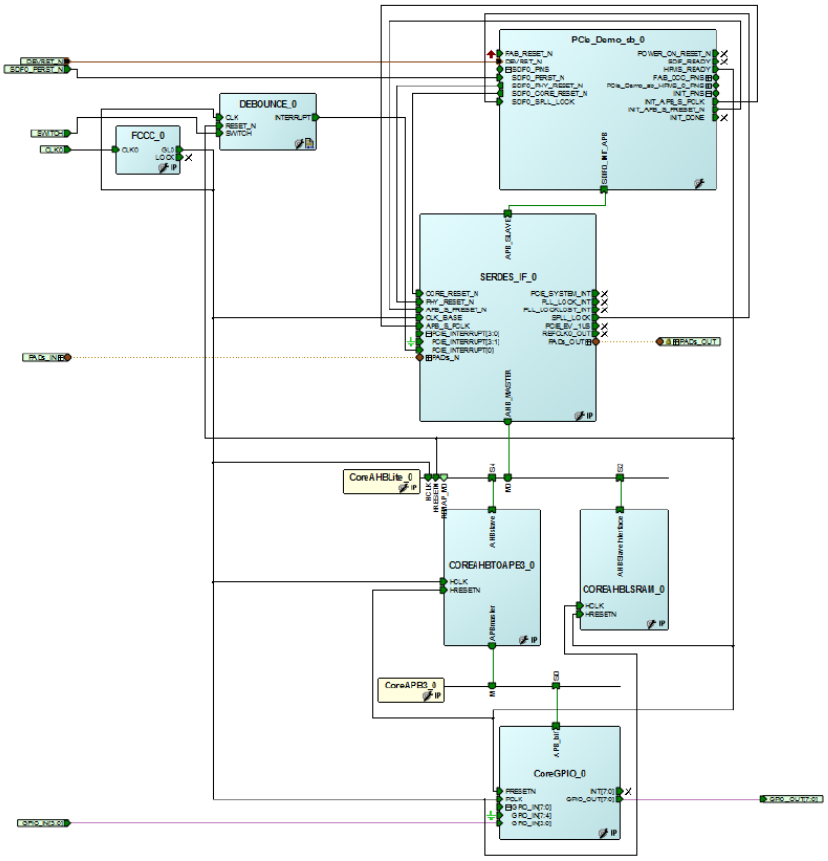
От		**До**	
PCIe_Demo_sb_0	SDIF0_PHY_RESET_N	SERDES_IF_0	PHY_RESET_N
	SDIF0_CORE_RESET_N		CORE_RESET_N
	SDIF0_SPLL_LOCK		SPLL_LOCK
	SDIF0_PERST_N	—	На верхний уровень (Promote to top level)
	INIT_APB_S_PCLK	SERDES_IF_0	APB_S_PCLK
	INIT_APB_S_PRESET_N		APB_S_PRESET_N
	INIT_DONE	—	Не используется (Mark Unused)
	HPMS_READY	CoreAHBLite_0,COREAHBTOAPB3_0, и COREAHBLSRAM_0	HRESETN
		CoreGPIO_0	PRESETN
		DEBOUNCE_0	RESET_N

	FAB_CCC_GL0	—	Не используется (Mark Unused)
	FAB_CCC_GL3	—	Не используется (Mark Unused)
	FAB_CCC_LOCK	—	Не используется (Mark Unused)
	POWER_ON_RESET_N	—	Не используется (Mark Unused)
	SDIF_READY	—	Не используется (Mark Unused)
	FAB_RESET_N	—	Подтяжка к "1" (Tie High)
	COMM_BLK_INT	—	Не используется (Mark Unused)
	HPMS_INT_M2F[15:0]	—	Не используется (Mark Unused)
	SDIF0_INIT_APB	SERDES_IF_0	APB_SLAVE
FCCC_0	GL0	CoreAHBLite_0,COREAHBTOAPB3_0 и COREAHBLSRAM_0	HCLK
		CoreGPIO_0	PCLK
		DEBOUNCE_0	CLK
		SERDES_IF_0	CLK_BASE
	CLK0	—	На верхний уровень (Promote to top level)
	LOCK	—	Не используется (Mark Unused)
CoreAHBLite_0	M0	SERDES_IF_0	AHB_MASTER
	S2	COREAHBLSRAM_0	AHBSlaveInterface
	S4	COREAHBTOAPB3_0	AHBslave
CoreAPB3_0	M	COREAHBTOAPB3_0	APBmaster
	S0	CoreGPIO_0	APB_bif
Debounce_0	SWITCH	—	На верхний уровень (Promote to top level)
CoreGPIO_0	INT[7:0]	—	Не используется (Mark Unused)
	GPIO_OUT[7:0]	—	На верхний уровень (Promote to top level)

	GPIO_IN [3:0]	—	На верхний уровень (Promote to top level)
	GPIO_IN [7:4]	—	Подтяжка к "0" (Tie Low)
SERDES_IF_0	PCIE_SYSTEM_INT	—	Не используется (Mark Unused)
	PLL_LOCK_INT	—	Не используется (Mark Unused)
	PLL_LOCKLOST_INT	—	Не используется (Mark Unused)
	PCIE_EV_1US	—	Не используется (Mark Unused)
	REFCLK0_OUT	—	Не используется (Mark Unused)
	PCIE_INTERRUPT[3:1]	—	Подтяжка к "0" (Tie Low)
Debounce_0	INTERRUPT	SERDES_IF_0	PCIE_INTERRUPT[0]

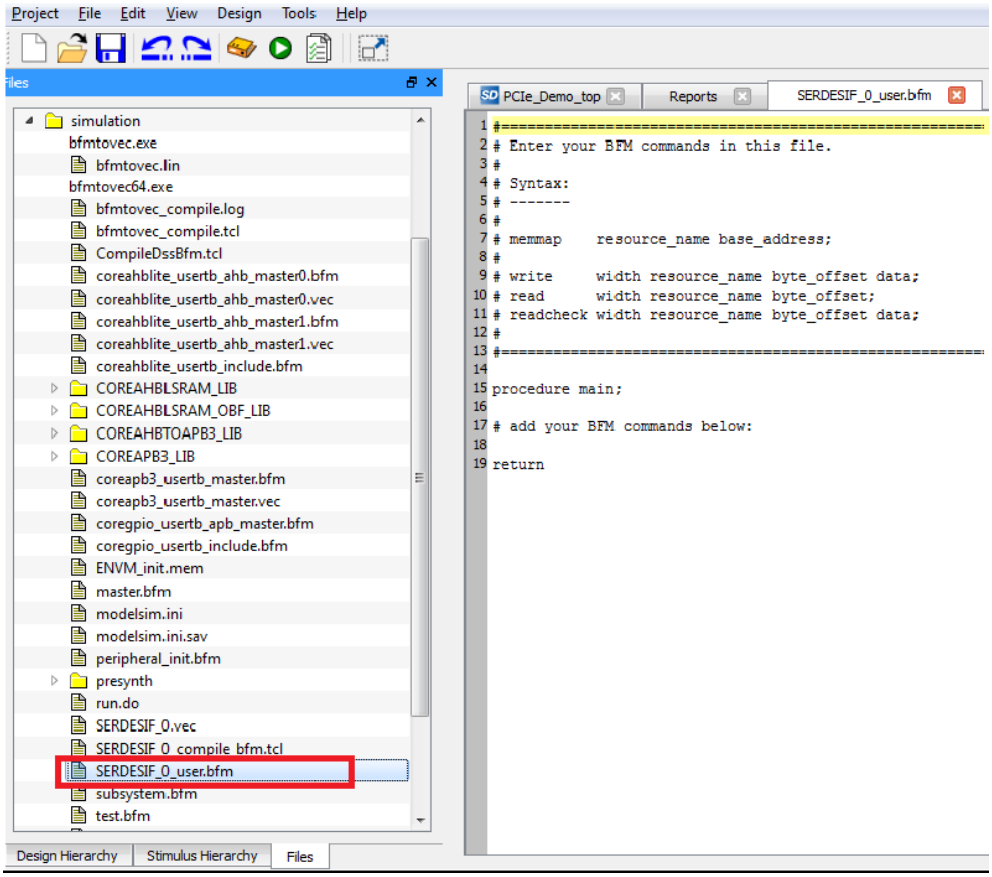
Для разделения GPIO_IN на части [3..0] и [7..4] и PCIE_INTERRUPT на [3..1] и [0] воспользуетесь инструментом Edit Slice (см. разд. 3.3.1).

После соединений получится схема, готовая к синтезу.



Следующим этапом проведём функциональное моделирование полученного устройства. Для этого нужно в первую очередь выработать тестовые воздействия. На этапе настройки SERDES мы указали уровень симуляции BFM (Bus Functional Model). При такой симуляции скрываются детали работы протокола и разработчику предоставляется удобный способ проверки внутренних соединений с блоком SERDESIF, при этом симуляция не перегружается деталями реализации нижних уровней, которые целиком выполнены в виде hard ip блока и прошли верификацию. Для проведения моделирования используется специальный BFM-скрипт, моделирующий транзакции чтения и записи из/в интерфейсы в матрице ПЛИС.

Откройте вкладку Files в области 1 главного окна Libero и найдите файл SERDES_0_user.bfm.



Данный файл генерируется средой автоматически и представляет собой шаблон, редактируемый пользователем. Добавьте в скрипт следующие команды чтения и записи:

```
memmap GPIO 0x40000000;
memmap LSRAM 0x20000000;
procedure main;

# add your BFM commands below:

write w GPIO 0xA0 0x00;
write w GPIO 0xA0 0x01;
write w GPIO 0xA0 0x02;
write w GPIO 0xA0 0x04;
write w GPIO 0xA0 0x08;
write w GPIO 0xA0 0x10;
```

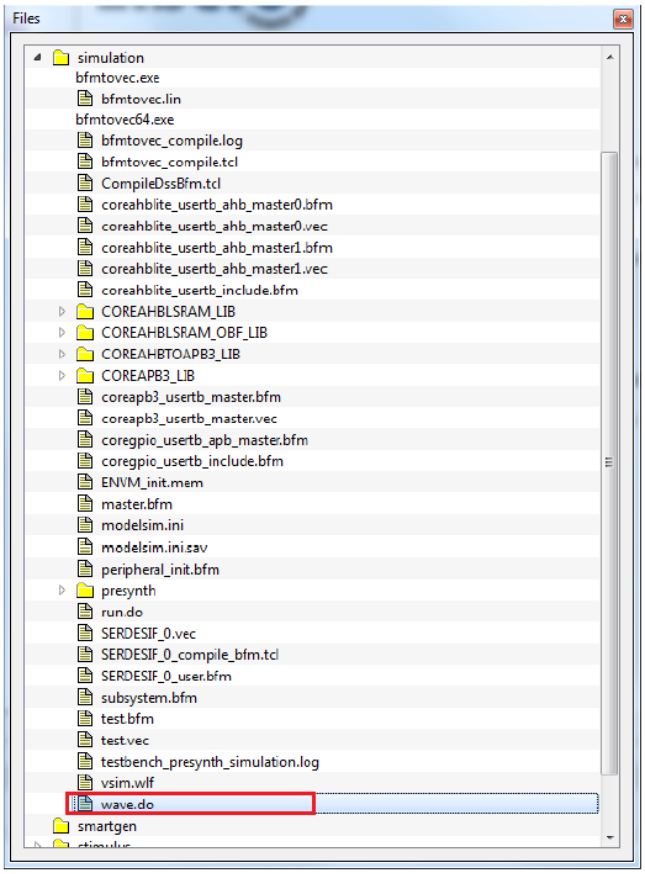
```
write w GPIO 0xA0 0x20;
write w GPIO 0xA0 0x40;
write w GPIO 0xA0 0x80;

write w LSRAM 0x00 0x12345678;
write w LSRAM 0x04 0x87654321;
write w LSRAM 0x08 0x9ABCDEF0;
write w LSRAM 0x0C 0x0FEDCBA9;

readcheck w LSRAM 0x00 0x12345678;
readcheck w LSRAM 0x04 0x87654321;
readcheck w LSRAM 0x08 0x9ABCDEF0;
readcheck w LSRAM 0x0C 0x0FEDCBA9;

return
```

Данные команды осуществляют:

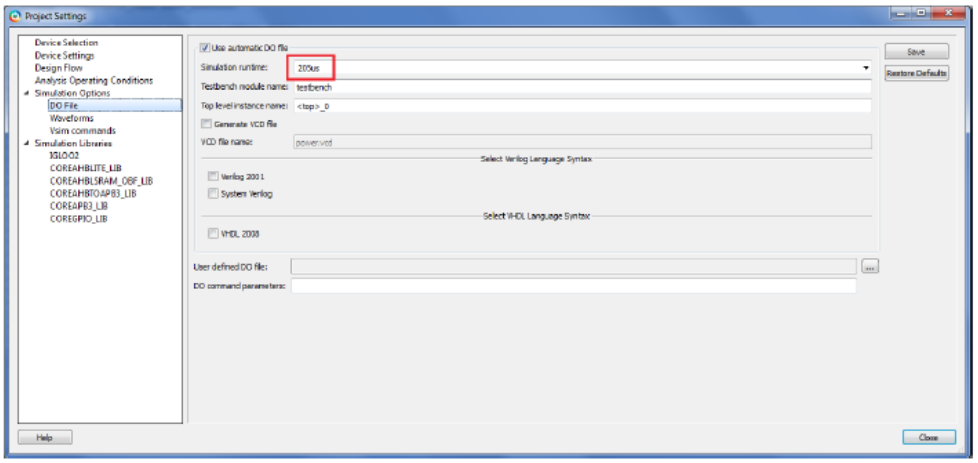
- Запись в GPIO[7..0];
- Запись в память LSRAM;
- Проверку записи в LSRAM.

При этом сценарии никакая информация не выходит и не входит в интерфейсные линии PCIe, проверяются исключительно внутренние интерфейсы.

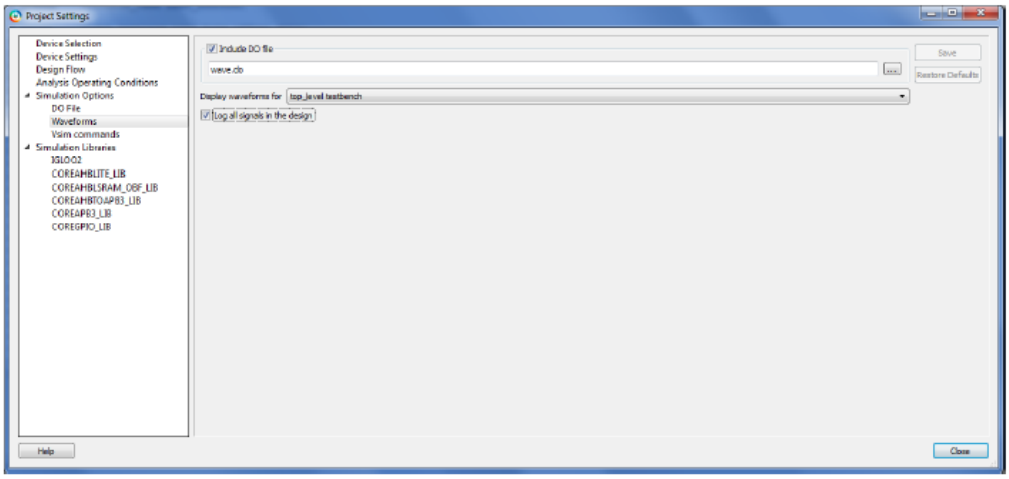
Следующим шагом найдите файл wave.do, отвечающий за сценарий проведения симуляции.



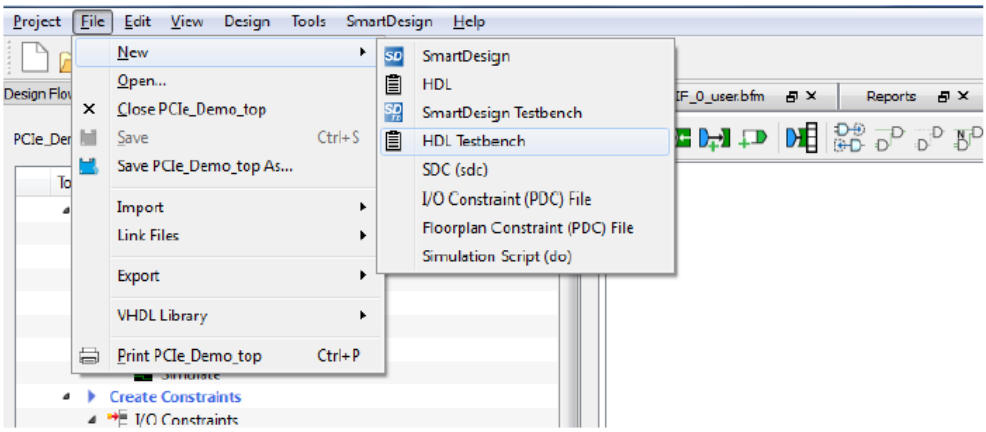
Откройте меню Project -> Project Settings и выберите пункт Simulation Options -> DO File. Введите в поле Simulation runtime “205us”.



