ТЕСТИРОВАНИЕ СНК NVIDIA TEGRA 2 И MICROSOFT WINDOWS EMBEDDED COMPACT 7. Часть 2 [1]

АРТЁМ СТОЛЯРОВ, ПАВЕЛ САВЫГИН, AXONIM Devices, Microsoft Embedded Partner

В предыдущей части статьи мы описывали систему-на-кристалле NVidia Tegra 2, функциональные блоки и тестировали блок кодирования/декодирования видео AVP. Сегодня разговор пойдёт о тестировании Windows Embedded Compact 7 на базе данной СнК и о применении новых технологий Windows Embedded Compact 7 при разработке ресурсоёмких приложений.

Для разработки сложных систем, требующих реального времени выполнения алгоритма с минимальными задержками, необходима ОС реального времени и, конечно, оптимизированный пакет аппаратной поддержки. Надо отметить, что сложные современные системы требуют мощных вычислительных ресурсов для работы специализированных алгоритмов, и не стоит забывать, что параллельно должны работать интерфейсы связи (WiFi, Bluetooth, Ethernet, и другие), а возможно потребуется и отображение текущей информации на ЖКИ-экране со временем обновления не менее 30 кадров в секунду.

С появлением многоядерных процессоров появились дополнительные возможности по разгрузке ядра, на котором производятся расчеты, от рутины, связанной с интерфейсной частью. Однако надо вернуться к самой ОС, т.к. она предоставляет основной инструмент для работы с ресурсами процессора. Необходимо описать особенности ОС Windows Embedded Compact 7, которые позволяют создавать на ее базе системы реального времени. В первую очередь это планировщик с отличным от настольных систем Windows алгоритмом работы, а также архитектура подсистемы обработки прерываний. Принцип работы планировщика в многопоточной среде Windows Embedded CE достаточно прост и основывается на приоритетах. Для каждого потока (в контексте каждого процесса может быть запущено несколько потоков) задается значение приоритета: от 0 — наивысший приоритет до 255 — наименьший приоритет. По умолчанию поток имеет значение приоритета, равное 251. В соответствии со значением приоритета готовых к исполнению потоков в каждый момент времени исполняется поток с наименьшим его значением. Этот простой принцип позволяет настроить систему для обеспечения детерминизма исполнения.

Другим, не менее важным показателем систем реального времени, является задержка при обработке прерываний. Обработка прерываний в Windows Embedded CE состоит в общем случае из двух этапов. Первый этап заключается в детектировании источника прерывания и выполняется в невытесняемом (non-preemptive) режиме работы системы. В соответствии с этим работа первого этапа может быть прервана только в случае возникновения более высокоприоритетного прерывания — другие потоки системы в этот промежуток времени не исполняются. После того как будет определен источник прерывания, система устанавливает связанное с ним событие. Установка события служит сигналом для второй части, выполняющей непосредственную обработку и являющейся для системы обычным потоком. Таким образом, задержка при обработке прерывания состоит из двух частей. Если первая составляющая зависит исключительно от аппаратной части, ее архитектуры и производительности, то задержка до запуска потока обработки прерывания помимо характеристик аппаратной платформы определяется тем, как долго система находится в невытесняемом режиме и тем, как настроены значения приоритетов других потоков в системе.

На рисунке 1 показан общий пример процедуры детектирования прерывания, где ISR (процедура детектирования источника прерывания), IST (поток обработки прерывания), OAL (слой абстракции ядра от конкретной платформы) являются частью BSP. Для тестирования задержек в поставке средств

