электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.31

Разработка радиационно-стойкого микроконтроллера на основе базового матричного кристалла

Селезнев А. А., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Специальная робототехника и мехатроника»

Научный руководитель: Кузин Ю. Р., зав. сектором отд. СМ4-4 НИИСМ
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
kafsm7@sm.bmstu.ru

Введение

Предпосылкой к написанию статьи стала существующая задача разработать для нужд российской оборонной промышленности систему управления, стойкую к радиационному воздействию. Специфика целевой области применения (оборонная промышленность) предполагает использование электронных компонентов исключительно отечественного производства. В связи с отсутствием удовлетворяющей требованиям задачи готовой БИС (микроконтроллера), было принято решение разработать специализированную систему на основе базового матричного кристалла (БМК). Для этого существуют удовлетворяющие требованиям радиационной стойкости достаточно вместительные кристаллы производства российской группы компаний «Ангстрем», а также находящиеся в открытом доступе soft-микропроцессорные ядра, на основе которых можно реализовать требуемый функционал в БМК.

БМК представляют собой универсальные кристаллы-заготовки, размещенные на полупроводниковой пластине. На кристалле в узлах прямоугольной матрицы располагаются простейшие элементы на основе транзисторов, которые коммутируются при производстве за счет нанесения маски соединений согласно предоставленному заказчиком описанию. В качестве описания может быть использован код на языке описания аппаратуры (HDL), например, на Verilog или VHDL. Одним из основных преимуществ БМК является возможность для разработчика спроектировать специализированную систему, по тем или иным причинам нереализуемую с помощью готовых БИС, а развитая технология и обширная библиотека логических элементов и схемотехнических решений позволяет снизить затраты на производство до приемлемых значений.

Soft-микропроцессор, или микропроцессор с программным ядром - это микропроцессорное ядро, которое можно создать с помощью логического синтеза, т. е. путем соединения логических вентилей БМК или ПЛИС согласно описанию системы. Такие ядра существуют в виде описаний на HDL и могут иметь как полностью закрытый (Xilinx MicroBlaze, Cortex) так и полностью или частично открытый исходный код (Aeroflex Gaisler LEON3, LatticeMico32, OpenRISC). Основными их преимуществами является наличие развитых средств разработки кода с широким спектром настроек и конфигурации и возможность использовать их в БМК и ПЛИС. В данной статье рассматривается бесплатный soft-процессор с открытым кодом LatticeMico32, разрабатываемый и поддерживаемый компанией Lattice Semiconductor как наиболее полно документированный, снабжаемый набором IP-ядер периферии и специализированным программным обеспечением для разработки.

В данной статье на примере демонстрационного проекта будут разобраны принцип разработки такой системы и все основные ее этапы.

Постановка задачи.

В рамках статьи были поставлены следующие задачи:

1. Конфигурация микроконтроллера на базе soft-процессора LatticeMico32 и прилагающихся периферийных блоков и портирование его на БМК;
2. Программирование полученного микроконтроллера на выполнение простых команд;
3. Тестирование и демонстрация полученного результата с помощью платы с ПЛИС.

Для решения данных задач будут использоваться: программные пакеты LatticeMico System и Xilinx ISE Design Suite, а также плата Xilinx Spartan-3AN Starter Kit с распаянной на ней ПЛИС Xilinx Spartan-3AN.

Обзор проекта.

LatticeMico32 представляет собой 32-разрядный открытый soft-процессор гарвардской RISC-архитектуры, использующий открытую параллельную шину Wishbone в качестве системной и периферийной. Ядро включает в себя опциональные кэши данных и инструкций, аппаратный модуль отладки, модули умножения и сдвига. В разработке проектов с этим ядром применяется программный пакет LatticeMico System, содержащий средства генерации Verilog-кода системы, написания программ для процессора, а также отладчик.

На плате Xilinx Spartan-3AN Starter Kit расположена ПЛИС Spartan-3AN, а также дополнительные элементы, такие как микросхемы памяти, порты RS-232, АЦП, ЦАП, и соединенные с пользовательскими ножками входа-выхода ПЛИС светодиоды. Для работы с ней будет использоваться программный пакет Xilinx ISE Design Studio, включающий в себя средства разработки и симуляции проектов на языках HDL, а также средства программирования ПЛИС производства компании Xilinx.

