XCore: A Formally Constructed Instruction Set Architecture Definition

Stephen Wright

Department of Computer Science, University of Bristol, UK*

stephen.wright@bris.ac.uk

*Funded by EPSRC Grant EP/H500316/1

Date	***Issue***	***Change***
26/9/2011	1	Original release

TABLE OF CONTENTS

1. Introduction 4
2. Archive Description 4
3. Host Environment 5
3.1.1. Operating System 5
3.1.2. Working Directory 5
4. External Tools Installation 5
4.1 Rodin 6
4.1.1. Java Installation 6
4.1.2. Font Installation 7
4.1.3. Platform Installation 7
4.1.4. Setup 8
4.1.5. Editor Installation 8
4.1.6. Prover Installation 8
4.1.7. Relevance Filter Installation 8
4.2 Microsoft Visual C++ 5.0 9
4.3 Eclipse SDK 9
4.3.1. Installation 9
4.3.2. Setup 9
5. XCoreIsa Installation 9
5.1 XCore Event-B Model 10
5.2 Execution Environment 10
5.3 Target Development Tools 10
5.4 B2C 11
5.4.1. Rodin Source Code 11
5.4.2. B2C Source Code 11
6. Demonstration Procedure 12
6.1 Demo Executable Build 12
6.2 Server Startup 12
6.3 Demo Program Execution 12
7. Build Procedure 16
7.1 Building Support Server 16
7.2 Debugging B2C 16
7.3 Installing B2C 17
7.4 Generating VM Source Code 17
7.5 Building VM 17

1. Introduction

The XCore microprocessor is an embedded device developed by XMOS Ltd of Bristol, UK. The Instruction Set Architecture (ISA) contains a range of typical instructions such as control-flow, register-to-register calculation and memory access, but also provides support for efficient multi-threaded programming and parallelism with other devices via fast interconnects. Support for these features is integrated into the ISA of the XCore, in contrast to a conventional memory-mapped device approach. This greatly improves run-time performance, at the cost of introducing specialist instructions to the ISA, which comprises 170 instructions. The ISA contains instructions of both two and four byte length, and implements a very compact encoding scheme.

The ISA has been embodied in the XS1-G4 XCore processor (a four-core device that can run up to 32 real time tasks), and theXS1-L1 (a single core device that can run up to 8 real time tasks). Although the XCore is used as the basis of multi-core devices, the individual cores are entirely symmetric and linked only via asynchronous communication links. Thus the single ISA model is appropriate to each individual core. XMOS provides a complete software development tool-chain that supports C, C++, and XC (a variant of C developed to best exploit the XCore architecture). The XCore is general-purpose and has been exploited in a range of different markets, including audio, display, communications, robotics and motor control.

A pdf document giving an informal specification of the ISA is available at: https://www.xmos.com/download/public/The-XMOS-XS1-Architecture%281%29.pdf.

As part of a Bristol University Knowledge Transfer Secondment (KTS) (Grant EP/H500316/1), a formal model of the complete ISA was constructed in the Event-B notation, using the Rodin toolset. This project applied and extended the Event-B and RODIN based techniques for Instruction Set Architecture (ISA) analysis, developed by Dr Stephen Wright during his doctoral research, to an industrial setting. To that end, XMOS Ltd hosted Dr Wright in the period October 2010 to October 2011.

This document gives complete instructions for installation, building and running of all models, tools, virtual machines and test programs developed during the project.

2. Archive Description

The accompanying archive *XCoreBundle.zip* contains a detailed Event-B model capable being automatically translated to C via a supplied Rodin translation plug-in. A server providing program load and text output functions to the VM via a common protocol is provided. A compiler targeting the ISA is available for free download from the XMOS website and is therefore not included in the archive.

The archive can be extracted using the generic zip tool supplied as part of Windows.