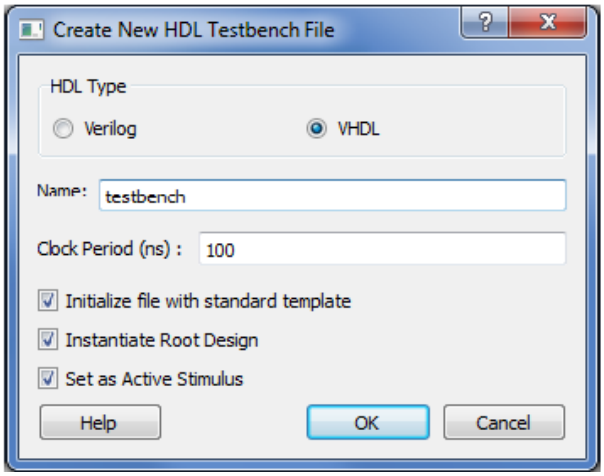
Нажмите Save и перейдите в пункт Waveforms. Поставьте галочку Include DO File и укажите рассмотренный выше файл wave.do, отметьте Log all signals in the design.



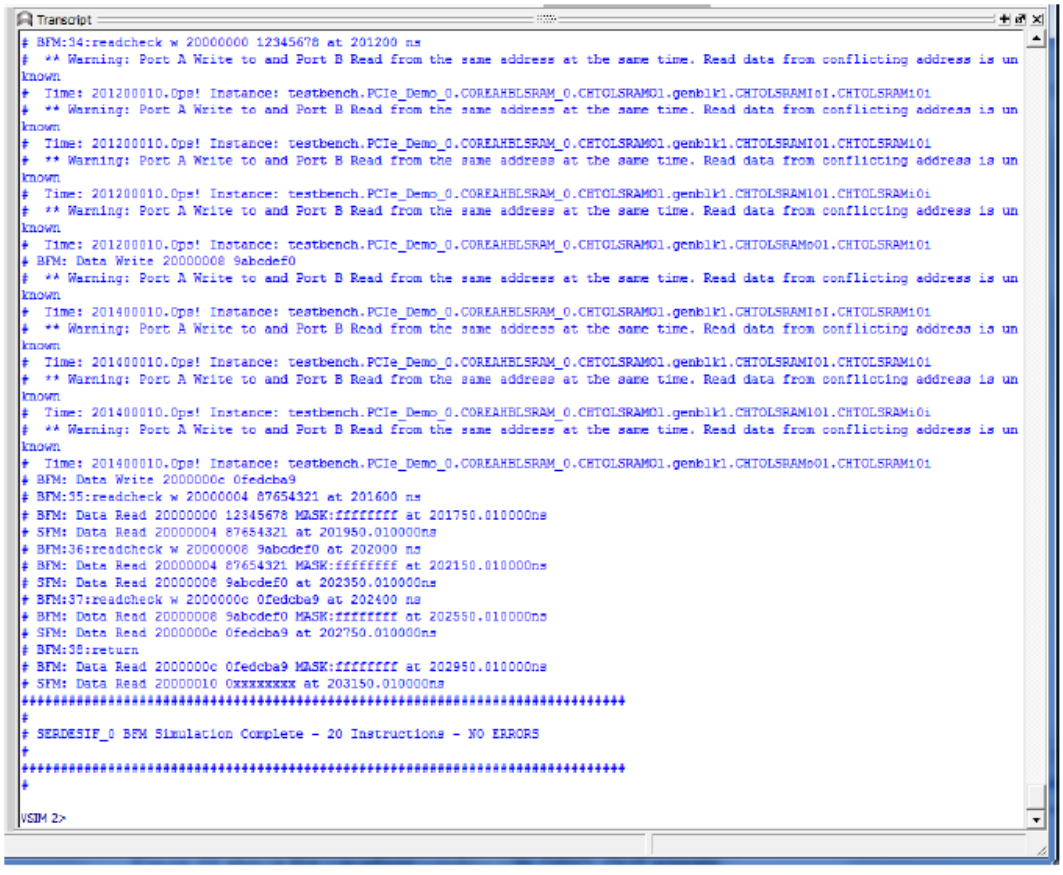
Нажмите Save, затем Close. Далее в главном окне Libero проследуйте в меню File -> New -> HDL Testbench.



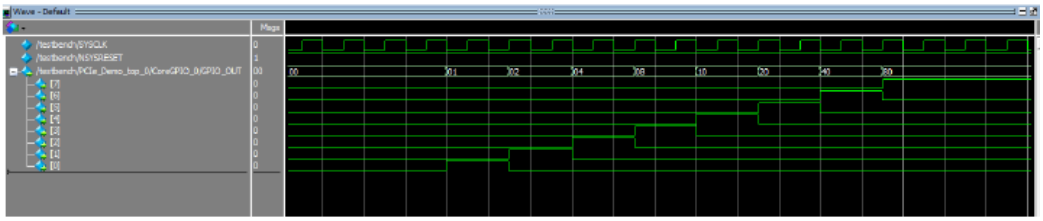
В диалоговом окне введите имя и выберите VHDL или Verilog.



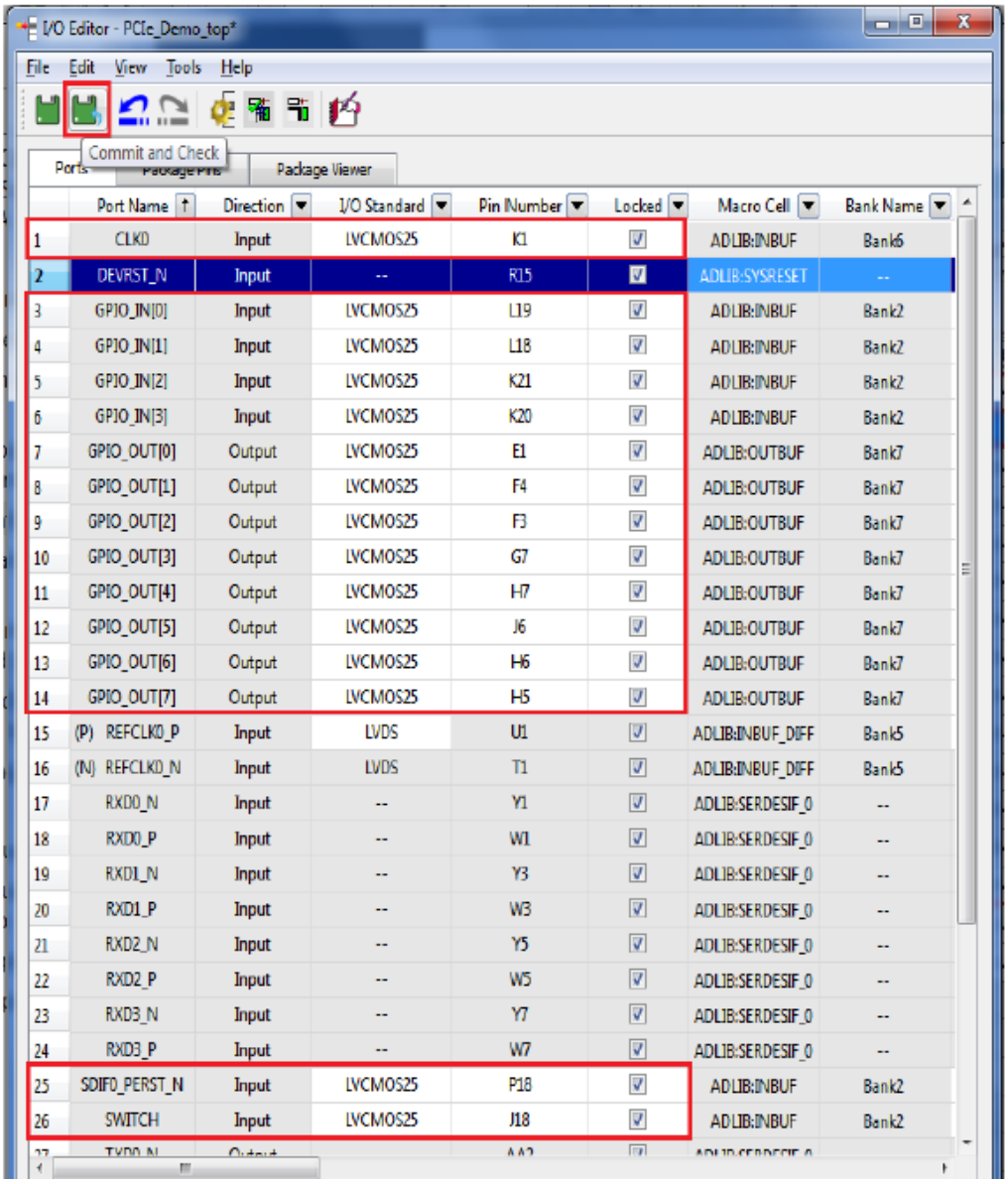
Нажмите ОК. Теперь можно запустить симуляцию, для этого сделайте двойной щелчок на Simulate в секции Verify Pre-Synthesized Design в Design Flow. В поле лога Modelsim можно наблюдать, как выполняются команды BFM.



На сформированной временной диаграмме после окончания процесса появятся сигналы GPIO.



После того, как требуемая временная диаграмма получена можно проверить дизайн на натурном образце — отладочном наборе IGLOO2 Evaluation Kit. Для начала укажем расположение внешних выводов в I/O Editor.



	Port Name	Direction	I/O Standard	Pin Number	Locked	Macro Cell	Bank Name
1	CLK0	Input	LVCMOS25	K1	☑	ADLIB:INBUF	Bank6
2	DEVRST_N	Input	--	R15	☑	ADLIB:SYSRESET	--
3	GPIO_IN[0]	Input	LVCMOS25	L19	☑	ADLIB:INBUF	Bank2
4	GPIO_IN[1]	Input	LVCMOS25	L18	☑	ADLIB:INBUF	Bank2
5	GPIO_IN[2]	Input	LVCMOS25	K21	☑	ADLIB:INBUF	Bank2
6	GPIO_IN[3]	Input	LVCMOS25	K20	☑	ADLIB:INBUF	Bank2
7	GPIO_OUT[0]	Output	LVCMOS25	E1	☑	ADLIB:OUTBUF	Bank7
8	GPIO_OUT[1]	Output	LVCMOS25	F4	☑	ADLIB:OUTBUF	Bank7
9	GPIO_OUT[2]	Output	LVCMOS25	F3	☑	ADLIB:OUTBUF	Bank7
10	GPIO_OUT[3]	Output	LVCMOS25	G7	☑	ADLIB:OUTBUF	Bank7
11	GPIO_OUT[4]	Output	LVCMOS25	H7	☑	ADLIB:OUTBUF	Bank7
12	GPIO_OUT[5]	Output	LVCMOS25	J6	☑	ADLIB:OUTBUF	Bank7
13	GPIO_OUT[6]	Output	LVCMOS25	H6	☑	ADLIB:OUTBUF	Bank7
14	GPIO_OUT[7]	Output	LVCMOS25	H5	☑	ADLIB:OUTBUF	Bank7
15	(P) REFCLK0_P	Input	LVDS	U1	☑	ADLIB:INBUF_DIFF	Bank5
16	(N) REFCLK0_N	Input	LVDS	T1	☑	ADLIB:INBUF_DIFF	Bank5
17	RXD0_N	Input	--	Y1	☑	ADLIB:SERDESIF_0	--
18	RXD0_P	Input	--	W1	☑	ADLIB:SERDESIF_0	--
19	RXD1_N	Input	--	Y3	☑	ADLIB:SERDESIF_0	--
20	RXD1_P	Input	--	W3	☑	ADLIB:SERDESIF_0	--
21	RXD2_N	Input	--	Y5	☑	ADLIB:SERDESIF_0	--
22	RXD2_P	Input	--	W5	☑	ADLIB:SERDESIF_0	--
23	RXD3_N	Input	--	Y7	☑	ADLIB:SERDESIF_0	--
24	RXD3_P	Input	--	W7	☑	ADLIB:SERDESIF_0	--
25	SDIF0_PERST_N	Input	LVCMOS25	P18	☑	ADLIB:INBUF	Bank2
26	SWITCH	Input	LVCMOS25	J18	☑	ADLIB:INBUF	Bank2

Далее запустим весь Design Flow до формирования файла прошивки (Generate Bitstream).

Перед выполнением следующих действий удостоверьтесь, что питание платы выключено. Проверьте конфигурацию джамперов на плате.

Jumper	Pin (from)	Pin (to)	Comments
J22	1	2	Default
J23	1	2	Default
J24	1	2	Default
J8	1	2	Default
J3	1	2	Default

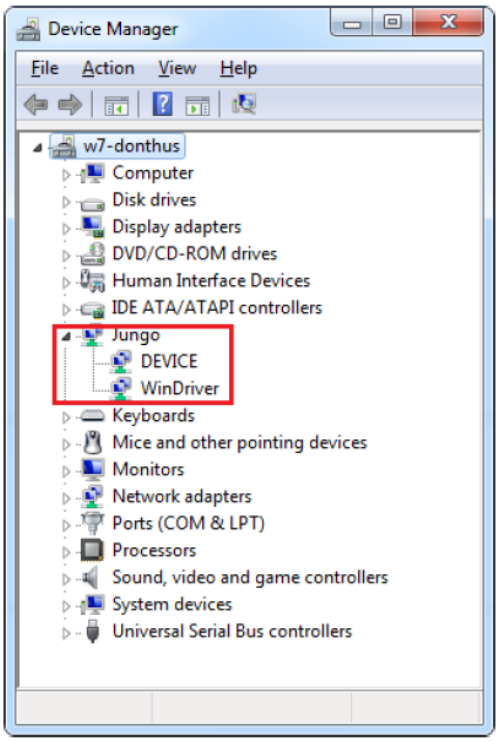
Подключите программатор FlashPro и подайте питание на плату. Запустите процесс программирования (Run PROGRAM Action).

После успешного программирования нужно соединить отладочную плату с ПК. Отладочная плата Evaluation Kit имеет форм-фактор платы PCIe и легко установится в один из слотов ПК. Перед установкой выключите питание ПК и отладочной платы. Найдите на материнской плате вашего ПК свободный слот PCIe (x1 и крупнее). Установите отладочную плату в слот, подайте на неё питание. Включите ПК.

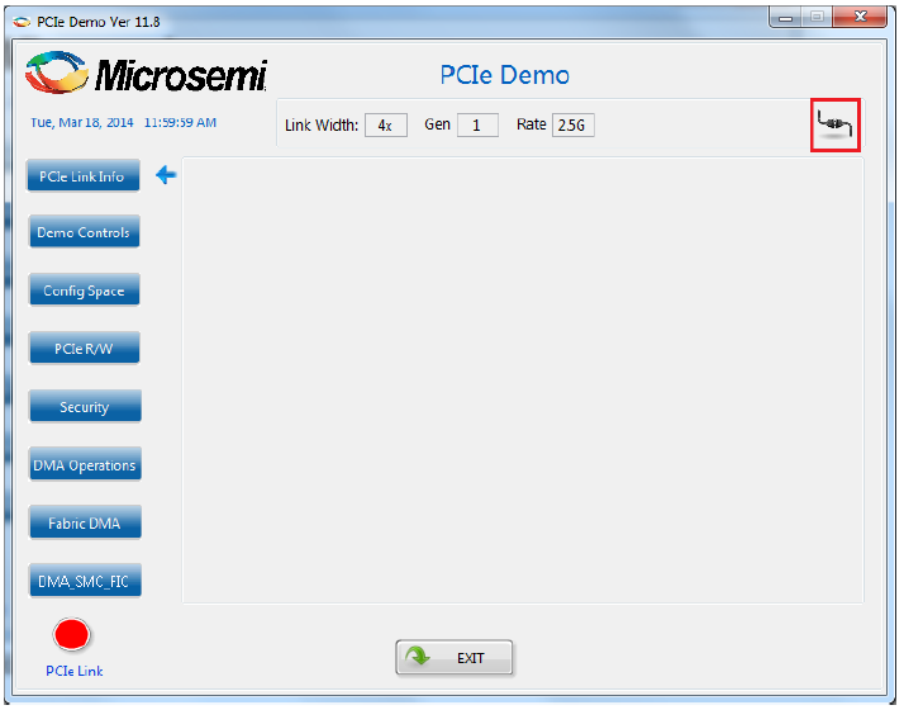
Для корректной работы устройства необходимо установить драйверы Jungo. Найдите в файлах демо-проекта папку Drivers\PCIe_Demo.rar и распакуйте содержимое на диск C:.

Замечание: действия по установке драйверов следует проводить обладая правами администратора.

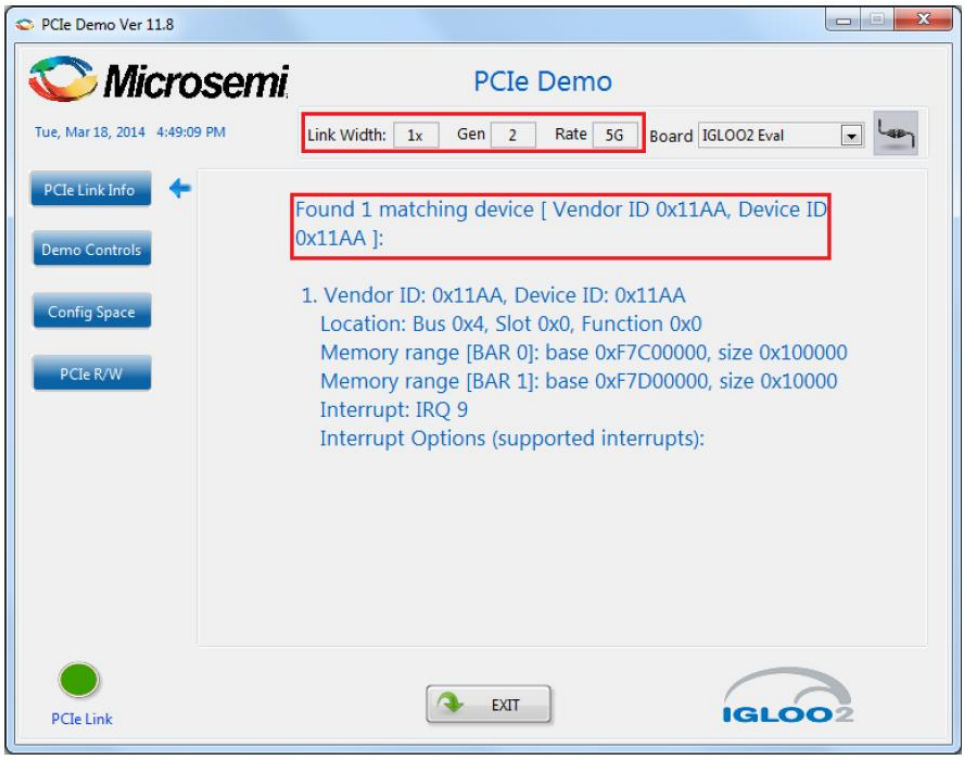
Запустите скрипт C:\PCIe_Demo\DriverInstall\Jungo_KP_install.bat. Нажмите «Установить», если появится диалоговое окно. Устройство с именем Jungo после этого должно появиться в диспетчере устройств.



