

SF2-Junior-KIT

Использование интерфейса I2C

в проектах пользователя

Введение

Шина I2C представляет собой двухпроводный интерфейс для передачи данными между несколькими устройствами, подключенными к шине. Микроконтроллерная подсистема SmartFusion2 имеет два идентичных аппаратно реализованных блока интерфейса I2C. Управление блоками I2C осуществляется процессором ARM Cortex-M3 микроконтроллерной подсистемы по шине Advanced peripheral bus (APB). Если в проекте пользователя необходимы более двух интерфейсов I2C, дополнительные контроллеры интерфейса могут быть реализованы в матрице ПЛИС СнК SmartFusion2. Для реализации в ПЛИС контроллеров I2C Microsemi предлагает IP-ядро coreI2C, присутствующее в каталоге среди разработки проектов СнК Libero SoC. Данное IP-ядро доступно обладателям, в том числе, бесплатной лицензии Libero SoC.

Как аппаратные блоки интерфейса I2C микроконтроллерной подсистемы, так и блоки I2C реализованные с помощью IP-ядра coreI2C могут быть сконфигурированы для работы, как в качестве мастера шины, так и в качестве ведомого устройства.

В данном руководстве описан пример создания системы включающей четыре контроллера интерфейса I2C, два из которых являются ведущими, а два соответственно ведомыми устройствами. При этом два контроллера (один ведущий и один ведомый) реализованы с помощью аппаратных блоков интерфейса I2C микроконтроллерной подсистемы, а два оставшихся – путем имплементации ядра coreI2C в матрице ПЛИС.

Проект, разработка которого описана в данном руководстве, аналогичен опорному проекту Microsemi [AC430: SmartFusion2 I2C Reference Design using Multiple Masters and Multiple Slaves - Libero SoC v11.7](#).

Необходимое программное обеспечение

Для изучения материала, изложенного в данном руководстве необходимо следующее программное обеспечение:

- среда разработки [Microsemi Libero SoC v11.8](#);
- утилита для программирования микросхем СнК и ПЛИС FlashPro v11.8 или более поздняя версия, которая может быть установлена как часть пакета программ Microsemi Libero SoC и может быть запущена внутри Libero SoC или отдельно;
- среда разработки встраиваемого программного обеспечения SoftConsole v3.4 или более поздняя, которая может быть установлена как часть пакета программ Microsemi Libero SoC или отдельно;
- программа-оболочка M2S_I2C.exe поставляемая в [архиве](#) с файлами проекта AC430: SmartFusion2 I2C Reference Design using Multiple Masters and Multiple Slaves - Libero SoC v11.7;
- драйверы шины USB, которые можно установить, пройдя по ссылке:
http://www.microsemi.com/document-portal/doc_download/131593-usb-uart-driver-files

Необходимое аппаратное обеспечение

Вам понадобится отладочный набор [SF2-Junior-KIT](#), включающий следующие компоненты:

- 1) Модуль SF2-Junior-KIT;
- 2) Жидкокристаллический дисплей 320x240 с интерфейсом SPI и сенсорной панелью (touchscreen);
- 3) Программатор FlashPro4;
- 4) USB – Bluetooth донгл;
- 5) Модуль приемопередатчика Bluetooth – UART;
- 6) Преобразователь напряжения AC-DC 9В 1А;
- 7) Кабель USB 2.0 A-male to mini-B.

Описание проекта

В проекте СнК, разработка которого описана в данном руководстве, задействованы четыре контроллера интерфейса I²C, два из которых являются ведущими, а два соответственно ведомыми устройствами. При этом два контроллера (один ведущий и один ведомый) реализованы с помощью аппаратных блоков интерфейса I²C микроконтроллерной подсистемы, а два оставшихся – путем имплементации ядра coreI²C в матрице ПЛИС (рис. 1). Пользователь имеет возможность читать и записывать содержимое буферов I²C интерфейсов используя персональный компьютер и программу-оболочку M2S_I²C.exe (рис. 2).

Сигналы SDA и SCL выведены на разъем X9 отладочного набора, таким образом, пользователь может подключить к отладочному комплекту дополнительные внешние устройства имеющие интерфейс I²C.

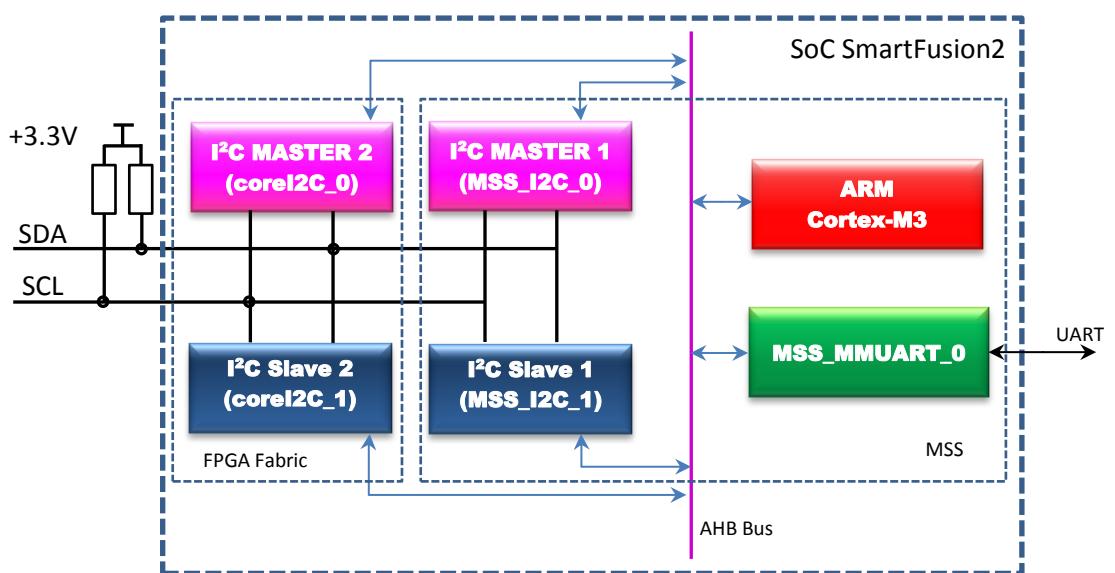


Рис. 1.



Рис. 2.

Разработка проекта системы на кристалле

Запустите приложение Libero SoC 11.8, дважды кликнув на ярлычок на рабочем столе или на аналогичный в меню Пуск > Все программы > Microsemi > Libero SoC v11.8.

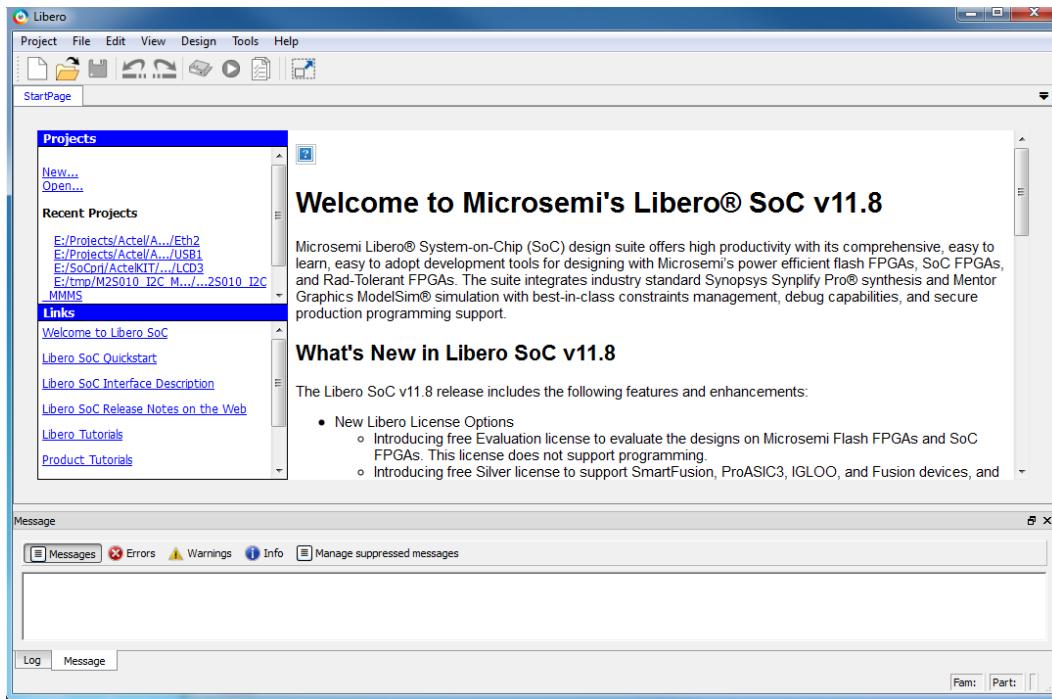


Рис. 3.

В главном меню Libero SoC (рис. 3) выполните команду **Project/New Project**, запустится мастер создания нового проекта. В появившемся окне укажите название проекта, например M2S010_MMMS, место расположения нашего проекта на диске и предпочтаемый язык проектирования – Verilog или VHDL. Опцию Enable block creation в рамках данного примера устанавливать не нужно. Нажмите кнопку «Next» (рис. 4).

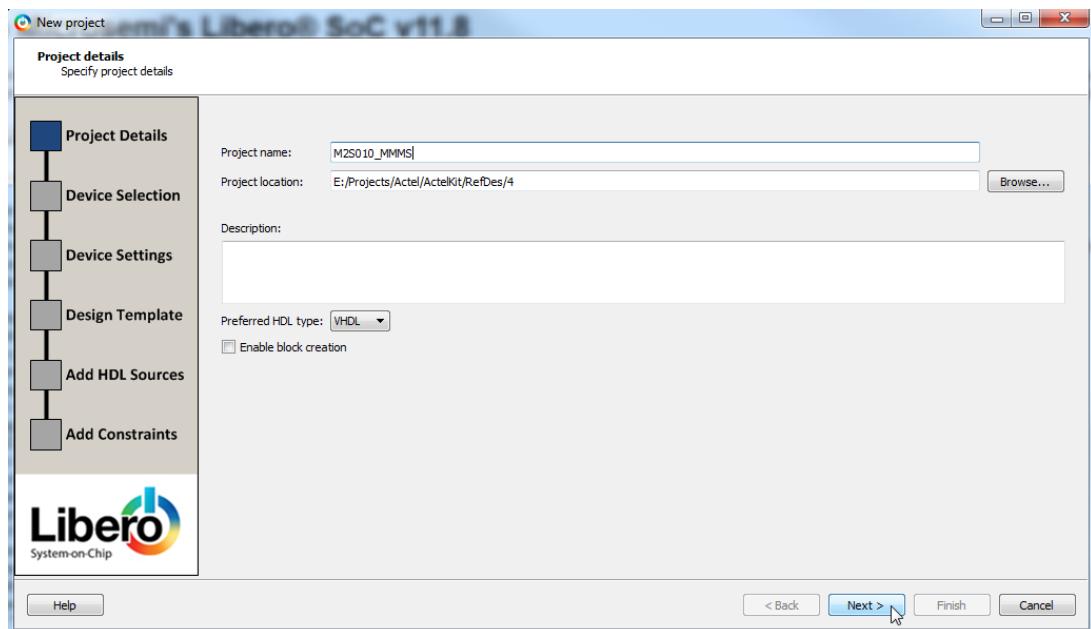


Рис. 4

В появившемся окне «Device selection» выбором желаемых параметров в выпадающих списках укажите PartNumber микросхемы, с которой будем работать. При работе с отладочным комплектом SF2-Junior-KIT необходимо выбрать вариант M2S010-TQ144, после чего нажмите «Next».

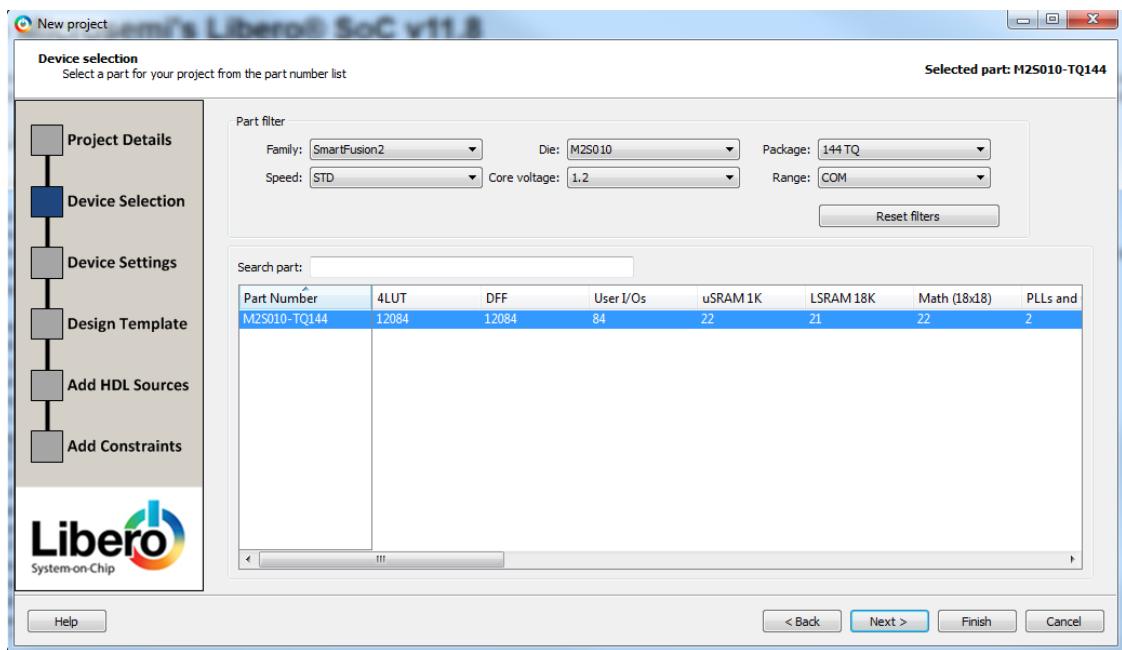


Рис. 5.

В следующем окне выберите настройки стандарта ввода-вывода по умолчанию LVCMOS 2.5V, напряжение питания PLL 2.5 V и задержку старта микросхемы после сигнала Reset 100 ms. Нажмите кнопку «Next» (рис. 6).

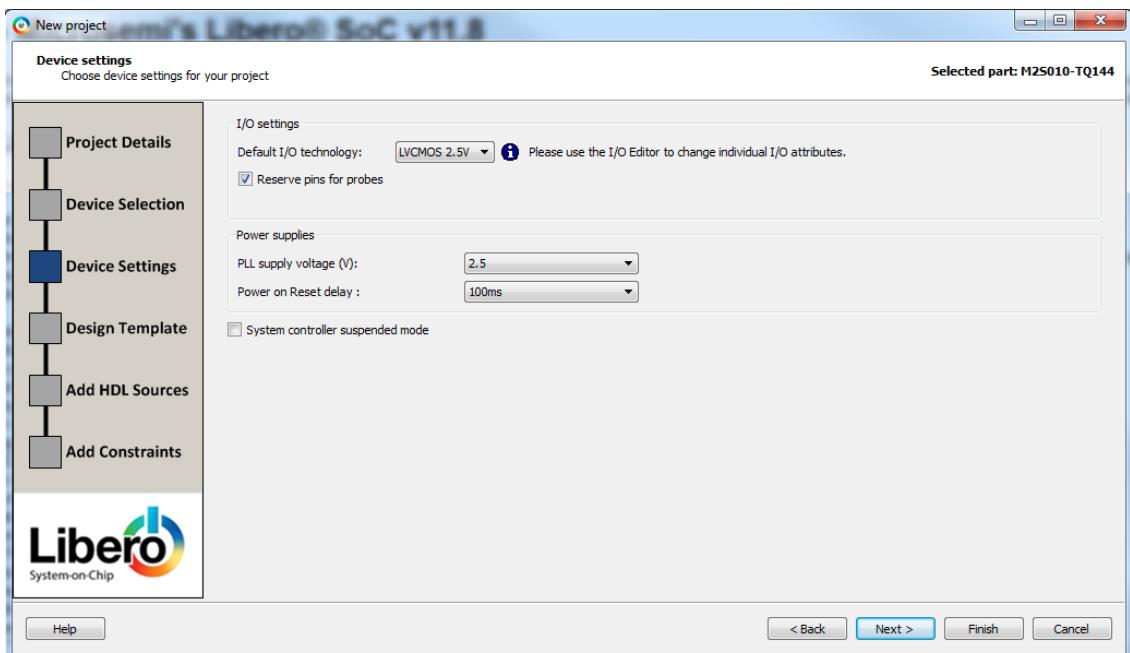


Рис. 6.

В следующем появившемся окне предлагается выбрать мастер, который будет использоваться для настройки микроконтроллерной подсистемы. Необходимо выбрать пункт «Create a microcontroller (MSS) based design», после чего нажать «Next» (рис. 7).

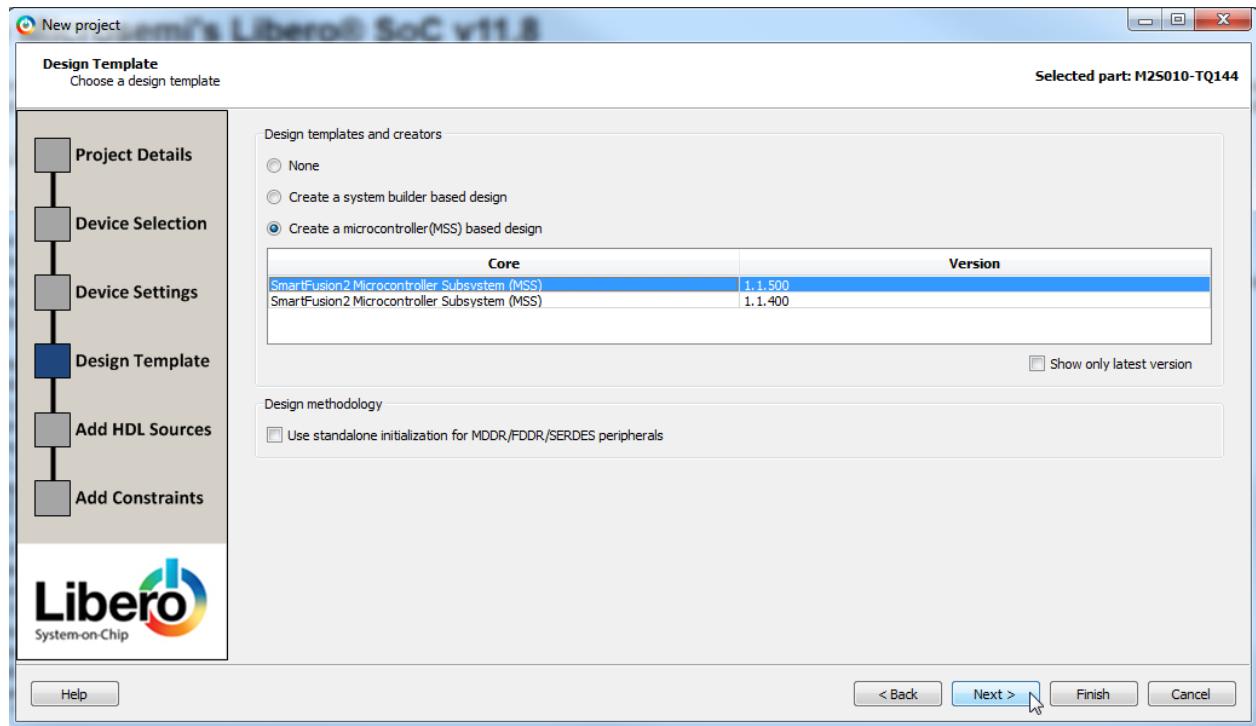


Рис. 7.

