Author: Aviv Burshtein Instructor: Oz Shmueli Academic supervisor: Prof. Tsahi Birk

Set Up an Epiphany Cluster

This article is meant to guide the user in recreating an Epiphany level cluster using multiple Parallella boards. It covers the steps in setting up the environments, manufacturing the required PCB connectors, and running a basic parallel software on it.

Introduction

Epiphany has a 1GB memory space, and every epiphany holds 4MB of this space (32 KB of which are actually shared memory and addressable). Theoretically this leads to the conclusion that it might be possible to connect 256 Epiphany boards together (4095 cores). While this might be possible in the future, it is far from the case with 2015 Parallella boards. These boards only have E-link connectors for the north and south ports of the Epiphany, which narrows the possible cluster to one ‘column’ of Parallella boards (16 boards). This number is narrowed by half if taking into account that all lower half of the memory space is used by the Parallella operating system as the FPGA logic and SDRAM’s address space. So for a simple application as is demonstrated here it is possible to use up to 8 Parallella boards in a column (128 cores).
This article does not discuss using SDRAM space for Parallella boards, even though it is definitely possible. The demonstration uses 2015.1 ESDK with 7020 ZYNQ and a corresponding FPGA build version. An attempt to apply it to other ESDK versions or builds might lead to driver mismatchs, and would probably require some adjustments to be made.

Setting up the environment

- Download this disk image : https://github.com/parallella/pubuntu/releases/tag/pubuntu-14.04-esdk.2015.1-20150130
  Make sure you choose the file that matches your board’s ZYNQ version (7010 or 7020).
- Write the image into a micro-USB card. You may use this software to do this: https://sourceforge.net/projects/win32diskimage/
- Connect to your Parallella using SSH. You may follow this Parallella tutorial (page 3): http://www.adapteva.com/wp-content/uploads/Parallella-Quick-Start-Guide_rev3.pdf
  I recommend using this software for your SSH connection: http://mobaxterm.mobatek.net/download-home-edition.html

Connecting multiple Parallella Boards

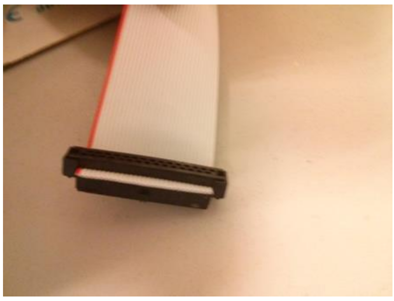
You may connect two or more boards via their north and south e-link connectors in one of two ways, both described below. The main advantage of the Porcupine connection is how comfortable and configurable it is, while the main disadvantage is it using large, slow and unreliable jumper cables. The main advantage of using my design is how small and fast it is, while the main disadvantage is the necessity of independently manufactured PCB.

Author: Aviv Burshtein Instructor: Oz Shmueli Academic supervisor: Prof. Tsahi Birk

1. Use Porcupine boards, and make your own connectors:

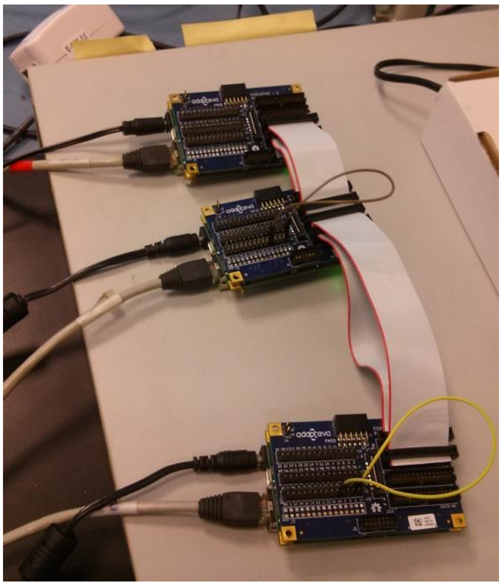
- Order a porcupine board for each Parallella: http://www.digikey.com/product-detail/en/ACC1600-01/1554-1003-ND/5048176
- Order flat cables and connectors: http://www.digikey.ca/product-search/en?keywords=MC30L-5-ND
  and get some jumper cables from the nearest electronics shop.
- Order flat cable connectors, two for each board you wish to connect: http://www.digikey.ca/product-search/en?keywords=WM14330-ND