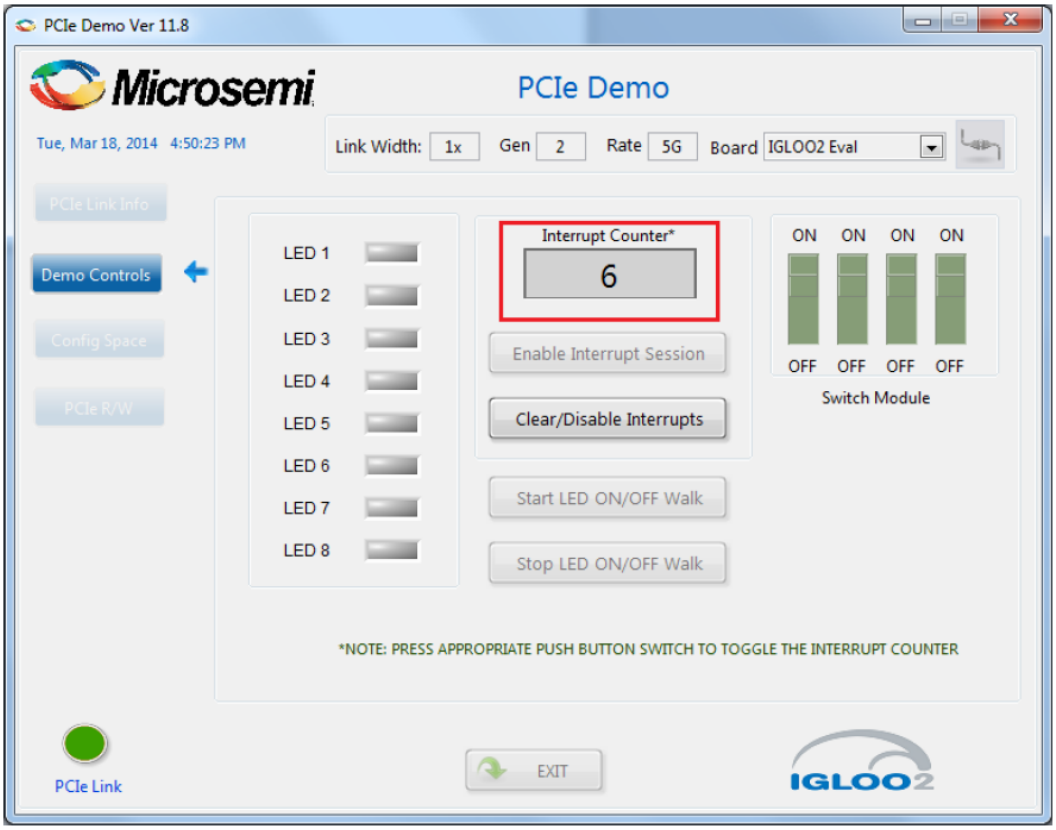
Для взаимодействия с пользователем ПК в файлах демонстрационного проекта предусмотрена специальная графическая среда, установщик которой находится в папке GUI. Нажмите setup.exe и установите программу с настройками по умолчанию. После этого в меню Пуск появится группа программ PCIe Demo, в которой располагается ярлык на исполняемый файл PCIe Demo GUI. Программа имеет простой интерфейс взаимодействия с созданным проектом в ПЛИС. Для установления соединения с платой следует нажать на значок



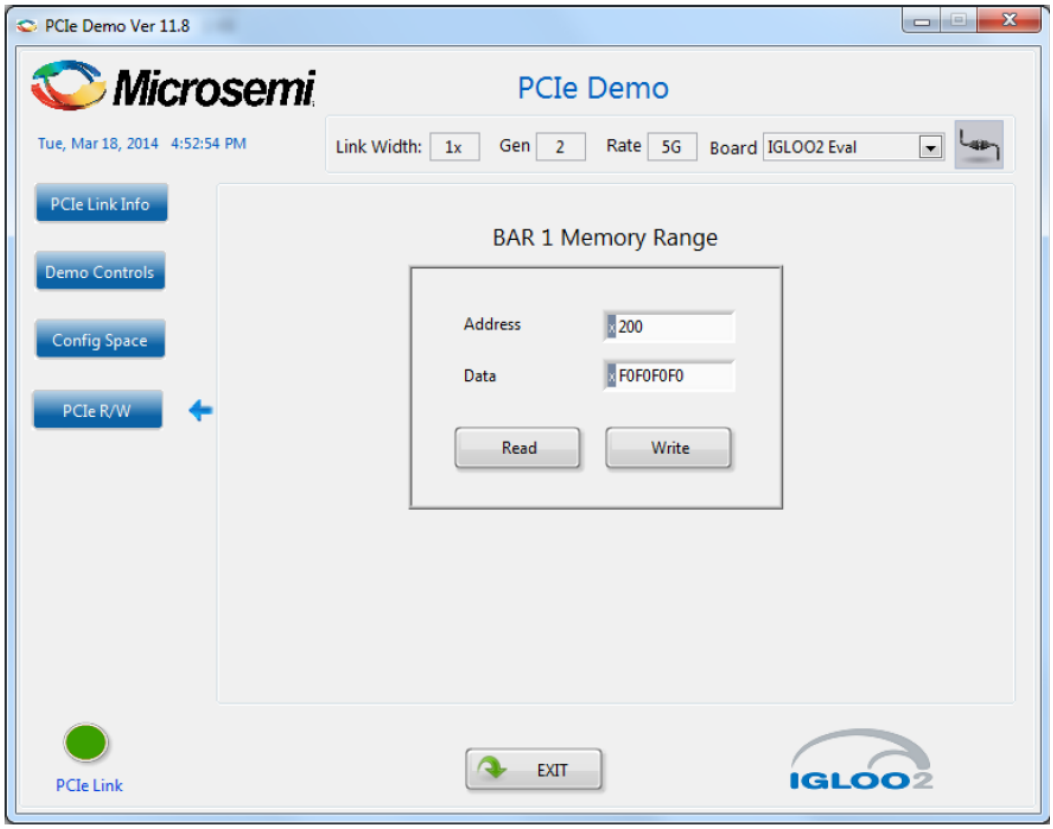
После этого индикатор PCIe Link сменит цвет с красного на зеленый и в окне появится информация об устройстве.



Щелкнув по Demo Controls пользователь переходит к просмотру и управлению состоянием GPIO и прерываний. Нажмите на обозначения светодиодов и удостоверьтесь, что они меняют своё состояние на плате. Измените состояние группы переключателей и просмотрите изменение в окне. Нажатием кнопки SW4 пользователь инициирует прерывание PCIe, счётчик которых отображается в окне.



Запись и чтение памяти LSRAM производится во вкладке PCIe R/W.



Измените состояние ячейки памяти и удостоверьтесь, что значение записалось. Для выхода из программы нажмите кнопку EXIT.

5.2. IGLOO2 Flash*Freeze Lab

Микросхемы SmartFusion2 и IGLOO2 включают в себя технологию и режим сверхнизкого потребления Flash*Freeze (сокр. F*F). Вход в режим F*F происходит с сохранением содержимого SRAM и триггеров, и выход из F*F обеспечивает оперативное восстановление работоспособности активного режима.

Одной из функций системного контроллера (System Controller) IGLOO2 является обработка системных запросов, поступающих через блок коммуникации (COMM_BLK). Системные запросы подразделяются на группы. Одной из таких услуг является вход в режим F*F. Некоторые настройки режима такие, как источник тактового сигнала для HPMS в режиме покоя или определение содержимого SRAM в течение F*F.

В этом руководстве рассказано как задать настройки F*F во время проектирования дизайна в Libero. Также будет описано, как войти в режим F*F используя System Services через IP-ядро CoreSysServices. Данное ядро взаимодействует с COMM_BLK через интерфейс FIC.

Архив с файлами проекта для последней версии Libero можно получить на сайте www.microsemi.com.

Структурная схема проекта показана на рисунке 5.3.

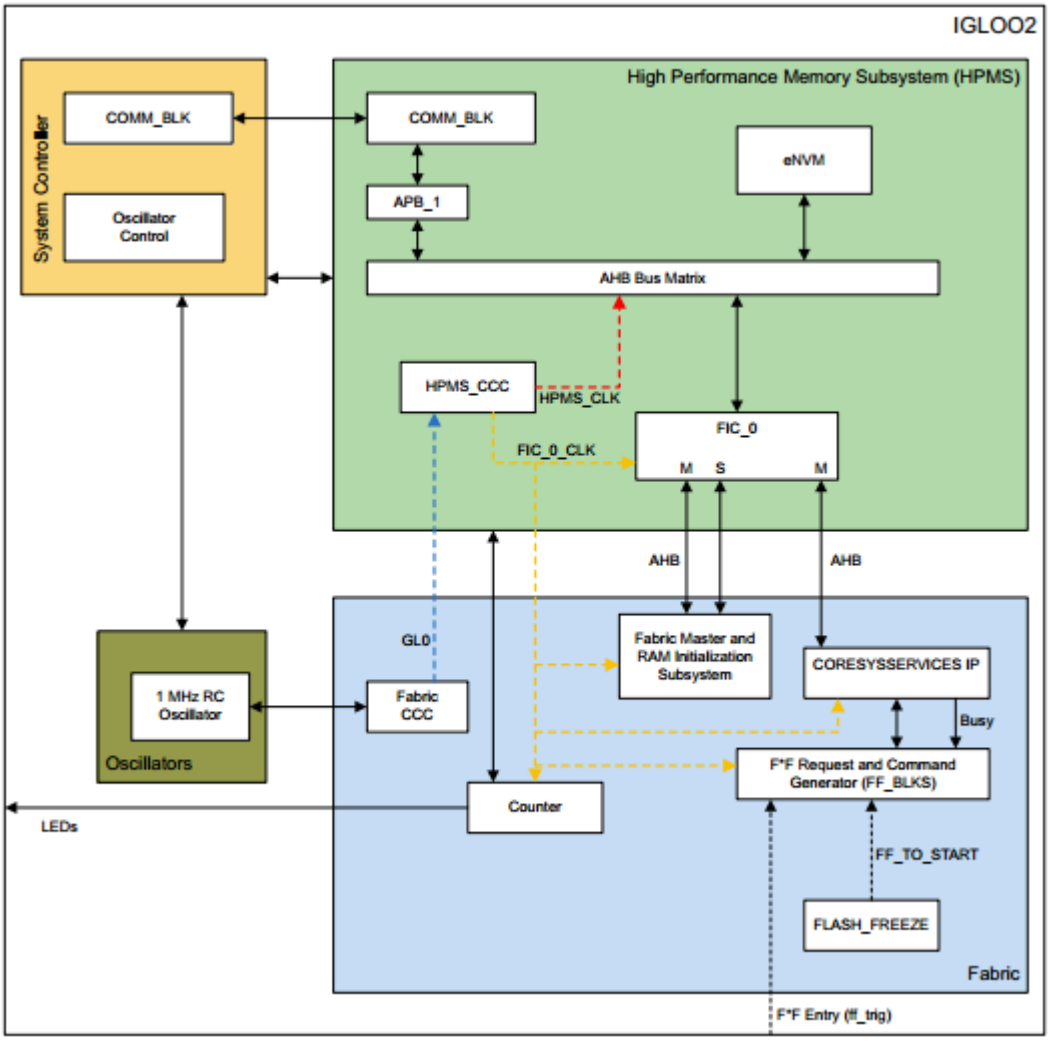


Рис. 5.3 Структурная схема проекта демонстрации режима FlashFreeze

Проект включается в себя сконфигурированную в SystemBuilder HPMS, счётчик, память SRAM, набор IP-ядер (CoreSysServices, CoreAHBLite, CoreAHBToAPB3, и CoreAPB3), FLASH_FREEZE макрос, AHB master, on-chip 1 MHz RC oscillator, fabric CCC (FCCC) и логика генерации запросов F*F (FF_BLKS).

Верхний уровень в Libero SmartDesign показан на рисунке 5.4.

Fabric AHB Master и SRAM Wrapper (AHBMASTER_FIC_RAM) используются для инициализации SRAM из энергонезависимой Flash созданным на этапе проектирования образом (Data Storage client). В Активном режиме (не F*F) HPMS_CCC сконфигурирована на выдачу тактового сигнала 100 МГц, источником для которой служит порт CLK_BASE. FCCC подает на CLK_BASE тактовый сигнал частотой 50 МГц. Источником сигнала для FCCC является встроенный генератор 1 МГц.

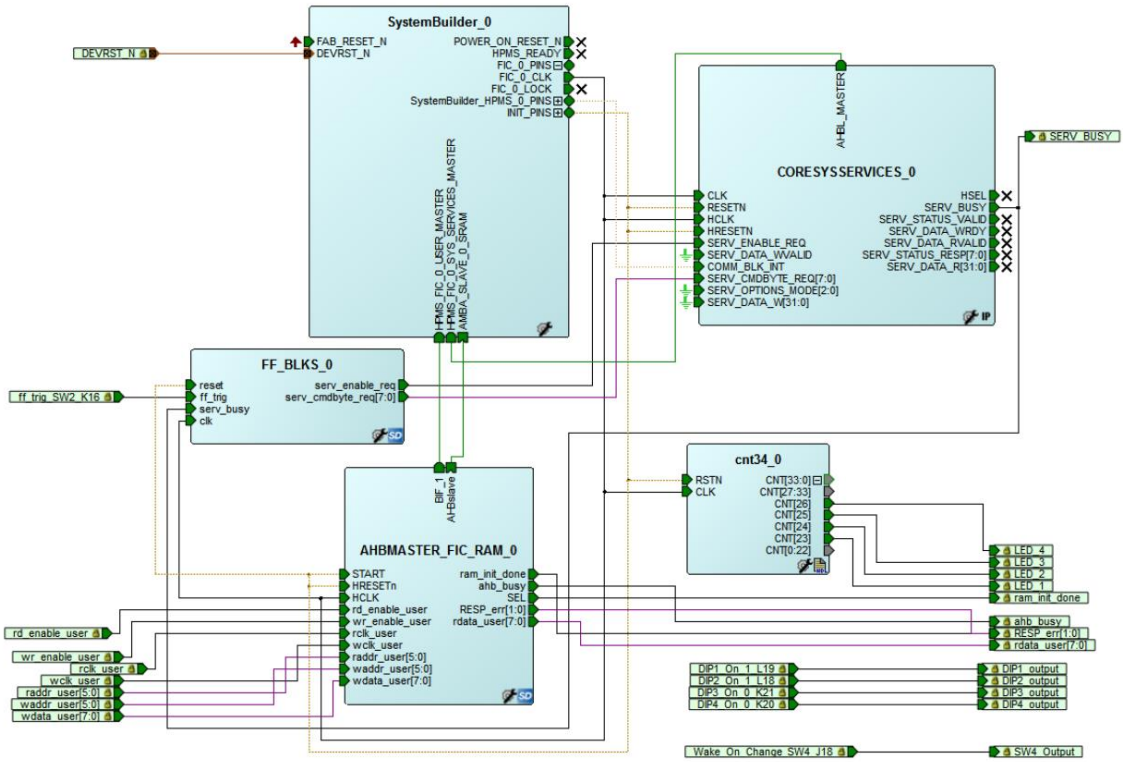


Рис. 5.4. Верхний уровень проекта FlashFreeze

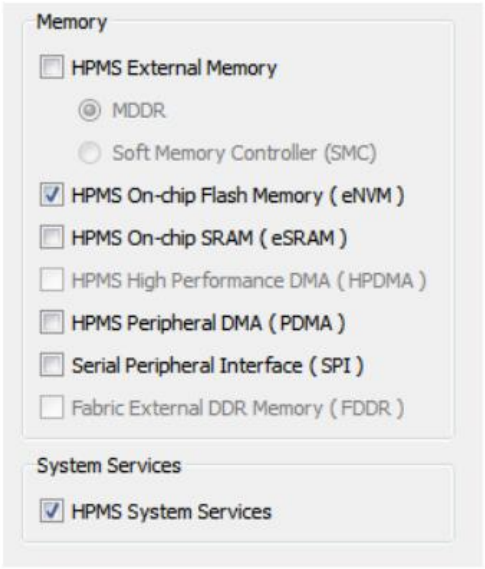
CoreSysServices IP использует только сервис F*F. Оно инициирует вход в режим, посылая команду системному контроллеру как только получит запрос F*F и команду из FF_BLKS. Последний посылает запрос и команду опираясь на сигнал ff_trig (поступает извне). FF_BLKS также проверяет сигналы занятости CoreSysServices и FF_TO_START из макроса FLASH_FREEZE.