В следующих появляющихся окнах ничего не меняем, просто нажимаем «Next» до появления окна изображенного на рис. 8. При выборе способа установки проектных ограничений нажмите кнопку «Use Enhanced Constraint Flow».

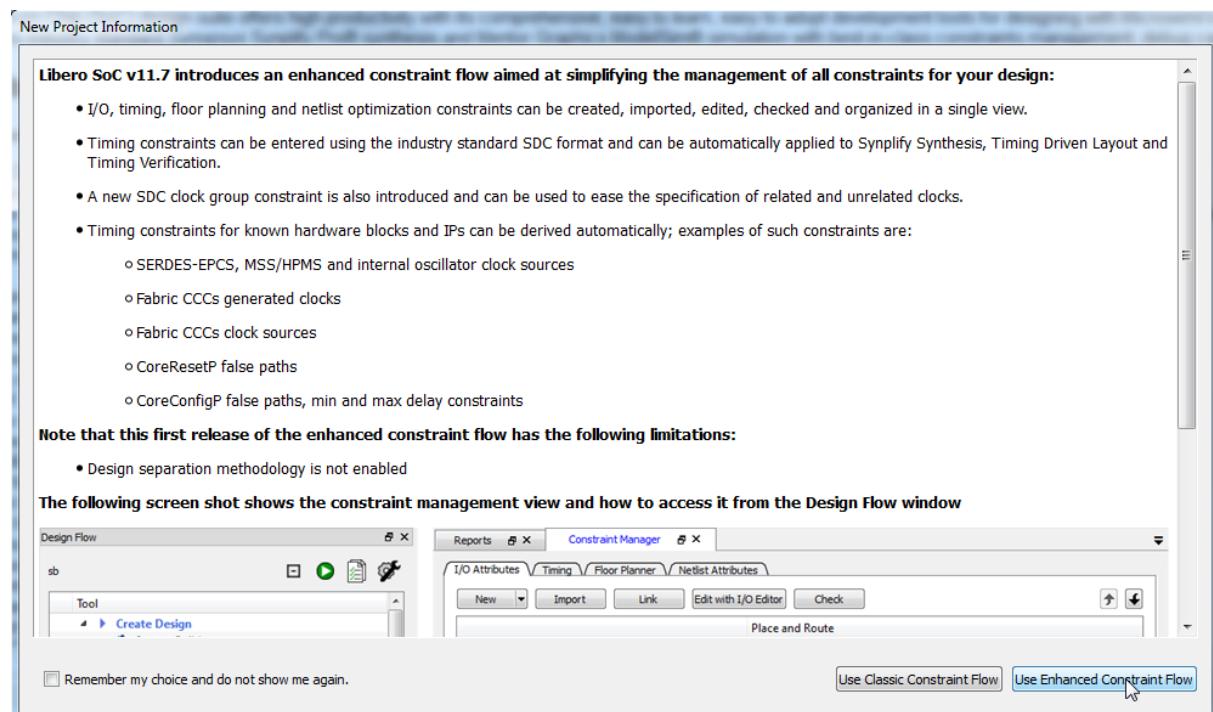


Рис. 8.

В результате выполнения описанных действий появится окно утилиты SmartDesign, в котором будет находиться один компонент M2S010_MMMS_MSS_0 – микроконтроллерная подсистема СнК SmartFusion2 (рис. 9).

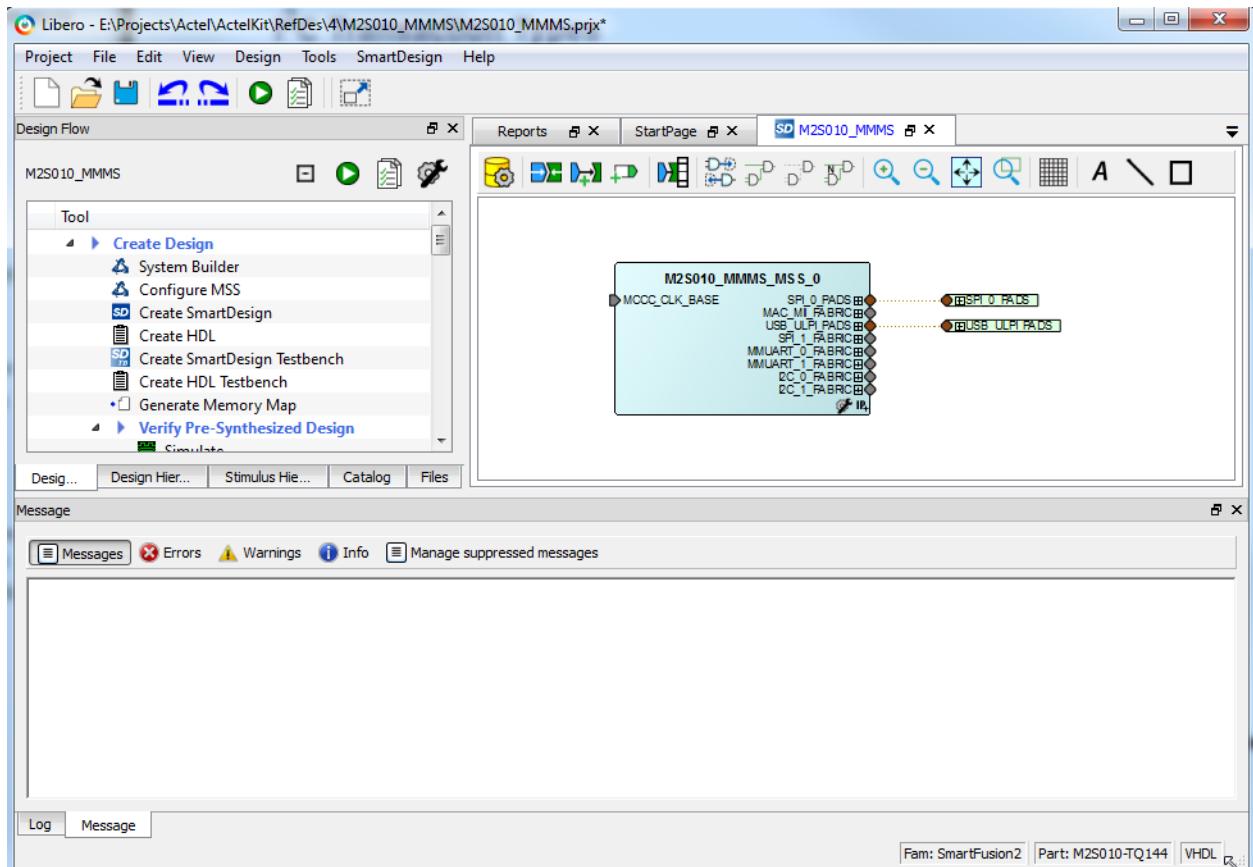


Рис. 9.

Созданный мастером компонент M2S010_MMMS_MSS_0 отражает состояние настроек микроконтроллерной подсистемы «по умолчанию». Нам необходимо изменить эти настройки в соответствии с задачами, решаемыми нашим приложением.

Наше приложение будет отправлять и получать сообщения от персонального компьютера используя аппаратный интерфейс UART микроконтроллерной подсистемы, и передавать и принимать данные по шине I2C используя два аппаратных блока интерфейса I2C и два блока реализованных с помощью ядра coreI2C в матрице ПЛИС.

Настроим соответствующие блоки архитектуры MSS, для этого дважды щелкнем на компоненте M2S010_MMMS_MSS_0. Откроется окно M2S010_MMMS_MSS_0 настроек микроконтроллерной подсистемы. В нем мы видим все доступные компоненты MSS (рис. 10).

В рамках функционала описанного проекта понадобится один контроллер интерфейса UART, например **MMUART_0**, и оба контроллера интерфейса I2C. Кроме того IP-ядра **coreI2C** будут подключаться к микроконтроллерной подсистеме через интерфейс Fabric Interface controller **FIC_0**. Все остальные компоненты MSS, а именно USB, Ethernet, MMUART_1, SPI_0, SPI_1, PDMA, CAN, WatchDog, GPIO, FIC_1 в разрабатываемом проекте задействованы не будут, их необходимо отключить, т. е. снять галочку в правом нижнем углу перечисленных компонентов (рис. 11).

Теперь настроим блоки микроконтроллерной подсистемы. Начнем с контроллера сброса. Для этого дважды щелкнем на блоке RESET Controller. Выберем опции настройки, представленные на рис. 12.

Настроим контроллер прерываний в соответствии с рис. 13 – внешние источники прерываний получат доступ к контроллеру прерываний микроконтроллерной подсистемы.

Настроим контроллер универсального приемопередатчика MMUART_0. В появившемся окне выберем следующие опции (рис. 14):

Duplex Mode: Full Duplex.

Async/Sync Mode: Asynchronous.

Use Modem Interface: снять галочку

RHD: Fabric

TXD: Fabric

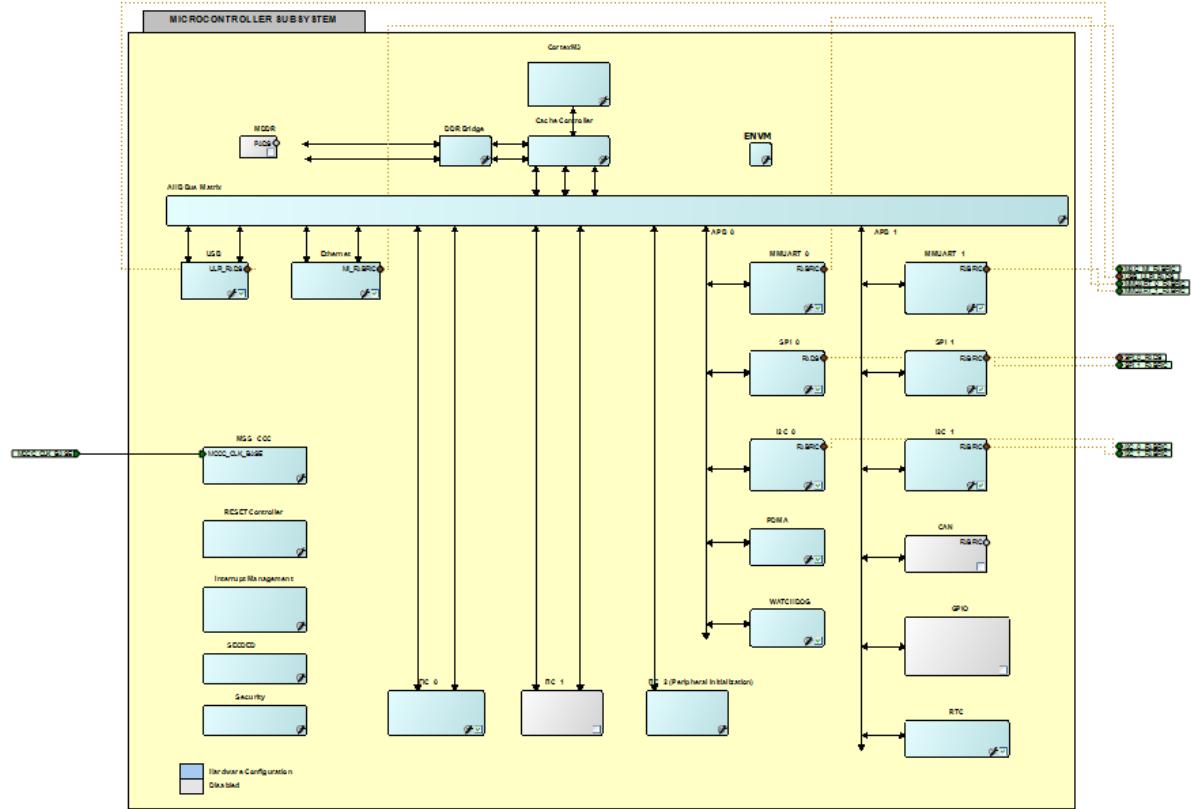


Рис. 10.

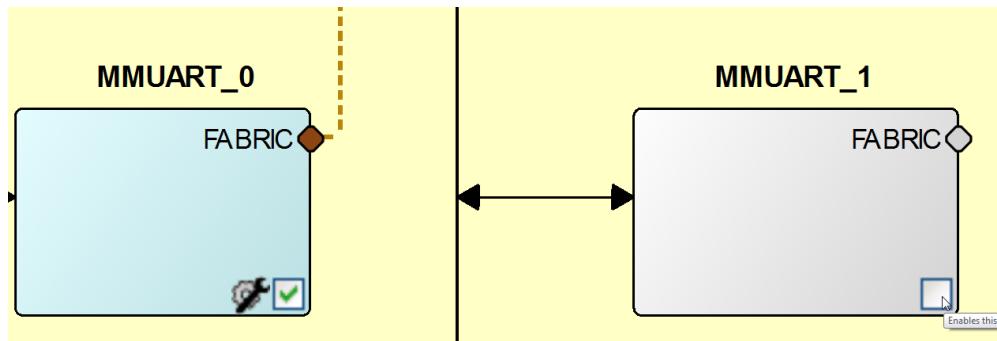


Рис. 11.

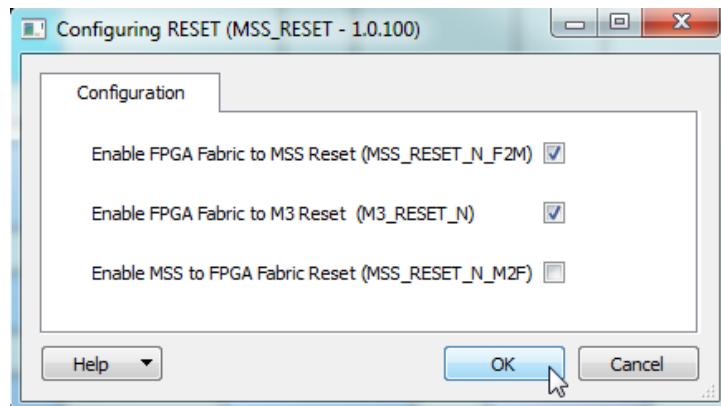


Рис. 12.

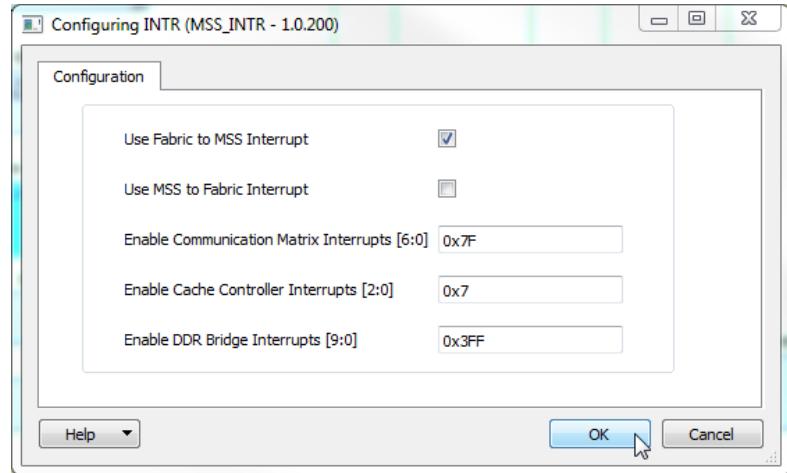


Рис. 13.

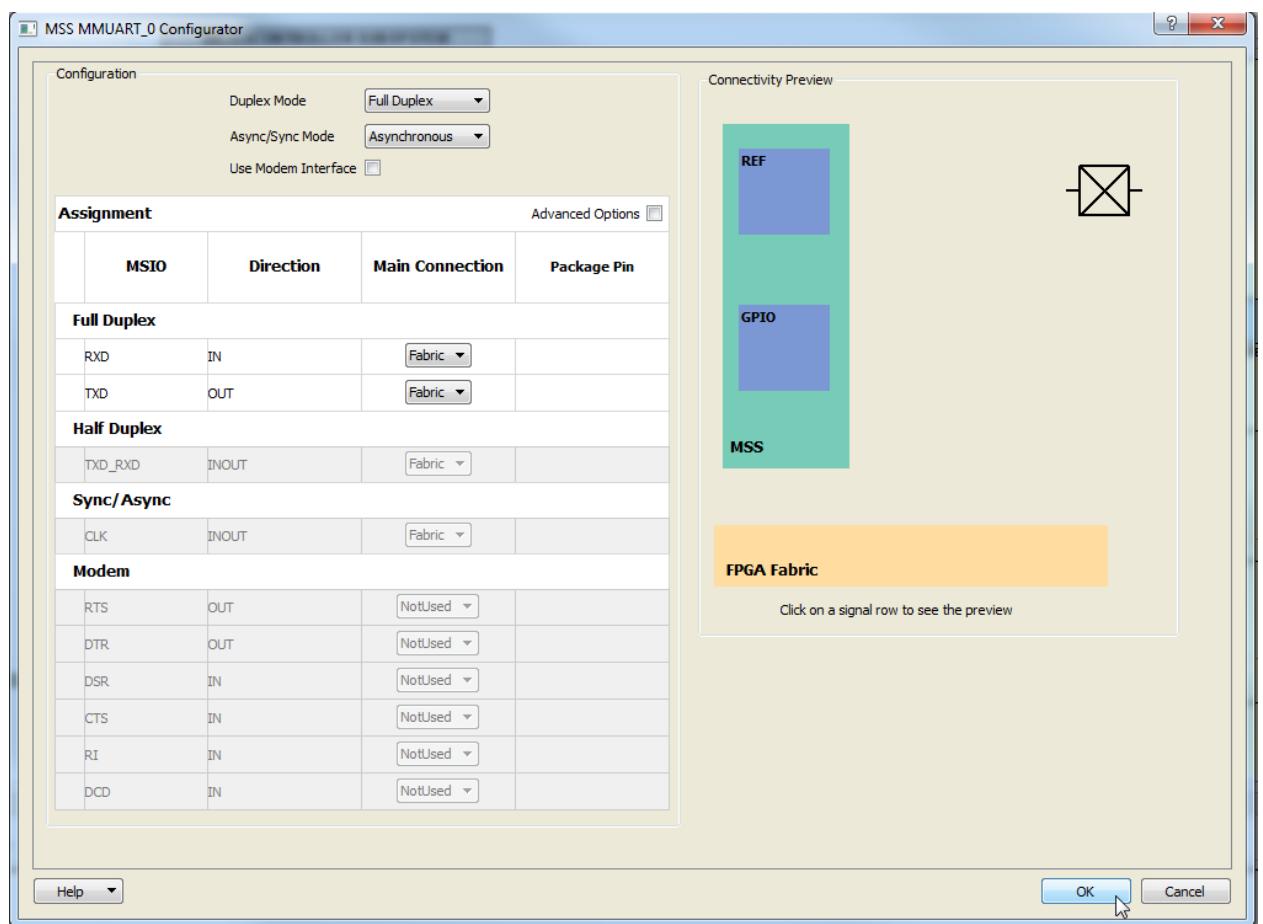


Рис. 14.