The archive contains:

1) *XCoreIsa_<date>.zip*, a Rodin 2.2 model of the XCore ISA.
2) *b2c_<date>..zip*, an Eclipse Java SDK project containing the Event-B to C translation tool "B2C".
3) *Execution_<date>.zip*, Visual Studio 5.0 projects, source code and batch files to compile the VM, and server source code to Windows. An XC/C test-suite and assembler bootstraps for compilation is included.

3. Host Environment

3.1.1. Operating System

The only currently supported host operating system is *Windows 7*, although porting to other platforms is practical for all the components.

3.1.2. Working Directory

In order to allow the user to locate all working files in a convenient location, all paths are specified relative to a virtual X-drive. This is configured by the following procedure:

- Create a working directory on the C-drive (for example C:\Development).
- Create a new file named Startup.bat in directory C:\.
- Use a text editor to insert the text "subst X: C:\ Development" into the file.
- Double-click C:\ Startup.bat to configure the X-drive.
- Right-click on C:\ Startup.bat and select "Create shortcut".
- Drag the shortcut into the Start\All Programs\Startup directory.

4. External Tools Installation

The flow of information through the various required tools is summarized in Figure 1.

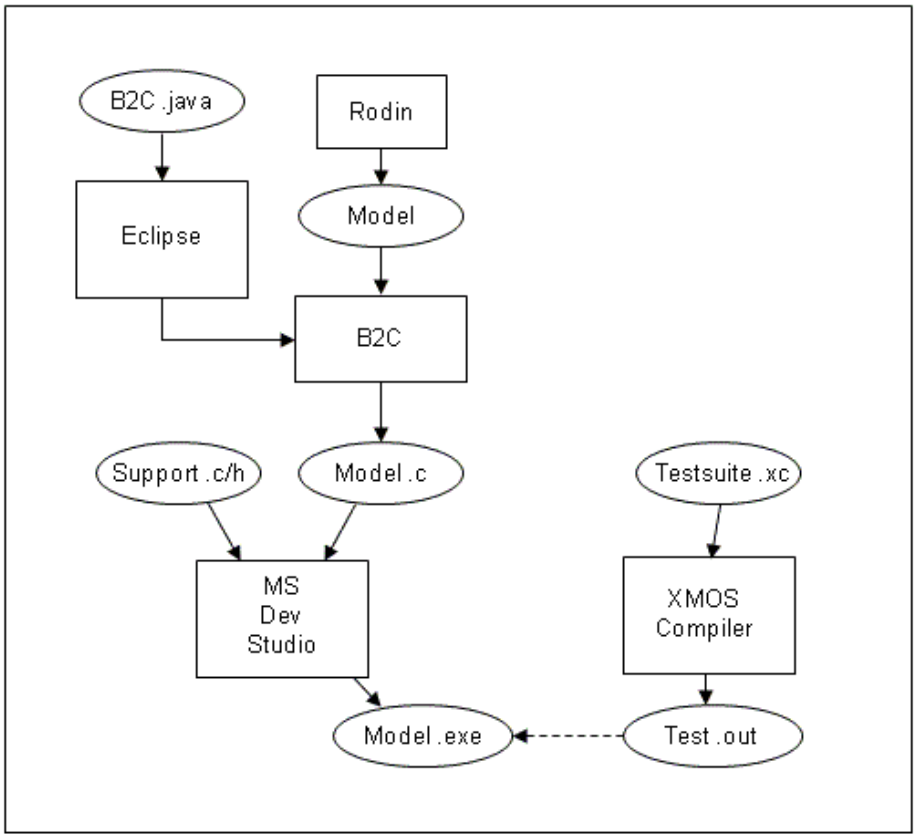

Figure 1: Development Tool Flow

The *Eclipse SDK* is used to compile Java source code to the *B2C* plug-in for *Rodin*. *Rodin* is then used to browse and edit the model and generate a C output file via *B2C*. This *B2C*-generated XCore C file is compiled along with various supporting C files by *Visual C++* to create a VM as a *Windows* console executable. The XMOS XCore tool chain is then used to compile the demonstration C and XC files to an XCore binary executable, allowing it to be executed on the *Event-B* derived XCore VM.