FF_TO_START выставляется системным контроллером и указывает на то, что сервис F*F сейчас запустится. Если сигнал в активном уровне, значит осталось около 10 мкс до отключения ядра.

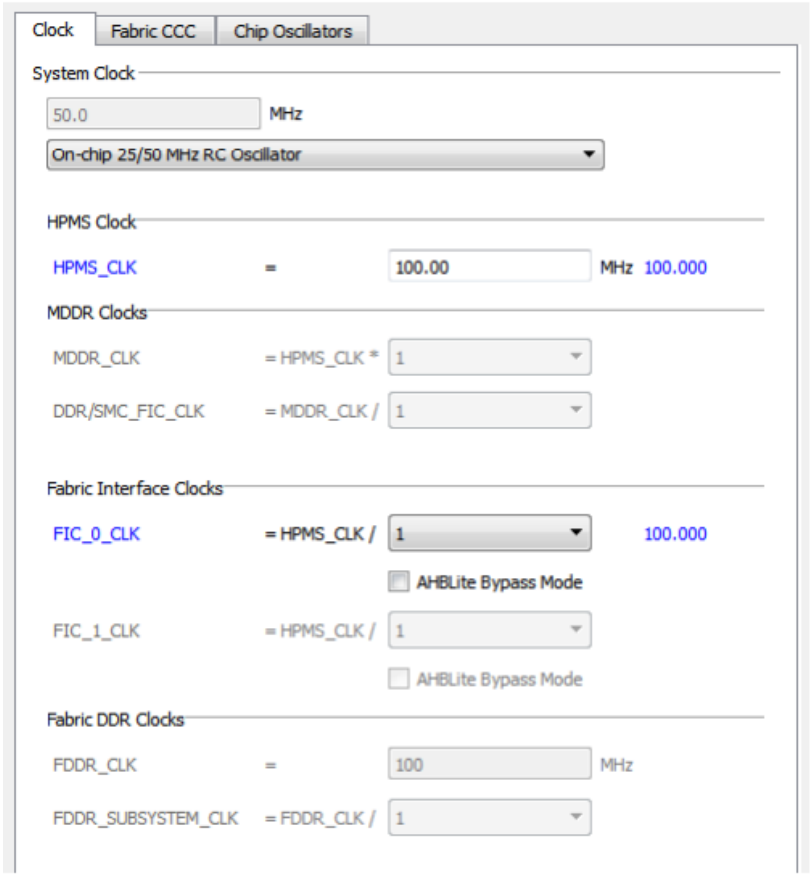
Выход из режима F*F можно осуществить разными способами:

- Активность I/O: любое изменение значений на указанных выводах;
- Определённое состояние I/O: выход по условию, задаваемому в виде двоичного кода на выводах;

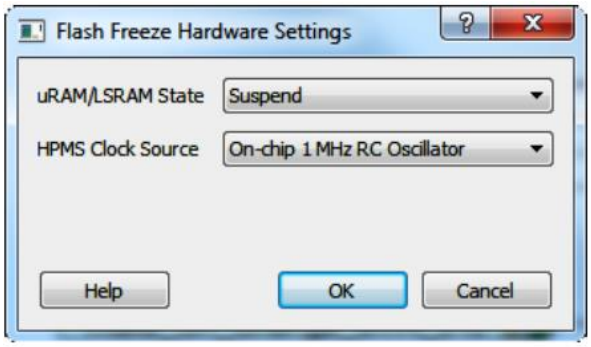
Для работы дизайна необходимо сконфигурировать HPMS включив внутрикристальную Flash-память и услуги системного контроллера HPMS System Services.



Источники тактового сигнала в HPMS следует настроить, как это сделано на рисунке.



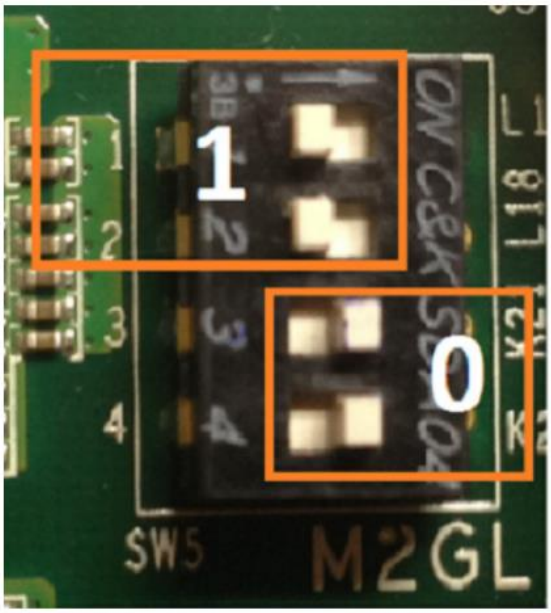
Источник тактового сигнала для HPMS в режиме F*F и состояние памяти указываются в диалоговом окне Flash*Freeze Hardware Settings.



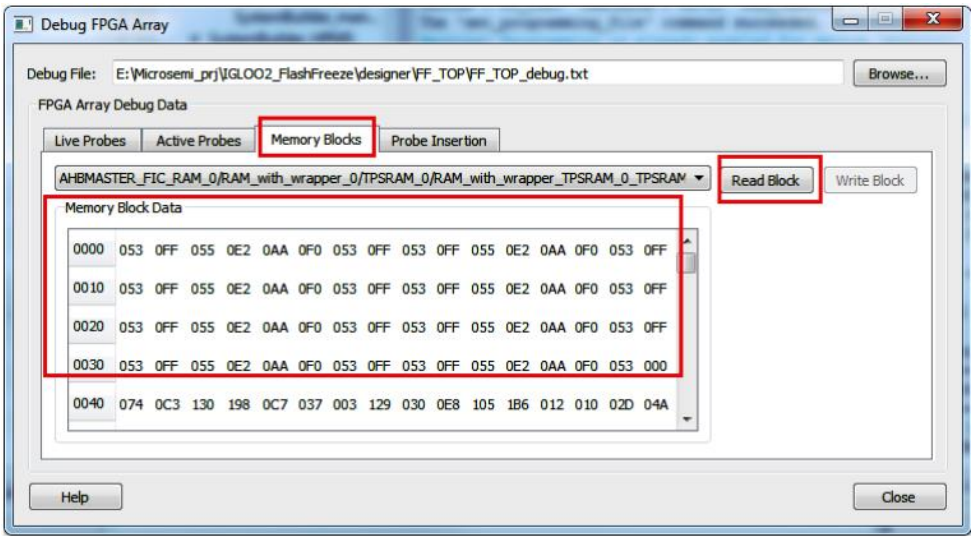
Настройки выхода из режима F*F задаются в I/O Editor. Wake_on_Change означает выход по изменению состояния сигнала, Wake_On_1/0 означает участие в составлении кодовой последовательности для выхода из режима.

Port Name	Pin Number	I/O state in Flash*Freeze mode	Resistor Pull	I/O available in Flash*Freeze mode	Low Power Exit
Wake_On_Change_SW4_	J18	TRISTATE	None	No	Wake_On_Change
DIP1_On_1_L19	L19	TRISTATE	Down	No	Wake_On_1
DIP2_On_1_L18	L18	TRISTATE	Down	No	Wake_On_1
DIP3_On_0_K21	K21	TRISTATE	Up	No	Wake_On_0
DIP4_On_0_K20	K20	TRISTATE	Up	No	Wake_On_0
ff_trig_SW2_K16	K16	TRISTATE	None	No	Off

Кодовая последовательность на отладочной плате Evaluation Kit задается с помощью блока DIP переключателей.



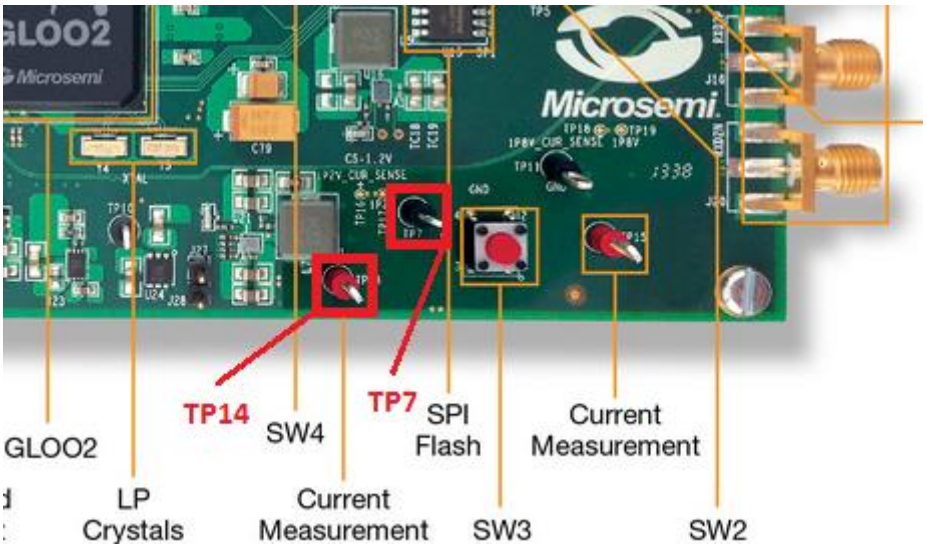
После прошивки ПЛИС можно экспериментировать с режимом F*F. Вход в режим осуществляется по состоянию переключателей либо по кнопке SW2. Выход из режима — SW4. Состояние памяти при этом можно посмотреть используя инструмент SmartDebug (см. разд. 3.10).



Для измерения тока потребления предусмотрены две точки — TP14 и TP7 между которыми установлен резистор с известной величиной сопротивления. Измеряя напряжение между ними, ток и мощность можно вычислить по следующим формулам:

$$Core\ Current\ (mA) = Measured\ Voltage\ (mV) \div 5\ (Scaling\ Factor)$$

$$Core\ Power\ (mW) = 1.2 \times Core\ Current$$



5.3.SmartFusion2 ARM Cortex-M3 Lab

Firmware Catalog в Libero SoC отображает список доступных возможностей подсистемы MSS в SmartFusion2. Для каждого из встроенных блоков может быть сгенерирован демонстрационный проект, который содержит пример использования блока в составе системы на SmartFusion2. В качестве IDE для разработки программного обеспечения может быть выбрана SoftConsole, IAR или Keil, в зависимости от предпочтений разработчика. Демонстрационный проект содержит только описание встраиваемого ПО и не содержит проект Libero. В данном

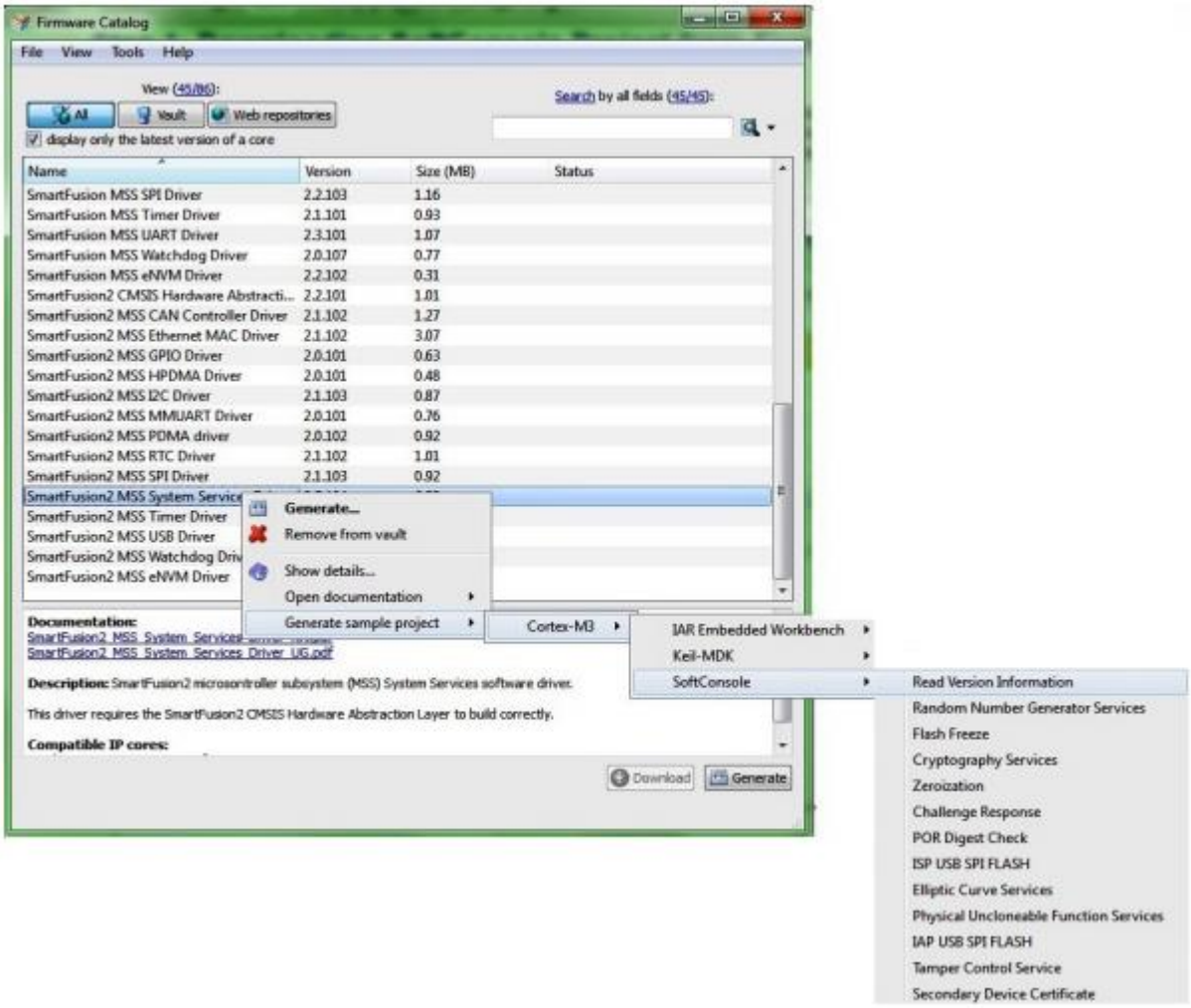
разделе предлагается пошаговое описание сборки проекта Libero и интеграции в него демонстрационного проекта встроенного ПО.

В данном примере будет описано создание системы, которая выдаёт по запросу следующую информацию:

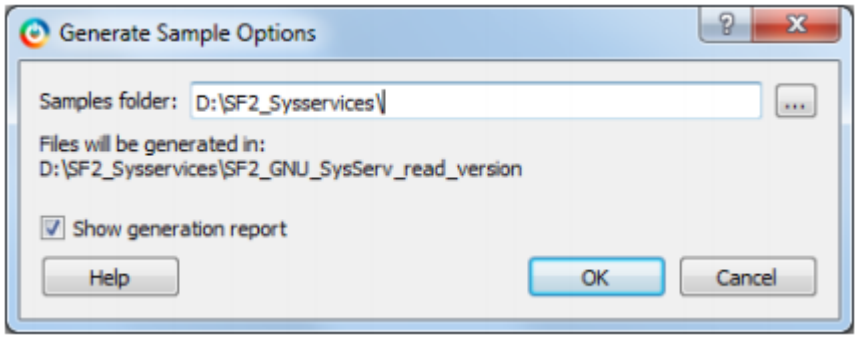
- Serial Number;
- USERCODE;
- User Design Version;
- NVM Data Integrity Check.

Для этого используются службы, которые предоставляет системный контроллер SmartFusion2. Для выполнения работы потребуется SmartFusion2 Security Kit, Rev D или старше. Все файлы проекта, описываемого в этом разделе можно получить на сайте www.microsemi.com.

Перед проведением работы нужно скачать из удаленного депозитария требуемый демонстрационный проект. Для этого нужно открыть Firmware Catalog в группе программ Microsemi в меню Пуск и проследовать по выпадающему меню, как на рисунке.



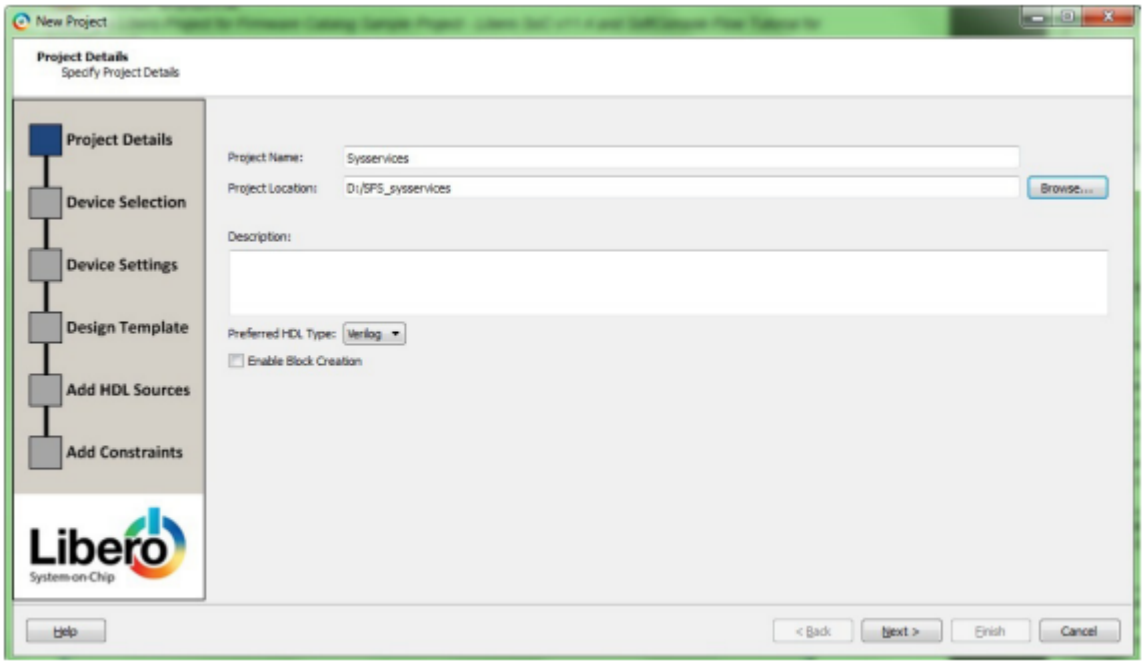
Откроется диалоговое окно, в котором следует указать расположение демонстрационного проекта.



