How to create a simple ColdFire and Altera FPGA IOC (Draft)

W. Eric Norum

August 18, 2005

Contents

Chapter 1

Introduction

This tutorial presents a step-by-step series of operations for creating a simple EPICS application for an Arcturus uCDIMM ColdFire 5282 module attached to an Altera FPGA development kit. The following software and hardware components are assumed to be in place:

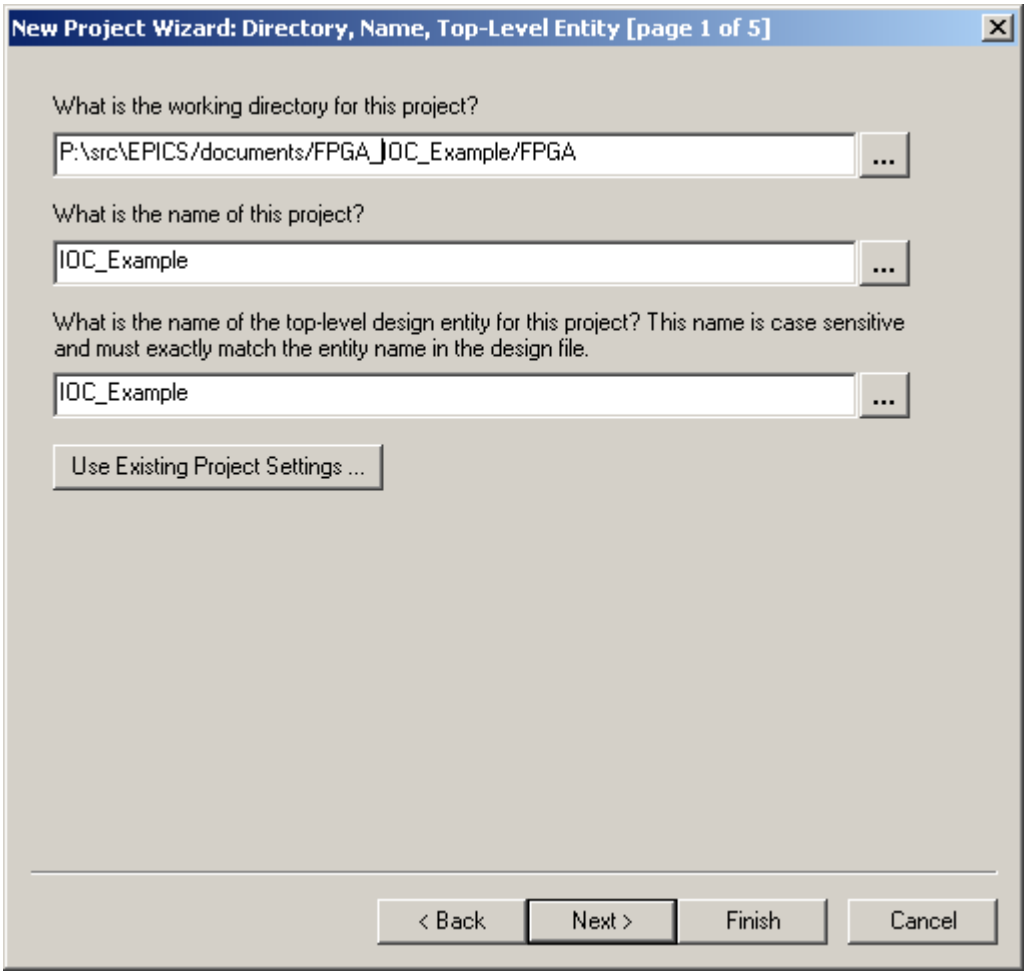
- Arcturus uCDIMM ColdFire 5282
- RTEMS with m68k tool chain and uC5282 board-support package
- EPICS version R3.14.7 or greater with RTEMS-uC5282 target
- Altera FPGA development kit with at least two sets of expansion prototype connectors.
- ColdFire/Stratix adapter card
- Quartus version 5.0 or greater and Altera SOPC builder
- ColdFire bridge SOPC component
- Console reset detect Quartus component (optional)

This tutorial is written for use with an Altera Stratix II DSP development kit. The changes required for use with other development kits are summarized in appendix B.