4.1 Rodin

Rodin 2.2 is needed for browsing of the model and its discharged proof obligations, and automatic generation of C VM source code. Note that the following procedures assume the use of *Windows 7* and the *Google Chrome* browser.

4.1.1. Java Installation

Rodin 2.2 requires installation of the *Java Runtime Environment 7*. This is achieved by the following procedure:

- Navigate to http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html.

- Tick the “Accept License Agreement” box.
- Select “Windows x86” from the “Product/File Description” drop-down, and select “Continue”.
- Select “jre-7-windows-i586.exe”. This will cause the installation to be downloaded and executed.
- Select the “Install” button.
- Select “Finish” when installation is complete.
- Verify that a new directory *C:\Program Files\Java\jre7* has been created.

4.1.2. Font Installation

The mathematical font used by Rodin is installed by the following procedure:

- Navigate to http://sourceforge.net/project/showfiles.php?group_id=108850.
- Locate the “Font: Brave Sans Mono” line and select “Download” on that line.
- Select “BraveSansMono-Roman-0.12.ttf”.
- Select “Save” in the download wizard.
- Select C:\ for the save location and select “Save”.
- Close the download wizard.
- Open the Windows Control Panel (in Start).
- Select the Fonts folder.
- Choose File > Install New Font.
- Select C:\ as the folder in the installation wizard.
- Select “Brave Sans Mono (TrueType)”.
- Make sure the “Copy fonts to the Fonts” check box is selected.
- Select “OK”.
- Verify that a new entry “Brave Sans Mono (TrueType)” has appeared in the Fonts folder.

4.1.3. Platform Installation

The *Rodin 2.2* main platform is now installed by the following procedure:

- Navigate to http://sourceforge.net/projects/rodin-b-sharp/files/Core_Rodin_Platform/2.2/rodin-2.2-win32.win32.x86.zip/download.
- Select “Download RODIN” from the top of the page. You will be invited to download a file “rodin-2.2-win32.zip”.
- Select “Save” in the download wizard.
- Select the Windows Desktop as the download destination.
- Select “Save”.
- Close the download wizard.
- Open *rodin-2.2-win32.zip* by clicking on it on the Windows Desktop. It contains a single folder *rodin*.
- Drag and drop the *rodin* folder into the “C:\Program Files” folder.
- Rename the new folder “rodin” in “C:\Program Files” to “rodin 2.2”.

4.1.4. Setup

In order to guarantee sufficient memory allocation for building of XCoreIsa under Rodin, a startup script is installed by the following procedure:

- Navigate to folder *C:\Program Files\rodin 2.2*.
- Right-click on the directory and create a new file *StartRodin.bat*.
- Right-click on *StartRodin.bat* and select "Edit".
- Insert the following line into the file: "C:\Program Files\rodin 2.2\rodin.exe" –vm java.exe –vmargs –Xms1450m –Xmx1450m
- Right-click on the file and select "Create Shortcut".
- Drag the new shortcut to the Windows Desktop.
- Create a new folder *X:\RodinWorkspace*.
- Select the shortcut from the Desktop to start Rodin.
- Browse to *X:\RodinWorkspace* in the "Workspace Launcher" wizard.
- Make sure the "Use this as the default and do not ask again" check box is selected.
- Select "OK" to start *Rodin*. The *Rodin* "Welcome" tab will be displayed.
- Close the "Welcome" tab by selecting the close cross.

4.1.5. Editor Installation

Additional proving plug-ins are required if Proof Obligation reproving is required. These are installed by the following procedure:

- In the *Rodin* environment, select "Help > Install New Software…". The "Available Software" wizard will appear.
- Select the "Work with:" drop-down and select "Camille - http://www.stups.uni-duesseldorf.de/camille_updates".
- Tick the "Camille Text Editor" box and click "Next".
- The plug-in will be installed. This may take several minutes. You will be prompted to "accept terms in the license agreement" during the process.