Далее перейдем к созданию проекта в Libero SoC. Откройте Libero, например, из группы программ Microsemi в меню Пуск. Создайте новый проект (щелчок по New или Project -> New Project).

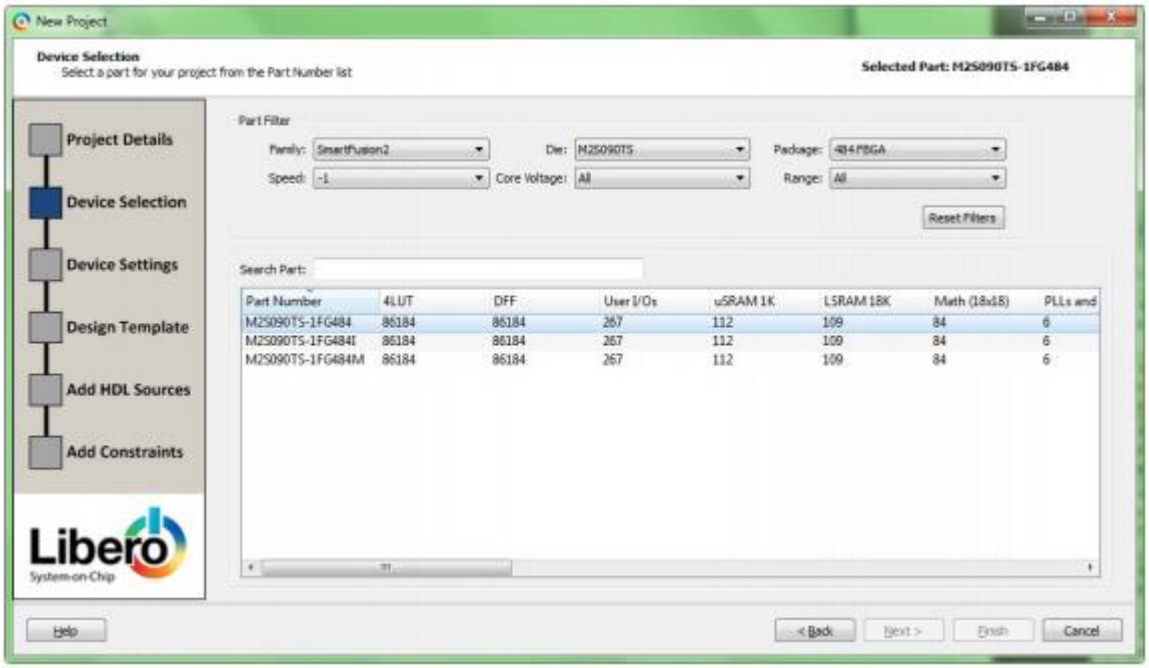
На первом этапе создания проекта введите следующую информацию:

- Project Name: Sysservices
- Project Location: Выберете расположение, например, D:/SF2_Sysservices
- Preferred HDL Type: Verilog

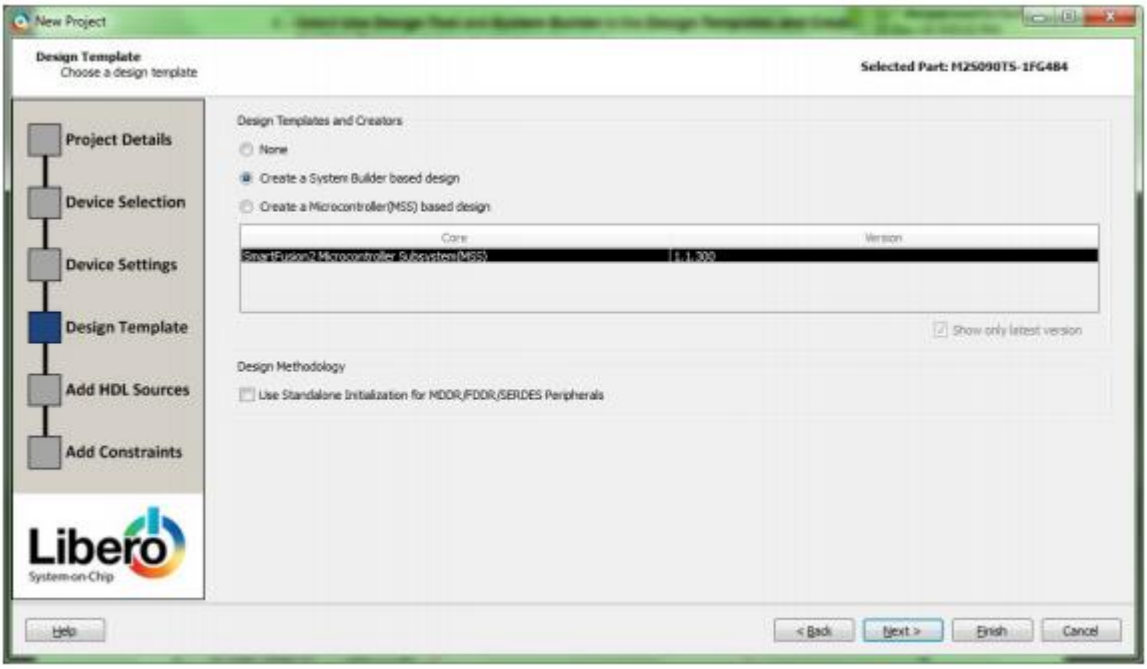


Нажмите Next. На следующей странице введите следующую информацию:

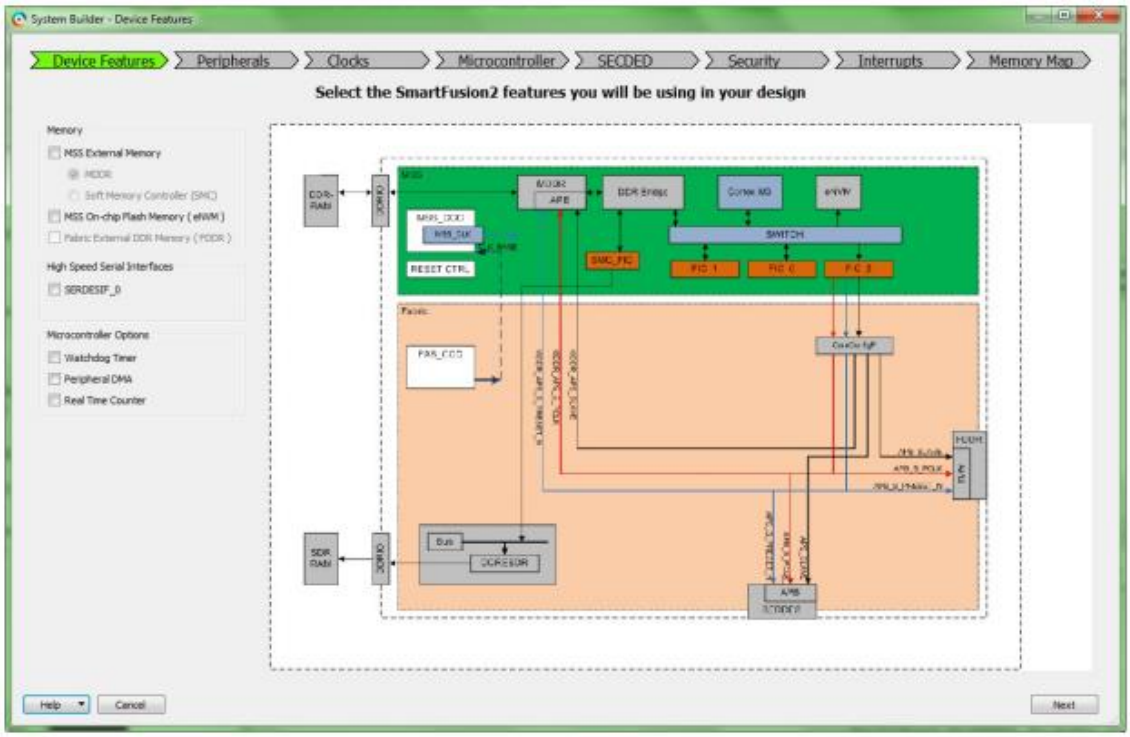
- Family: SmartFusion2
- Die: M2S090TS
- Package: 484 FBGA
- Speed: -1
- Core Voltage: All
- Range: All



Нажмите Next. На следующем шаге оставьте настройки по умолчанию. Нажмите Next, отметьте Create a System Builder based design.



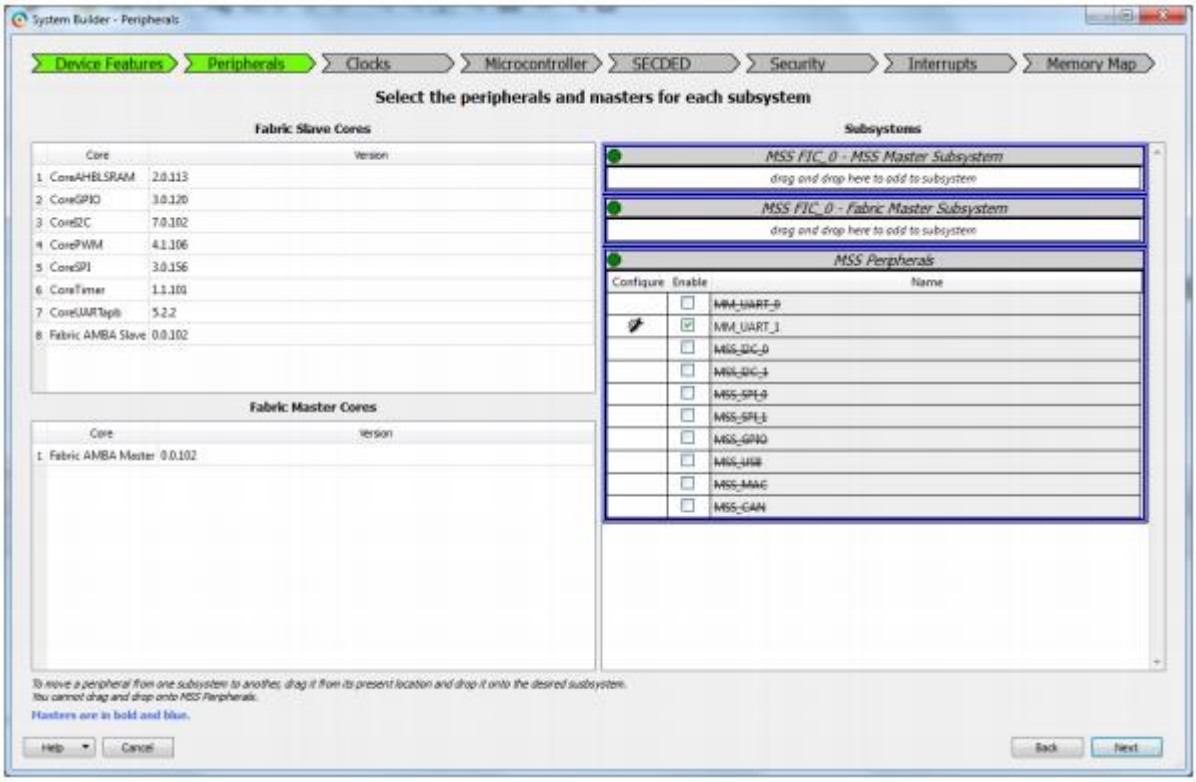
Нажмите Finish. Откроется диалоговое окно ввода имени компонента SystemBuilder. Введите имя Sysservices и нажмите ОК. Откроется окно конфигурации SystemBuilder.



Нажмите Next, на следующей странице снимите галочки со следующих намиенований периферийных устройств:

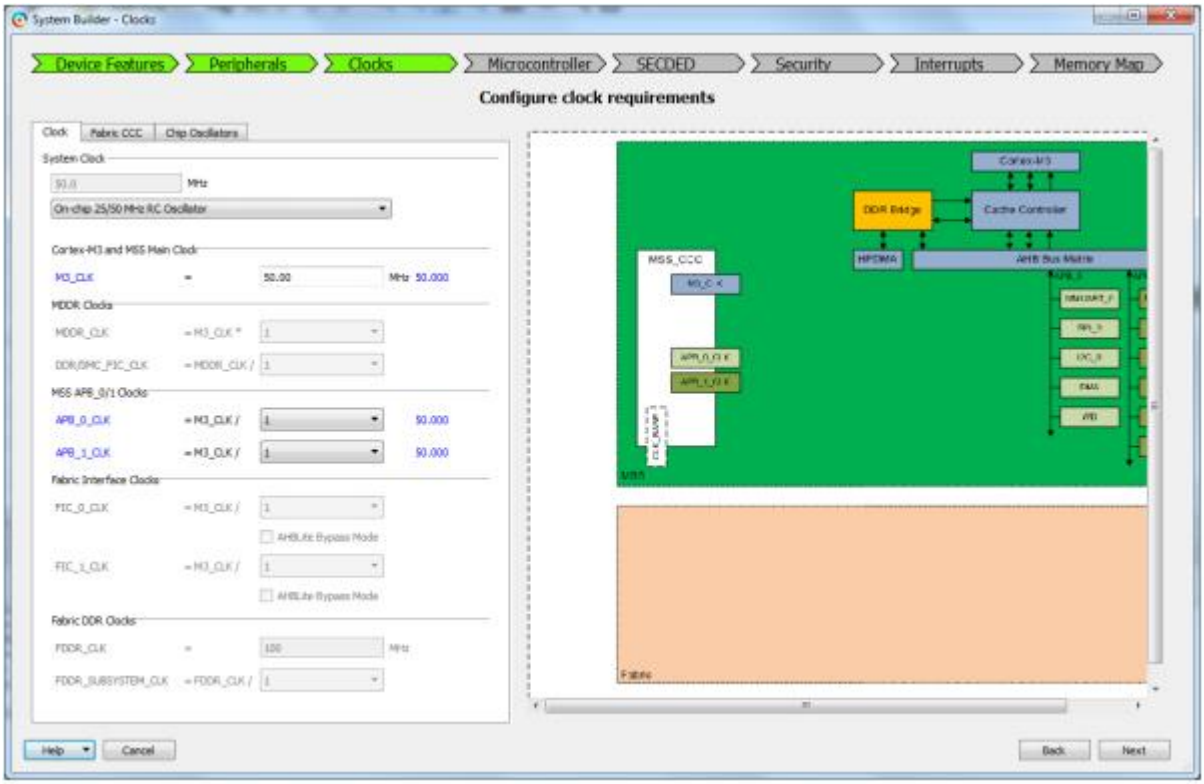
- MMUART_0
- SPI_0 и SPI_1
- I2C_0 и I2C_1
- USB

- Ethernet
- CAN



Нажмите Next. На следующей странице введите следующие настройки:

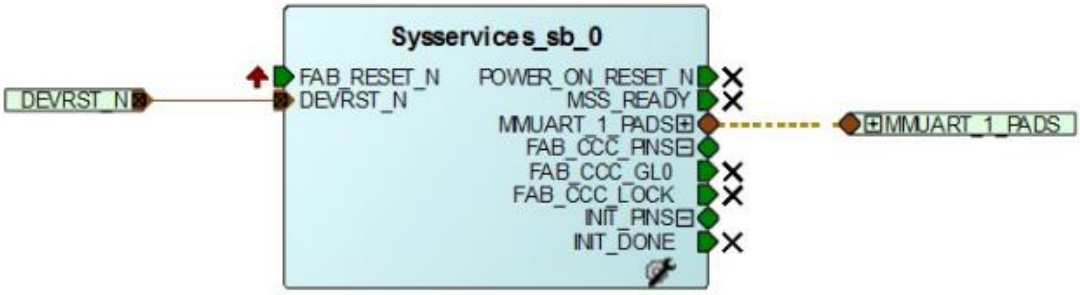
- Выберите частоту System Clock 50 MHz и источник On-chip 25/50 MHz RC Oscillator
- Введите в поле M3_CLK 50 MHz
- Настройте частоты APB_0_CLK и APB_1_CLK как M3_CLK/1
- Остальные настройки оставьте как есть



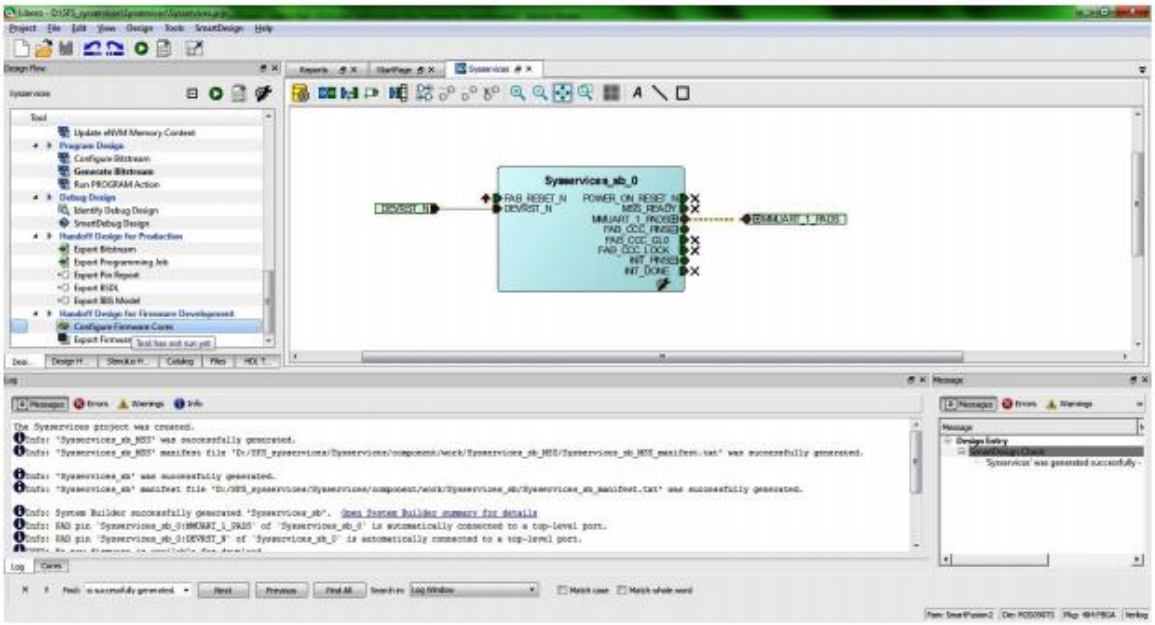
Нажимайте Next и оставляйте настройки по умолчанию на остальных страницах, на странице Memory Page нажмите Finish. В рабочей области SmartDesign появится символическое обозначение подсистемы MSS.