На рис. 1 представлена блок-схема, отражающая порядок разработки демонстрационного проекта. С помощью средства System Builder пакета LatticeMico System осуществляется конфигурация микропроцессорной системы. Полученные в результате Verilog-файлы используются в пакете Xilinx ISE Design Suite для создания на их основе бит-файла программирования ПЛИС Spartan-3AN. Полученный также в System Builder файл описания системы .msb используется для написания программы для полученного микроконтроллера в среде разработки на основе Eclipse, входящей в LatticeMico System. Эта программа в виде полученного после компиляции и компоновки кода будет записана во flash-память STM M29DW323DT, присутствующую на плате [5]. В начале своей работы процессор обратится к flash-памяти, считает инструкции и выполнит программу. Flash-память будет использована только на этапе тестирования с помощью ПЛИС, а в реальном проекте на основе БМК для соблюдения требований к системе по радиационной стойкости будет использоваться радиационно стойкая память типа Mask ROM.

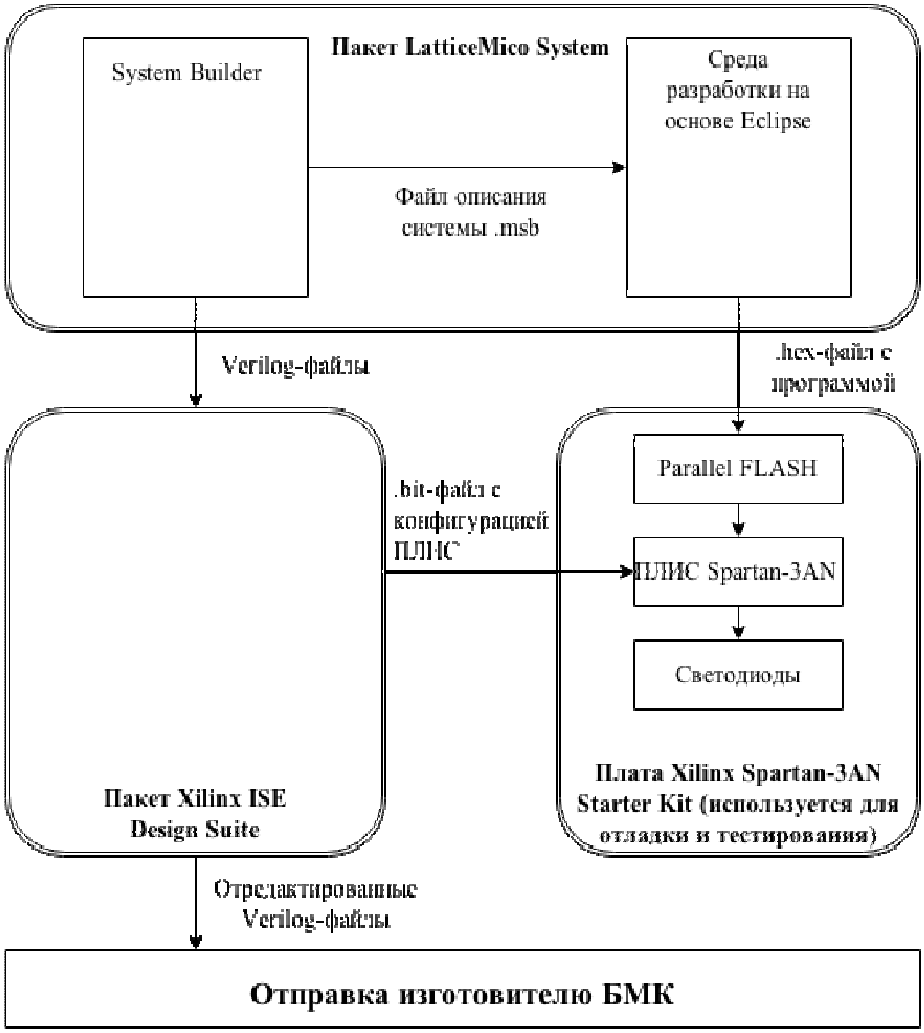

Рис. 1. Порядок разработки проекта

Конфигурация микропроцессорной системы.

Конечным результатом конфигурации системы на основе soft-процессора является дерево файлов с HDL-кодом, описывающих и объявляющих все модули и их взаимосвязи. Этого дерева достаточно как для получения бит-файла, программирующего ПЛИС, так и для синтеза маски соединений БМК.