4.1.6. Prover Installation

Additional proving plug-ins are required if Proof Obligation reproving is required. These are installed by the following procedure:

- In the *Rodin* environment, select "Help > Install New Software…". The "Available Software" wizard will appear.
- Select the "Work with:" drop-down and select "Atelier B Provers - http://bmethod.com/update_site/atelierb_provers".
- Tick the "Atelier B Provers" box and click "Next".
- The plug-in will be installed by the usual process.

4.1.7. Relevance Filter Installation

Additional proving plug-ins are required if Proof Obligation reproving is required. These are installed by the following procedure:

- In the *Rodin* environment, select "Help > Install New Software…". The "Available Software" wizard will appear.

- Select the “Work with:” drop-down and select “Rodin - http://rodin-b-sharp.sourceforge.net/updates”.
- Expand the “Prover Extensions” line.
- Tick the “Relevance Filter” box and click “Next”.
- The plug-in will be installed by the usual process.

4.2 Microsoft Visual C++ 5.0

Microsoft Visual C++ 5.0 is required for compilation of the VMs and support server. Installation is performed by inserting of a *Visual C++ 5.0* installation CD and following the automatically executed installation procedure.

4.3 Eclipse SDK

The *Eclipse Software Development Kit* is required for compilation of the *B2C* plug-in for Rodin. This is installed by the following procedure:

4.3.1. Installation

- Navigate to http://www.eclipse.org/downloads.
- Locate the line “Eclipse IDE for Java Developers” and select the “Windows” link from this.
- Select a mirror site from those presented. The “File Download” wizard for a *.zip* file will appear.
- Select “Save” and select the Windows Desktop as the destination directory.
- Select “Save”. The download will commence.
- Open the *.zip* file by clicking on it. It contains a single folder *eclipse*.
- Drag and drop the eclipse folder into the *C:\Program Files* folder.
- Right-click on the file and select “Create Shortcut”.
- Drag the new shortcut to the Windows Desktop.

4.3.2. Setup

- Create a new folder *X:\EclipseWorkspace*.
- Select the shortcut from the Desktop to start Eclipse.
- Browse to *X:\EclipseWorkspace* in the “Workspace Launcher” wizard.
- Make sure the “Use this as the default and do not ask again” check box is selected.
- Select “OK” to start Eclipse. The Eclipse “Welcome” tab will be displayed.
- Close the “Welcome” tab by selecting the close cross.

5. XCoreIsa Installation

Source code for all XCore ISA components is supplied in the *XCoreBundle.zip* archive described in 2. This file should be downloaded and saved to the Windows Desktop.

5.1 XCore Event-B Model

Importing of the XCore model into Rodin is required for inspection and modification. This is achieved by the following procedure:

- Extract the *XCoreIsa_<date>.zip* file from *XCoreBundle.zip* to the Windows desktop.
- Run *Rodin 2.2* using the procedure described in 4.1.4.
- Select "File > Import" in the Rodin environment. The "Import" wizard will appear.
- Expand the "General" line and select the "Existing projects into workspace" line.
- Select "Next".
- Select the "Select Archive" radio button and browse to *XCoreIsa_<date>.zip* on the Windows Desktop.
- Select "Open" and then "Finish". The import will commence. The model will be automatically rebuilt: this will take several minutes.

5.2 Execution Environment

The Execution bundle is required for execution of the XCore VM. Importing this is achieved by the following procedure:

- Extract the *Execution_<date>.zip* file from *XCoreBundle.zip* to the Windows desktop.
- Open *Execution_<date>.zip* by clicking on it. It contains a single folder *Execution*.
- Drag and drop the *Execution* folder into the "X:\" folder.
- Open the Visual C++ environment.
- Select "File > Open Workspace". The "Open Workspace" wizard will appear.
- Browse to *X:\Execution\WindowsProj* and select *Execution.dsw*. The component projects of the workspace will be loaded.