Chapter 2

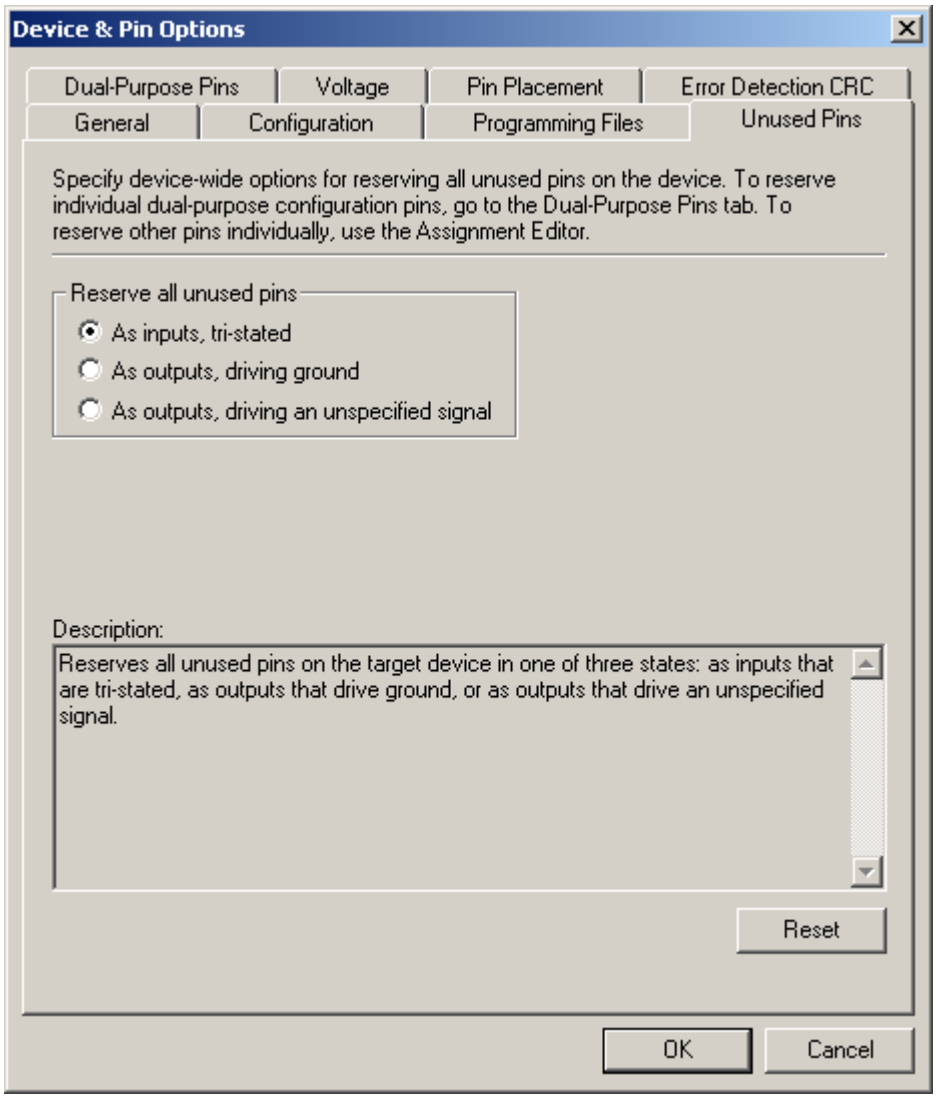
FPGA application

1. Use the Quartus ”New Project Wizard” to create a new project. I called the project `IOC_Example` with same name for the top-level entity:

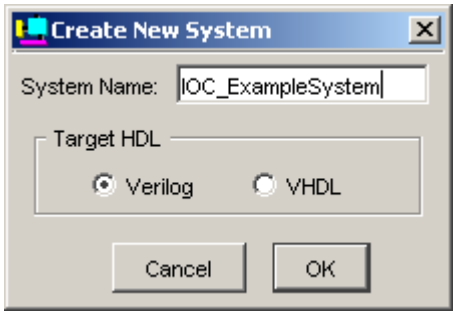


2. Ensure that unused pins are treated as inputs (Assignments→Device…, Device & Pin Options, Unused Pins tab). Not all the ColdFire signals are used by this example and Quartus helpfully grounds all the corresponding

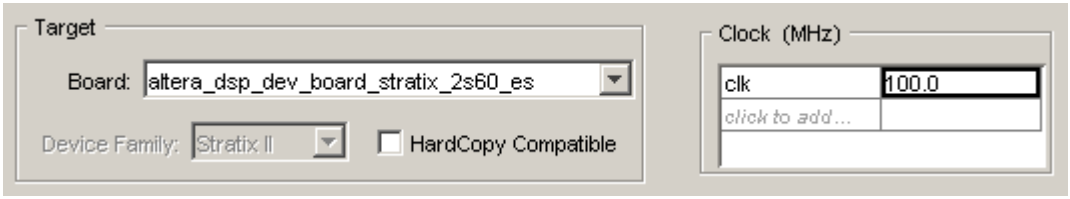
pins if this step is omitted!



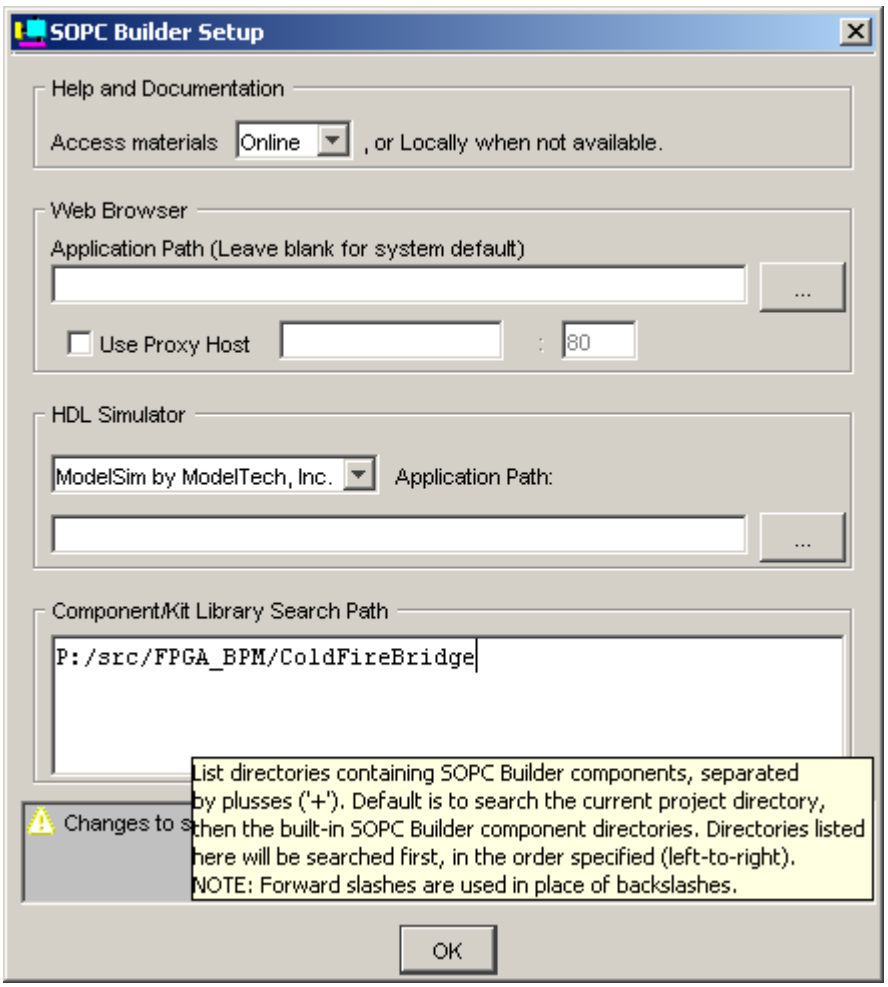
3. Start the SOPC Builder (Tools→SOPC Builder...) and create a new SOPC system. I called the SOPC system `IOC_ExampleSystem`:



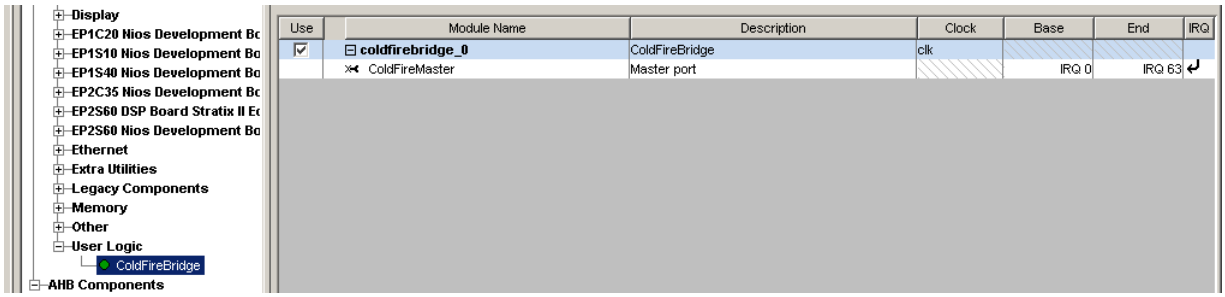
4. Set the SOPC Board and Clock parameters:



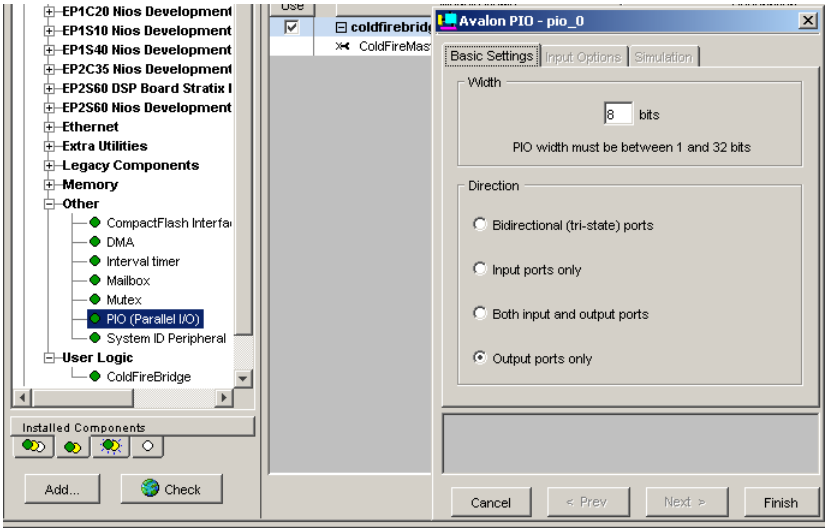
5. If you have not done so in a previous SOPC project, add the directory containing the ColdFire Bridge SOPC component to the list of directories which the SOPC Builder will search (File→SOPC Builder Setup...). Note that this list of directories applies to SOPC Builder sessions in **all** projects. Note also that changes to this list do not take effect until the SOPC Builder is next started. When the SOPC Builder is started and searches the directory containing the ColdFire Bridge SOPC component it will add the ColdFire Bridge component to the list of User Logic components.



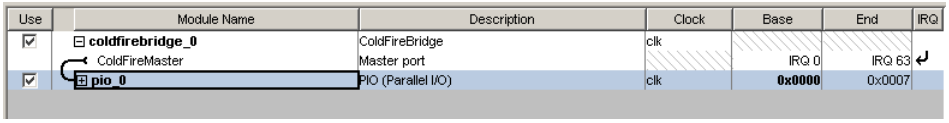
6. Double-click the ColdFire Bridge component to add it to the SOPC design:



7. Add an 8-bit output port to the SOPC design:



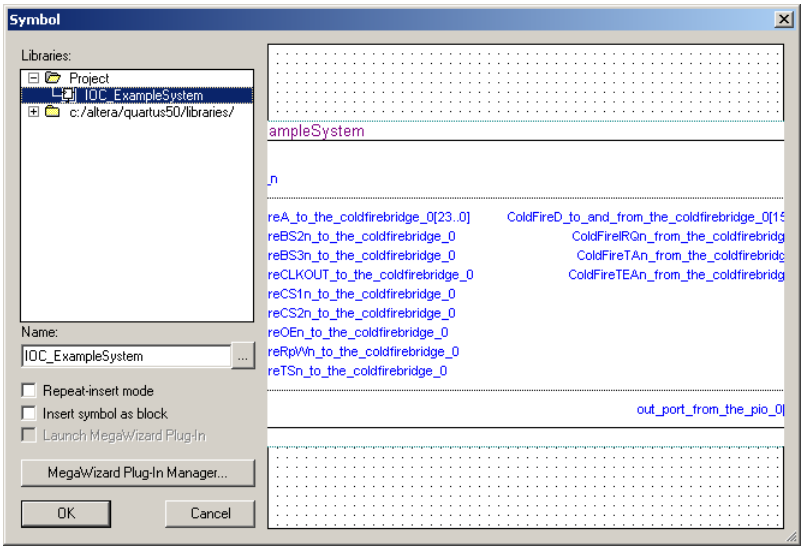
The SOPC components and connections should then be:



8. Click the `Generate` button to create the SOPC block.

9. Create the top-level entity design file. In the Quartus window select File→New, create a new Block Diagram/Schematic File and save it as `IOC_Example`.