В случае с LatticeMico32 – это Verilog-файлы .v. Для их первоначальной генерации используется средство System Builder программного пакета LatticeMico System. На рис. 2 представлено окно System Builder. В изображенном на рисунке окне открыт проект системы, содержащий ядро LatticeMico32 (LM32), контроллер параллельной flash-памяти (flash) и модуль портов входов-выходов (gpio). В каждом из компонентов могут быть настроены различные параметры, такие как разрядности шин, размеры блоков памяти и т.д. System Builder предоставляет также возможность добавления собственных компонентов и создания для них пользовательского интерфейса настройки, при условии наличия их HDL-кода и правильно описанного в нем Wishbone-интерфейса [2].

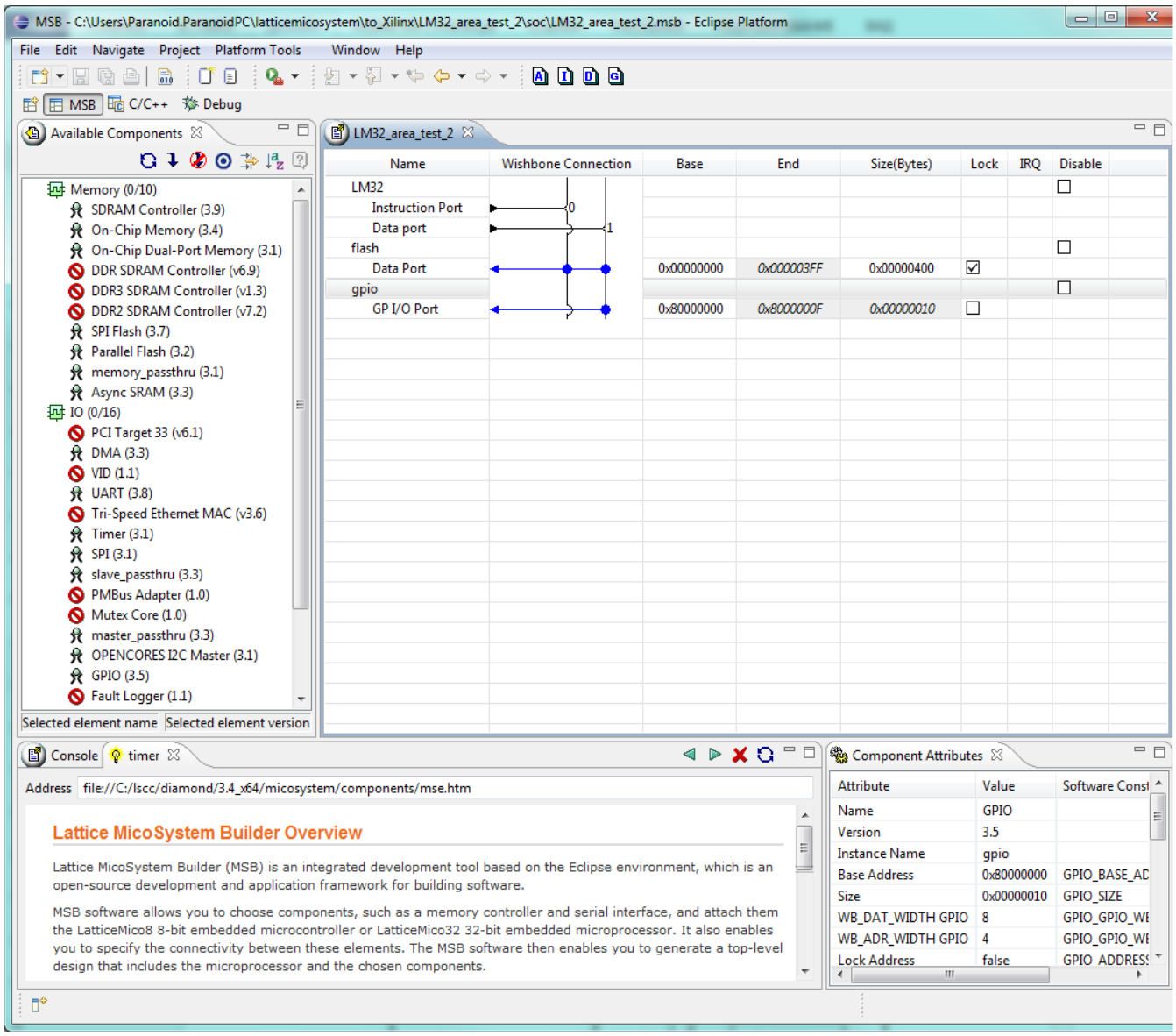

Рис. 2. Пример окна LatticeMico System Builder

На рис. 3 в качестве примера изображено диалоговое окно настройки ядра LM32. Параметр «Location of Exception Handlers» задает адрес, в котором должен находиться обработчик сброса, то есть инструкции, которую ядро считает первой [2,3]. В данном случае этот адрес совпадает с адресом начала флэш-памяти.

С помощью System Builder задаются также соединения с шиной всех компонентов, их адреса, приоритеты обработки прерываний, поступающих с них; осуществляется проверка правильности компоновки системы. После проверки по нажатию кнопки «Generate» программа сгенерирует, во-первых, полное дерево Verilog-файлов, и, во-вторых, файл описания системы .msb, который используется вторым компонентом пакета – средой разработки на основе Eclipse, к которой мы вернемся позже.