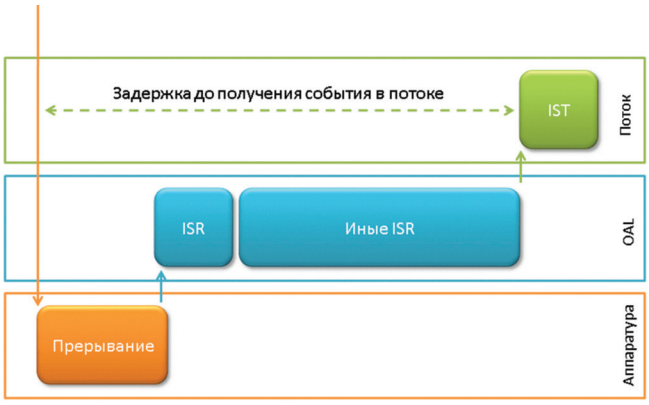

Рис. 1. **Примерный вид обработки прерывания до получения события в потоке**

[1] Первая часть статьи опубликована в «ЭК» 2, 2012.

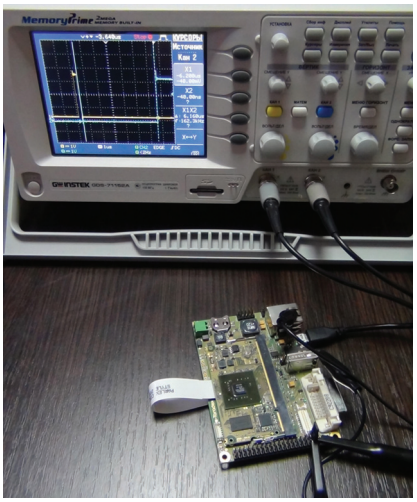

Рис. 2. **Тестовая система**

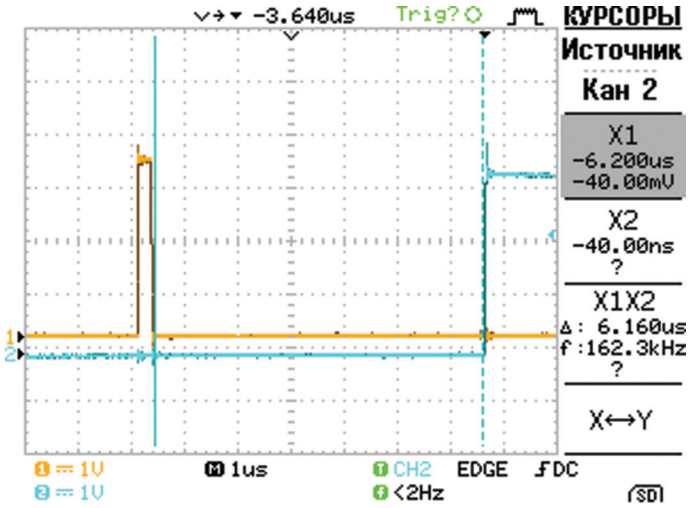

Рис. 3. **Осциллограмма задержки получения события от внешнего прерывания**

разработки Windows Embedded CE есть набор утилит. Поскольку принцип работы этих утилит основан на считывании показаний системного таймера, то полученные при их помощи результаты служат исключительно для предварительной оценки. Для тестирования задержек обработки прерываний в данной системе с NVidia Tegra2 (см. рис. 2) использовался аппаратный метод. Суть этого метода заключается в измерении задержек с использованием внутреннего генератора и вводов/выводов общего назначения (GPIO). Для реализации данного метода была разработана программа, настраивающая периферию PWFM в Tegra2 для генерации сигналов с заданным периодом и скважностью, а для захвата прерывания и сигнализации о получении события в пользовательском потоке использовались вводы/выводы общего назначения GPIO.

Для разработки программы была использована среда Visual Studio 2005 SP1, проект создан на основе Microsoft Windows Mobile 6 Professional SDK с использованием инструкций ARMv5. Конфигурация — Release с оптимизацией O2.

Была измерена задержка на установку и сброс состояния GPIO, которая составила 384 нс. Чтобы избавиться от влияния задержек на GPIO, решено было использовать вход GPIO_B5 как источник прерывания по заднему фронту, а выход GPIO_A6 — как сигнал получения события в потоке по переднему фронту. Поэтому задержки на GPIO взаимно вычитаются, и время между задним фронтом импульса PWFM и передним фронтом сигнала GPIO_A6 показывает, насколько долго задерживается момент получения события в пользовательском приложении. На рисунке 3 показана осциллограмма, на которой канал 1 (оранжевый цвет) показывает выход PWFM импульса для генерации прерывания, и канал 2 (голубой цвет) показывает состояние выхода GPIO_A6, который устанавливается в «1» после получения события в потоке, ожидающем событие.