Настроим контроллер FIC_0, необходимый для связи по шине APB3 с IP-ядрами coreI2C, имплементируемыми в матрице ПЛИС. Необходимые параметры настройки FIC_0 представлены на рис. 15.

Настройки аппаратных контроллеров шины I2C остаются без изменений (рис. 16).

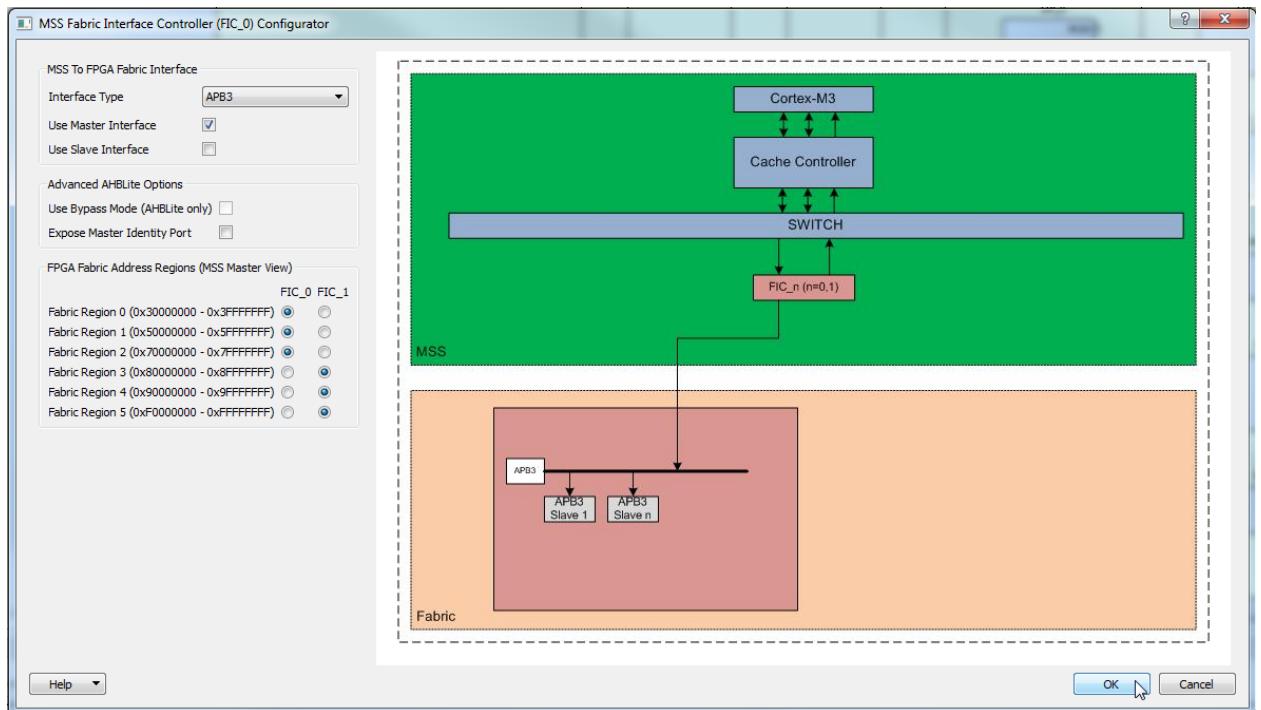


Рис. 15.

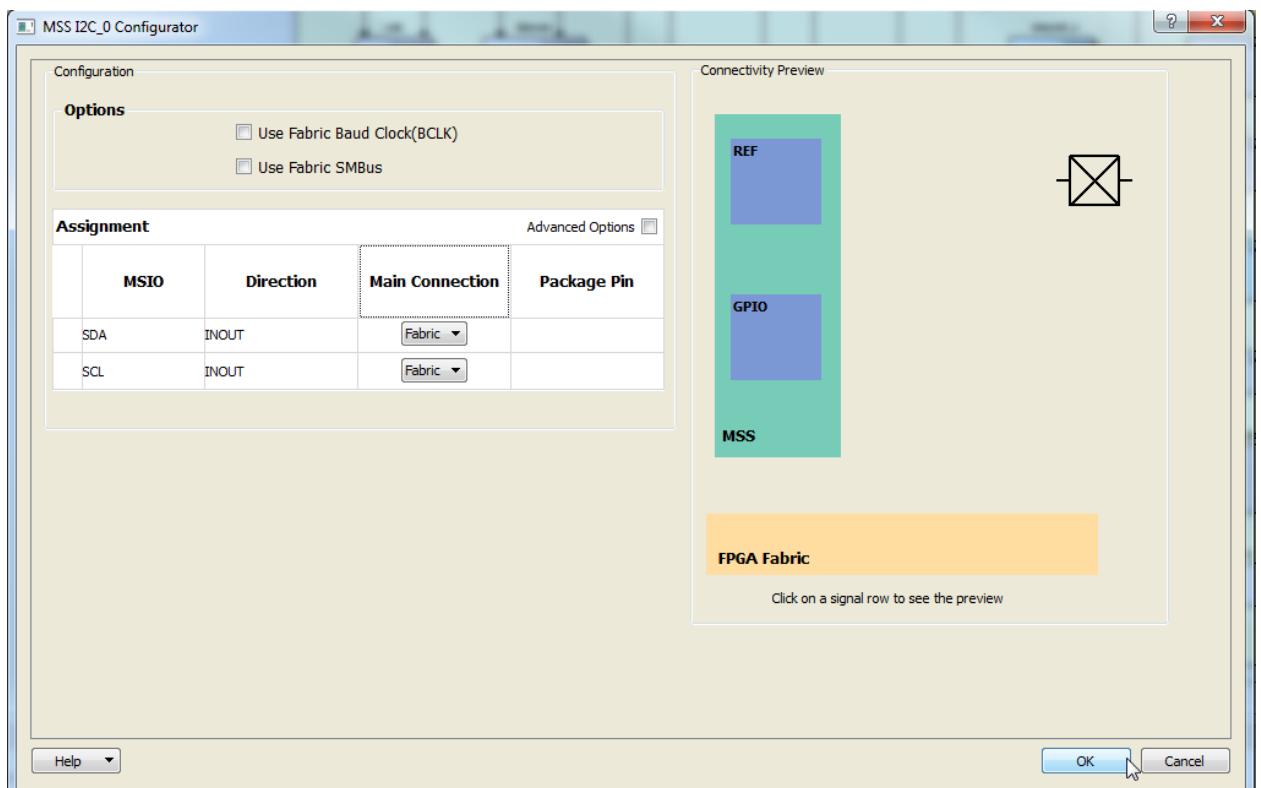


Рис. 16.

В результате окно настроек микроконтроллерной подсистемы примет вид, представленный на рис. 17.

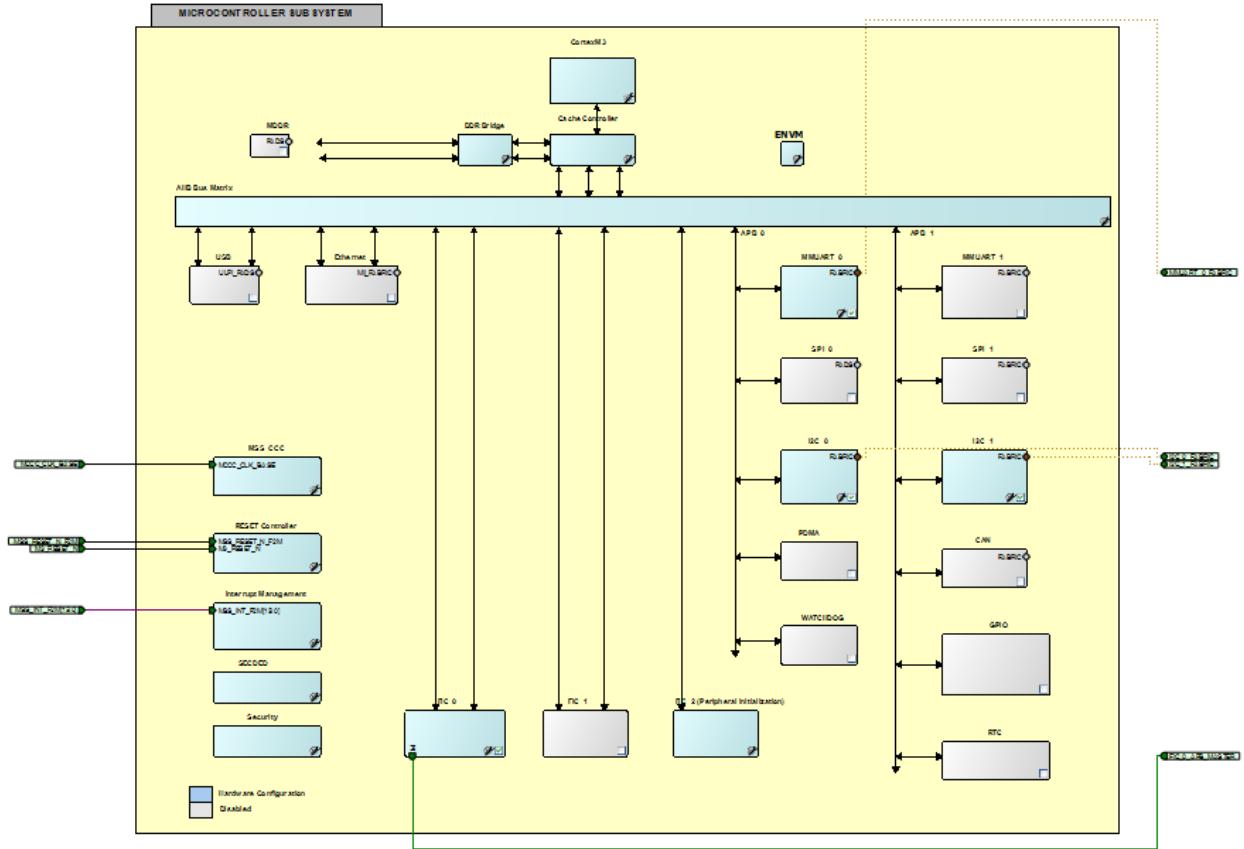


Рис. 17.

Сохраним изменения и вернемся во вкладку M2S010_MMMS редактора SmartDesign (рис. 18). Внешний вид компонента микроконтроллерной подсистемы изменился – в правом верхнем углу появился восклицательный знак на желтом фоне.

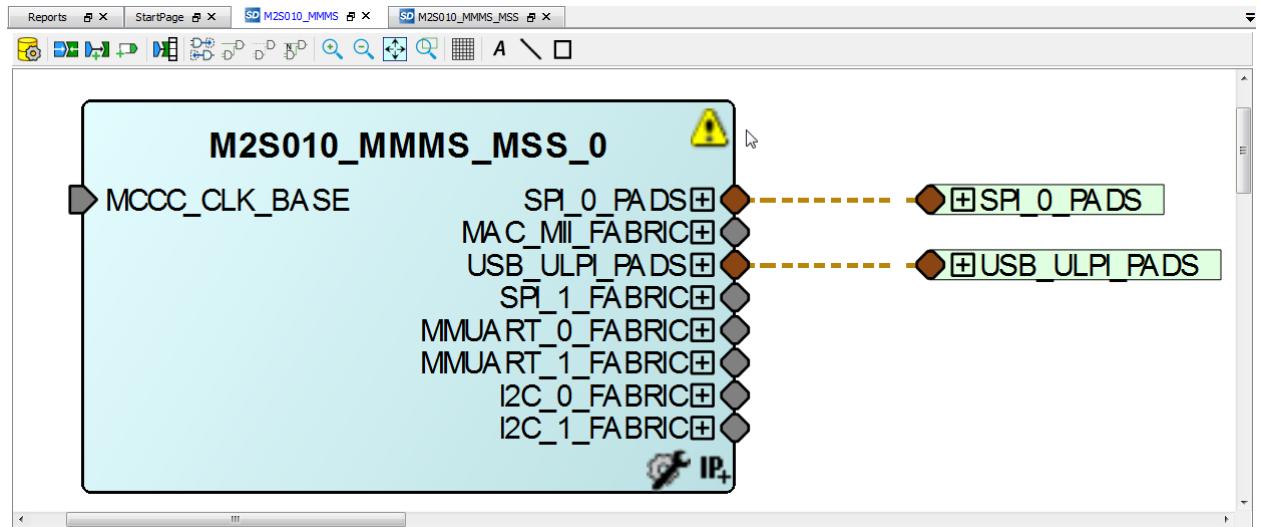


Рис. 18.

Это означает, что свойства компонента изменились и его необходимо обновить. Для обновления необходимо щелкнуть по нему правой кнопкой мыши и в появившемся меню выбрать команду **Update Instance(s) with Latest Component...** (рис. 19).

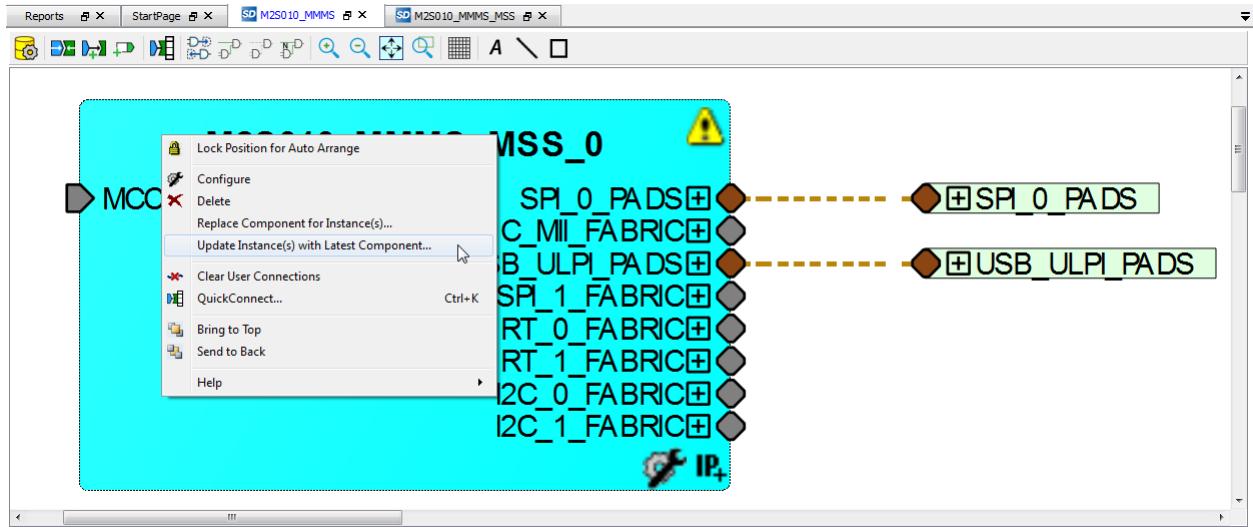


Рис. 19.

После обновления компонент должен принять вид, подобный представленному на рис. 20.

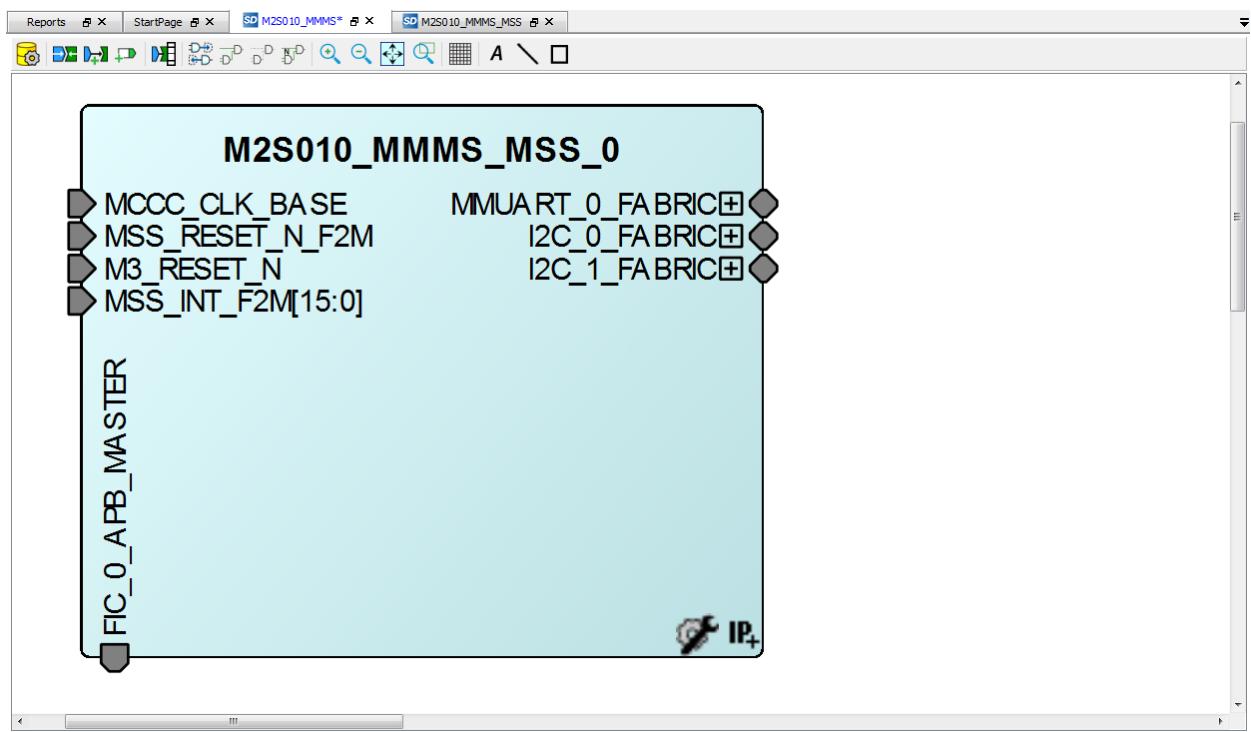


Рис. 20.

Теперь в дополнение к компоненту микроконтроллерной подсистемы из стандартного каталога Libero SoC на рабочее поле проекта нужно добавить IP-ядра и компоненты, отвечающие за тактирование, системный сброс и обмен по шине I2C.

Для реализации описанной выше функциональности проекта необходимы ядра и компоненты стандартной библиотеки компонент Libero SoC, указанные в таблице 1.

Таблица 1.

Ядра и компоненты стандартного каталога Libero SoC, используемые в проекте.

№ п/п	Раздел стандартного каталога Libero SoC	Название ядра/компоненты в каталоге Libero SoC	Название в проекте	Количество в проекте	Назначение
1	Processors	SmartFusion2 Microcontroller Subsystem (MSS)	M2S010_MMMS_MSS_0	1	Конфигуратор микроконтроллерной подсистемы MSS SmartFusion2
2	Clock & Management	Chip Oscillators	OSC_0	1	Источник сигнала тактирования
3	Clock & Management	Clock Conditioning Circuitry (CCC)	FCCC_0	1	Формирование сетки тактовых частот
4	Macro Library	SYSRESET	SYSRESET_0	1	Формирование сигнала сброса
5	Bus Interfaces	CoreAPB3	CoreAPB3_0	1	Коммуникация IP-ядер FPGA с MSS SmartFusion2
6	Peripherals	CoreI2C	CoreI2C_0, CoreI2C_1	2	Ядро интерфейса I2C с подключением к MSS по шине APB3
7	Macro Library	BIBUF	BIBUF_0 ... BIBUF_7	8	Двунаправленный буфер ввода-вывода
8	Macro Library	AND3	AND3_0	1	Логический элемент AND на 3 входа

Для использования нужного компонента в проекте необходимо перейти во вкладку Catalog, раскрыть нужный раздел каталога и мышью перетащить компонент на рабочее поле проекта (рис. 21).