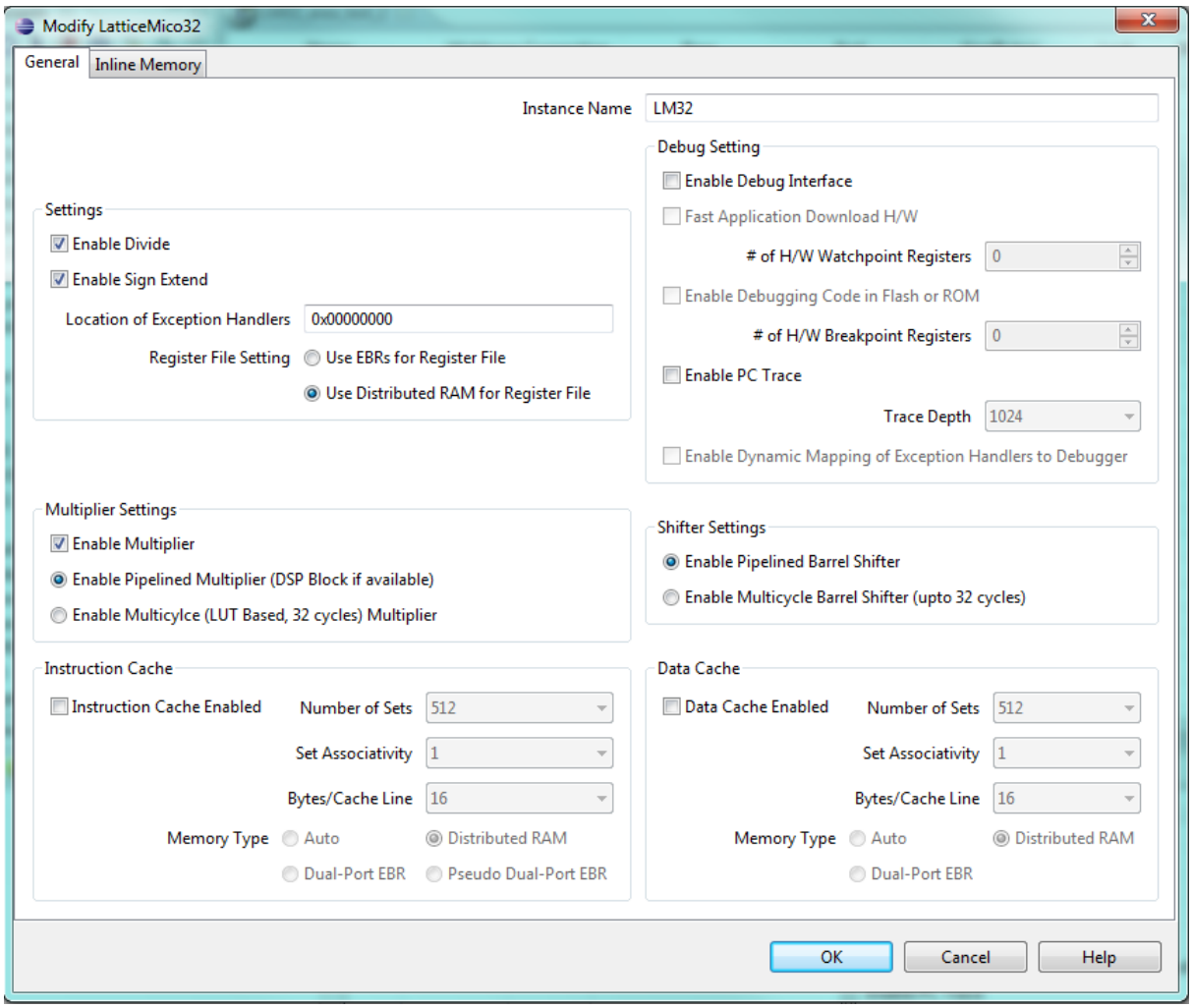

Рис. 3. Пример окна настройки ядра LatticeMico32

Следующим шагом будет импорт Verilog-файлов в проект программного пакета Xilinx ISE Design Suite. С его помощью будут решаться две задачи: редактирование Verilog-файлов и программирование ПЛИС Xilinx Spartan-3AN на плате.

Необходимость редактирования файлов возникает в связи со следующим недостатком LatticeMico32: его исходный код написан в расчете на использование ПЛИС фирмы Lattice Semiconductor, и поэтому содержит конструкции, объявляющие специфические для этих ПЛИС компоненты. Вследствие этого синтезатор, отличный от синтезатора Lattice, не сможет распознать и синтезировать эти компоненты, что затрудняет портирование на другие платформы. Однако, все такие конструкции достаточно легко заменимы вручную либо на универсальные, либо на специфические уже для целевой платформы компоненты. Причем стоит отметить, что такие универсальные конструкции иногда находятся непосредственно в коде LM32, и для их использования требуется лишь убрать if-ветвления, их обходящие. Например, одной из необходимых

правок в демонстрационном проекте была замена буферов с третьим состоянием, объявленных в коде как готовые модули, распознаваемые синтезатором Lattice, на аналогичные, написанные вручную. Для редактирования кода можно использовать любой текстовый редактор, но ISE Design Suite, во-первых, предоставляет удобную среду разработки для этого, и, во-вторых, позволяет сразу проверить использованные в коде универсальные конструкции на синтезируемость.

