# Static and Dynamic Reconfigurable Computing with Real-Time Infrastructure

EE 4982V

Senior Honors Progress Report

University of Minnesota, Twin Cities

by

Nathan Kro

Advisor

Prof. Richard Voyles

May 5, 2006

Updated: May 15, 2011, Rev E

## *Abstract*

In this report, the research completed about reconfigurable computing is presented. A description for a morphing bus to help implement reconfigurable computing is given. An example application for reconfigurable computing, the TerminatorBot, is also provided. Considerations for both design and layout for the TerminatorBot PCB's are given.

# *Table of Contents*

# List of Figures

# 1 Introduction

The idea for reconfigurable computing has been around since the 1960s, when Gerald Estrin published a paper that proposed a method of implementing it. The method proposed by Estrin was to have a computer that consisted of a standard processor and an array of reconfigurable hardware. The processor would control the function of the reconfigurable hardware. The reconfigurable hardware would be setup to perform a specific task, like image sensing, as fast as it would be performed by a dedicated piece of hardware. Once the task was completed, the hardware could be adjusted to perform a new function. This would allow a computer structure that had better flexibility with the same hardware speed. Estrin's idea was ahead of the technology of the 1960s, but his idea has become popular again as the technology is now becoming available [1].

Xilinx was one of the first major companies to research and start development on reconfigurable computing. In the mid-1990s Xilinx started developing a new FPGA family specifically designed for reconfigurable computing. The part had "deep configuration memory, the ability to partially reconfigure the chip on the fly and numerous other features" [2]. This new FPGA family was deemed too difficult to implement for production by Xilinx, but some of the features are available in other Xilinx families now, like the Virtex-II family [2].

Reconfigurable computing could be used for many different applications, such as robotics or in an assembly line of a factory; there are countless potential uses. The reason for so many uses is because of the benefits of reconfigurable computing. Less hardware is needed because the same hardware can be reconfigured to perform different tasks.

Because there is less hardware, it takes up less space and products can be smaller. The hardware can be easily upgraded and reused for different projects because it only needs to be reprogrammed. These benefits are something that a team of researchers from the University of Minnesota are looking to implement in their robot called the TerminatorBot.

The TerminatorBot has been around since 2000 with many modifications throughout its life. It was created to be a search-and-rescue robot funded by the DARPA Distributed Robotics contract. It is capable of manipulating objects and locomotion through the use of two arms. Because of the small size of the robot, reconfigurable computing would allow the TerminatorBot to have more functionality through the use of less hardware, which takes up less space in its small body. This would allow the TerminatorBot to adapt more to its environment.

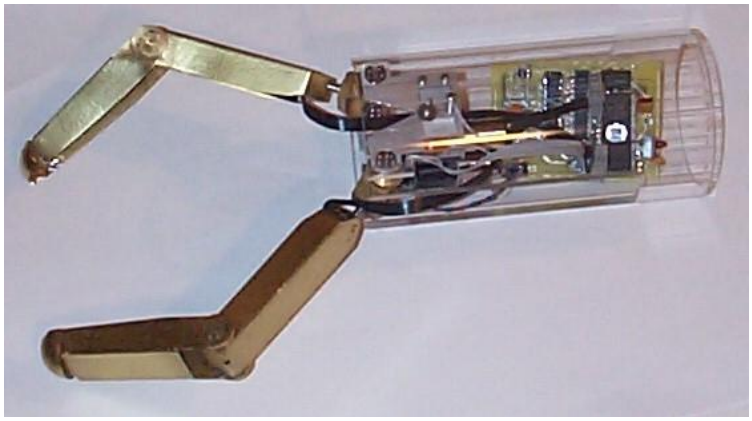


**Figure 1: The TerminatorBot [3]**

# 2 Work Performed

To implement reconfigurable computing on the TerminatorBot there are three main things that are needed: the CPU/reconfigurable hardware interface, the morphing bus, and the add-on cards that perform specialized functions.

The CPU/reconfigurable hardware interface for the TerminatorBot will be the Xilinx Virtex II family of processors. The Virtex II has an embedded CPU core, which results in a System-On-Chip design [4]. This will allow for better integration between the CPU/FPGA interface and the rest of the system.