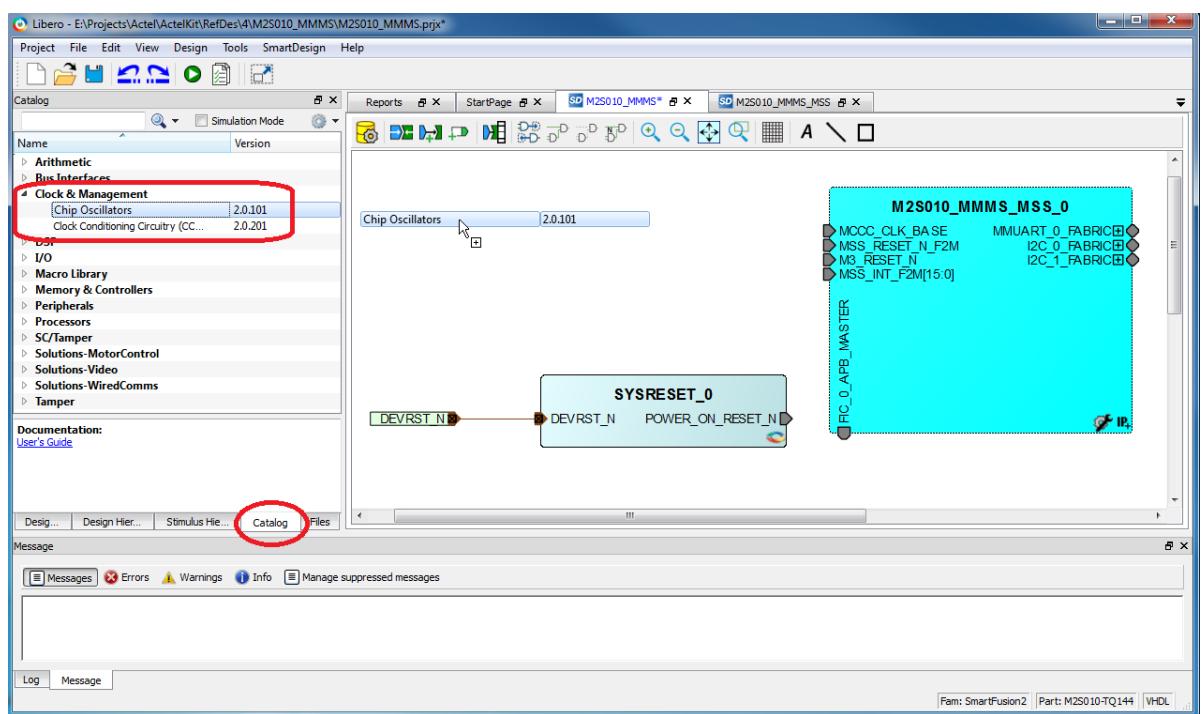


Рис. 21.

Параметры настройки источника сигнала тактирования приведены на рис. 22. Будем использовать генератор тактового сигнала, реализованный в СнК, частота которого определяется внешним кварцевым резонатором (20 МГц), установленным на плате отладочного набора.

Возможные альтернативы генератору на основе внешнего кварцевого резонатора в данном проекте это внутренние, то есть выполненные полностью на кристалле СнК SmartFusion2, RC-генераторы с частотой колебаний 50 МГц и 1МГц.

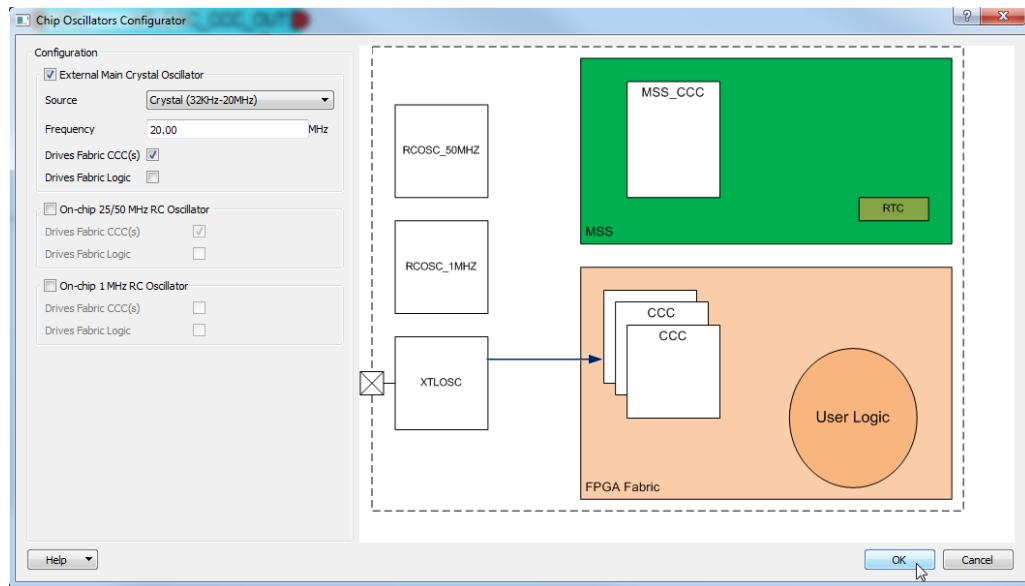


Рис. 22.

Сигнал тактирования 20 МГц с выхода генератора попадает на схему формирования тактового сигнала (Clock Condition Circuit, CCC) на основе блока фазовой автоподстройки частоты, ФАПЧ или PLL. С помощью CCC и PLL мы можем сформировать до 4 включительно высокостабильных сигналов опорных частот на основе единственного высокостабильного входного сигнала. Причем, частоты на выходе блока CCC могут сильно отличаться как от входной частоты, так и друг от друга. Таким образом, различные блоки проекта могут тактироваться существенно различными тактовыми частотами. В нашем проекте будем использовать один сигнал тактирования для всего устройства 25 МГц. Параметры настройки схемы формирования сигналов тактовых частот представлены на рис. 23.

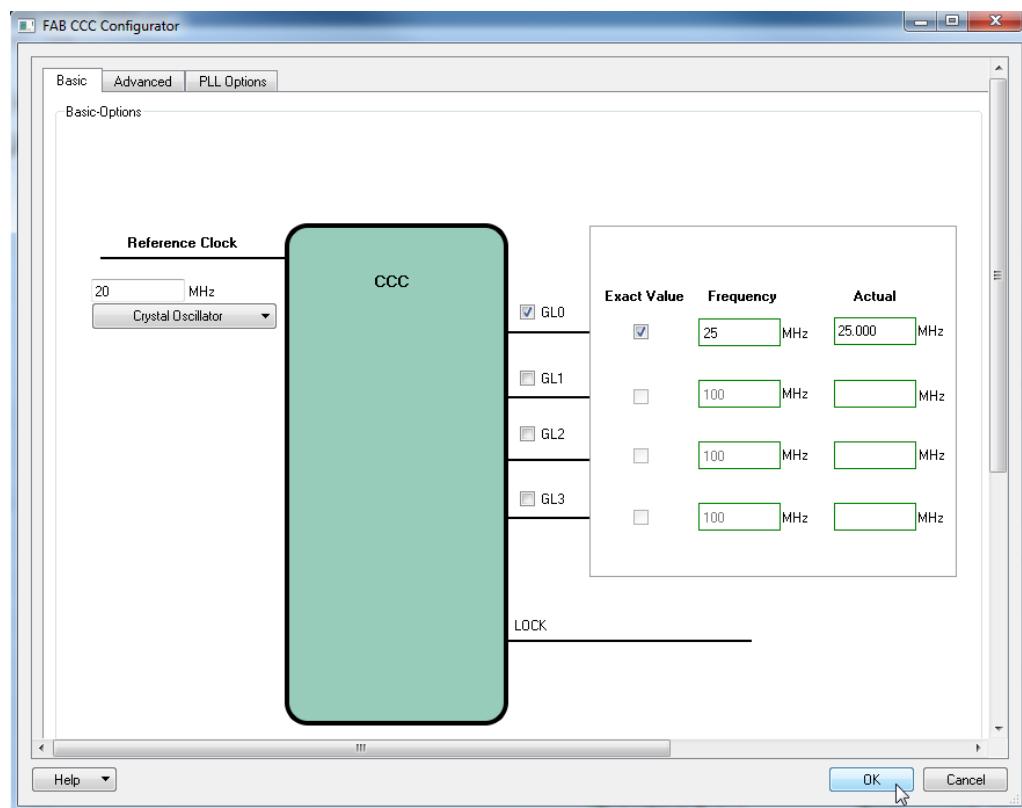


Рис. 23.

Сигнал тактирования с выхода CCC в нашем проекте подается на соответствующие входы IP-ядер и на вход предназначенный для сигнала тактирования микроконтроллерной подсистемы M2S010_MMMS_MSS_0. Внутри микроконтроллерной подсистемы имеется своя собственная схема формирования сигнала тактирования MSS_CCC. Параметры настройки MSS_CCC представлены на рис. 24.

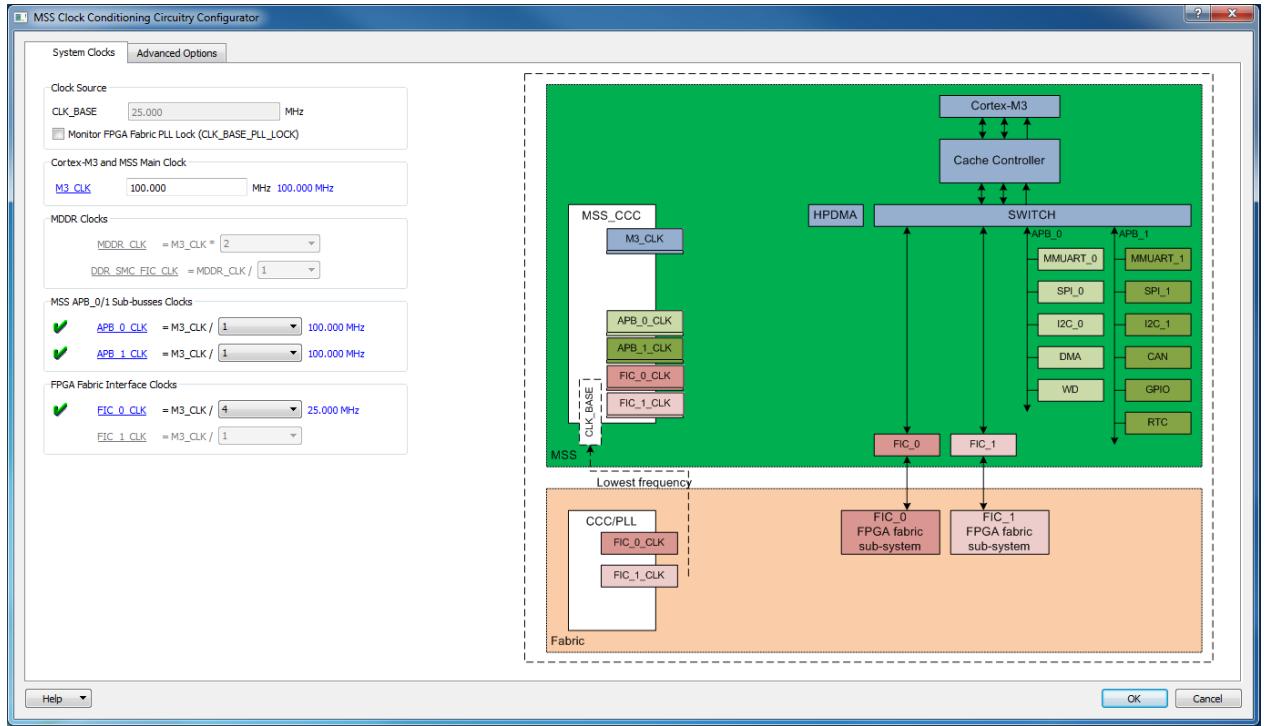


Рис. 24.

Параметры настройки ядра coreAPB3 представлены на рис. 25. Обмен будет происходить 32-битными словами между микроконтроллерной подсистемой и двумя ядрами coreI2C. Во избежание путаницы стоит уточнить, что по отношению к шине APB3 оба ядра coreI2C являются ведомыми, а мастером на шине APB3 является микроконтроллерная подсистема с процессором Cortex_M3. При этом, если смотреть на те же IP-ядра со стороны интерфейса I2C, то одно из ядер (coreI2C_0) на шине в нашем проекте будет ведущим интерфейсом I2C (I2C Master), а второе (coreI2C_1) ведомым (I2C slave).

Параметры настройки ядер coreI2C показаны на рис. 26.

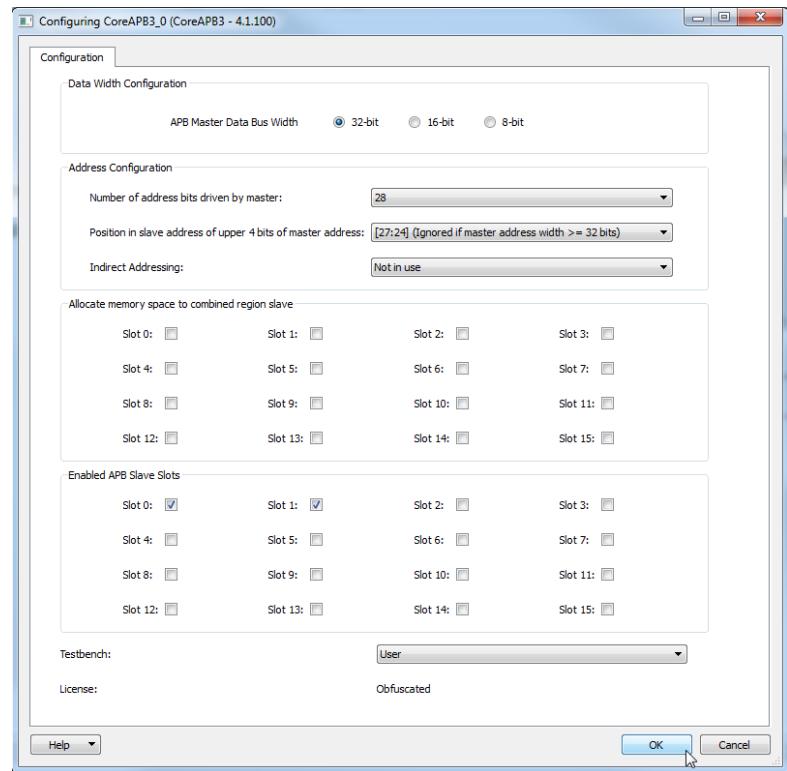


Рис. 25.

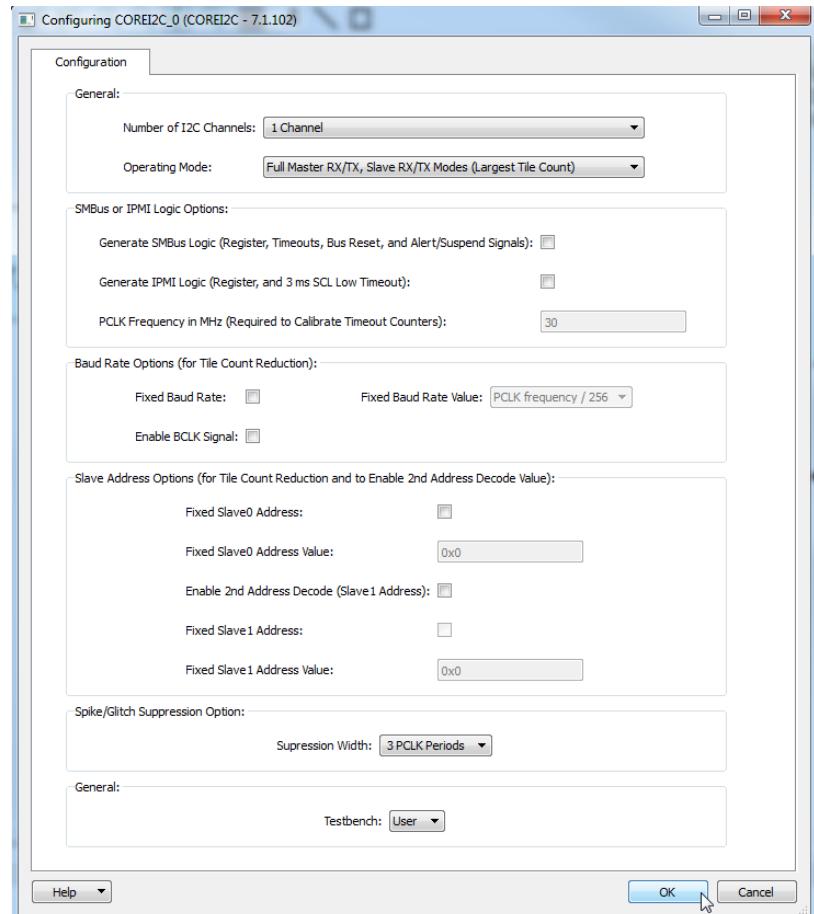


Рис. 26.

Теперь выполним соединения. Ниже приведен список соединений проекта в формате IP-ядро.Контакт – IP-ядро.Контакт:

1. OSC_0. OSC_0_XTLOSC_CCC_OUT – FCCC_0. XTLOSC_CCC_IN.
2. CoreAPB3_0.M – M2S010_MMMS_MSS_0.FIC_0_APB_MASTER.
3. CoreAPB3_0.S0 – COREI2C_0.APBSlave.
4. CoreAPB3_0.S1 – COREI2C_1.APBSlave.
5. FCCC_0.GL0 – M2S010_MMMS_MSS_0.MCCC_CLK_BASE.
6. FCCC_0.GL0 – COREI2C_0.PCLK.
7. FCCC_0.GL0 – COREI2C_1.PCLK.
8. FCCC_0.LOCK – AND3_0.A.
9. SYSRESET_0.POWER_ON_RESETN – AND3_0.C
10. AND3_0.Y – M2S010_MMMS_MSS_0.MSS_RESET_N_F2M.
11. AND3_0.Y – COREI2C_0.PRESETN.
12. AND3_0.Y – COREI2C_1.PRESETN.
13. M2S010_MMMS_MSS_0.MSS_INT_F2M[1] – COREI2C_1.INT[0].
14. M2S010_MMMS_MSS_0.MSS_INT_F2M[0] – COREI2C_0.INT[0].

Пункты, имеющие общий контакт образуют единую цепь прохождения сигнала, например пп. 5, 6, 7 и пп. 10, 11, 12 являются соответственно цепями прохождения сигнала тактирования и сброса.