Произведите следующие настройки портов подсистемы:

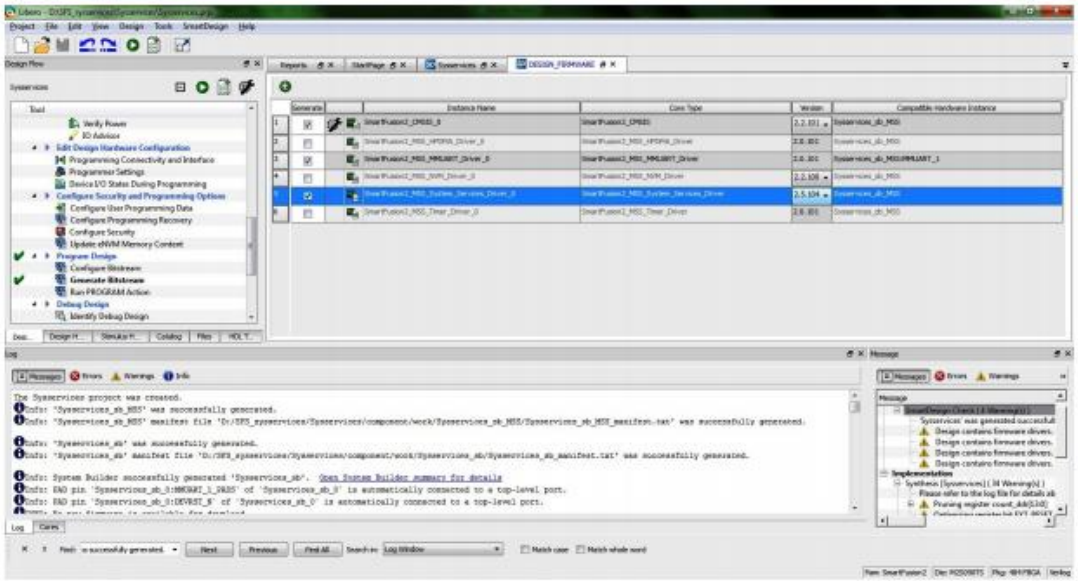
- POWER_ON_RESET_N -> Mark Unused.
- MSS_READY -> Mark Unused.
- В группе INIT_PINS: INIT_DONE -> Mark Unused.
- В группе FAB_CCC_PINS: FAB_CCC_GL0 -> Mark Unused.
- FAB_CCC_LOCK -> Mark Unused.
- FAB_RESET_N -> Tie High.
- Нажмите File -> Save.



Таким образом, MSS настроена и готова к работе. Далее сконфигурируем и сгенерируем проект встраиваемого ПО. В Design Flow сделайте двойной щелчок на Configure Firmware Cores в секции Handoff design for Firmware Development.

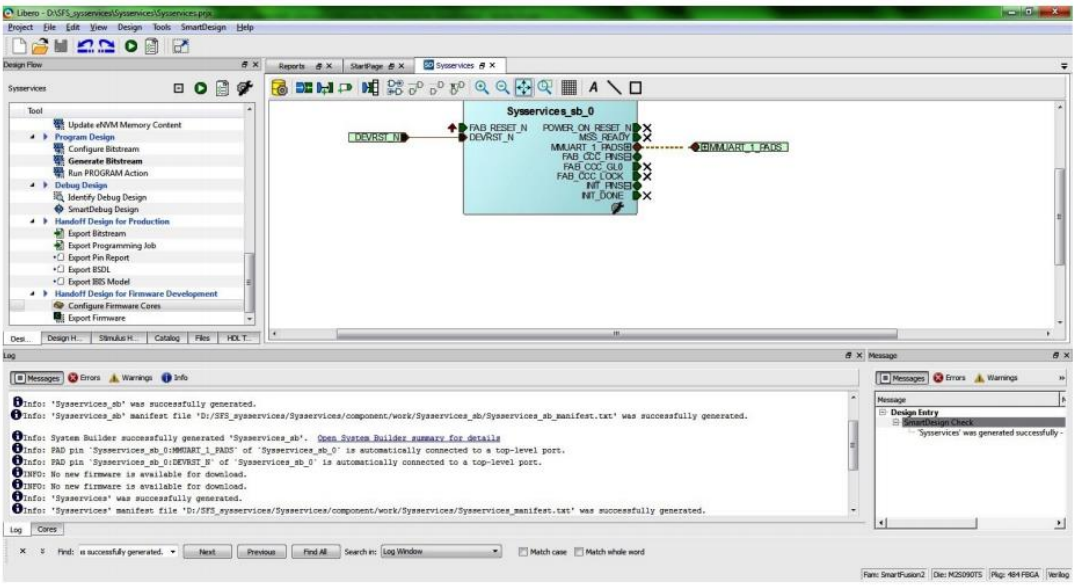


Откроется окно DESIGN_FIRMWARE. В этом окне снимите галочки напротив всех пунктов кроме CMSIS, MMUART_0 и System Services.

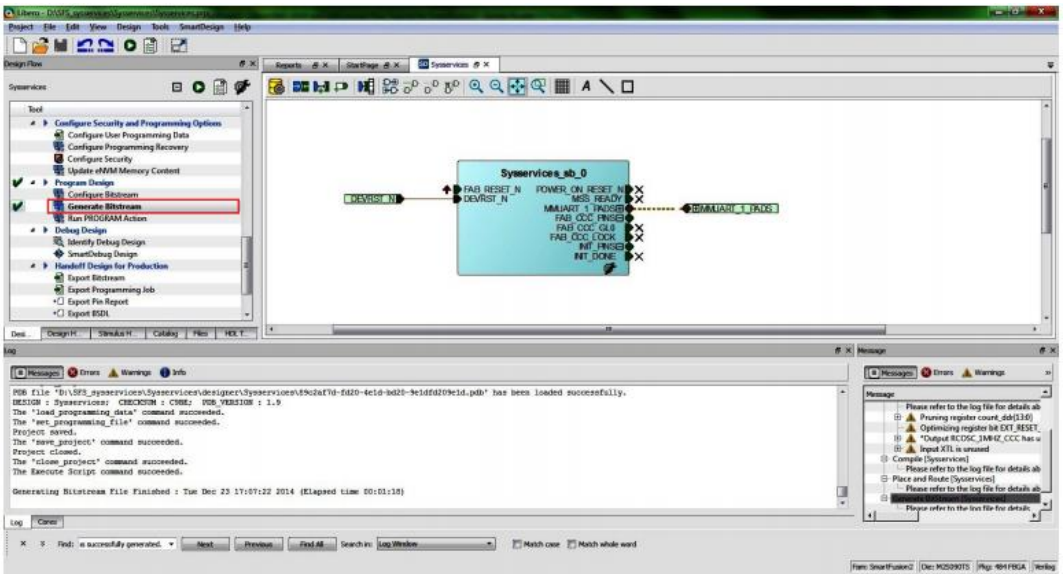


Далее сгенерируйте компонент SmartDesign, нажав . Появится сообщение об успешной генерации компонента:

```
Info: 'Sysservices' was successfully generated.
```



Нажмите Generate Bitstream, чтобы получить файл прошивки микросхемы.



Нажмите Export Firmware в Design Flow, откроется соответствующее диалоговое окно. Введите следующие настройки:

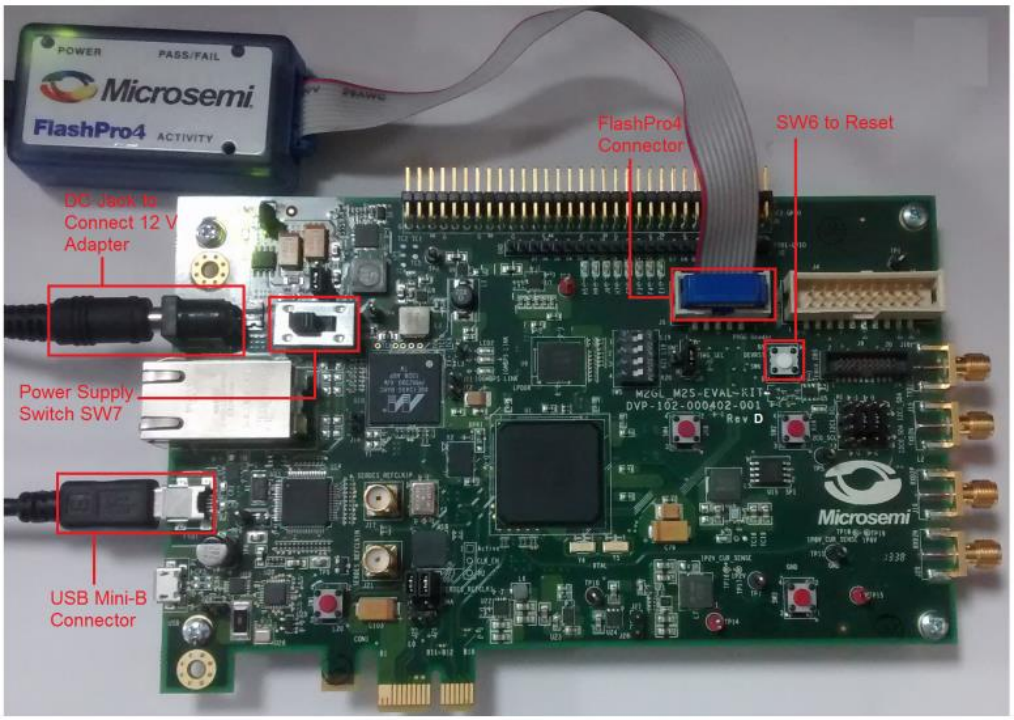
- Поставьте галочку Create project for selected Software Tool Chain.
- Выберите SoftConsole3.4 из выпадающего списка.

Нажмите ОК, при этом появится окно, сообщающее об успешной генерации проекта.

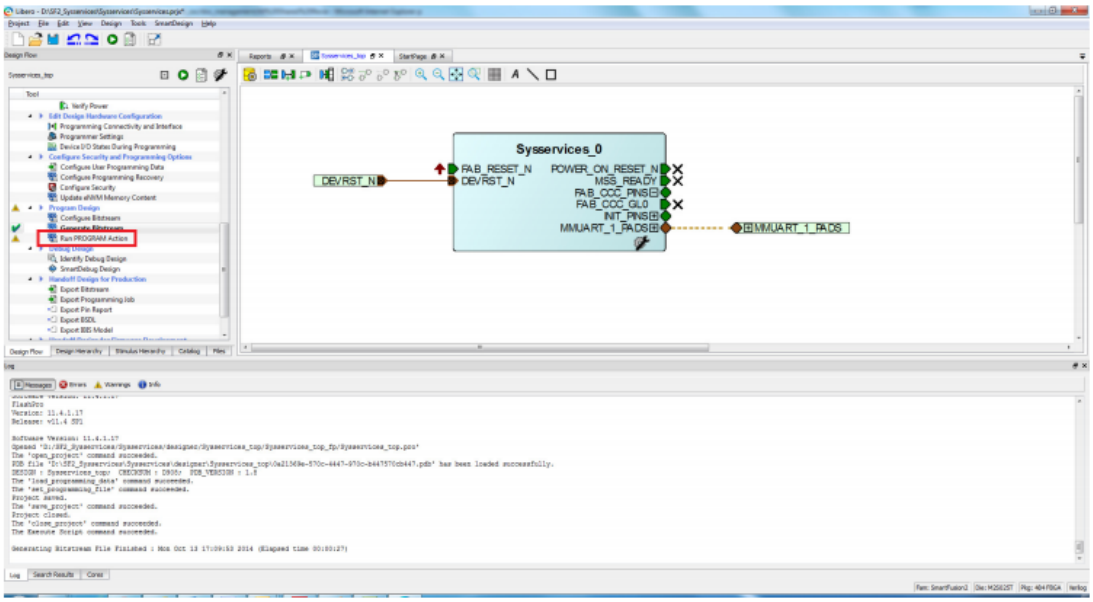
Следующий шаг — программирование микросхемы SmartFusion2. Для проведения программирования присоедините программатор FlashPro к разъему J5 на плате и удостоверьтесь в правильной расстановке джамперов:

Номер джампера	От	До	Комментарий
J22, J23, J24, J8, J3	1	2	Это стандартная конфигурация джамперов при поставке. Удостоверьтесь, что эта конфигурация не нарушена.

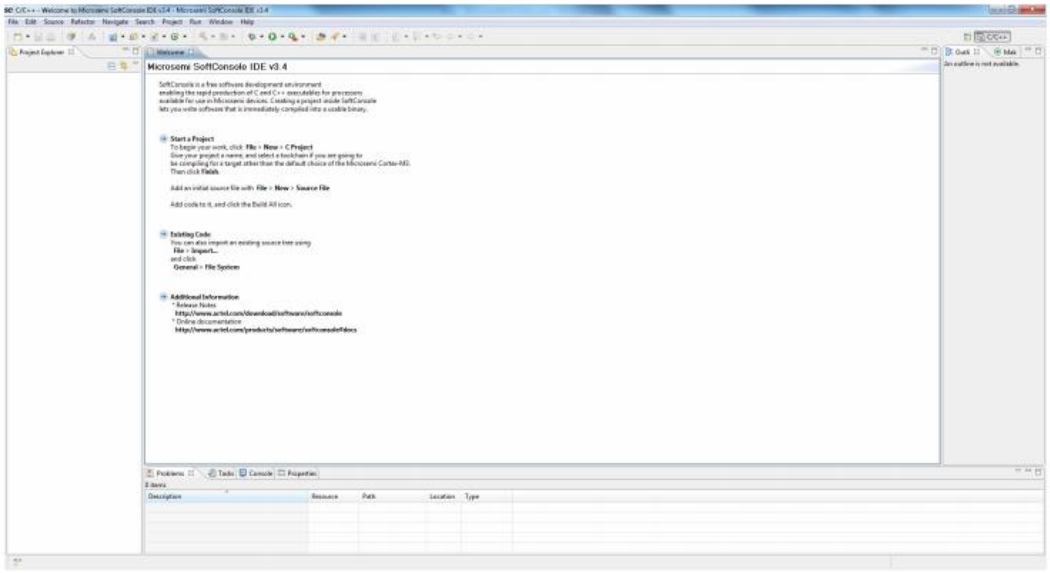
Подключите питание к разъему J6, включите питание платы переключателем SW7.



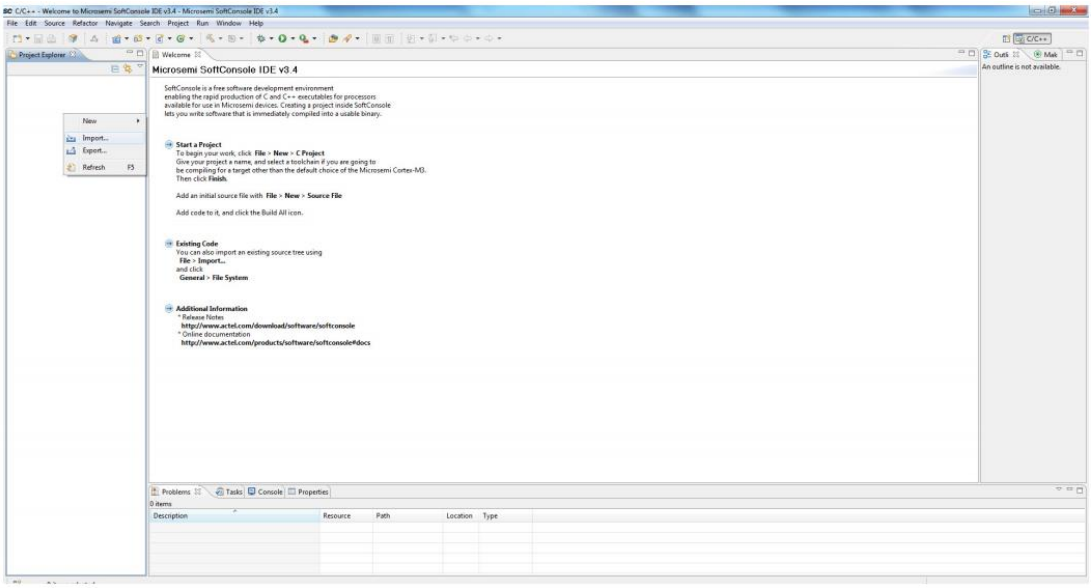
Нажмите Run PROGRAM Action в Design Flow.



