МЕТОДЫ ВНЕДРЕНИЯ ИНФОРМАЦИИ В ПРОГРАММНЫЙ КОД ЦИФРОВЫХ УСТРОЙСТВ С МПУ

А.В. Брель
Кафедра ПОИТ, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: alexandervbrel@gmail.com

Эта статья описывает методы внедрения информации в программный код цифровых устройств с микропрограммным управлением, а также приводит методы защиты от обнаружения и извлечения внедрённой информации.

Введение

В настоящее время наряду со специализированной цифровой логикой для решения задач широкое применение получили устройства с микропрограммным управлением (МПУ). Такие устройства позволяют совместить как оптимизацию логики при цифровом проектировании, так и универсальность, и гибкость микропрограммного управления.

При решении задач с помощью устройств с МПУ затраты на разработку кода микропрограммы сопоставимы и могут превышать затраты на разработку проекта устройства. Также как и проект цифрового устройства, микропрограммный код может быть неправомерно использован третьими лицами. В настоящее время для подтверждения авторских прав, а также для подтверждения прав на использование продуктов в цифровом проектировании широко применяется технология внедрения «водяных знаков» и «отпечатков пальцев» [1].

I. Основные задачи

Основными задачами при разработке данных методов внедрения информации являются: минимизация воздействие на исходный микрокод, а также описание и функционирование устройства, простота извлечения данных, вместе с возможностью автоматизации внедрения и извлечения информации.

II. Избыточность микропрограммного кода

При проектировании набора микрокоманд устройства с МПУ выгодно оптимизировать формат инструкций для меньшей избыточности кода, так как это экономит размеры памяти, требуемой для хранения микропрограммы. С другой стороны, такая оптимизация может привести к усложнению формата инструкций, а значит и устройства декодирования, что увеличивает ресурсы, потребляемые устройством. Поэтому, обычно инструкции имеют однотипный формат, в котором имеется поле кода операции и прочие поля - операнды этой операции, а для сокращения длины поля кода операции для однотипных инструкций дополнительная идентификация инструкции выносится за пределы этого поля - в поля операндов операции. Иногда операции, имеющие разные мнемоники могут иметь один и тот же код операции, но внедрять информацию в некоторые неиспользуемые биты из полей операндов инструкции. Например, в soft-процессоре фирмы Xilinx PicoBlaze типичный размер кода операции 6 бит, но операции условного вызова(CALL X,Sub), условного перехода(JUMP X,Adr), управления разрешения прерываний, условного возврата (RETURN X), а также операции сдвигов сгруппированы под одинаковыми кодами инструкций, но различаются битами в полях операндов. В архитектуре soft-процессора OpenRISC типичный размер поля инструкции кода операции 6, но за счёт группирования операций и использования полей операндов для идентификации количество реализуемых операций в этой архитектуре достигает 235.

Примером архитектуры, в которой избыточность набора инструкций была сведена практически к нулю можно назвать Atmel AVR. Это архитектура с размером инструкции, в которой имеется 136 мнемоник операций. Однако, за счёт того, что некоторые мнемоники являются "виртуальными» (реализованы через другие операции) количество операций в наборе команд меньше. Например, мнемоники add(сложение регистров) и lsl(логический сдвиг регистра влево) имеют одинаковый код операции, а операция lsl реализуется через сложение регистра с самим собой. Также в этой архитектуре нет чёткого понятия поля кода операции, а также общего формата команд. При этом интересно кодирование операций: операции делятся на группы, в которых коды операций, как и коды групп, формируются с помощью префиксного кодирования. Также для некоторых операций биты отдельного операнда могут находиться не последовательно, а быть разбросаны по инструкции.

III. Внедрение информации с использованием избыточности микропрограммного кода

Избыточность кода присуща практически всем архитектурам с МПУ и позволяет внедрять достаточные для доказательства авторских прав

данные. В данном случае можно говорить о статическом водяном знаке, так как он никак не влияет на функционирование устройства. Рассмотрим формат операции SRR, которая сдвигает регистр вправо на одну позицию в архитектуре с четырьмя адресуемыми регистрами с длиной команды 8: 3 старших бита отводятся для кода операции, затем два бита - для обозначения инкрементируемого регистра. Остальные же три бита никак не задействованы. Эти биты могут быть использованы для внедрения части информации.