5.3 Target Development Tools

The *XMOS XCore* development tools are required for building the XCore test executable that is to be executed by the VM. Importing these is performed by the following procedure:

- Navigate to http://www.xmos.com/products/development-tools.
- Use the "register" function on this page to create an XMOS account.
- Use the "login" function on this page to enter the XMOS account.
- Click on "Free Download" for the "Microsoft Windows XP SP3 (32-Bit)" line on this page (this installation is also compatible with Windows 7).
- Run the installation wizard that is downloaded by this link.
- On completion of the installation, a shortcut to "XMOS Command Prompt (<version>)" will appear on the Windows Desktop.
- Right click on the shortcut and select "Properties".
- Edit the "Start in:" field to "X:\Execution" and click "OK".

5.4 B2C

Importing of the *B2C* plug-in project into the Eclipse SDK is required for insertion into *Rodin*, inspection and modification.

5.4.1. Rodin Source Code

The B2C source code has dependencies within the Rodin core platform source, so this must be installed. This is achieved by the following procedure:

- Navigate to http://sourceforge.net/projects/rodin-b-sharp/files/Core_Rodin_Platform/2.2/.
- Select "rodin-2.2.r11814-sources.zip" within this. The "File Download" wizard will appear.
- Select the Windows Desktop as the destination directory and select "Save". The download will commence.
- Run *Eclipse* using the procedure described in 4.3.2.
- Select File > Import in the Eclipse environment. The "Import" wizard will appear.
- Expand the "General" line and select the "Existing projects into workspace" line.
- Select "Next".
- Select the "Select Archive" radio button and browse to "rodin-2.2.r11814-sources.zip" on the Windows Desktop.
- Select "Open" and then "Finish". The import will commence. The source code will be automatically rebuilt, yielding several warnings.

5.4.2. B2C Source Code

The B2C source code has may now be installed. This is achieved by the following procedure:

- Extract the *b2c_<date>.zip* file from *XCoreBundle.zip* to the Windows desktop.
- Select File > Import in the Eclipse environment. The "Import" wizard will appear.
- Expand the "General" line and select the "Existing projects into workspace" line.
- Select "Next".
- Select the "Select Archive" radio button and browse to b2c.zip on the Windows Desktop.
- Select "Open" and then "Finish". The import will commence. The source code will be automatically rebuilt: this will take several minutes.
- Select "Open" and then "Finish". The import will commence. The plugin will be automatically rebuilt.

6. Demonstration Procedure

6.1 Demo Executable Build

The XCore demo program source code is given in *Testsuite.c* in *X:\Execution\TargetTestsuite*. The file may also be accessed via the *Testsuite files* project in the *Visual C++* workspace imported in 5.2.

- Run the XMOS development tools command prompt by clicking the Desktop link created in Section 5.3.
- Change directory to *X:\Execution\Scripts* (with the DOS command "cd X:\Execution\Scripts").
- Run the DOS batch file *BuildXCTestsuite.bat* (with the command "BuildXCTestsuite").
- Display directory *X:\Execution\TargetTestsuite* (with the command "dir X:\Execution\TargetTestsuite").
- Confirm that the timestamps for files *Testsuite.out* and *Testsuite.txt* have been updated.

6.2 Server Startup

The server provides executable loading and text output services for the executing VM, and must therefore be running before target execution is started. The procedure for starting it is as follows:

- Open a DOS console and change directory to *X:\Execution\Server\bin*.
- Run the server executable with the command "Server". The message "Server created OK." will appear.