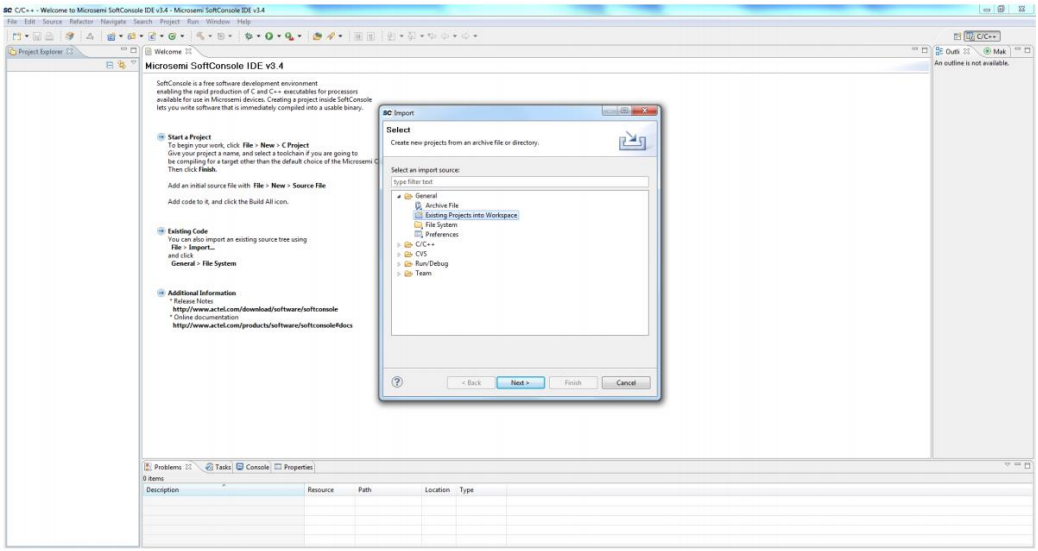
Перейдем к разработки встроенного ПО в SoftConsole. Откройте SoftConsole в автономном режиме.



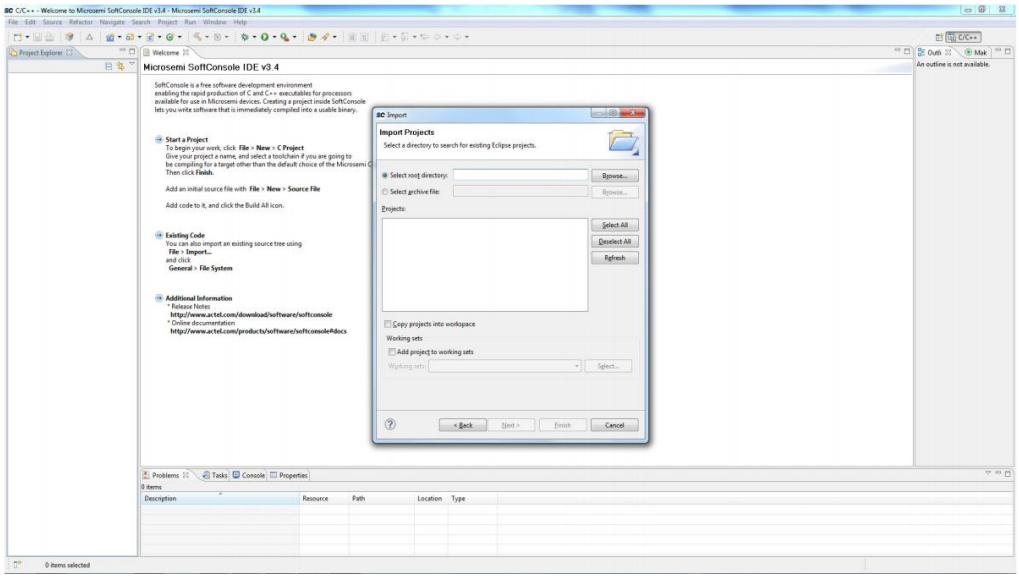
Сделайте щелчок правой кнопкой мыши в Project Explorer и выберите Import.



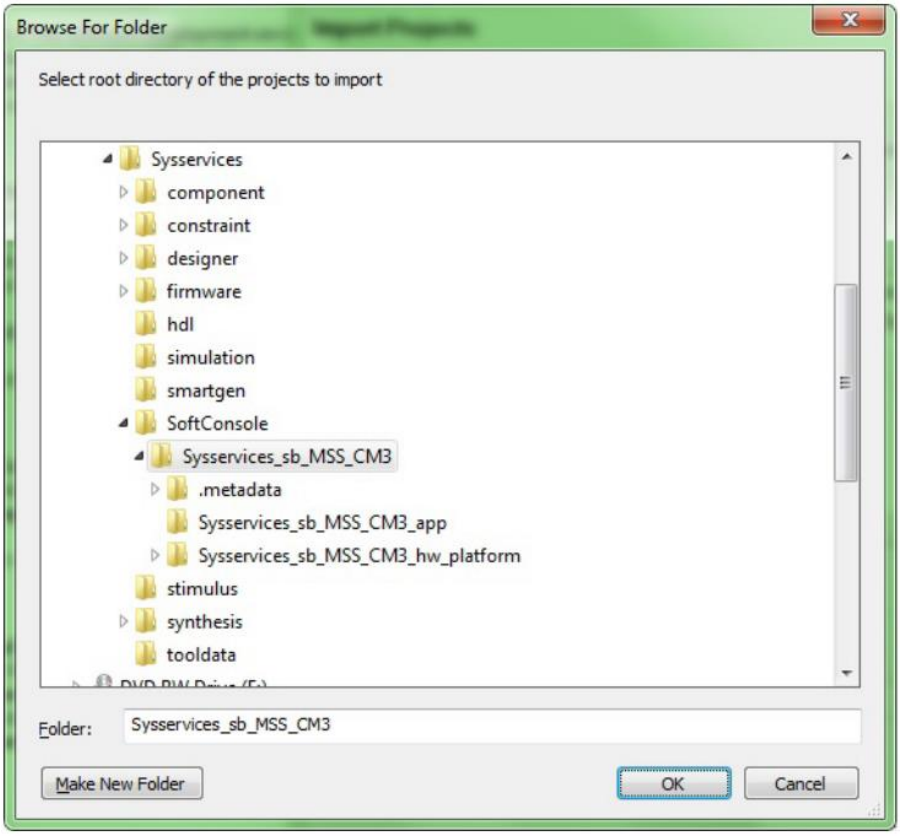
В открывшемся диалоговом окне выберите General -> Existing Projects into Workspace. Нажмите Next.



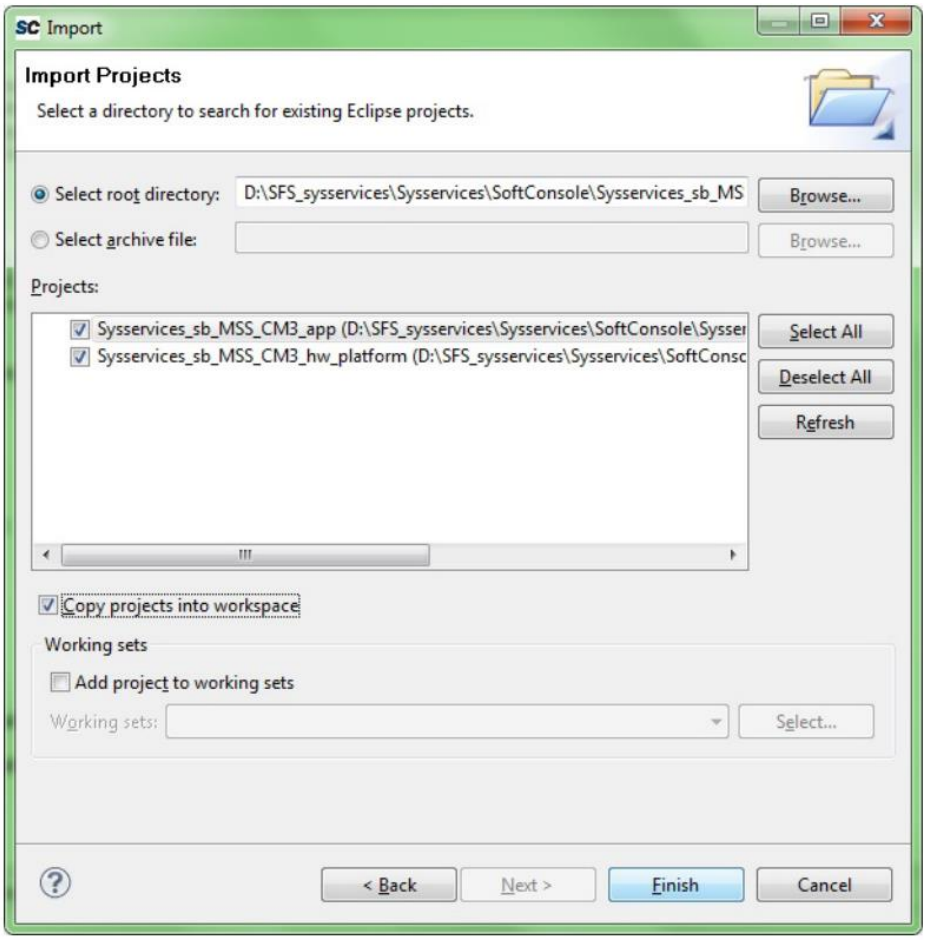
Откроется диалоговое окно Import.



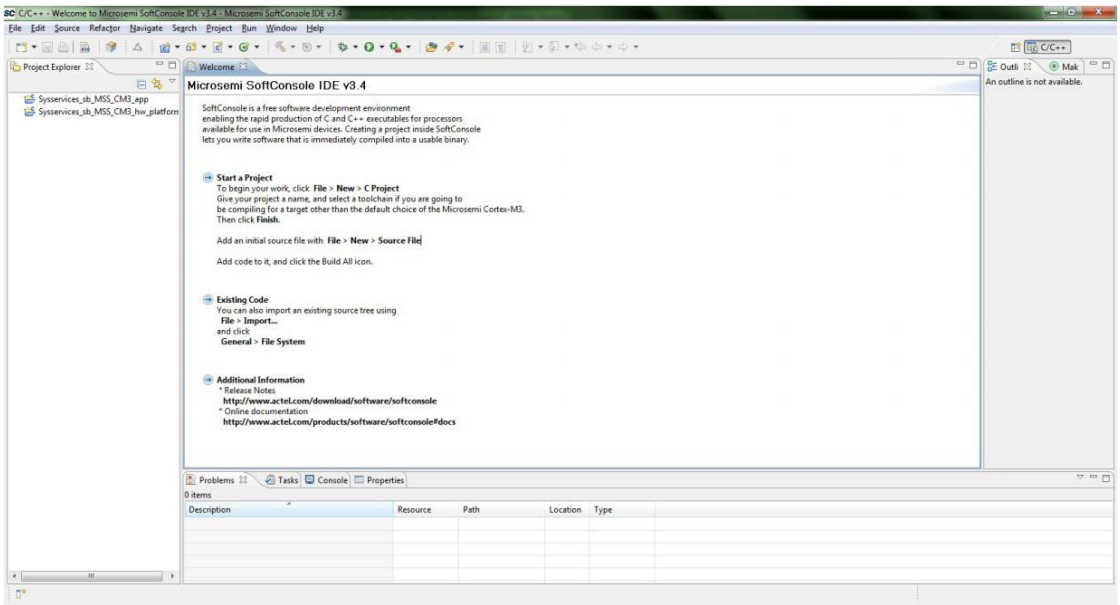
Нажмите Browse и найдите проект для SoftConsole в папке с проектом Libero. Нажмите OK.



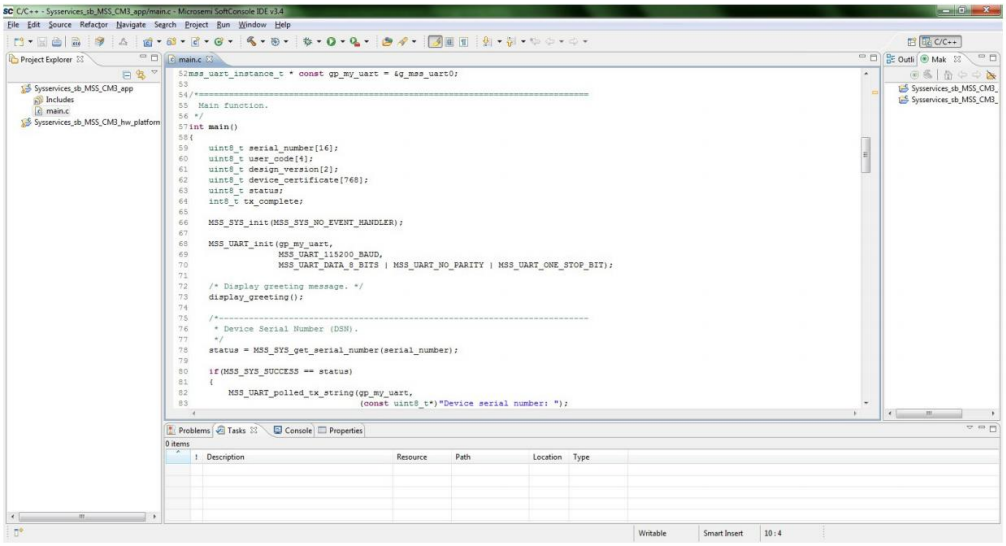
Проекты будут отображены в поле Projects. Нажмите Finish.



Выбранные проекты будут отображены в главном окне SoftConsole в Project Explorer.

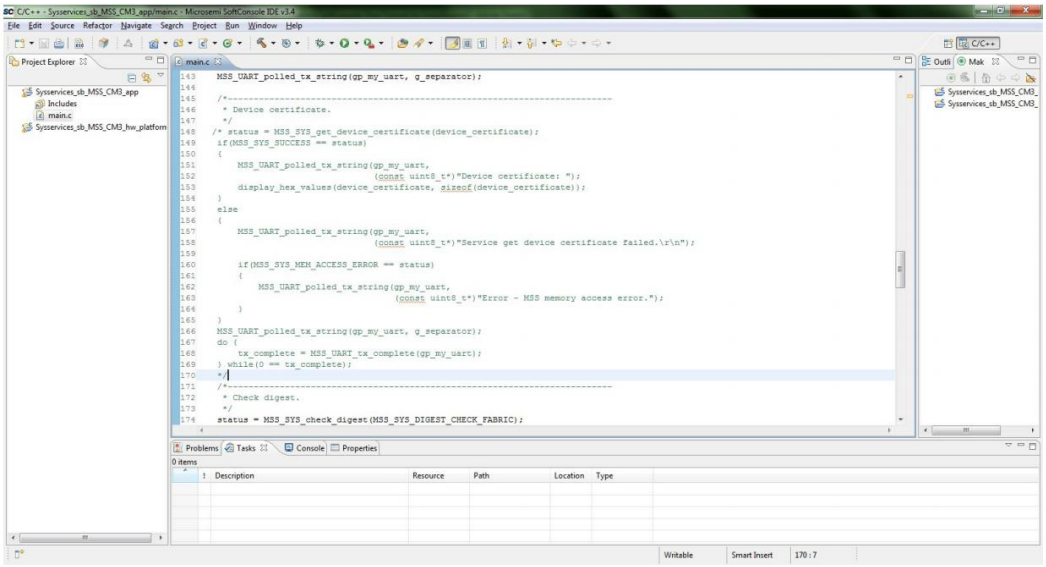


Найдите папку с проектом, сгенерированным из Firmware Catalog, найдите в ней файл main.c и замените им аналогичный файл в папке проекта Sysservices_sb_MSS_CM3. Откройте файл в SoftConsole из Project Explorer.