Как видно из рисунка 3 измеренная задержка составляет ~6,2 мкс и остаётся неизменной даже при запуске многопоточной задачи. При тестировании был отключен модуль динамической подстройки частоты (DVFS) и установлена постоянная частота ARM ядер — 1 ГГц. Из проведённого эксперимента следует вывод о том, что данная ОС вместе с СнК NVidia Tegra2 и BSP от NVidia отлично подходит для построения систем жесткого реального времени.

Следующий немаловажный элемент ОС Windows Embedded Compact 7 — это готовое API (кстати, очень схожее с настольными ОС) для управления потоками в системах с симметричной многопроцессорной архитектурой. В такой системе потоки могут параллельно выполняться на нескольких процессорах. Общая идеология переключения задач в таких системах имеет ряд особенностей. В Windows Embedded Compact 7 поток имеет дополнительный атрибут — маску процессоров (affinity). Маска определяет подмножество микропроцессоров, на которых может исполняться поток. Этот атрибут задаётся функцией CeSetThreadAffinity. Маска может быть задана сразу для всех потоков, принадлежащих одному процессу. Для этого используется функция CeSetProcessAffinityMask. В случае переноса уже готового приложения на новую платформу на базе Windows Embedded Compact 7 и NVidia Tegra2 можно установить маску процессоров (affinity) при работающем приложении, используя готовую утилиту от Toradex под названием Colibri Monitor (см. рис. 4).