- Connect both ends of the cable to the connectors you purchased:

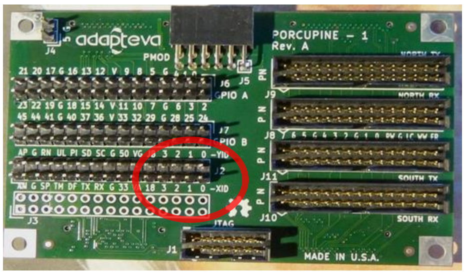
You must make two of these for every pair of Parallella boards.

- Connect Parallella boards as follows:



- Hardwire core-id:
  The core-id can be reconfigured by hardwiring the PEC_POWER connector.
  The pins representing the core-id are indicated in this image:

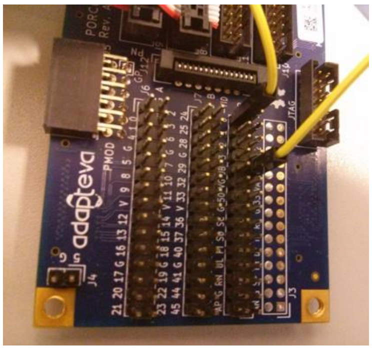
Author: Aviv Burshtein Instructor: Oz Shmueli Academic supervisor: Prof. Tsahi Birk



The core-id vector is 6 bits for Yid (columns) and 6 bits for Xid (rows). The last two bits of both are always 00 for the upper left core in each board (row and column are multiples of 4) and therefore 8 bits are enough to represent the address. The default core-id coordinates are (32,8), which translate to 808 in hexadecimal representation, and are represented as follows:

Yid 3	Yid 2	Yid 1	Yid 0	Xid 3	Xid 2	Xid 1	Xid 0
0	1	0	0	0	1	0	0

In order to set the core-id, jumpers must be used to connect GND or 1.8 Volts outputs to the correct coordinates. For example, coordinates (36,8) would translate to 908 in hex representation, and would be wired like this:

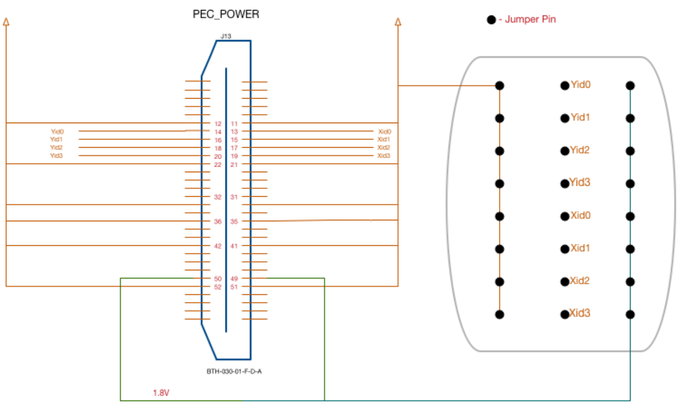


2. Produce PCB connectors and jumper sockets using my design:

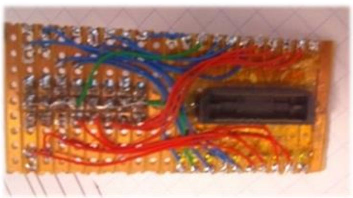
- Download the connector design files from [here].
- Send to reproduction in a fab. I recommend: https://www.itead.cc/
- Order connectors: http://www.toby.co.uk/content/catalogue/products.aspx?series=BTH-xxx-01-x-D-A-xx
  Make sure you order some spares for hardwiring core-ids.
- Solder the connectors to the PCB:

- Connect as follows, preferably under static pressure or bolted to a surface.