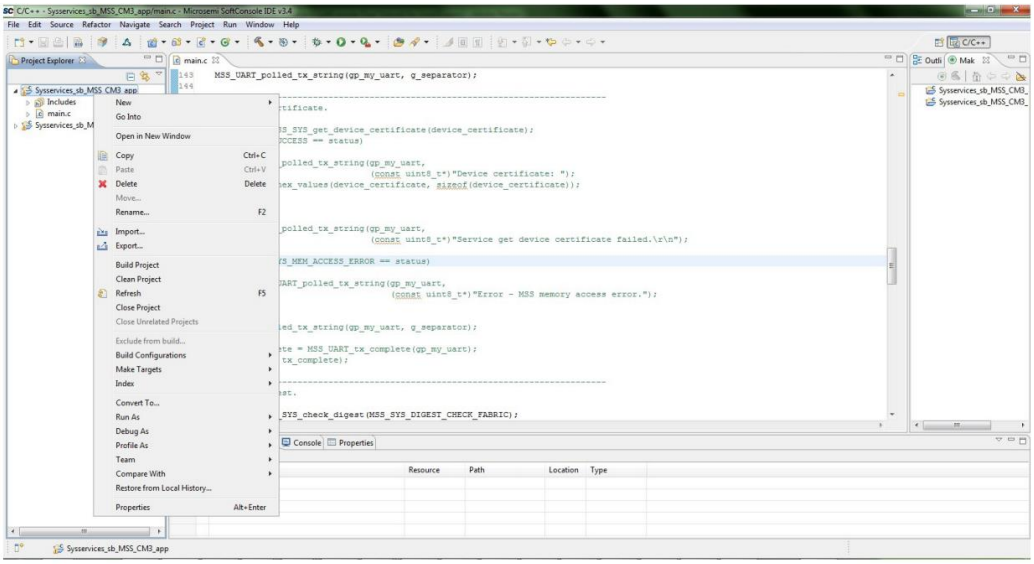


В данном руководстве не рассматривается сервис Device certificate, он будет доступен в следующих выпусках. В связи с этим следует немного изменить программу.

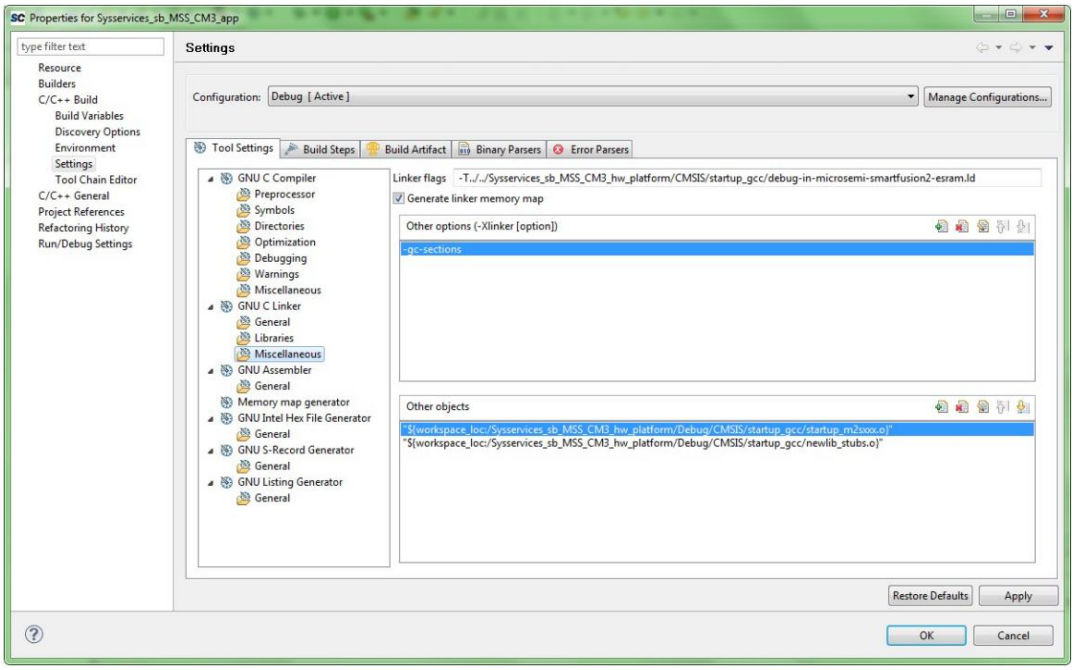
- Закомментируйте в main.c строки, в которых вызывается сервис Device Certificate.
- Измените gp_my_uart на &g_mss_uart1.



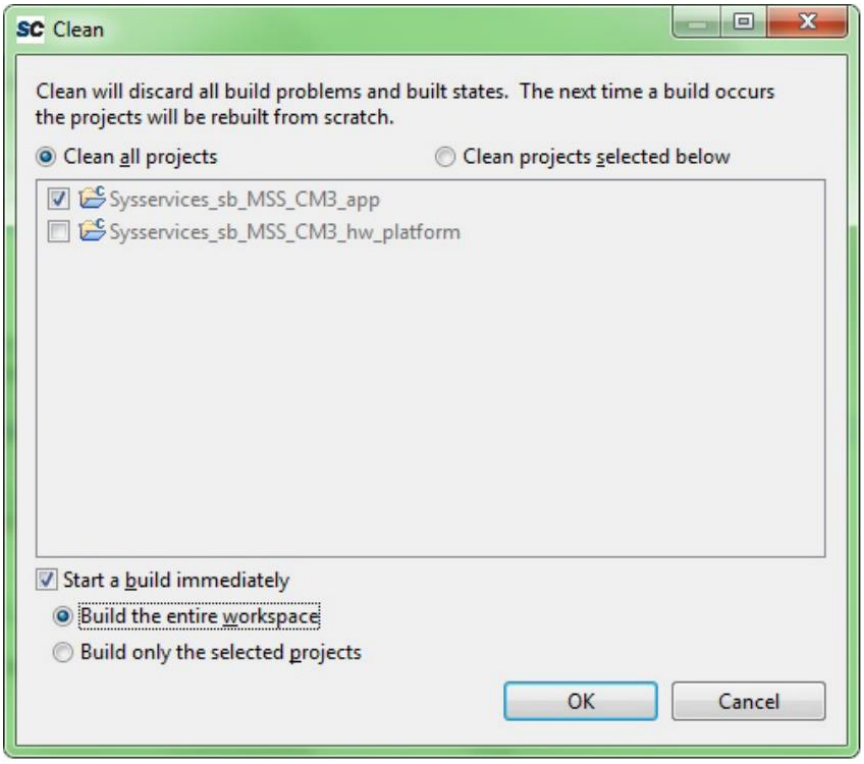
Щелкните правой кнопкой мыши на Sysservices_sb_MSS_CM3 и выберите Propeties из выпадающего меню.



Произведите настройкb GNU C Linker в C/C++ Build -> Settings -> GNU C Linker -> Miscellaneous: в поле Linker flags задайте debug-in-microsemi-smartfusion2-esarm.ld.

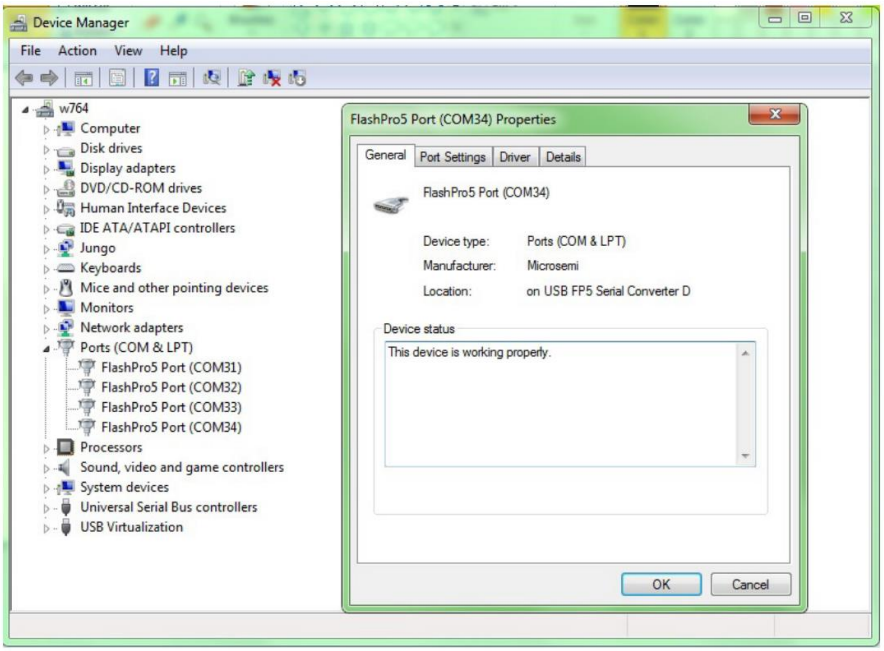


Произвдеите сборку проекта «с нуля» в меню Project -> Clean. В диалоговом окне оставьте все настройки по умолчанию и нажмите ОК.



Далее установите драйвер переходника COM-USB (FTDI), драйвер можно получить по ссылке: www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip

Кабель mini-USB для связи с ПК подключается к разъему J18, мост USB to UART автоматически определится. Проверим это в диспетчере устройств системы.

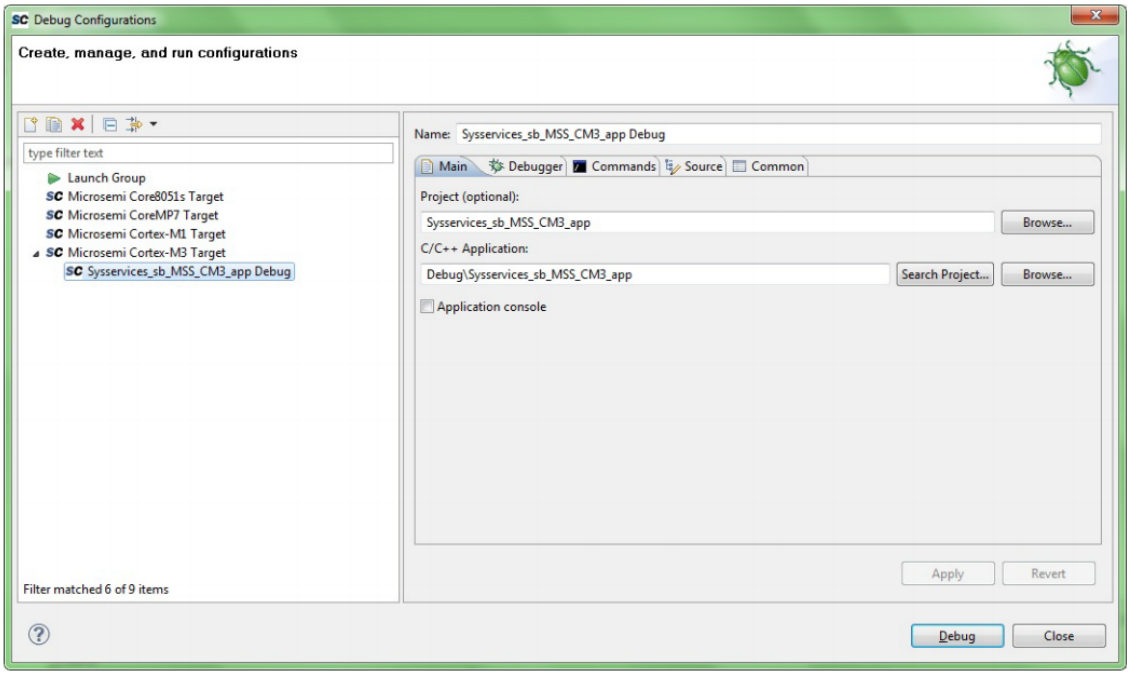


Для связи с ПК понадобится программа PuTTY либо любая другая программа-эмулятор теримнала. Для связи нужно ввести следующие настройки сессии:

- 115,200 baud rate
- 8 data bits

- 1 stop bit
- No parity
- No flow control

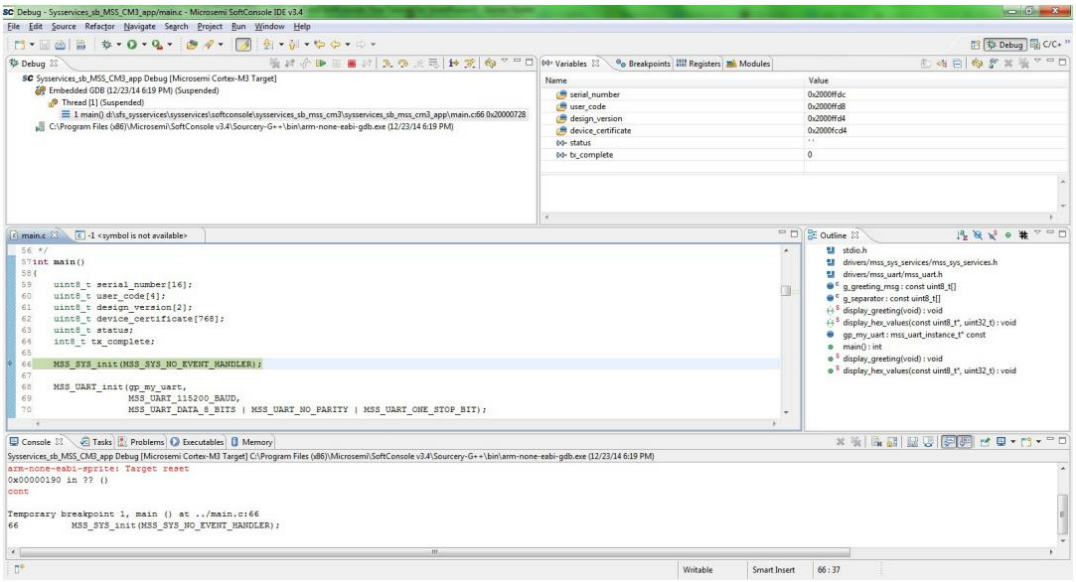
Вернемся в SoftConsole и проследуем в меню Run -> Debug Configurations. Откроется соответствующее диалоговое окно. Сделайте двойной щелчок по Microsemi Cortex-M3 Target.



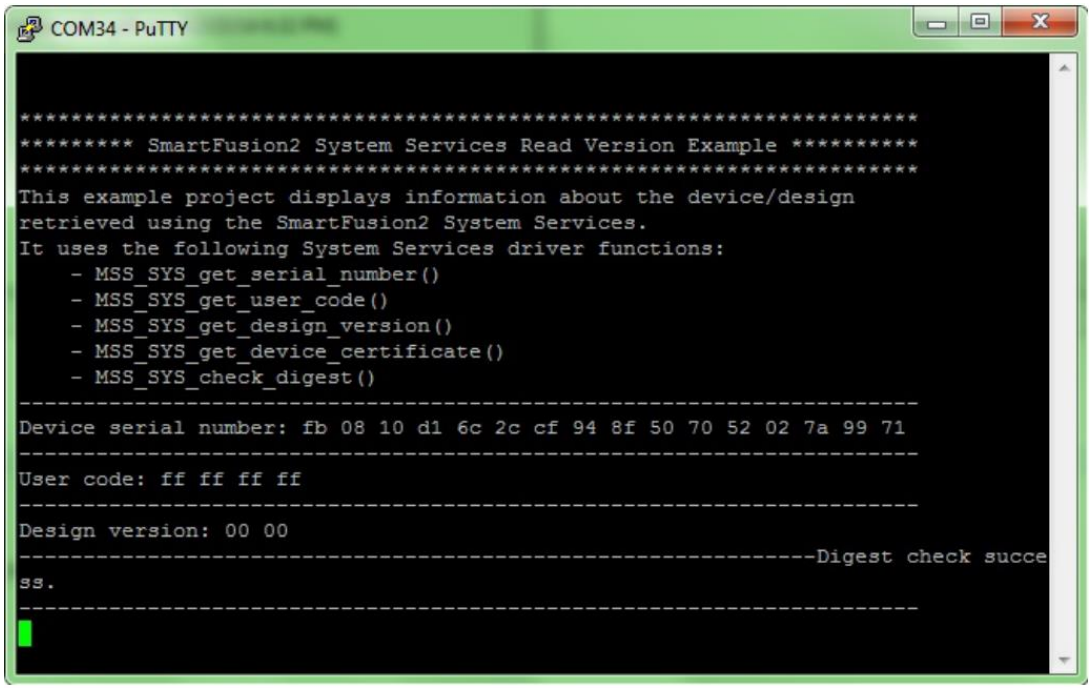
Удостоверьтесь, что настройки соответствуют следующим:

- Name: Sysservices_sb_MSS_ CM3 Debug
- Project: Sysservices_sb_MSS_ CM3
- C/C++ Application: Debug\Sysservices_MSS_MSS_CM3_app

Нажмите Debug и подтвердите переход в перспективу отладки. Программа загрузится во внутреннюю память микросхемы и остановится на первой контрольной точке.



Нажмите Run -> Resume для продолжения выполнения. При этом в окне PuTTY появится информация, предоставляемая системным контроллером по вызову с помощью функций в программе.



Разработчик может попробовать изменить демонстрационную программу и разработать собственную на её основе.

Список литературы

1. Libero Software Installation and Licensing Guide: http://www.microsemi.com/document-portal/doc_download/131602-libero-ide-and-software-installation-and-licensing-guide
2. Руководство по Libero SoC v11.4: http://www.microsemi.com/document-portal/doc_download/130850-libero-soc-v11-0-user
3. Implementing PCIe Control Plane Design in IGLOO2 FPGA - Libero SoC v11.5 Tutorial: http://www.microsemi.com/document-portal/doc_download/132748-implementing-pcie-control-plane-design-libero-soc-flow-tutorial-for-igloo2-fpga
4. IGLOO2 Flash*Freeze Entry and Exit - Libero SoC v11.5 Application Note: http://www.microsemi.com/document-portal/doc_download/133022-ac412-igloo2-flash-freeze-entry-and-exit
5. Creating a Libero Project for Firmware Catalog Sample Project Libero SoC v11.5 and SoftConsole Flow Tutorial for SmartFusion2: http://www.microsemi.com/document-portal/doc_download/133764-tu0487-creating-a-libero-project-for-firmware-catalog-sample-project-libero-soc-v11-5-and-softconsole-flow-tutorial-for-smartfusion2