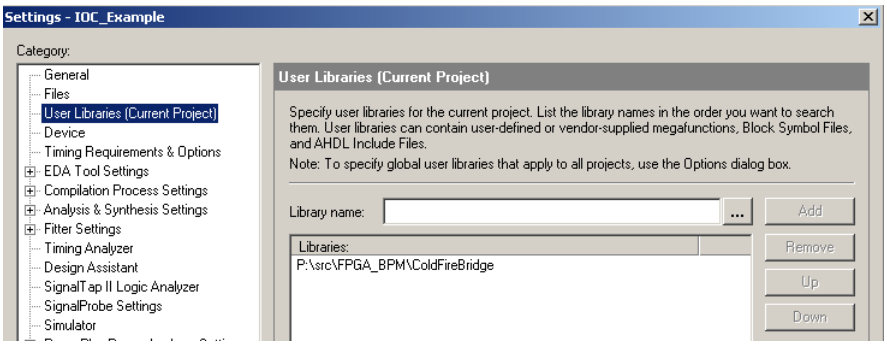
10. Double-click in the `IOC_Example.bdf` design window and add the `IOC_ExampleSystem` SOPC block to the design:



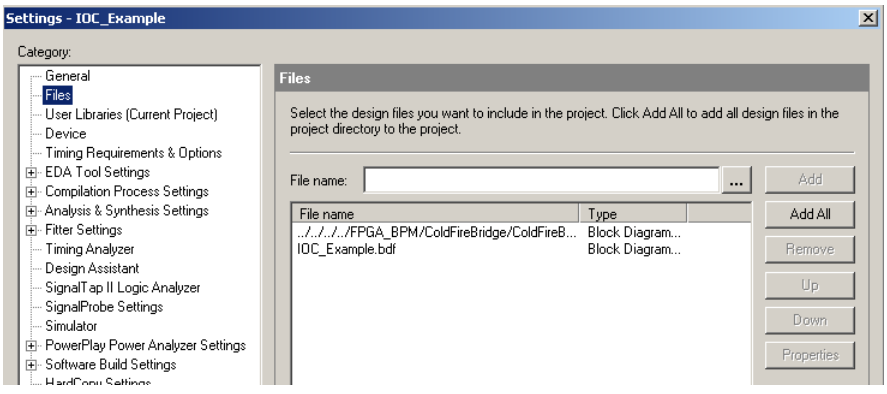
11. Add I/O pins and assign them to the appropriate FPGA pins. The complete system should then appear as shown on the following page.

12. Make some changes to some settings (Assignments→Settings...):

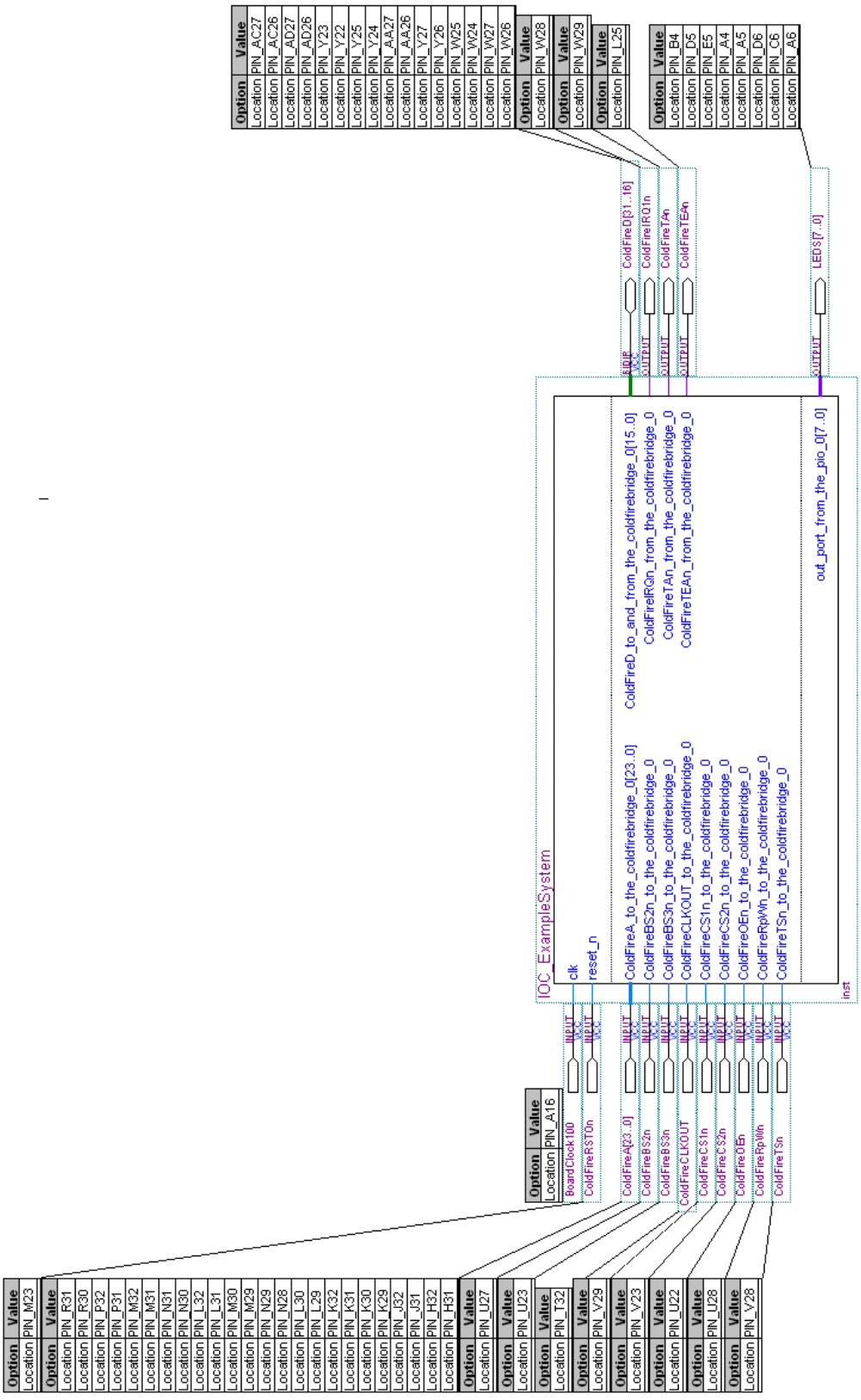
    User Libraries (Current Project) Add the path to the directory containing the ColdFire Bridge SOPC component:



    Files Quartus II has a bug which causes it to generate a bad system unless the `ColdFireBridge.bdf` design file is explicitly mentioned in the list of application files:



13. Compile the project (you'll see lots of warnings...) and load it into the FPGA.


IOC_ExampleSystem
clk
reset_n
ColdFireA_to_the_coldfirebridge_0[23..0]
ColdFireBS2n_to_the_coldfirebridge_0
ColdFireBS3n_to_the_coldfirebridge_0
ColdFireCLKOUT_to_the_coldfirebridge_0
ColdFireCS1n_to_the_coldfirebridge_0
ColdFireCS2n_to_the_coldfirebridge_0
ColdFireOEn_to_the_coldfirebridge_0
ColdFireRpWn_to_the_coldfirebridge_0
ColdFireTSn_to_the_coldfirebridge_0
ColdFireD_to_and_from_the_coldfirebridge_0[15..0]
ColdFireIRQn_from_the_coldfirebridge_0
ColdFireTAn_from_the_coldfirebridge_0
ColdFireTEAn_from_the_coldfirebridge_0
out_port_from_the_pio_0[7..0]
inst
BoardClock100
ColdFireRSTOn
ColdFireA[23..0]
ColdFireBS2n
ColdFireBS3n
ColdFireCLKOUT
ColdFireCS1n
ColdFireCS2n
ColdFireOEn
ColdFireRpWn
ColdFireTSn
ColdFireD[31..16]
ColdFireIRQ1n
ColdFireTAn
ColdFireTEAn
LEDS[7..0]
INPUT
OUTPUT
BIDIR
VCC
Option Value
Location PIN_A16
Location PIN_T32
Location PIN_U23
Location PIN_U27
Location PIN_V29
Location PIN_V23
Location PIN_U22
Location PIN_U28
Location PIN_V28
Location PIN_M23
Location PIN_R31
Location PIN_R30
Location PIN_P32
Location PIN_P31
Location PIN_M32
Location PIN_M31
Location PIN_N31
Location PIN_N30
Location PIN_L32
Location PIN_L31
Location PIN_M30
Location PIN_M29
Location PIN_N29
Location PIN_N28
Location PIN_L30
Location PIN_L29
Location PIN_K32
Location PIN_K31
Location PIN_K30
Location PIN_K29
Location PIN_J32
Location PIN_J31
Location PIN_H32
Location PIN_H31
Location PIN_AC27
Location PIN_AC26
Location PIN_AD27
Location PIN_AD26
Location PIN_Y23
Location PIN_Y22
Location PIN_Y25
Location PIN_Y24
Location PIN_AA27
Location PIN_AA26
Location PIN_Y27
Location PIN_Y26
Location PIN_W25
Location PIN_W24
Location PIN_W27
Location PIN_W26
Location PIN_W28
Location PIN_W29
Location PIN_L25
Location PIN_B4
Location PIN_D5
Location PIN_E5
Location PIN_A4
Location PIN_A5
Location PIN_D6
Location PIN_C6
Location PIN_A6

Chapter 3

EPICS application

The steps listed below show how to create an example EPICS IOC application which uses the ASYN I/O environment to control the LEDs on the FPGA card.

1. Create a new EPICS `<TOP>` directory and make a new application in it:

   ```
   .../makeBaseApp.pl -t ioc ledDriver
   .../makeBaseApp.pl -t ioc -i -a RTEMS-uC5282 ledDriver
   ```

2. Edit configure/CONFIG to enable only the RTEMS-uC5282 target:

   ```
   CROSS_COMPILER_TARGET_ARCHS = RTEMS-uC5282
   ```

3. Edit configure/RELEASE to specify the location of ASYN support:

   ```
   ASYN=.../modules/soft/asyn
   ```

4. Edit ledDriverApp/src/Makefile:

   - Build only for RTEMS IOC targets (change the `PROD_IOC` line to `PROD_RTEMS`):

     ```
     PROD_RTEMS = ledDriver
     ```

   - Add asyn support:

     ```
     ledDriver_DBD += asyn.dbd
     ```

   - Add the asyn library the the `ledDriver_LIBS` (before the `EPICS_BASE_IOC_LIBS` line):

     ```
     ledDriver_LIBS += asyn
     ledDriver_LIBS += $(EPICS_BASE_IOC_LIBS)
     ```

   - Add the FPGA device support dbd file (to be written in a following step):

     ```
     ledDriver_DBD += ledDeviceSupport.dbd
     ```

   - Add the FPGA device support source file (to be written in a following step):

     ```
     ledDriver_SRCS += ledDeviceSupport.c
     ```

5. Edit ledDriverApp/src/Makefile and add the line:

   ```
   DB += ledDriver.db
   ```

6. Create the ledDriverApp/Db/ledDriver.db file referred to in the previous step. The file should contain:

```
record(longout,"leds") {
    field(DTYP,"asynInt32")
    field(OUT,"@asyn(ledDriver 0 0)")
}
```

   The three values in the `OUT` field are:

   (a) The port name.

   (b) The address (unused by this driver, but still needed)

   (c) The timeout value, in seconds (unused by this driver, but still needed).