После редактирования кода можно сгенерировать бит-файл конфигурации, который через USB-кабель JTAG, разъем для которого присутствует на плате, загружается в ПЛИС. Перед генерацией бит-файла в ISE Design Suite следует обозначить, на какие выводы ПЛИС будут выведены входные и выходные сигналы. В данном случае в систему нужно завести тактовый сигнал (для него используется присутствующий на плате генератор тактовой частоты 50 МГц) и сигнал сброса, который назначается на любую кнопку или переключатель на плате. Выходом системы будут являться сигналы модуля входов-выходов, которые следует вывести на светодиоды на плате. Также необходимо соединить шины адреса и данных flash-контроллера и flash-памяти на плате.

Написание программы для микроконтроллера.

Теперь для проверки работоспособности полученного микроконтроллера было решено написать для него исполняемую программу и загрузить ее во flash-память. Для этой цели используется специализированная среда разработки, являющаяся частью пакета LatticeMico System. Она сделана на основе открытой среды разработки Eclipse и опытный разработчик программ для микроконтроллеров без труда освоится в ней. На рис. 4 представлен пример окна среды.

В изображенном окне можно увидеть код на языке программирования C, реализующий попеременное мигание светодиодов на плате. Использованные в этом коде функции MicoGetDevice, MICO_GPIO_WRITE_DATA и MicoSleepMilliSecs предоставляются драйверами, которыми в виде файлов на языках C и ассемблера снабжены все стандартные компоненты LatticeMico32. В случае разработки собственных компонентов разработчик также может снабдить их драйверами, файлы с кодом которых будут автоматически генерироваться при создании системы с этими компонентами [3].