Для выполнения пунктов 11 и 12 необходимо предварительно вычленить нужные сигналы из массива, т.е. щёлкнуть правой кнопкой мыши на контакте MSS_INT_F2M[15:0] и выполнить команду Edit Slice выпадающего меню, как показано на рис. 27.

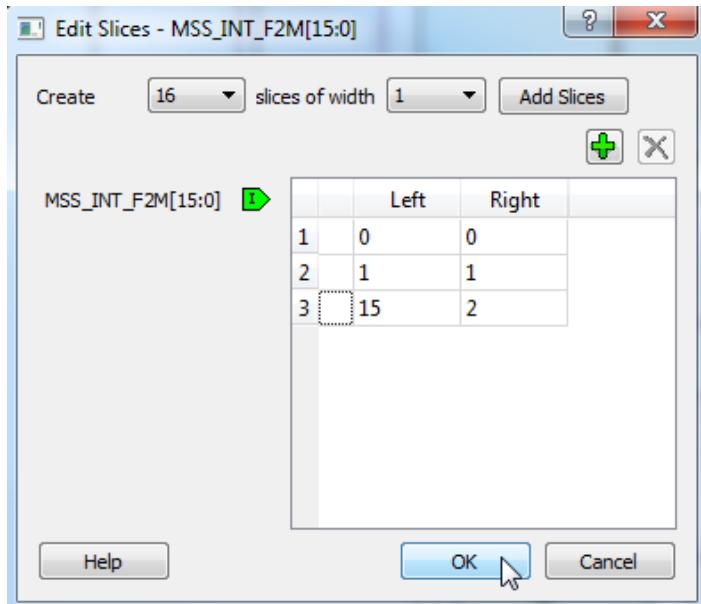


Рис. 27.

Контакты M2S010_MMMS_MSS_0.MSS_INT_F2M[15:2] подтягиваем к “вниз” – **Tie Low**. Контакты M2S010_MMMS_MSS_0.MMUART_0_FABRIC и AND3_0.B выводим на верхний уровень, т.е. нажимаем на контактах правую кнопку мыши и выбираем команду **Promote to Top Level**.

Промежуточный результат проектирования (без подключённых двунаправленных буферов BIBUF) представлен на рис. 28.

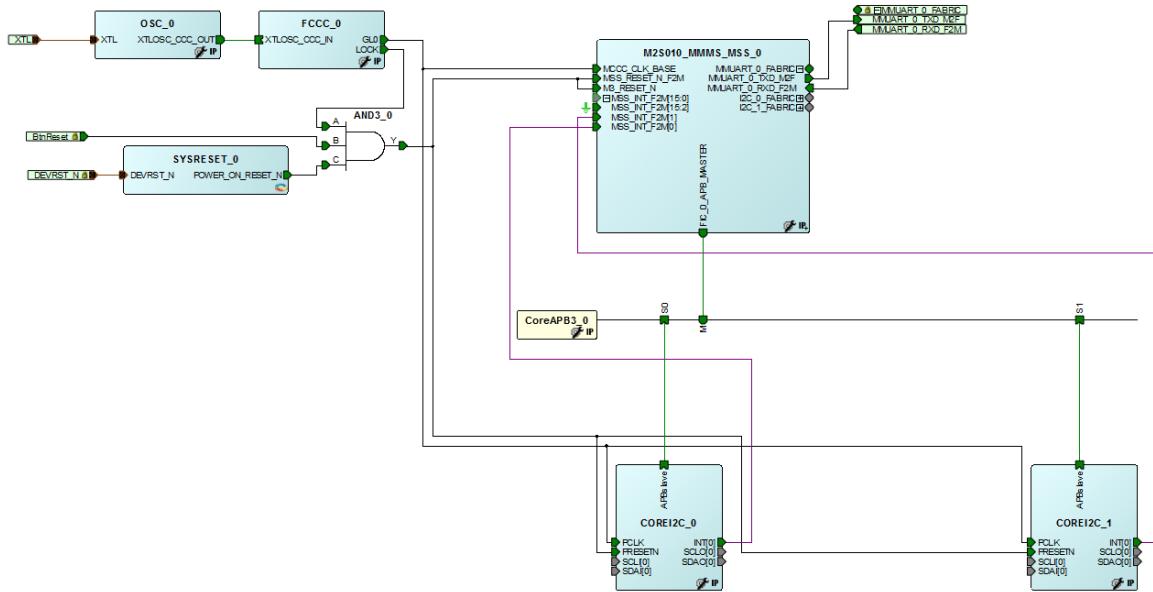


Рис. 28.

Шина I2C согласно спецификации является двунаправленной, т.е. контакт микросхемы, к которому подключена линия связи интерфейса (SDA или SCL), должен «уметь» поочередно работать и на прием и на передачу. Для реализации двунаправленного режима работы контактов ввода-вывода в стандартной библиотеке ядер Libero SoC доступен компонент BIBUF. Способ подключения данного ядра зависит от источника/приемника сигналов. Возможны следующие варианты подключения:

1. Два двунаправленных буфера BIBUF подключаются к ядру coreI2C. Способ подключения контактов IP-ядер представлен в таблице 2.

Таблица 2. Подключение coreI2C и BIBUF

№ п/п	Компонент.Сигнал	Компонент.Сигнал
1	coreI2C_0.SCLI[0]	BIBUF_0.Y
2	coreI2C_0.SDAI[0]	BIBUF_1.Y
3	coreI2C_0.SCLO[0]	(Not) BIBUF_0.E
4	coreI2C_0.SDAO[0]	(Not) BIBUF_1.E
5	coreI2C_1.SCLI[0]	BIBUF_3.Y
6	coreI2C_1.SDAI[0]	BIBUF_2.Y
7	coreI2C_1.SCLO[0]	(Not) BIBUF_3.E
8	coreI2C_1.SDAO[0]	(Not) BIBUF_2.E

Префикс (Not) перед входом разрешения двунаправленного буфера означает, что сигнал перед подачей на соответствующий контакт нужно проинвертировать, что можно выполнить, нажав правую кнопку мыши на контакте и выбрать «Invert» в появившемся меню. Входы данных BIBUF.D двунаправленного буфера необходимо подтянуть «вниз», выбрав «Tie Low» в выпадающем меню при щелчке правой кнопкой мыши на соответствующем контакте.

2. Два компонента BIBUF подключаются к аппаратно реализованному блоку интерфейса I2C микроконтроллерной подсистемы MSS SmartFusion2. В этом случае со стороны микроконтроллерной подсистемы пользователю становятся доступны 6 контактов для подключения. Способ их коммутации с контактами двунаправленных буферов представлен в таблице 3.

Таблица 3. Подключение MSS и BIBUF

№ п/п	Компонент.Сигнал	Компонент.Сигнал
1	M2S010_MMMS_MSS_O.I2C_0_SDA_F2M	BIBUF_4.Y
2	M2S010_MMMS_MSS_O.I2C_0_SDA_M2F	BIBUF_4.D
3	M2S010_MMMS_MSS_O.I2C_0_SDA_M2F_OE	BIBUF_4.E
4	M2S010_MMMS_MSS_O.I2C_0_SCL_F2M	BIBUF_5.Y
5	M2S010_MMMS_MSS_O.I2C_0_SCL_M2F	BIBUF_5.D
6	M2S010_MMMS_MSS_O.I2C_0_SCL_M2F_OE	BIBUF_5.E
7	M2S010_MMMS_MSS_O.I2C_1_SDA_F2M	BIBUF_6.Y
8	M2S010_MMMS_MSS_O.I2C_1_SDA_M2F	BIBUF_6.D
9	M2S010_MMMS_MSS_O.I2C_1_SDA_M2F_OE	BIBUF_6.E
10	M2S010_MMMS_MSS_O.I2C_1_SCL_F2M	BIBUF_7.Y
11	M2S010_MMMS_MSS_O.I2C_1_SCL_M2F	BIBUF_7.D
12	M2S010_MMMS_MSS_O.I2C_1_SCL_M2F_OE	BIBUF_7.E

3. В случае выбора варианта подключения аппаратного интерфейса I2C к выделенным для данного интерфейса контактам микросхемы, двунаправленные буферы выделенных контактов инициализируются автоматически, разработчику сразу становятся доступны двунаправленные контакты SDA, SCL компонента-конфигуратора микроконтроллерной подсистемы, которые остается вывести «наружу» выбрав «Promote to Top Level» при правом клике мыши на соответствующем контакте. Однако данный вариант доступен для корпусов с большим количеством контактов, наш TQ144 к таковым не относится.

Итак, подключаем четыре экземпляра BIBUF к ядрам COREI2C_0 и COREI2C_1 в соответствии с таблицей 2, четыре экземпляра BIBUF к компоненту микроконтроллерной подсистеме в соответствии с таблицей 3. Переименуем контакты «PAD» в соответствии с их функциональным назначением для улучшения читабельности схемы. Верхний уровень проекта в окончательном виде представлен на рис. 29.

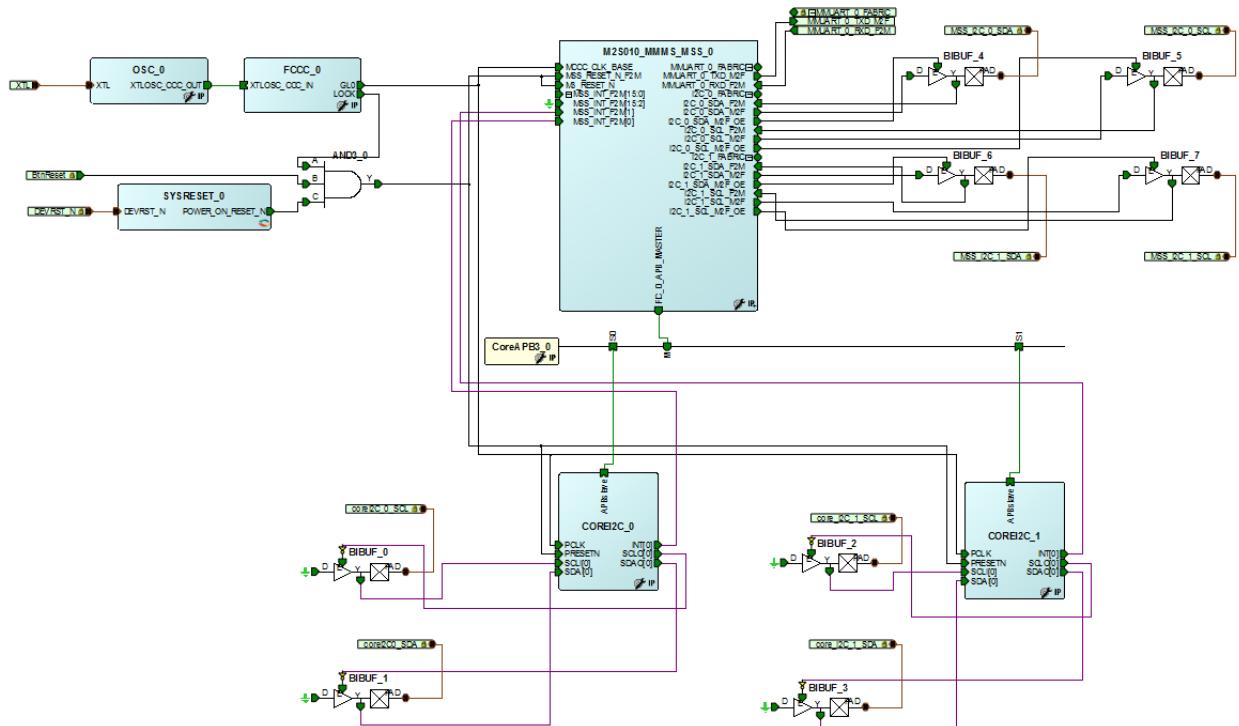


Рис. 29.

Сохраняем настройки и выполняем команду **Generate Component** (рис. 30).

После выполнения генерации компонента создадим карту памяти блоков архитекторы нашего проекта, для чего выполним команду **Generate Memory Map** вкладки **Design Flow**. Результат работы команды – таблицу адресов всех используемых блоков архитектуры проекта можно увидеть во вкладке Reports (рис. 31). Начальные адреса аппаратно реализованных блоков микроконтроллерной подсистемы фиксированы, в то время как начальные адреса IP-ядер (в нашем проекте это coreI2C_0 и coreI2C_1) могут отличаться от проекта к проекту и изменяться при доработке проекта СнК, т.е. добавлении/удалении из проекта IP-ядер, изменении их свойств. По этой причине важно контролировать совпадение значений начальных адресов указанных в коде ВПО со значениями, приведенными в отчете, появляющемся после выполнения команды **Generate Memory Map**. В отчете для IP-ядер coreI2C_0 и coreI2C_1 пользователю предлагаются по 3 диапазона адресов (рис. 31). В проекте можно использовать любой из предложенных диапазонов.

Выполним синтез проекта (рис. 32) и назначим контакты микросхемы входным и выходным сигналам нашего проекта для чего выполним команду **Manage Constraints** (рис. 33), в появившейся вкладке **Constraints Manager** выполним команду **Edit > Edit with I/O Editor** (рис. 34). В открывшемся окне утилиты I/O Editor назначаем контакты микросхемы M2S010-TQ144 входящим и выходящим сигналам нашего проекта (рис. 35). Сохраним изменения и закроем I/O Editor.

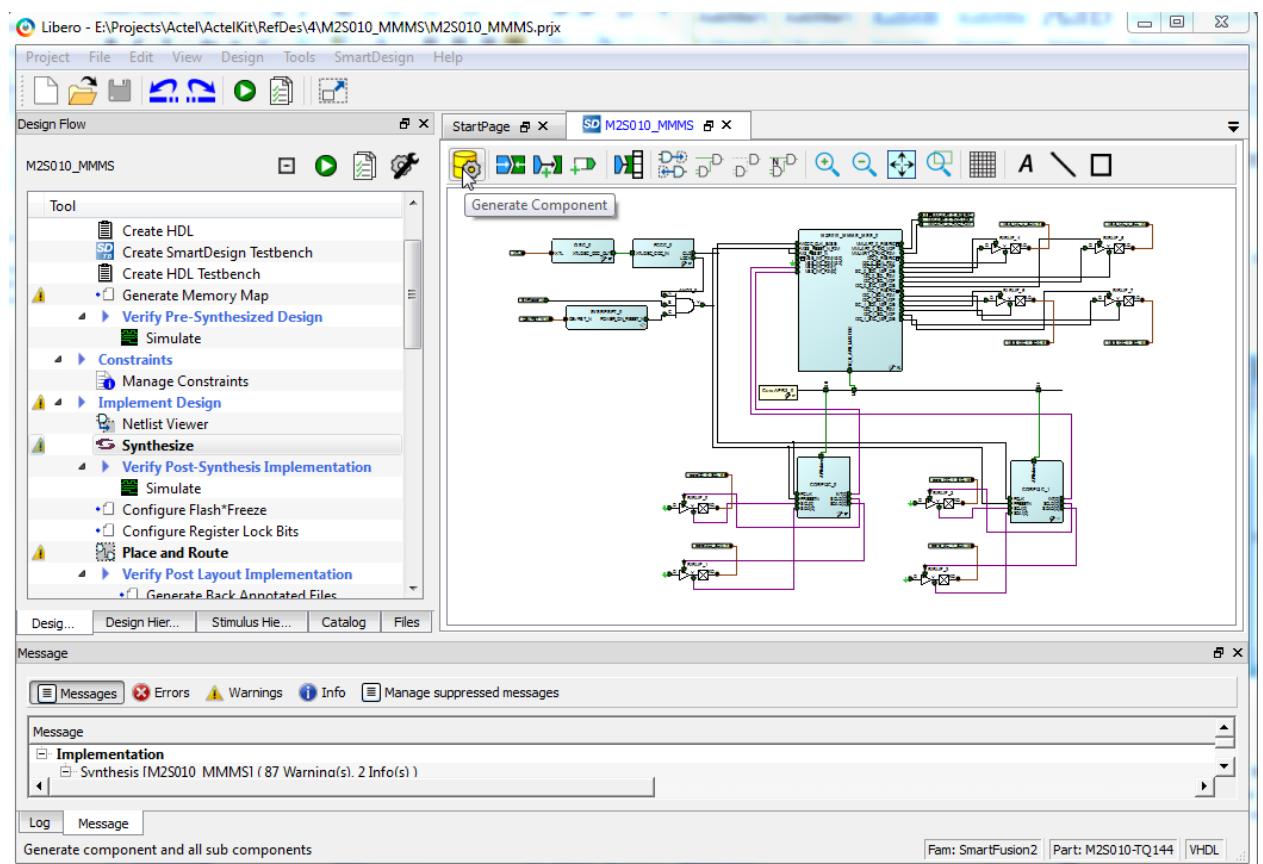


Рис. 30.

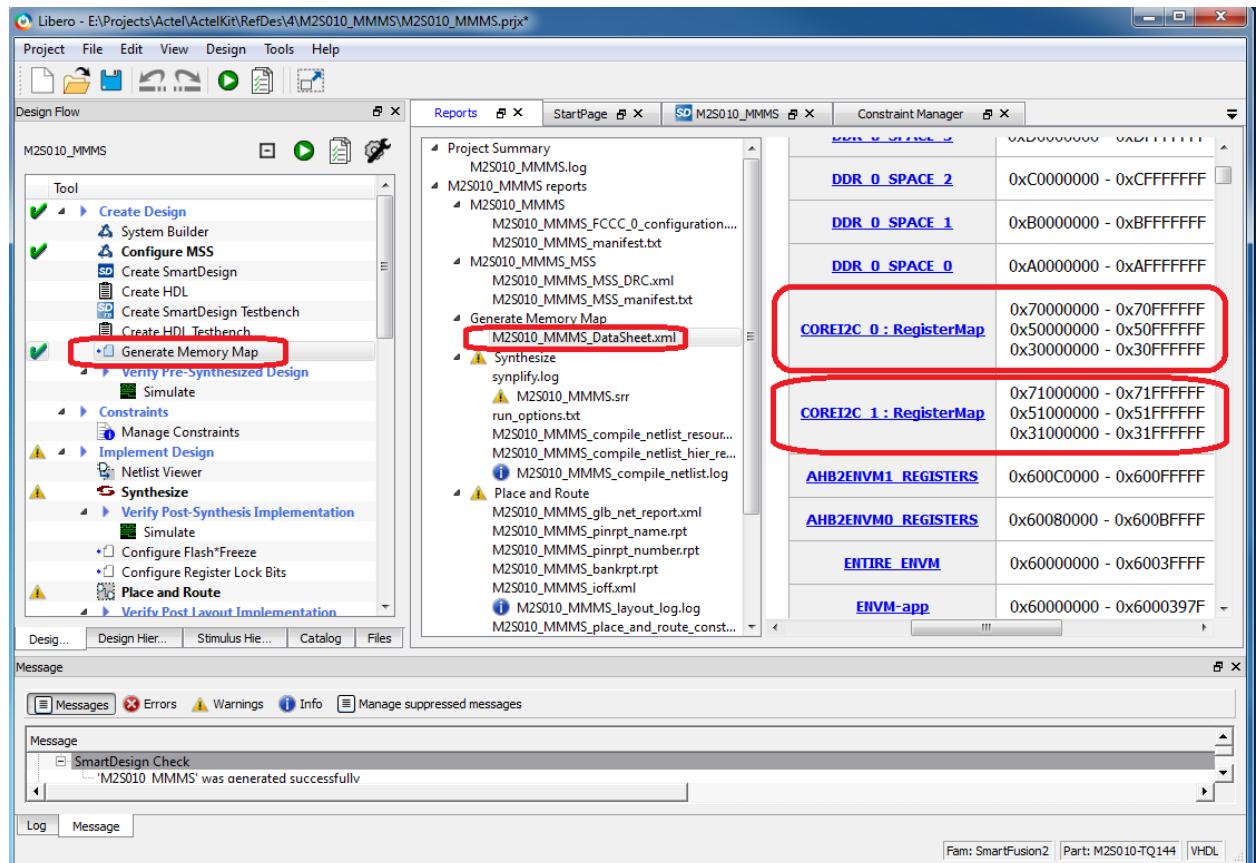


Рис. 31.