7. Create ledDriverApp/src/ledDeviceSupport.dbd with the contents:

```
registrar(ledDriverDeviceSupportRegistrar)
```

8. Create ledDriverApp/src/ledDeviceSupport.c. A complete listing of is included in appendix A. Much of this file is common to all ASYN drivers. The following points describe the lines of particular interest to this application.

   12 When the SOPC Builder generates a system it creates a C header file describing the system components. This header file can be included explicitly or, as I have done, can be symbolically linked to a file in the application source directory:

   `ledDriverApp/src/SOPC.h -> ../../../FPGA/coldfirebridge_0_map/system.h`

   14 The SOPC address space appears int the ColdFire address space at the locations mapped to $\overline{\text{CS1}}$ and $\overline{\text{CS2}}$. The RTEMS board-support package sets up these chip selects at locations 30000000_{16} and 31000000_{16}, respectively.

   15 The '+1' is required because the ColdFire bus is 16-bit big-endian so the least-significant byte on the data bus appears at an odd address.

   23 The table of asynInt32 methods can't be statically initialized (like the asynCommon methods are at line 58) since the asynInt32Base initialize method (invoked at line 118) writes to the table to override some methods.

   41 The check for the existence of the I/O port turns off all of the LEDs. A real application might want to do something different.

   65 The line of code that actually performs the I/O operation.

   72 More complete device support would probably do nothing more in this routine than register an iocsh command. The actual ASYN registration calls would then be called from this command when invoked by the st.cmd script. This would allow the port name to be set from the st.cmd script rather than being burned into the program.

   80 The arguments to the registerPort method are:

   (a) The port name.

   (b) The port attributes. This driver is is not 'multi device' and does not block.

   (c) The autoconnect flag. This driver wants to be automatically reconnected.

   (d) The priority of the I/O thread (unused for this driver).

   (e) The stack size of the I/O thread (unused for this driver).

9. Run `make` to compile the application.

10. Set the uCDIMM ColdFire 5282 card environment variables as shown below. The exact values will differ as appropriate for your network numbers and NFS server:

```
FACTORY=Arcturus Networks Inc.
REVISION=uC5282 Rev 1.0 4MB External Flash
SERIAL=X42B20ADC-0130C
CONSOLE=ttyS0
KERNEL=0:linux.bin
KERNEL_ARGS=root=/dev/rom0
HWADDR0=00:06:3B:00:53:0C
FW_VERSION=180001
_0=10000000:400000:RW
RAMIMAGE=yes
IPADDR0=www.xxx.yyy.56
HOSTNAME=ioccoldfire2
BOOTFILE=epics/ucdimm/bin/RTEMS-uC5282/ucdimm.boot
NAMESERVER=www.xxx.yyy.167
NETMASK=255.255.252.0
SERVER=www.xxx.yyy.167
CACHE=on
NFSMOUNT=nfssrv:/export/homes:/home
CMDLINE=.../FPGA_IOC_Example/EPICS/iocBoot/iocledDriver/st.cmd
```

11. Download and execute the application:

    - Start the TFTP server on the ColdFire:

      ```
      B$ tftp
      uCTFTP Console 1.0 is running ...
      ```

    - Use the tftp program on your workstation to transfer the executable image to the ColdFire:

      ```
      tftp> connect www.xxx.yyy.56
      tftp> put ledDriver.boot
      ```

    - When the executable image has been transferred type an ¡ESC¿ to the ColdFire to terminate the TFTP server. Use the `goram` command to start the IOC:

      ```
      Downloading.............................................
      B$ goram
      Go from RAM!
      Go from 0x40000
      NTPSERVER environment variable missing -- using www.xxx.yyy.167

      ***** Initializing network *****
      Startup after External reset.
      fs1: Ethernet address: 00:06:3b:00:53:0c
      ***** Initializing NFS *****
      This is RTEMS-RPCIOD Release $Name:  $
      ($Id: tutorial.tex,v 1.6 2005/08/17 13:50:51 norume Exp $)

      Till Straumann, Stanford/SLAC/SSRL 2002
      See LICENSE file for licensing info
      This is RTEMS-NFS $Name:  $
      ($Id: tutorial.tex,v 1.6 2005/08/17 13:50:51 norume Exp $)

      Till Straumann, Stanford/SLAC/SSRL 2002
      See LICENSE file for licensing info
      ```