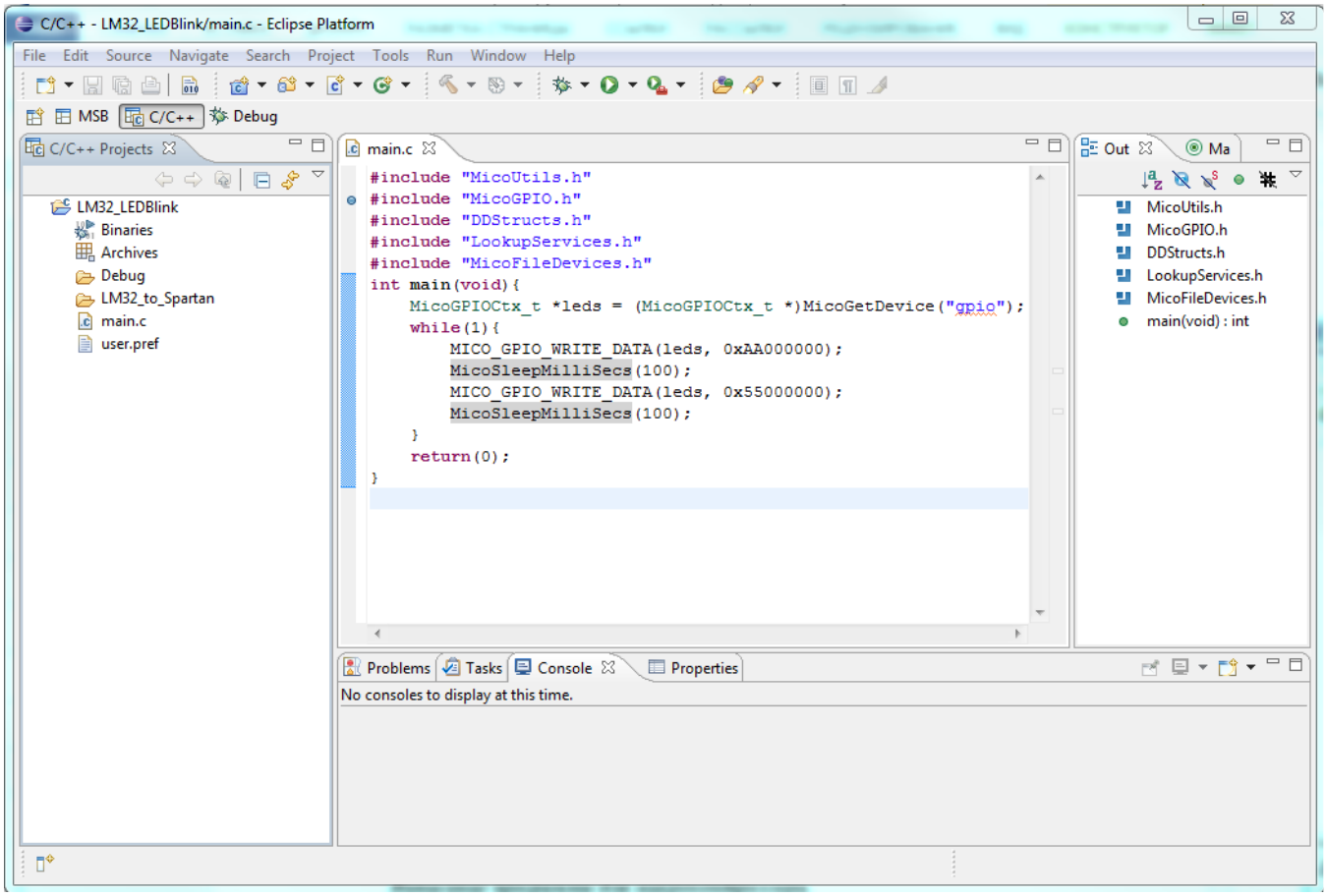

Рис. 4. Пример окна LatticeMico Eclipse C/C++

При создании C/C++ проекта в среде необходимо указать файл описания системы .msb, полученный при генерации системы в средстве System Builder. От выбора этого файла зависит работа компоновщика кода при сборке проекта, так как в нем содержится информация о типе, размере и адресах существующих в системе устройств памяти. На основе этой информации компоновщику кода необходимо будет указать, какие устройства использовать в качестве программной памяти (program memory), памяти только для чтения (read-only memory) и памяти для записи и чтения (read/write memory). Программная память служит для хранения инструкций, в памяти для записи и чтения располагается оперативная память системы, а в памяти только для чтения могут хранится константы, используемые в программе. На рис. 5 изображен пример окна настройки параметров проекта, связанных с целевой системой. Настройки предусматривают возможность предварительного программирования (deployment) одного или нескольких устройств памяти, используемых в программе [1, 4].