Данный тип внедрения информации в микропрограммный код легко поддаётся автоматизации. Для внедрения данных необходимо задать формат и описание команд с указанием неиспользуемых битов в инструкциях, информацию для внедрения и начальную микропрограмму в бинарном виде. Результатом работы этого этапа внедрения является микропрограмма в бинарном виде с внедрённой на уровне инструкций информацией.

IV. Внедрение в микрокод неиспользуемых инструкций

В большинстве наборов команд устройств с микропрограммным управлением имеются команды условных и безусловных переходов. Это означает, что часть микропрограммы может быть не исполнена и даже не прочитана из ПЗУ при определённых условиях. Используя это свойство, можно осуществить внедрение информации в микропрограмму добавлением не исполняемых участков, ограниченных безусловным (или контролируемым условным) переходом. Для считывания такого водяного знака требуется вычитать всю память микрокода, а также идентифицировать места внедрения информации. При возможности внесения изменений в устройство и наличия у него порта вывода предлагается создать режим считывания водяного знака, в котором команды переходов исполняться не будут, при этом во внедрённых участках поместить инструкции вывода частей водяного знака в порт.

Очевидным плюсом такого подхода является то, что таким образом можно внедрить практически неограниченное количество данных, и отсутствие или минимальное изменение управляющего устройства. Но отрицательной стороной подхода является то, что он увеличивает требования по объёму ПЗУ для устройства.

V. Внедрение в ПЗУ микрокода на уровне LUT

Для хранения микропрограммного кода могут применяться различные типы памяти. В случае если это ROM-память, строящаяся на основе массивов комбинационной логики, можно применить метод внедрения информации в комбинационную логику FPGA. Комбинационная логика в FPGA реализуется на Look-Up-Table(LUT) блоках, которые являются аппаратными реализациями таблиц истинности логических функций, описанных в текстовых HDL–описаниях. В составе CLB–блоков FPGA имеется возможность использовать блоки с различным числом входов в зависимости от используемой модели устройства. При реализации с помощью одного такого LUT–блока функции с меньшим числом аргументов, чем число его входов, часть ресурсов этого блока остается невостребованной, и нет возможности эти ресурсы применить для других функций. Таким образом, эти незадействованные ресурсы LUT–блока могут быть использованы для сохранения водяного знака [1]. При внедрении информации на уровне LUT имеется возможность не только использовать избыточные ресурсы LUT, но и расширять массив LUT для внедрения большего количества информации.

VI. Защита внедрённых в микрокод данных

При использовании собственного устройства управления и набора команд, злоумышленнику для определения и удаления внедрённой информации понадобится анализ построения и функционирования устройства. В то время как при использовании общеизвестного устройства с МПУ, например OpenRISC или Xilinx PicolBlaze, злоумышленник сможет легко определить внедренную информацию и модифицировать её также как и микропрограмму. Для защиты внедрённой информации предлагается использовать шифрование микрокода, а дешифрирование инструкций производить на лету, непосредственно при выборке. Следует учитывать следующие требования: максимальная скорость дешифрирования для минимального воздействия на выборку инструкций, изменение порядка выполнения инструкций в зависимости от данных (наличие условных переходов), минимизация аппаратных затрат на дешифрирование.

1. Иванюк, А. А. Проектирование встраиваемых цифровых устройств и систем /А. А.Иванюк //Издательство Бестпринт, 2012. – С. 262–265.
2. PicoBlaze 8-bit Embedded Microcontroller User Guide [Electronic resource] / Xilinx, Inc, 2004-2005, 2008-2011 – Mode of access: www.xilinx.com/support/documentation/ip_documentation/ug129.pdf – Date of access: 05.09.2014.
3. OpenRISC 1000 Architecture Manual [Electronic resource] / OPENCORES.ORG 2014 – Mode of access: https://raw.githubusercontent.com/openrisc/doc/master/openrisc-arch-1.1-rev0.pdf – Date of access: 05.09.2014.
4. AVR Microarchitecture [Electronic resource] / Prof. Ben Lee – Mode of access: http://web.engr.oregonstate.edu/ sinky/teaching/4.AVR_Microarchitecture.pdf – Date of access: 05.09.2014.
5. Huffman D. A. A method for the construction of minimum-redundancy codes / D. A. Huffman // Proceedings of the I.R.E., Sept. 1952, pp. 1098–1102