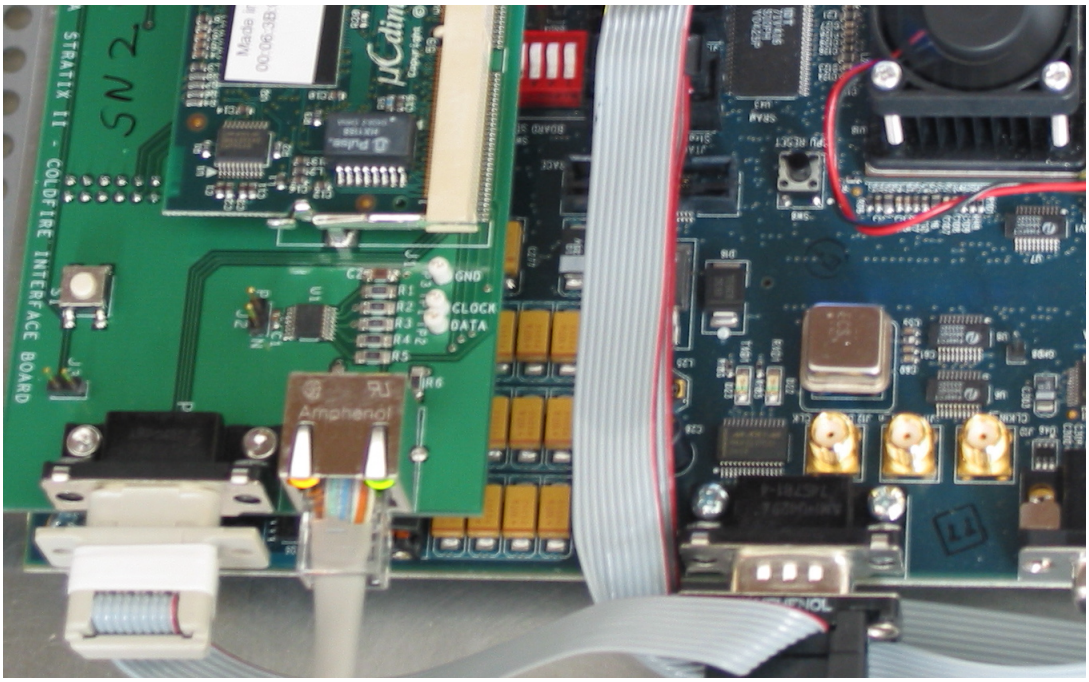
```
Trying to mount www.xxx.yyy.167:/export/homes on /home
***** Initializing NTP *****
***** Starting EPICS application *****
## Example RTEMS startup script
## You may have to change ledDriver to something else
## everywhere it appears in this file
#< envPaths
## Register all support components
dbLoadDatabase("../../dbd/ledDriver.dbd",0,0)
ledDriver_registerRecordDeviceDriver(pdbbase)
## Load record instances
dbLoadRecords("../../db/ledDriver.db","user=norume")
iocInit()
Starting iocInit
############################################################################
###  EPICS IOC CORE built on Jul 25 2005
###  EPICS R3.14.7 $$Name:  $$ $$Date: 2005/08/17 13:50:51 $$
############################################################################
iocInit: All initialization complete
## Start any sequence programs
#seq sncledDriver,"user=norume"
ioccoldfire2>
```

Chapter 4

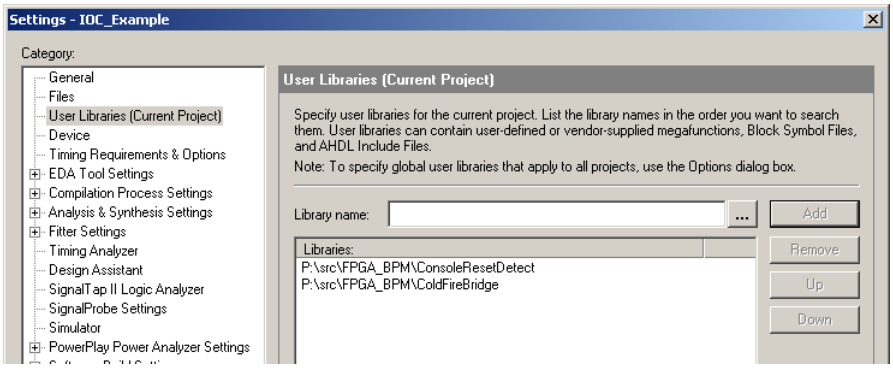
Remote Reset

All VME IOCs at the APS have a card which monitors the console received-data line and generates a system reset when a particular sequence (three consecutive control-X, control-Y or control-Z characters or any consecutive combinations of these characters) of characters is detected. It is quite easy to add this capability to the ColdFire/FPGA IOC.

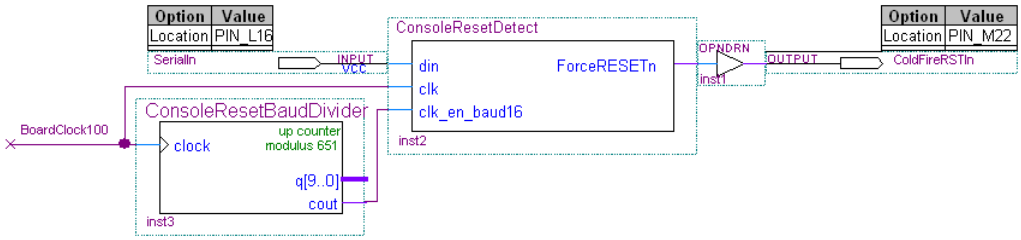
1. Make a special serial line cable which will connect the serial received-data and ground lines of the ColdFire console port to the 9-pin connector on the FPGA development kit. The following picture shows an example. The 9-pin connector plugged in to the FPGA development kit has only two pins (received-data and ground).



2. Add the path to the directory containing the Console Reset Detect component to the list of application user libraries:



3. Add the following components to the application top-level entity (`IOC_Example.bdf`). The ConsoleReset-BaudDivider is a simple modulus-651 counter which sets the serial line speed at 9600 baud ($100000000 \div (9600 \times 16) = 651$). If you're using a different system clock or a different serial line speed you'll have to replace this component with your own counter.