6.3 Demo Program Execution

- Make sure the test suite executable has been compiled for the stack variant using the procedure in 6.1 and the server is running using the procedure in 6.2
- Change directory in the VM DOS console to *X:\Execution\BVirtualMachine\BXCoreBin*.
- Run the VM with the command "BXCore".
- Confirm that VM event instrumentation is displayed in the VM DOS console, and test suite output text is displayed in the Server DOS console.
- The console output for a successful run of the test-suite is as follows:

```
Client accepted on [124].
Loading [X:\Execution\TargetTestsuite\Testsuite.txt], searching for [_DoSyscall].
Found [.text] section.
Found [.init] section.
Found [.fini] section.
Found [.globcode] section.
Found [.gnu.linkonce.t.__call_exitprocs_impl] section.
Found [.rodata] section.
```

```
Found [.cp.rodata] section.
Found [.cp.const4] section.
Found [.cp.string] section.
Found [.ctors] section.
Found [.dtors] section.
Found [.dp.data] section.
Found [.dp.rodata] section.
Found [.eh_frame] section.
Found [.netinfo] section.
Found [.xtabranch] section.
Found [.xtacalltable] section.
Found [.xtaendpointtable] section.
Found [.xtalabeltable] section.
Found [.xtasystem] section.
Found [.xtathreadstart] section.
Found [.xtathreadstop] section.
Found [.xmosnote] section.
Found [.xmosnote] section.
Found [.xmosnote] section.
Found [.expr] section.
Found [.symtab] section.
Found [.strtab] section.
All object lines loaded.
Sending image: size[22705] breakpoint[00010eba]

[XCore ISA test]

[Basic output test]
Count[100]
Count[101]
Count[102]
Count[103]
Count[104]
Count[105]
Count[106]
Count[107]
Count[108]
Count[109]

[Logic test]
Or [0/0] result[0]
Or [1/2] result[3]
Xor [0/0] result[0]
Xor [0/1] result[1]
Xor [1/0] result[1]
Xor [1/1] result[0]
```

```
[Arithmetic test]
Mod [5/3] result[2]
Mod [-5/3] result[-2]
Mod [5/-3] result[2]
Mod [-5/-3] result[-2]

[Basic parallel threads test]
Task1 - 0
Task3 - 0
Task1 - 1
Task3 - 1
Task1 - 2
Task2 - 0
Task3 - 2
Task2 - 1
Task2 - 2
t1[3] t2[3] t3[3]

[Channel test]
Channel Tx[1000]
Channel Rx[1000]
Channel Tx[1001]
Channel Rx[1001]
Channel Tx[1002]
Channel Rx[1002]

[Streaming channel test]
Stream Tx[5000]
Stream Rx[5000]
Stream Tx[5001]
Stream Rx[5001]
Stream Tx[5002]
Stream Rx[5002]

[Port test]
Port Tx[6000]
Port Rx[6000]
Port Tx[6001]
Port Tx[6002]
Port Rx[6001]
Port Rx[6002]

[Timer event test]
Initial time [1622]
Timer triggered at time [2622]
Timer triggered at time [3622]
Timer triggered at time [4622]
```

```
Timer triggered at time [5622]
Timer triggered at time [6622]

[Channel event test]
Channel Tx[1000]
Channel Rx[1000]
Channel Tx[1001]
Channel Rx[1001]
Channel Tx[1002]
Channel Rx[1002]
Channel Tx[1003]
Channel Rx[1003]
Channel Tx[1004]
Channel Rx[1004]

[Combined channel/timer event test]
Channel Tx[1000]
Initial time [8060]
Channel Rx[1000]
Channel Tx[1001]
Channel Rx[1001]
Channel Tx[1002]
Channel Rx[1002]
Channel Tx[1003]
Channel Rx[1003]
Channel Tx[1004]
Channel Rx[1004]
Timer triggered at time[9060]
Timer triggered at time[10060]
Timer triggered at time[11060]
Timer triggered at time[12060]
Timer triggered at time[13060]

[Interrupt test]
Setting interrupt vector on port[00010100]
Port Tx[12345]
Interrupt detected
Port Rx[12345]
Interrupt test complete

[Exception test]
Setting exception vector.
Dividing-by-zero...
Exception detected
Instruction[f80447ec] Location[0001028a] Length[4]
Type[7] Data[0]
Exception handled.
```

[End of XCore ISA test]
Cannot read message header.
Cannot read packet.