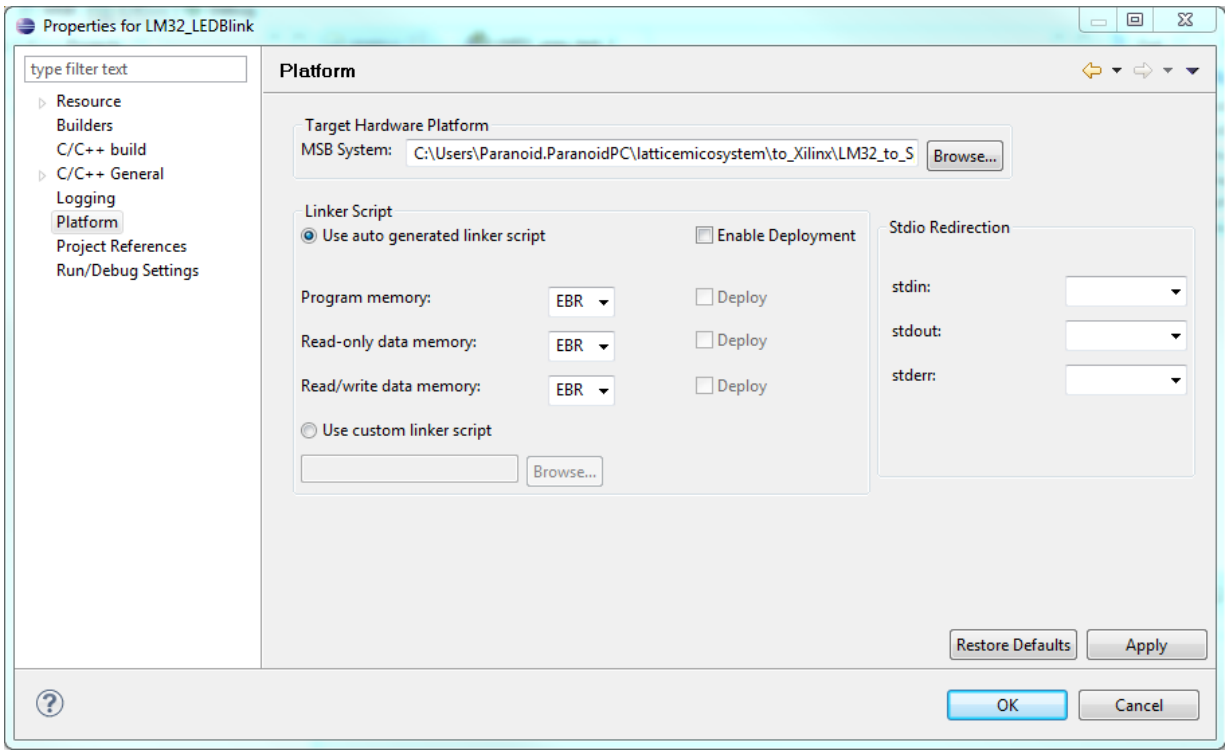

Рис. 5. Пример окна параметров проекта

Однако, в системе, собранной ранее, присутствует только flash-память, которая не может быть использована для быстрой записи данных. Это означает отсутствие в системе оперативной памяти. Решение о неиспользовании других видов памяти было принято по следующей причине: на плате отсутствует внешняя RAM, находящаяся на общих с используемой flash-памятью шинах, а использование внутренней блочной RAM ПЛИС, имеющейся в Spartan-3AN, некорректно с точки зрения тестирования возможностей по портированию LM32 на БМК, так как в предполагаемых целевых БМК внутренняя RAM отсутствует. Оперативная память необходима для работы программ на языке С, и поэтому в рамках рассматриваемого проекта от использования С было решено отказаться в пользу написания простой программы на языке ассемблера LM32. Описание всех команд языка ассемблера LM32 присутствует в документации. Код программы приведен ниже:

```
        xor  r0, r0, r0
        mvi  r9,  0x0000
        mvhi r9,  0x8000
        mvi  r10, 0x5555
        mvhi r10, 0x5555
        sw   (r9+0), r10
lp1:    nop
```