- Hardwire PCB: It is possible to solder the male connectors themselves to a constant core-id, and simply place the hardwired connector in the PEC_POWER socket. However, there is a preferable solution that is more flexible. Use this design and connect the relevant pins in the PEC_POWER socket to jumper pins.



Here are two examples:



Author: Aviv Burshtein Instructor: Oz Shmueli Academic supervisor: Prof. Tsahi Birk

- If anyone could pick this up and add a PCB design for this I would gladly add it to the files here!

- When connecting the boards connect the correct pins for the core-id according to the explanation in the previous section.

Set up and run example code

The code uses the 2015 ESDK interrupts library to prove basic communication between the different Parallella boards. The communication is in the Epiphany level, meaning there is no read-write connection from a ZYNQ to a distant epiphany, and communication between OS and distant Epiphany boards is done via mailboxes in the local Epiphany board.

- Download the code folder from this link and unzip it.
- Copy the folder to your Parallella machines.
- Set up your HDF file using the edit_hdf script:
  From the first board (with a single slave board):
  Sudo ./edit_hdf.bash master 1
  Here the second argument represents the number of slave boards in the system.
  From a single slave board:
  Sudo ./edit_hdf.bash slave 1
  In this example the second argument represents the current slave board number.
  Note that if you want to use more than two boards you must change the *maxcorenum* macro in the files *int-test.c* and *e-int-test.master.c* to match the amount of cores you need, and use edit_hdf accordingly.
  For example, if you wish to use three boards in total (1 master and 2 slaves), you should set *maxcorenum* to 48, type "**Sudo ./edit_hdf.bash slave 1**" in the first slave Parallella, and type "**Sudo ./edit_hdf.bash slave 2**" in the second slave Parallella. In the master Parallella type "**Sudo ./edit_hdf.bash master 2**".
- Run ./build.sh from the slave directory on all slave boards and run it from the master directory on the master board.
- Run ./run.sh from the slave directory on all slave boards and then run it from the master directory on the master board.
  The output should be the text in each board's space in the mailbox (within the first core's memory space).
  Note that it might be wise to set up each core on its own at first and run "hello world" on it with the new core-id configuration.

Author: Aviv Burshtein Instructor: Oz Shmueli Academic supervisor: Prof. Tsahi Birk

Key Design Decisions and Barriers

Core-id

To my knowledge Adepteva plans to add core-id configuration support using FPGA register and next generation Epiphany Config-Registers in the future. As these are not yet available I decided to hardwire the core-id. A possible alternative would be to change FPGA design and set the core-id.

E-Link Enabling

The reset_system() function in 2015.1 ESDK locks the communication via E-Link connections. I therefore had to write a similar function with a slight difference- it doesn't lock the E-Link connection. I called it "reset_connected_systems" and I believe adding it to the next ESDK might be a good idea. This solution was made possible by the help of peter (Parallella forum profile).

User Interrupts

The 2015 ESDK added extraordinary support of user interrupt functions that allow synchronized communication between the cores. The interrupts based code is based on Yaniv Sapir's great interrupts code example. This is a good solution, but I believe it must be wrapped in more convenient user interface in order to scale.

Known Issues

- Loose connectors: The connected boards must be aligned well and not bend the connector in order for the system to work. This is difficult to maintain and uncomfortable. Possible solutions could be adding bolt holes to the design or using a mounting surface.

Next Steps

- Add a bolt to the connector design.
- Create a PCB design for the core-id setting connector.
- Add a library to support ZYNQ to distant communication via mailboxes, without dealing with space allocation and interrupts
- Use GPIO and FPGA routing configuration to close loop for read-write connection between from ZYNQ to remote Epiphanies.

Useful links

Project Repository
Relevant forum discussions:
Epiphany Mesh1
Epiphany Mesh2

Guides and Starting:
Setup tutorial

Author: Aviv Burshtein Instructor: Oz Shmueli Academic supervisor: Prof. Tsahi Birk

Code repository and design files:
[Our Repository]