4. Recompile the project and load it into the FPGA. You should now have the ability to remotely reset the ColdFire.

Appendix A

ledDeviceSupport.c

```
/*
 * ASYN Int32 driver for simple FPGA application
 */
#include <epicsStdio.h>
#include <epicsExport.h>
#include <cantProceed.h>

#include <asynDriver.h>
#include <asynInt32.h>
#include <devLib.h>

#include "SOPC.h"   /* Symbolic link to SOPC-generated system header file */

#define AVALON_BASE  0x30000000    /* Base of Avalon space in ColdFire space */
#define OPTR ((epicsUInt8 *)(AVALON_BASE+PIO_0_BASE+1))

/*
 * Driver private storage
 */
typedef struct drvPvt {
    asynInterface common;
    asynInterface asynInt32;
    asynInt32     asynInt32Methods;
    volatile epicsUInt8 *optr;
} drvPvt;

/*
 * asynCommon methods
 */
static void
report(void *pvt, FILE *fp, int details)
{
}

static asynStatus
connect(void *pvt, asynUser *pasynUser)
```

```
37  {
38      drvPvt *pdrvPvt = (drvPvt *)pvt;
39      epicsUInt8 dummy = 0;
40
41      if (devWriteProbe(sizeof(dummy), pdrvPvt->optr, &dummy) != 0) {
42          asynPrint(pasynUser, ASYN_TRACE_ERROR, "ledDriver: memory probe failed\n");
43          return asynError;
44      }
45      pasynManager->exceptionConnect(pasynUser);
46      return asynSuccess;
47  }
48
49  static asynStatus
50  disconnect(void *pvt, asynUser *pasynUser)
51  {
52      pasynManager->exceptionDisconnect(pasynUser);
53      return asynSuccess;
54  }
55  static asynCommon common = { report, connect, disconnect };
56
57  /*
58   * asynInt32 methods
59   */
60  static asynStatus
61  int32Write(void *pvt, asynUser *pasynUser, epicsInt32 value)
62  {
63      drvPvt *pdrvPvt = (drvPvt *)pvt;
64
65      *pdrvPvt->optr = value;
66      return asynSuccess;
67  }
68
69  /*
70   * Register ourself with ASYN
71   */
72  static void ledDriverDeviceSupportRegistrar(void)
73  {
74      drvPvt *pdrvPvt;
75      asynStatus status;
76      const char *portName = "ledDriver";
77
78      pdrvPvt = callocMustSucceed(sizeof(drvPvt), 1, "ledDriver");
79
80      status = pasynManager->registerPort(portName, 0, 1, 0, 0);
81      if(status != asynSuccess) {
82          printf("ledDriver registerDriver failed\n");
83          return;
84      }
85
86      pdrvPvt->common.interfaceType = asynCommonType;
87      pdrvPvt->common.pinterface  = (void *)&common;
```

```
88      pdrvPvt->common.drvPvt = pdrvPvt;
89      status = pasynManager->registerInterface(portName, &pdrvPvt->common);
90      if (status != asynSuccess) {
91          printf("ledDriver registerInterface failed\n");
92          return;
93      }
94
95      pdrvPvt->asynInt32Methods.write = int32Write;
96      pdrvPvt->asynInt32.interfaceType = asynInt32Type;
97      pdrvPvt->asynInt32.pinterface  = &pdrvPvt->asynInt32Methods;
98      pdrvPvt->asynInt32.drvPvt = pdrvPvt;
99      status = pasynInt32Base->initialize(portName, &pdrvPvt->asynInt32);
100     if (status != asynSuccess) {
101         printf("ledDriver pasynInt32Base->initialize failed\n");
102         return;
103     }
104     pdrvPvt->optr = OPTR;
105 }
106 epicsExportRegistrar(ledDriverDeviceSupportRegistrar);
```

Appendix B

Alternate Pinouts

The information in this tutorial applies to several Altera development kits, including:

1. Stratix II DSP development kit(EP2S60), 100 MHz clock.
2. Stratix II NIOS development kit (EP2S30), ”BoardClock100” input is a 50 MHz clock.
3. Cyclone II NIOS development kit (EP2C35), ”BoardClock100” input is a 50 MHz clock, expansion prototype connectors require extensions to clear tall components on development kit.

The pin assignments for each of these kits are presented in the following table.

Signal	1	2	3
BoardClock100	A16	AF15	????
ColdFireA[0]	R31	E8	????
ColdFireA[1]	R30	J9	????
ColdFireA[2]	P32	F8	????
ColdFireA[3]	P31	A3	????
ColdFireA[4]	M32	C4	????
ColdFireA[5]	M31	C3	????
ColdFireA[6]	N31	C5	????
ColdFireA[7]	N30	K10	????
ColdFireA[8]	L32	H9	????
ColdFireA[9]	L31	G9	????
ColdFireA[10]	M30	A5	????
ColdFireA[11]	M29	B5	????
ColdFireA[12]	N29	D6	????
ColdFireA[13]	N28	A6	????
ColdFireA[14]	L30	H10	????
ColdFireA[15]	L29	K11	????
ColdFireA[16]	K32	F10	????
ColdFireA[17]	K31	A7	????
ColdFireA[18]	K30	C7	????
ColdFireA[19]	K29	D7	????
ColdFireA[20]	J32	A8	????
ColdFireA[21]	J31	G10	????

ColdFireA[22]	H32	J11	????
ColdFireA[23]	H31	F11	????
ColdFireBS2n	U27	A24	????
ColdFireBS3n	U23	B24	????
ColdFireCLKOUT	T32	AC14	????
ColdFireCS1n	V29	D18	????
ColdFireCS2n	V23	J18	????
ColdFireD[16]	AC27	H15	????
ColdFireD[17]	AC26	J15	????
ColdFireD[18]	AD27	C16	????
ColdFireD[19]	AD26	A17	????
ColdFireD[20]	Y23	C17	????
ColdFireD[21]	Y22	A18	????
ColdFireD[22]	Y25	F17	????
ColdFireD[23]	Y24	K16	????
ColdFireD[24]	AA27	G17	????
ColdFireD[25]	AA26	A19	????
ColdFireD[26]	Y27	C18	????
ColdFireD[27]	Y26	C19	????
ColdFireD[28]	W25	A20	????
ColdFireD[29]	W24	J17	????
ColdFireD[30]	W27	A21	????
ColdFireD[31]	W26	C20	????
ColdFireIRQ1n	W28	A22	????
ColdFireOEn	U22	K17	????
ColdFireRSTIn	M22	H18	????
ColdFireRSTOn	M23	J14	????
ColdFireRpWn	U28	B22	????
ColdFireTAn	W29	C21	????
ColdFireTEAn	L25	H17	????
ColdFireTSn	V28	C22	????
LEDS[0]	B4	AD26	????
LEDS[1]	D5	AD25	????
LEDS[2]	E5	AC25	????
LEDS[3]	A4	AC24	????
LEDS[4]	A5	AB24	????
LEDS[5]	D6	AB23	????
LEDS[6]	C6	AB26	????
LEDS[7]	A6	AB25	????
SerialIn	L16	H7	????