bi lp1

Данная программа первой инструкцией обнулит в ядре регистр общего назначения r0, который для правильной работы некоторых псевдо-инструкций языка должен всегда содержать 32-разрядный ноль. Примерами таких псевдо-инструкций служат три из четырех других инструкций (кроме sw), использованных в программе, в частности, инструкция записи в регистр mvi r9, 0x0000 в действительности реализуется как сложение addi r9, r0, 0x0000. По этой причине обнуление этого регистра при сбросе системы рекомендовано производителем, а его использование в качестве пользовательского регистра может нарушить работу программы [1]. Далее, с помощью двух других регистров общего назначения, по адресу портов ввода-вывода будет записано значение 0x55555555, благодаря чему светодиоды на плате загорятся через один. И, наконец, с помощью двух последних инструкций код зациклится между двумя соседними адресами.

С помощью предоставляемых пакетом LatticeMico System средств приведенный код был скомпилирован и скомпонован в .elf-файл, а затем преобразован в файл формата Intel HEX. В таком виде им можно программировать flash-память. Для программирования памяти использовался flash-программатор на основе Spartan-3AN, предоставляемый компанией Xilinx как демонстрационный проект для плат Spartan-3A/3AN Starter Kit. В нем сама ПЛИС конфигурируется в программатор, который связывается с компьютером через интерфейс RS-232. Через интерфейс передается .hex-файл с программой, и программатор записывает данные из него в схему памяти STM M29DW323DT по указанному адресу. В данном случае указывается нулевой адрес, так как именно к нему ядро обратится при запуске. Эти действия необходимо совершить до конфигурации ПЛИС в микроконтроллер, так как иначе bit-файл программатора изменит конфигурацию и ее потребуется провести заново.

После программирования flash-памяти и конфигурации ПЛИС по отпусканию кнопки сброса можно наблюдать, как на плате загораются светодиоды. Это свидетельствует об исполнении сконфигурированным микроконтроллером программы.

Заключение.

В статье было проведено создание демонстрационного проекта, и описаны все этапы этого процесса. По завершению тестирования полученный Verilog-код был отправлен на предприятие «ОАО «Ангстрем». Проект был отправлен в двух вариантах: минимальном, в котором в настройках ядра были отключены все опциональные аппаратные модули, такие как умножитель, и расширенном, где присутствовали

аппаратные делитель и умножитель, а также конвейерный модуль сдвига. На предприятии с помощью специализированного синтезатора был проведен тест, показывающий сколько эквивалентных вентилей в БМК предприятия серии 5522 займет синтезированный проект. Тест показал, что минимальный проект занимает 34 343 вентиля, а расширенный – 41 272 вентиля. Максимальная допустимая сложность БМК 5522 составляет приблизительно 100 000 вентилей. Результаты теста говорят о том, что проект реализуем в БМК ОАО «Ангстрем», и на кристалле остается достаточно места для реализации периферийных модулей в более сложных проектах.

Таким образом, рассмотренная в статье методика позволяет разрабатывать микропроцессорные системы на основе LatticeMico32 для БМК и ПЛИС, в частности, радиационно-стойких.

Список литературы

1. LatticeMico32 Processor Reference Manual. © Lattice Semiconductor Corporation, 2012. Available at: http://www.latticesemi.com/~/media/Documents/UserManuals/JL/LatticeMico32ProcessorReferenceManual21.PDF, accessed 23.03.2015.
2. LatticeMico32 Hardware Developer User Guide. © Lattice Semiconductor Corporation, 2014.Available at: http://www.latticesemi.com/~/media/Documents/UserManuals/JL/LatticeMico32HWUG32.pdf, accessed 24.03.2015.
3. LatticeMico32 Software Developer User Guide. © Lattice Semiconductor Corporation, 2014. Available at: http://www.latticesemi.com/~/media/Documents/UserManuals/JL/LatticeMico32SWDeveloperUG32.pdf, accessed 24.03.2015).
4. LatticeMico32 Tutorial. © Lattice Semiconductor Corporation, 2012. Available at:: http://www.latticesemi.com/~/media/Documents/Tutorials/LZ/LatticeMico32Tutorial.PDF, accessed 23.03.2015).
5. Spartan-3A/3AN FPGA Starter Kit Board User Guide. © Xilinx, Inc., 2008. Режим доступа: http://www.xilinx.com/support/documentation/boards_and_kits/ug334.pdf (дата обращения 22.03.2015).