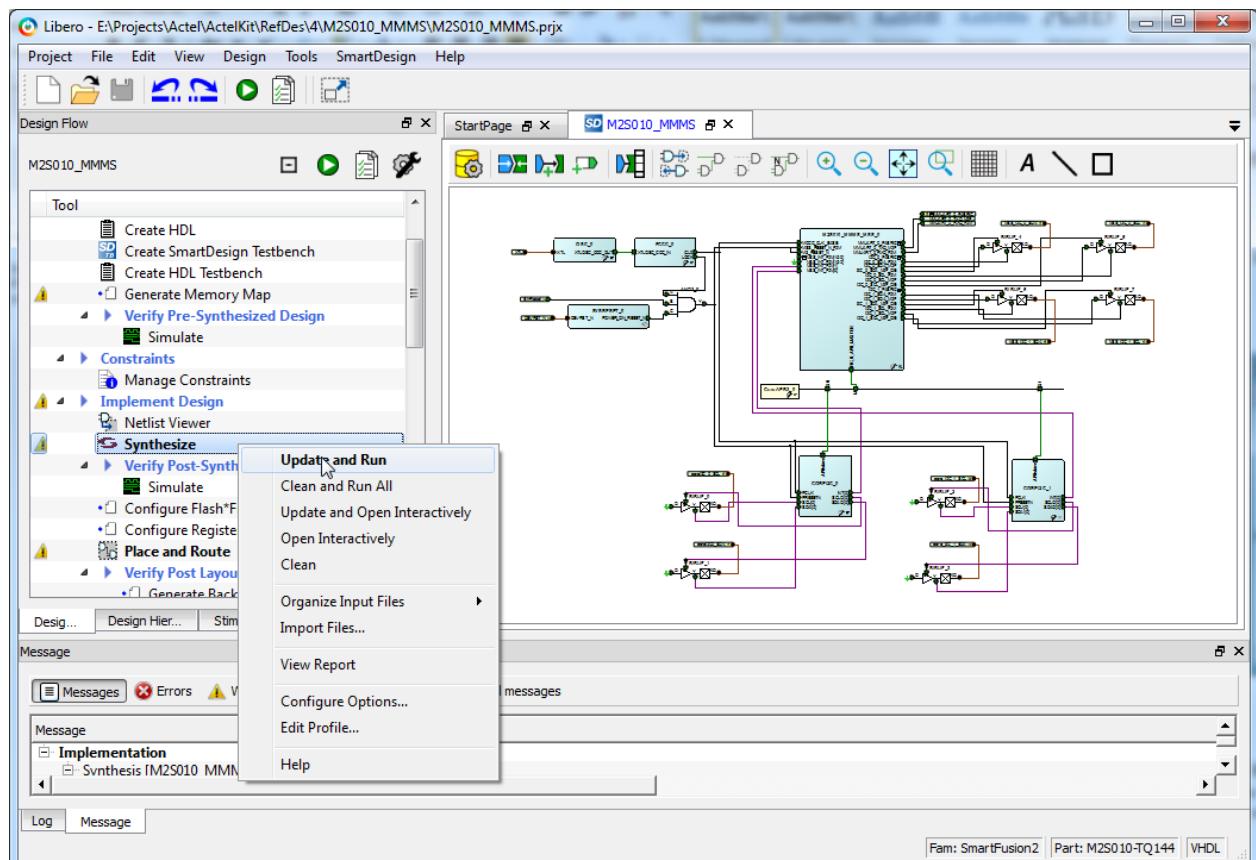


Рис. 32.

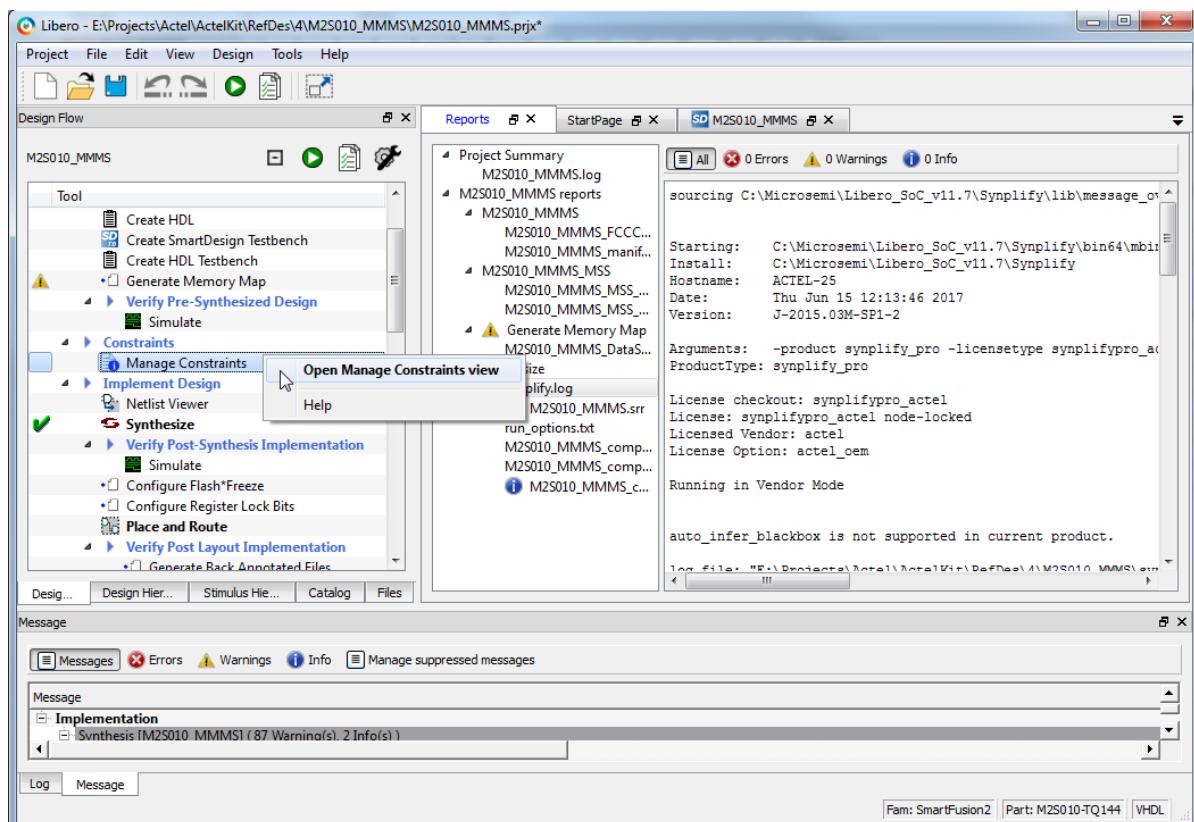


Рис. 33.

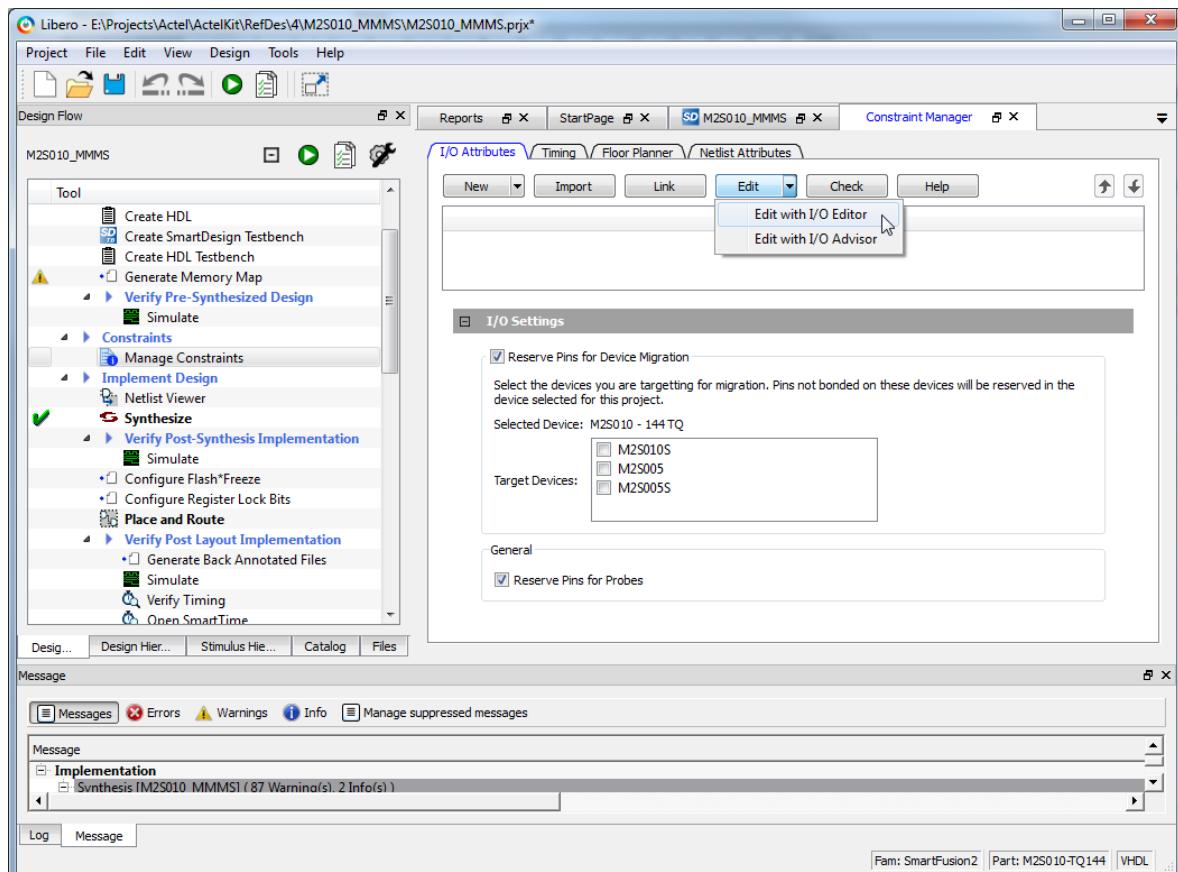


Рис. 34.

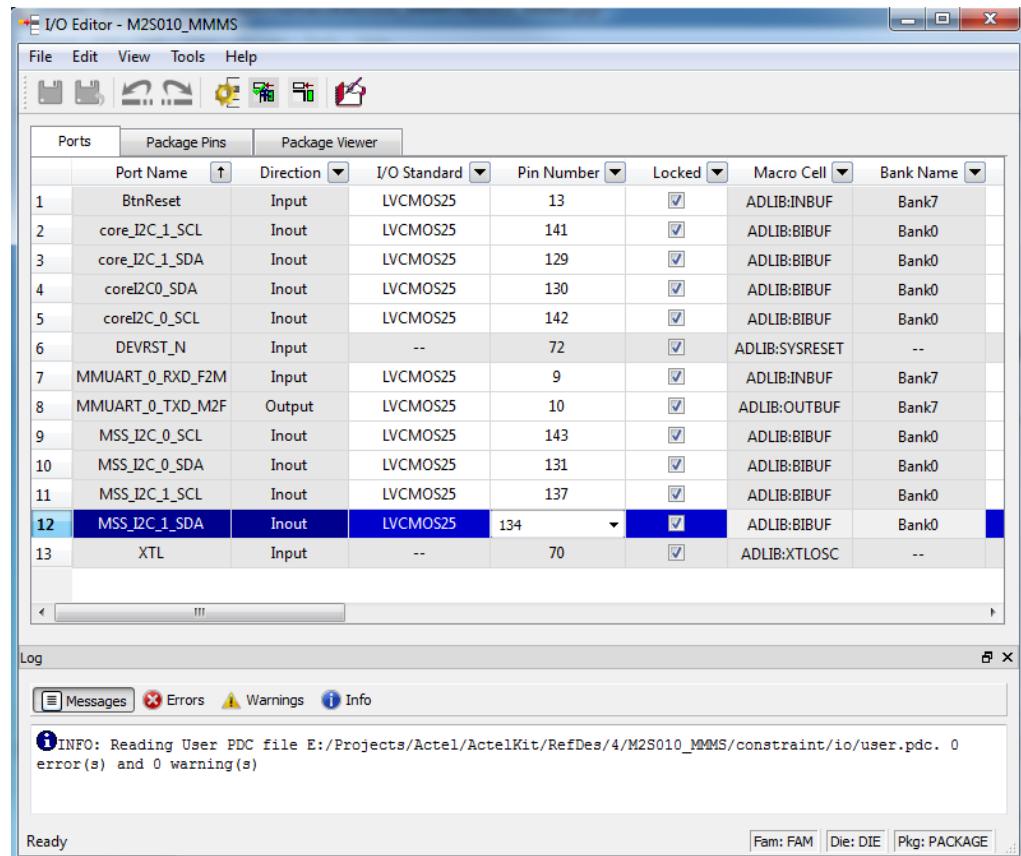


Рис. 35.

Выполним команду **Configure Firmware Cores** во вкладке **Design Flow** (рис. 36).

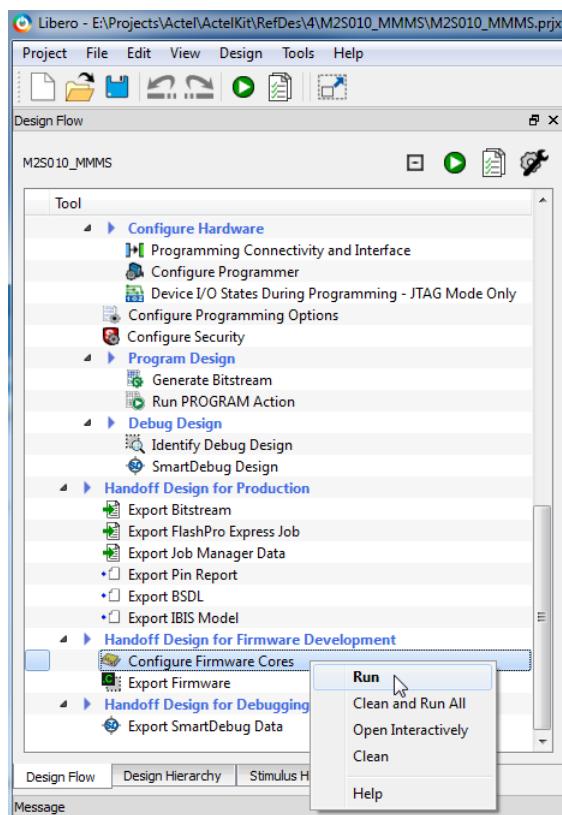


Рис. 36.

Теперь создадим проект встроенного программного обеспечения (ВПО) процессора ARM Cortex-M3 выполнив команду **Export Firmware**. В появившемся окне выберем опцию **Create project including hardware configuration and firmware drivers** (рис. 37).

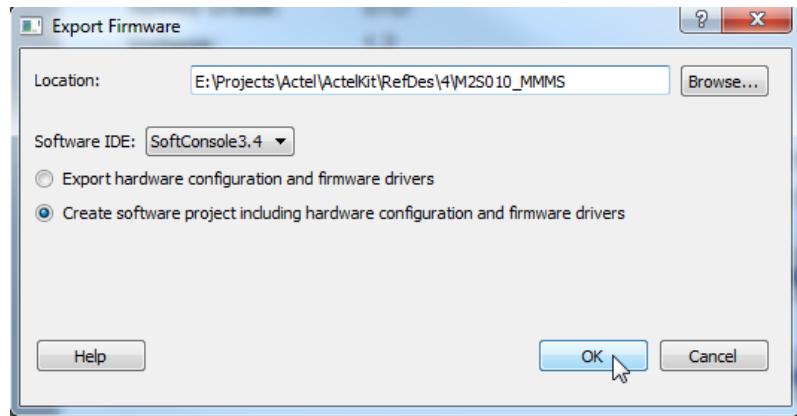


Рис. 37.

Откроем созданный проект ВПО в среде SoftConsole v.3.4 (рис. 38). Заменим содержимое файлов main.c и i2c_interrupt.c исходным кодом содержащимся в файлах из каталога /Source в архиве с файлами проекта, прилагаемом к данному руководству (либо заменим файлы в проекте файлами из архива) (рис. 39). Файлы с исходным кодом на языке «С» приложения можно скачать с сайта корпорации Microsemi по [ссылке](#) или с сайта компании ООО «ПСР Актел» в [архиве](#) с примером проекта.

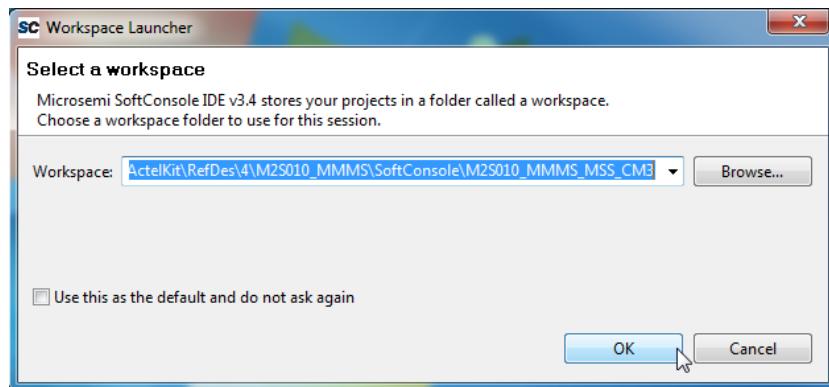


Рис. 38.