7. Build Procedure

All source code for the Event-B model, B2C C auto-generator, XCore test-suite, and execution environment are provided in XCoreBundle.zip, in order to allow modification of any of these components. The following procedures enable rebuilding and testing after modification.

7.1 Building Support Server

All source code for the support server may be accessed and built via the *Server files* project in the *Visual C++* workspace described in 5.2. Rebuilding of the server executable used in 6.2 is achieved using the following procedure:

- If the server is running in a DOS console after the procedure in 6.2, make sure that it has been stopped by selecting that console and entering “Ctrl-C”.
- Open *Visual C++*.
- If the “Workspace” explorer is not already open in Visual C++, open it using “View > Workspace”.
- Locate the “Server files” project in the Workspace explorer.
- Right-click on the project and select “Clean (selection only)”.
- Right-click on the project and select “Build (selection only)”.
- Navigate to *X:\Execution\Server\bin* and confirm that the timestamp for file Server.exe has been updated.

7.2 Debugging B2C

- Open the Eclipse environment described in 4.3 and import the *B2C* source code using the procedure in 5.4.
- Open the "Debug Configurations..." window in the "Run" drop-down in the Eclipse environment.
- Right-click on the "Eclipse Application" entry in the box on the left and click "New". This creates a new launch configuration.
- Open out the "Eclipse Application" entries and select the newly created configuration (by default named "New_configuration (1)".
- Open the drop-down by the "Run a product" radio button and select "org.rodinp.platform.product". This informs Eclipse to use Rodin as the plug-in host.
- Select a workspace area on the X-drive via the "File System" button in the "Workspace Data" area.
- Select Run->Debug in the Eclipse environment, to run up the locally built Rodin, with B2C installed.

7.3 Installing B2C

Installation of a modified version of B2C into a release installation of Rodin is performed by the following procedure:

- In the Eclipse environment select “File > Export”.
- Expand the cross of “Plug-in Development” and select “Deployable plug-ins and fragments”.
- Select “Next”. The “Export” dialog will appear.
- Select only the “b2c (0.1.0)” check box and select *C:\Program Files\rodin 2.2* as the destination directory.
- Select “Finish”. Build and installation of the plug-in will commence.
- Open *Rodin* using the procedure in 4.1. If it was currently open, close it and re-open it.
- Confirm that a *B2C* icon and drop-down have been installed in the Rodin toolbar.

7.4 Generating VM Source Code

C source files are automatically generated from the XCore model via *B2C*, for compilation into the Event-B VMs, achieved using the following procedure:

- Open *Rodin* using the procedure in 4.1.
- Ensure that the XCore model has been imported using the procedure from 5.1.
- Ensure that *B2C* has been installed using the procedure in 7.2.
- Select the *B2C* icon in the *Rodin* environment. Instrumentation will be displayed in the *DOS* console started by the batch file used to start *Rodin*.
- Navigate to *X:\RodinWorkspace* and confirm that three new files *XCoreIsa.c*, *XCoreIsa.h* and *XCoreIsa.txt* have been generated.

7.5 Building VM

All source code for the prototype VMs may be accessed and built via the *BXCore files* project in the *Visual C++* workspace. Rebuilding of the VM executable used in 6.3 is achieved using the following procedure:

- Open *Visual C++*.
- Ensure that the file *XCoreIsa.c* has been generated using the procedure in 7.4.
- Locate the *BXCore files* project in the Workspace explorer.
- Right-click on the project and select “Clean (selection only)”.
- Right-click on the project and select “Build (selection only)”.
- Navigate to *X:\Execution\BVirtualMachine\BXCoreBin* and confirm that the timestamp for *BXCore.exe* has been updated.