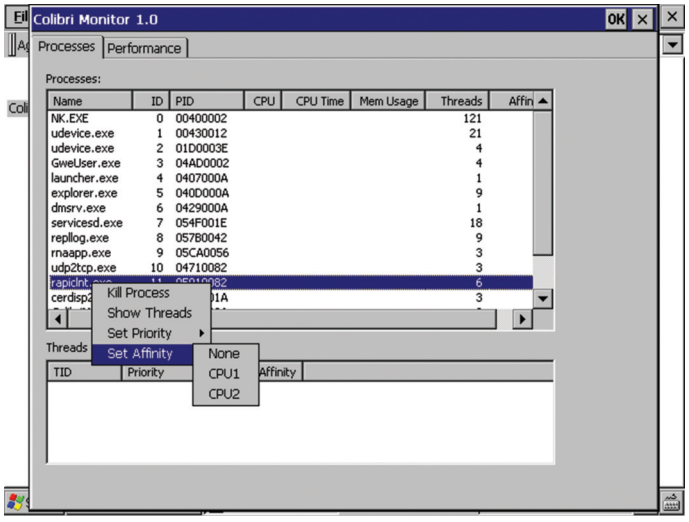

Рис. 4. **Утилита Colibri Monitor**

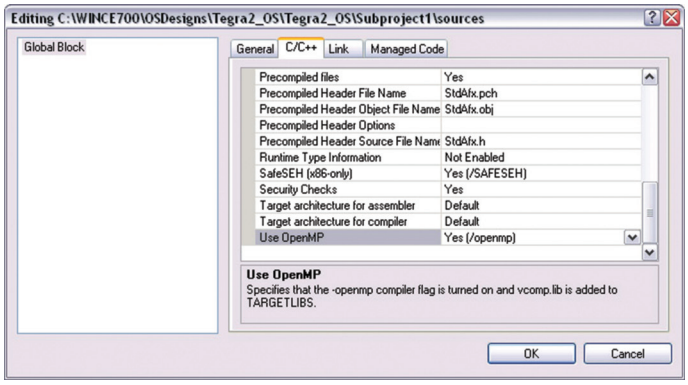

Рис. 5. **Окно свойств подпроекта в среде Platform Builder 7.0**

Рис. 6. **Комплексные решения на базе Windows Embedded**

Надо отметить, что с появлением систем с симметричной многопроцессорной архитектурой у разработчиков возникла необходимость рационально использовать все ядра процессора. В случае сложных либо многоитерационных процессов ОС Windows Embedded Compact 7 имеет возможность запуска приложения с поддержкой замечательного стандарта OpenMP. Данный стандарт почти полностью совместим с настольной версией OpenMP за исключением нескольких моментов (отсутствие поддержки прагм omp dynamic и omp threadprivate). OpenMP реализует параллельные вычисления с помощью многопоточности, в которой главный (master) поток создает набор подчиненных (slave) потоков, и задача распределяется между ними. Предполагается, что потоки выполняются параллельно на машине с несколькими процессорами (количество процессоров не обязательно должно быть больше или равно количеству потоков).

Задачи, выполняемые потоками параллельно, также как и данные, требуемые для выполнения этих задач, описываются с помощью специальных директив препроцессора соответствующего языка — прагм. Ниже приведены примеры программ с использованием директив OpenMP. (В этой программе сложение данных массива (a и b) происходит параллельно двумя потоками, что сокращает время работы алгоритма почти в 2 раза, по сравнению с выполнением данного кода на одном ядре. Как следствие, при появлении многоядерных (3 и более ядер) СнК прирост производительности увеличится соответственно):

```
#include <stdio.h>
#include <omp.h>

#define N 100

int main(int argc, char *argv[])
{
 float a[N], b[N], c[N];
 int i;

 // установить число потоков в 2
 omp_set_num_threads(2);

 // инициализируем массивы
 for (i = 0; i < N; i++)
 {
  a[i] = i * 1.0;
  b[i] = i * 2.0;
 }
 // вычисляем сумму массивов
 #pragma omp parallel shared(a, b, c) private(i)
 {
 #pragma omp for
  for (i = 0; i < N; i++)
   c[i] = a[i] + b[i];
 }
 printf («%f\n», c[2]);
 return 0;
}
```

Разработать приложения можно, добавив подпроект в Platfrom Builder 7.0 (см. рис. 5) и включив поддержку OpenMP в свойствах подпроекта. Хорошие примеры систем жесткого реального времени с применением Windows Embedded Compact 7 демонстрирует компания Beckhoff's: Print Mark Demo, Schtick pendulum и другие.

СнК компании NVidia показала себя с наилучшей стороны, открыв возможности для реализации различных устройств, требующих значительных вычислительных ресурсов. Линейка Tegra от NVidia, на наш взгляд, открывает большие перспективы. Появление модулей на базе четырёхъядерного СнК Tegra3 с поддержкой памяти DDR3 позволяет решать задачи, с которыми мог справиться только настольный ПК. На рисунке 6 показано комплексное решение с применением различных ОС из Windows Embedded канала.

Windows® Embedded представляет портфель платформ, инструментов и технологий, способствующих преодолению технологических проблем. OEM-изготовители различного оборудования выбирают Windows Embedded, чтобы сократить время выхода продуктов на рынок, повысить уровень доверия и поддержки, а также расширить возможности подключения к ИТ-сетям и различным системам. При поддержке множества разработчиков, партнеров и независимых поставщиков программных продуктов Windows Embedded помогает OEM-изготовителям быстро и эффективно выводить новейшие продукты на рынок.

ЛИТЕРАТУРА

1. Benchmarking Real-time Determinism in Microsoft Windows CE//http://msdn.microsoft.com/en-us/library/ms836535.aspx.

2. П. Белевский. Windows Embedded CE 6.0 в системах реального времени//ИСУП, № 2(26), 2010.

3. C/C++ Libraries for Windows Embedded Compact (Windows Embedded Compact 7)// http://msdn.microsoft.com.

4. Kurt Kennett. Symmetric Multiprocessing in Windows Embedded Compact 7// Windows Embedded Core Team, Microsoft Corporation.