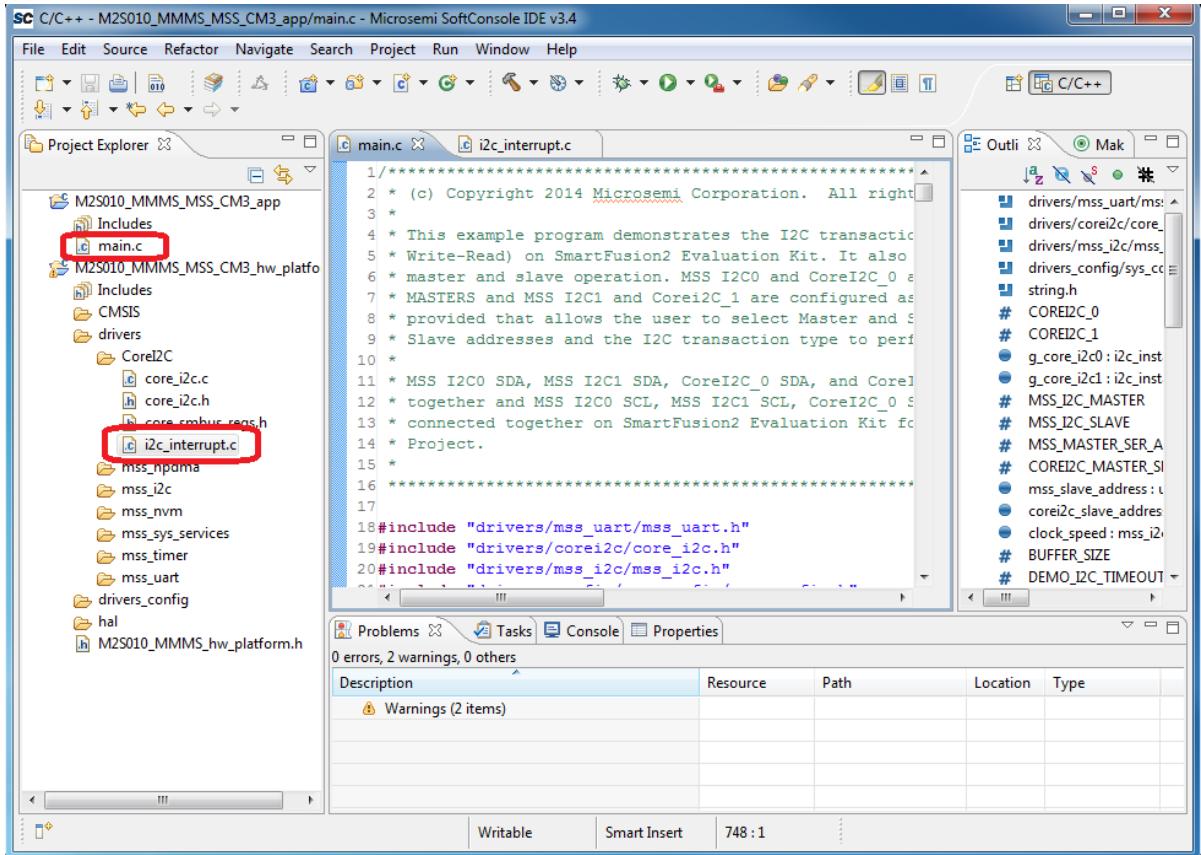


Рис. 39.

Откроем файл **M2S010_MMMS_hw_platform.h** (рис. 40). Проконтролируем совпадение начальных адресов IP-ядер coreI2C_0 и coreI2C_1 с одним из предложенных вариантов начальных адресов указанных в отчете **Generate Memory Map / M2S010_MMMS_DataSheet.xml** (рис. 31).

В модуле main.c добавим директиву

```
#include "M2S010_MMMS_hw_platform.h"
```

со ссылкой на файл M2S010_MMMS_hw_platform.h в котором содержатся актуальные значения начальных адресов IP-ядер coreI2C_0 и coreI2C_1 и закомментируем значения адресов указанных IP-ядер, присутствующих в проекте по умолчанию (рис. 41).

SC C/C++ - M2S010_MMMS_MSS_CM3_hw_platform/M2S010_MMMS_hw_platform.h - Microsemi SoftConsole IDE v3.4

File Edit Source Refactor Navigate Search Project Run Window Help

C/C++

Project Explorer

- M2S010_MMMS_MSS_CM3_app
- M2S010_MMMS_MSS_CM3_hw_platform
 - Archives
 - Includes
 - CMSIS
 - drivers
 - drivers_config
 - hal
 - Release
 - M2S010_MMMS_hw_platform.h**

main.c i2c_interrupt.c M2S010_MMMS_hw_platform.h

```

1#ifndef M2S010_MMMS_HW_PLATFORM_H_
2#define M2S010_MMMS_HW_PLATFORM_H_
3/*********************************************************************
4*
5*Created by Microsemi SmartDesign Wed Jun 21 09:36:04
6*
7*Memory map specification for peripherals in M2S010_MM
8*/
9
10/*
11* CM3 subsystem memory map
12* Master(s) for this subsystem: CM3
13*/
14#define COREI2C_0 0x70000000U
15#define COREI2C_1 0x71000000U
16
17
18#endif /* M2S010_MMMS_HW_PLATFORM_H */
19

```

Outliner

- # M2S010_MMMS_HW_PL
- # COREI2C_0
- # COREI2C_1

Problems Tasks Console Properties

0 items

Description	Resource	Path	Location	Type

Writable Smart Insert 14 : 52

Рис. 40.

SC C/C++ - M2S010_MMMS_MSS_CM3_app/main.c - Microsemi SoftConsole IDE v3.4

File Edit Source Refactor Navigate Search Project Run Window Help

C/C++

Project Explorer

- M2S010_MMMS_MSS_CM3_app
 - Binaries
 - Includes
 - Release
 - main.c**
- M2S010_MMMS_MSS_CM3_hw_platform
 - Archives
 - Includes
 - CMSIS
 - drivers
 - drivers_config
 - hal
 - Release
 - M2S010_MMMS_hw_platform.h

main.c i2c_interrupt.c M2S010_MMMS_hw_platform.h

```

16 ****
17
18#include "drivers/mss_uart/mss_uart.h"
19#include "drivers/corei2c/core_i2c.h"
20#include "drivers/mss_i2c/mss_i2c.h"
21#include "drivers_config/sys_config/sys_config.h"
22#include "string.h"
23#include "M2S010_MMMS_hw_platform.h"
24
25/*
26 * Register Base Address for two CoreI2C devices
27 */
28#define COREI2C_0 0x30000000U
29#define COREI2C_1 0x30001000U
30
31/*
32 * Instance data for two CoreI2C devices and MSS I2C
33 */
34i2c_instance_t g_core_i2c0;

```

Outliner

- drivers/mss_uart/ms...
- drivers/corei2c/c...
- drivers/mss_i2c/mss...
- drivers_config/sys...
- string.h
- M2S010_MMMS_hw...
- g_core_i2c0 : i2c_inst
- g_core_i2c1 : i2c_inst
- # MSS_I2C_MASTER
- # MSS_I2C_SLAVE
- # MSS_MASTER_SER_A
- # COREI2C_MASTER_SI
- mss_slave_address : u...
- corei2c_slave_address : u...
- clock_speed : mss_i2...
- # BUFFER_SIZE
- # DEMO_DC_TIMEOUT

Problems Tasks Console Properties

0 items

Description	Resource	Path	Location	Type

Writable Smart Insert 23 : 37

Рис. 41.

Установим активной конфигурацию Release (рис. 42).

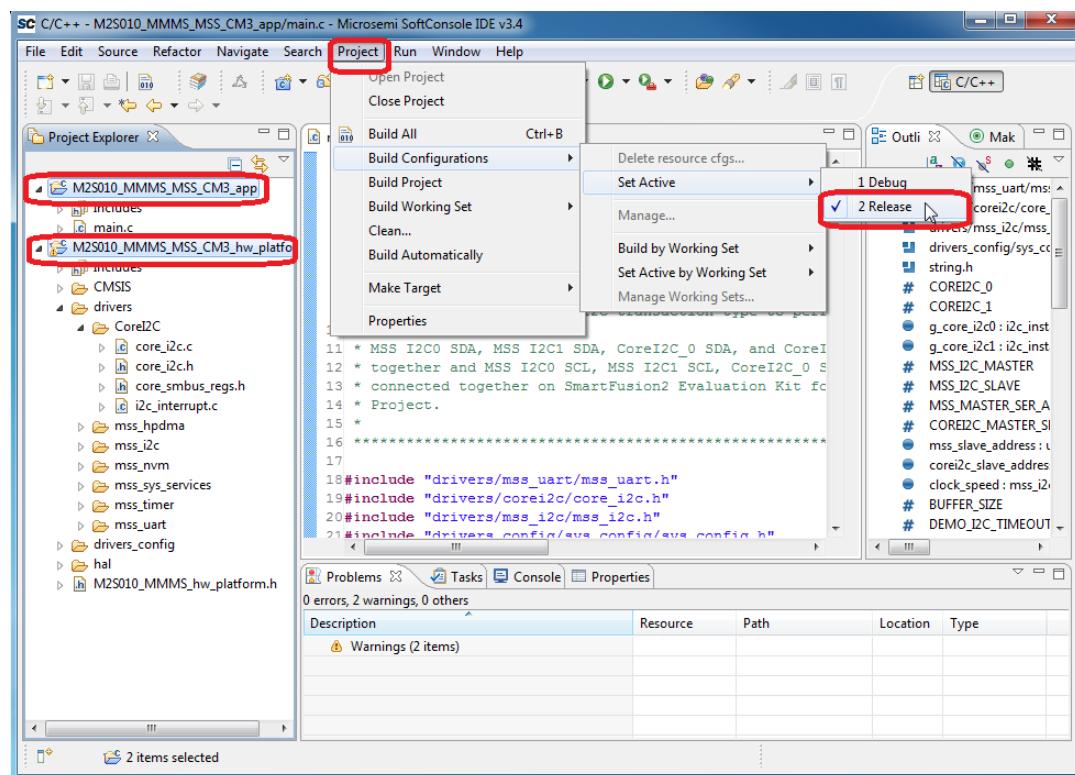


Рис. 42.

Выделив с помощью мыши проект M2S010_MMMS_MSS_CM3_app, отредактируем его свойства (рис. 43). Во вкладке C/C++ Build > Settings перейдем в окно Tool Settings и в настройках GNU C Linker > Miscellaneous (рис. 44) изменим значение параметра Linker flags на
-T\${workspace_loc:/M2S010_MMMS_MSS_CM3_hw_platform/CMSIS/startup_gcc/production-smartfusion2-execute-in-place.ld}

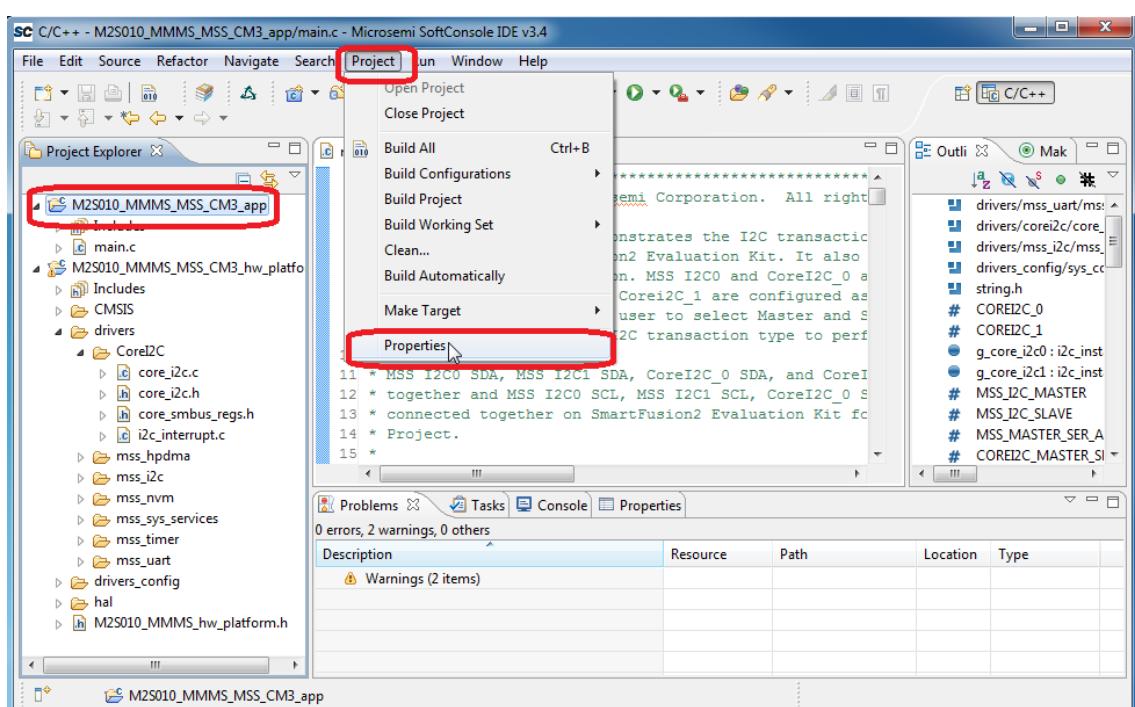


Рис. 43.

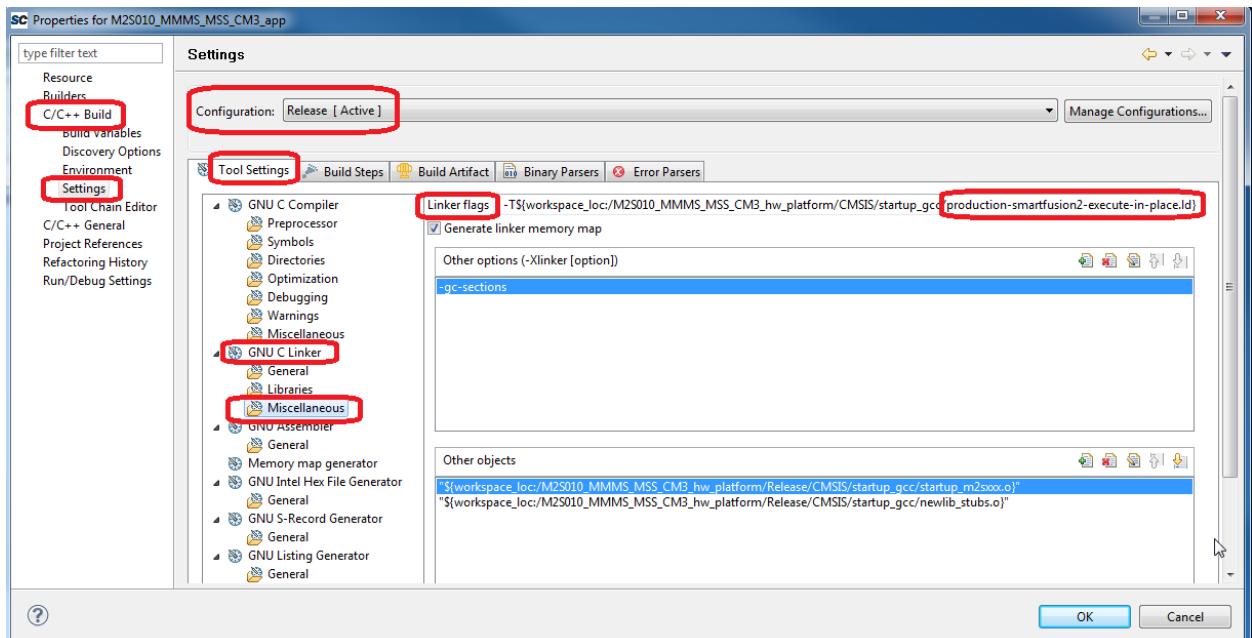


Рис. 44.

Выполним команду основного меню **Project > Clean**. В результате описанных действий в папке Release проекта ВПО появится файл с исполняемым кодом нашего приложения M2S010_MMMS_CM3_app.hex (рис. 45).

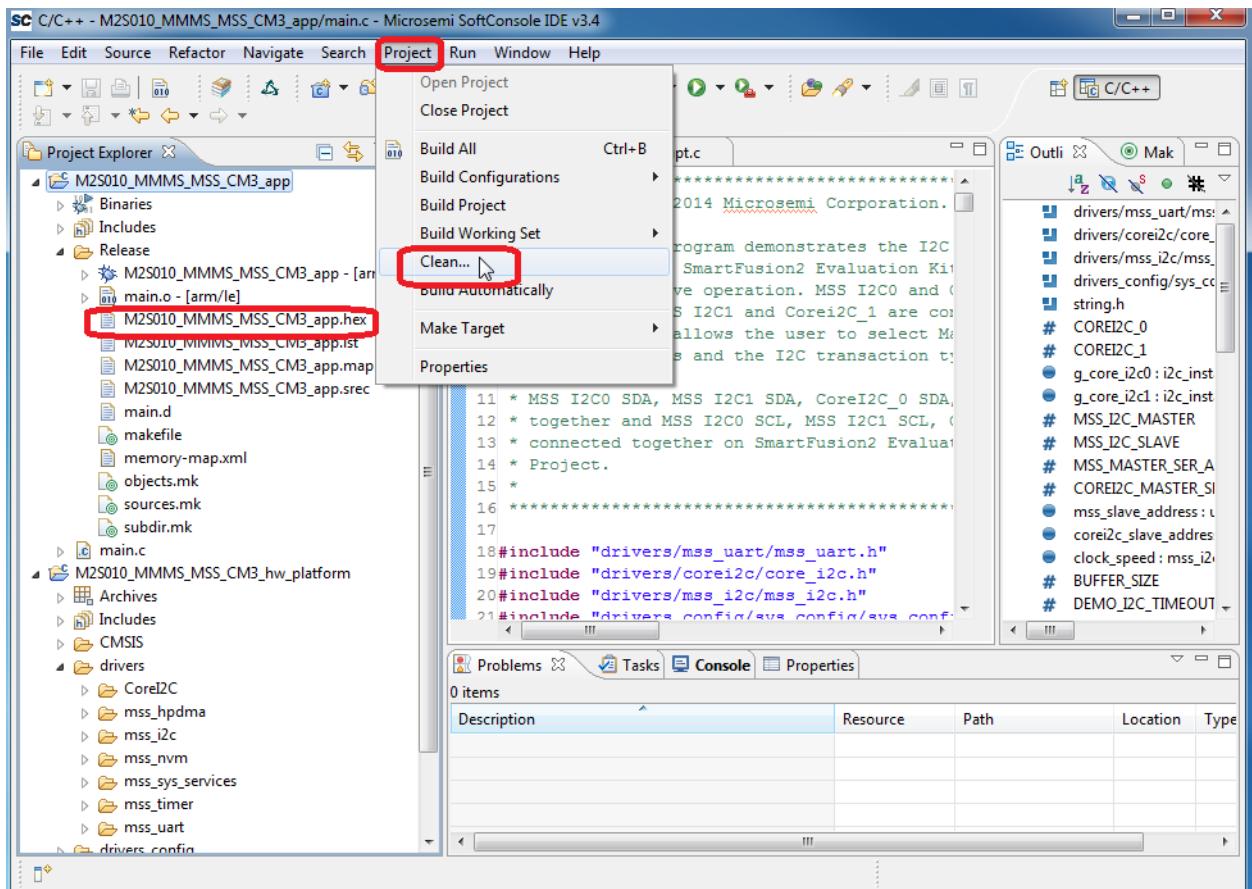


Рис. 45.

Теперь создадим файл конфигурационной последовательности, включающий битстрим матрицы ПЛИС и исполняемый образ встроенного программного обеспечения процессора. Для этого в проекте СнК в среде *Libero SoC* 11.8 в окне настроек микроконтроллерной подсистемы отредактируем свойства блока eNVM, для чего дважды щелкнем на соответствующем элементе графического интерфейса (рис. 46).

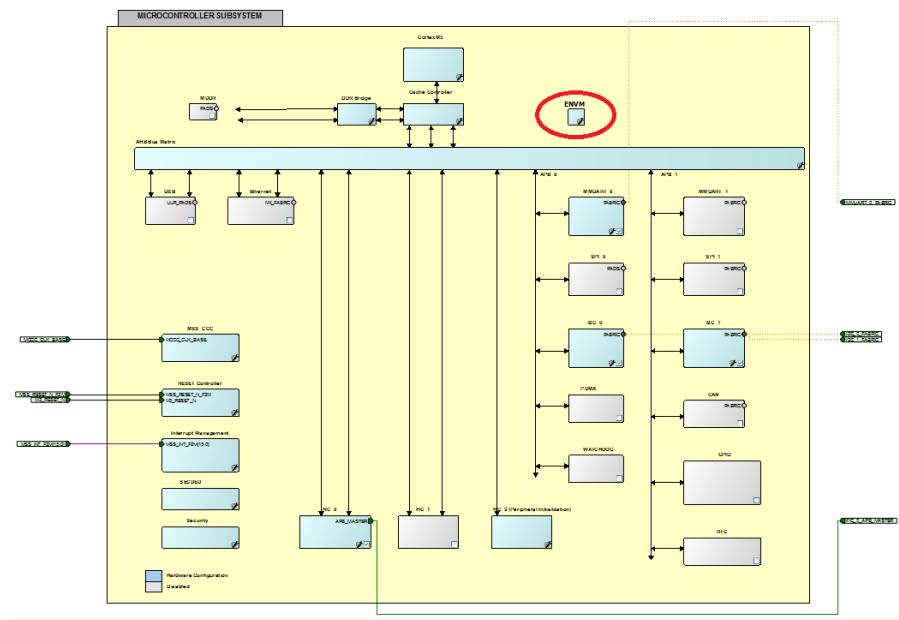


Рис. 46.

В появившемся окне **eNVM Configurator** дважды щелкнем на пункте **Data Storage** и в следующем появившемся окне **Add Data Storage Client** укажем условное название нашего приложения, а также пути к hex-файлу образа прошивки ВПО (рис. 47).

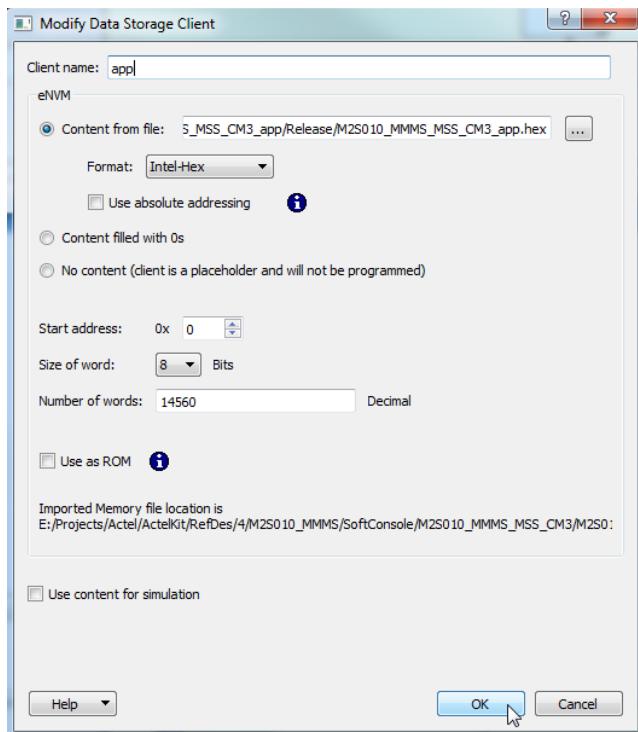


Рис. 47.

После чего окно **eNVM Configurator** примет вид, представленный на рис. 48. В окне отобразится строка с основными характеристиками загруженного файла.

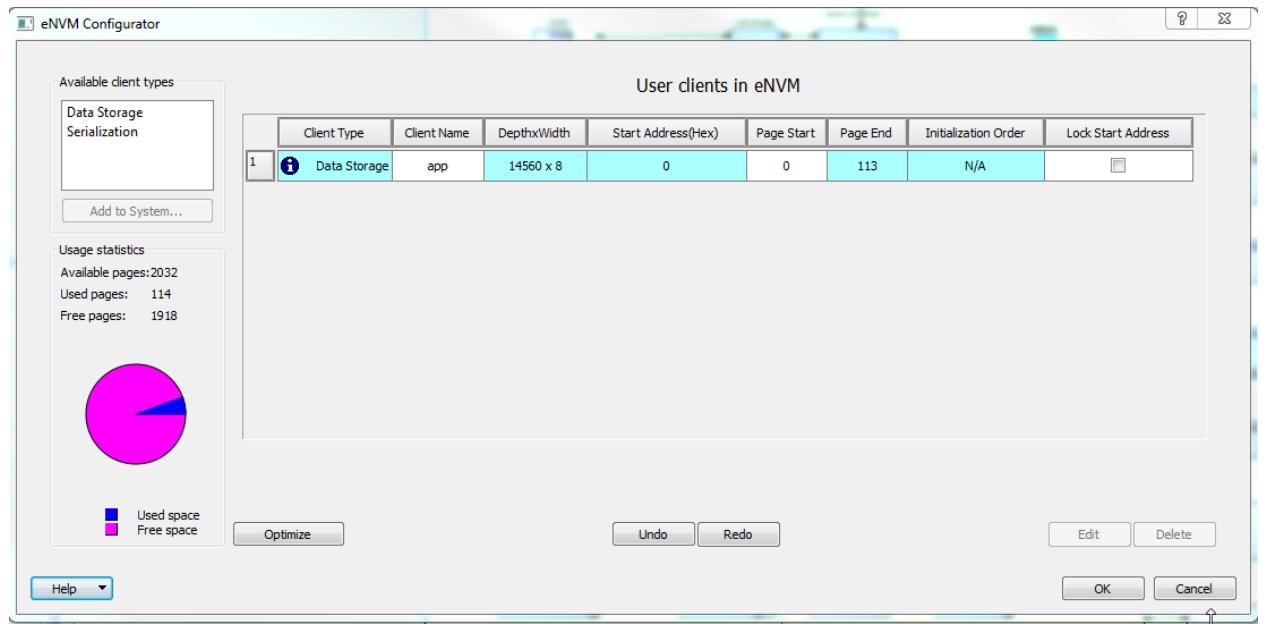


Рис. 48.

Сохраним изменения, выполним процедуру генерации компонента (рис. 49).

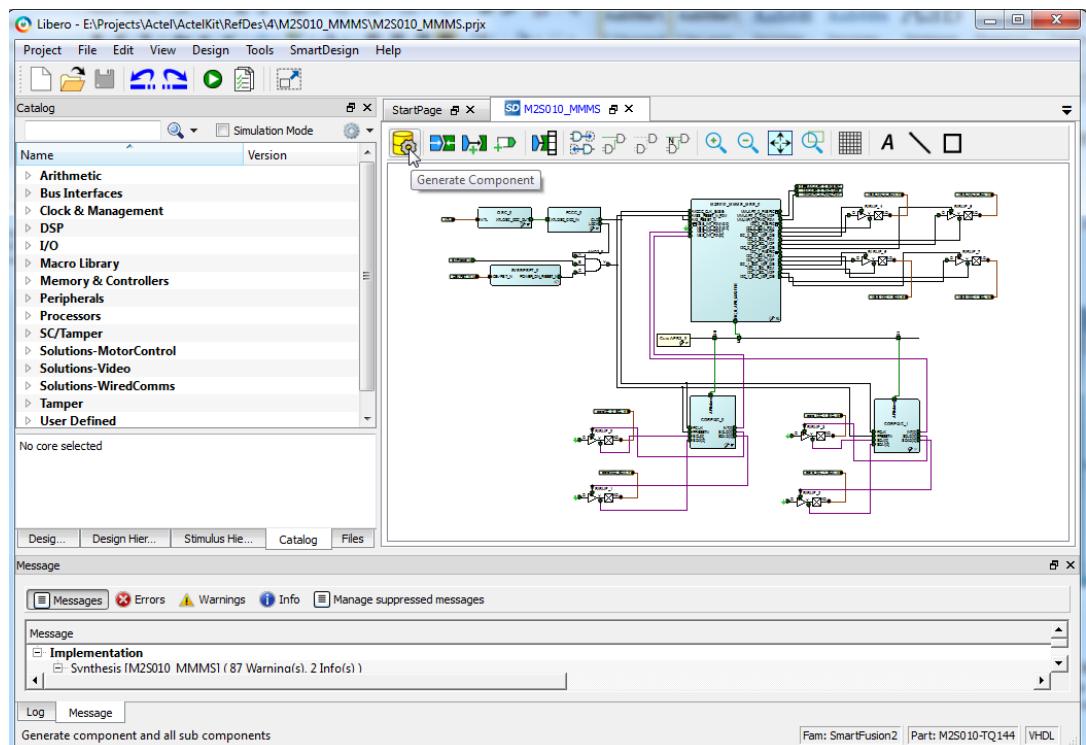


Рис. 49.

Подключим программатор к отладочному набору и персональному компьютеру, запустим процесс генерации файла прошивки и программирования (рис. 50).

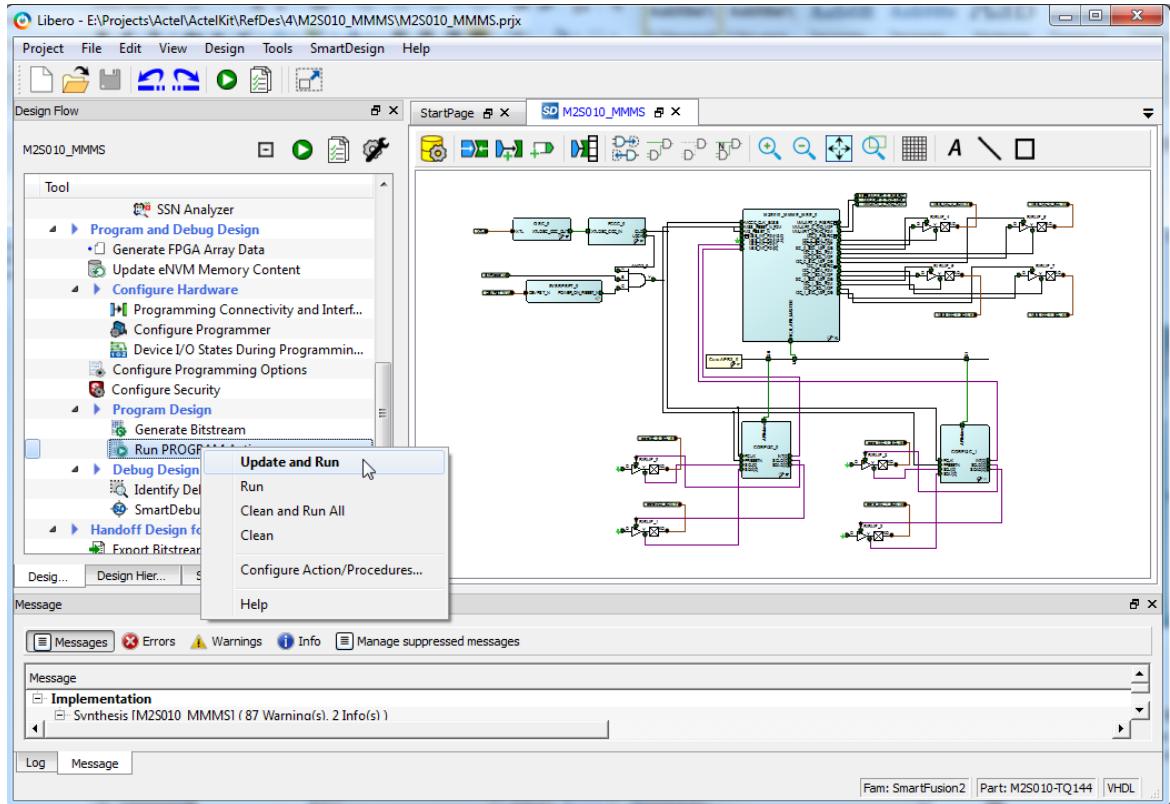


Рис. 50.

На плате отладочного набора SF2-Junior-KIT располагаются два разъема GPIO для подключения сигналов внешних устройств к контактам M2S010-TQ144 отладочного набора. В данном проекте контакты микросхемы M2S010-TQ144 назначены таким образом, чтобы вывести все входящие и выходящие сигналы проекта (x_SCL, x_SDA, UART_Rx, UART_Tx) на указанные разъемы GPIO. Теперь необходимо соединить между собой в пределах групп SCL и SDA все сигналы проекта x_SCL и x_SDA, то есть электрически соединить контакты разъема в две группы по четыре контакта каждая. Группа SCL: контакты 5, 6, 7, 8 разъема X9. Группа SDA: контакты 11, 12, 13, 14 разъема X9. Выполнить соединения можно с помощью соединительных проводов, входящих в состав набора и двух перемычек на 4 контакта, которые необходимо предварительно подготовить (табл. 4).

В состав набора SF2-Junior-KIT входят приемопередатчики USB – Bluetooth dongle и модуль UART- Bluetooth HC-06. Для связи отладочного набора с персональным компьютером (ПК) необходимо в USB разъем ПК включить USB – Bluetooth dongle, а к разъему X10 отладочного набора подключить модуль HC-06 (табл. 5).

Таблица 4. Подключение перемычек к разъему X9 отладочного набора SF2-Junior-KIT

№ контакта разъема X9 SF2-Junior-KIT	Номер контакта M2S010-TQ144	Название цепи проекта СнК	Название группы
5	142	coreI2C_0_SCL	SCL
6	143	MSS_I2C_0_SCL	SCL
7	141	coreI2C_1_SCL	SCL
8	137	MSS_I2C_0_SCL	SCL
11	134	MSS_I2C_1_SDA	SDA
12	131	MSS_I2C_0_SDA	SDA
13	130	coreI2C_0_SDA	SDA
14	129	coreI2C_1_SDA	SDA

Таблица 5. Подключение модуля HC-06 к разъему X10 отладочного набора SF2-Junior-KIT

№ контакта разъема X10 SF2-Junior-KIT	Номер контакта M2S010-TQ144	Название цепи проекта СнК / принципиальной схемы	Название цепи модуля HC-06
5	9	MMUART_0_RXD_F2M	RXD
6	10	MMUART_0_TXD_M2F	TXD
25		+3.3V	VCC
29		GND	GND

Выполняем подключения, описанные в таблицах 4, 5 (рис. 51). Прошиваем СнК конфигурационной последовательностью нашего проекта.

Запускаем программу оболочку M2S_I2C.exe из архива с файлами проекта. После открытия окна программы M2S_I2C.exe нажимаем кнопку BtnReset на плате отладочного набора SF2-Junior-KIT. После этого нажимаем кнопку «Connect» графического интерфейса программы. После выполнения соединения по串-порту выполняем транзакции чтения, записи, записи-чтения интерфейса I2C. Результат выполнения транзакций отображается в окне Write/Read Data (ASCII) графического интерфейса (рис. 48).

Проверяем корректность работы процедур чтения и записи по всем реализованным интерфейсам I2C (рис. 52).

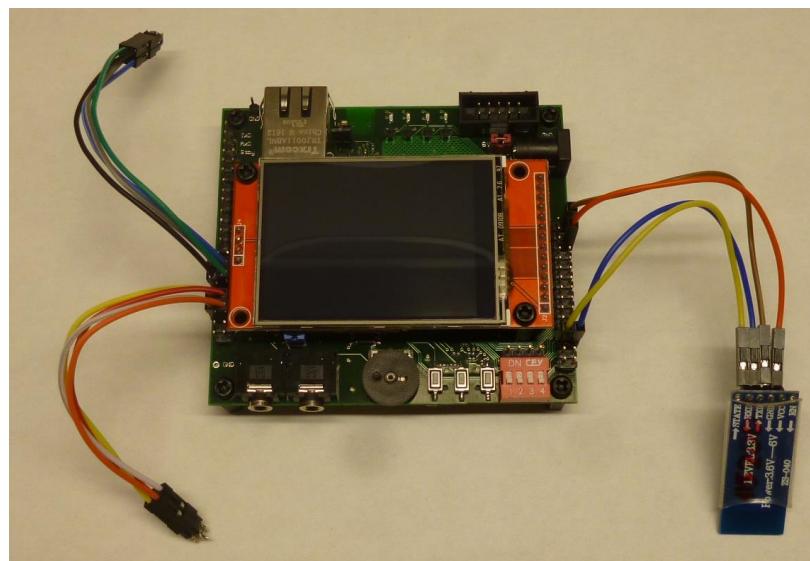


Рис. 51.

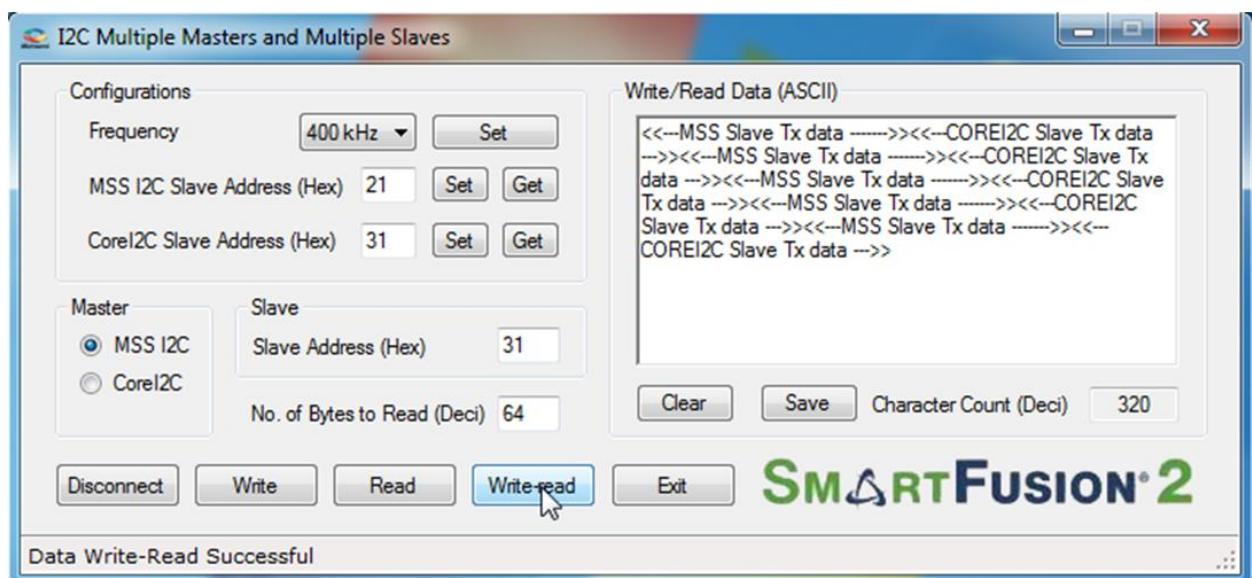


Рис. 52.

Проекты Снк и ВПО, разработка которых описана в данном руководстве, файлы с исходным кодом встроенного программного обеспечения, а также утилитой для визуализации работы приложения на экране ПК можно скачать по [ссылке](#).

Вопросы по материалу, изложенному в данном руководстве, можно задать сотрудникам службы технической поддержки компании ООО «ПСР Актел» по телефону **+7 (812) 740-60-09**, или по электронной почте support@actel.ru.