The add-on cards are used to perform specialized functions like camera operation, motion sensors, smell sensors, etc. They plug into the main circuit board of the TerminatorBot and they are able to be added or removed when desired. Only the camera interface has been implemented for this project, but the use and interface to the cards and the morphing bus are designed to accommodate a wide range of expansion cards.

The morphing bus connects the CPU/reconfigurable hardware interface to the add-on cards. The bus needs to be able to handle all the different types of I/O cards that can be attached to the system. One of the goals when designing this morphing bus was to allow it to work for any kind of system. This bus will work regardless of the CPU/reconfigurable hardware interface or the I/O card(s) attached.

## 2.1 Morphing Bus

The morphing bus is different from other bus architectures because there is no extra conversion logic when communicating with the I/O cards. For standard buses, there is a protocol that needs to be followed by each device on the bus. The expansion card then needs to have logic onboard to convert from this standard to its interface format. This is shown in Figure 2 below.

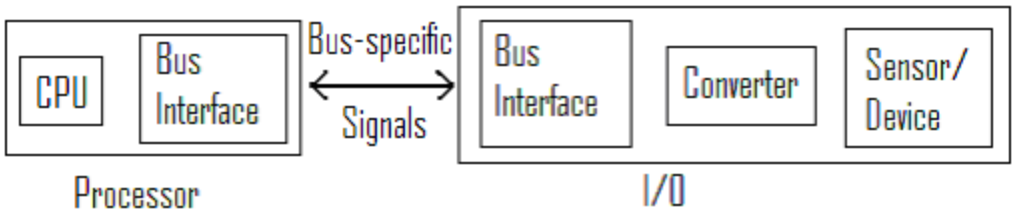



**Figure 2: Standard bus (static)**

The way the morphing bus works is to send the signals directly to the sensor. The conversion logic is incorporated into the bus interface when it is statically reconfigured for the sensor. This is because the data sent on the bus to the device is already tailored to that particular device. This setup can be seen in Figure 3 below.

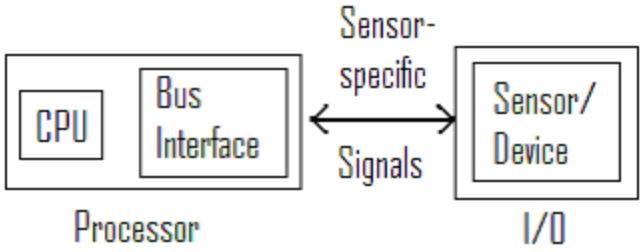



**Figure 3: Morphing bus (dynamic)**

Note, however, that at the present time there will still need to be logic to do analog conversion on expansion boards. The current FPGAs are not capable of both digital and analog signaling. Xilinx's product roadmap includes FPGAs capable of both analog and digital programmability in the next year or so.

The work performed in researching configurations of the morphing bus was concentrated on configurations that would allow for the best expansion in the future. The morphing bus will have 50 signal lines available. The possibility of having 60 signals was investigated, but 50 signal lines was selected due to size constraints of the boards (see Section 2.2). Four of the signal lines will be dedicated to power and ground (two

ground, one 5V, and one 3.3V), leaving 46 lines available for other connections. The signals that are not used on the individual wedges are passed through to the next wedge. They are passed in a way such that the next wedge will use the first signals on the morphing bus as if they were connected to the first pins on the baseboard connection. This configuration can be seen in Figure 4 below. By passing the unused signals in this way, each wedge does not need to know/care what other wedges may be connected before it. However, the CPU does need to know in what order the wedges are connected so that it can be configured to interface with the wedges as they are connected. In this way the design is statically reconfigurable. When a new wedge is added to the current system, the CPU just needs to know what wedge was added, and the signals associated with the wedge can be configured into the FPGA pinout. Similarly, if a new wedge is swapped with a wedge that is currently in the system, the CPU needs to know which wedge was taken out and what it was replaced with, then it can reconfigure the FPGA pinout for the new wedge that has been added.

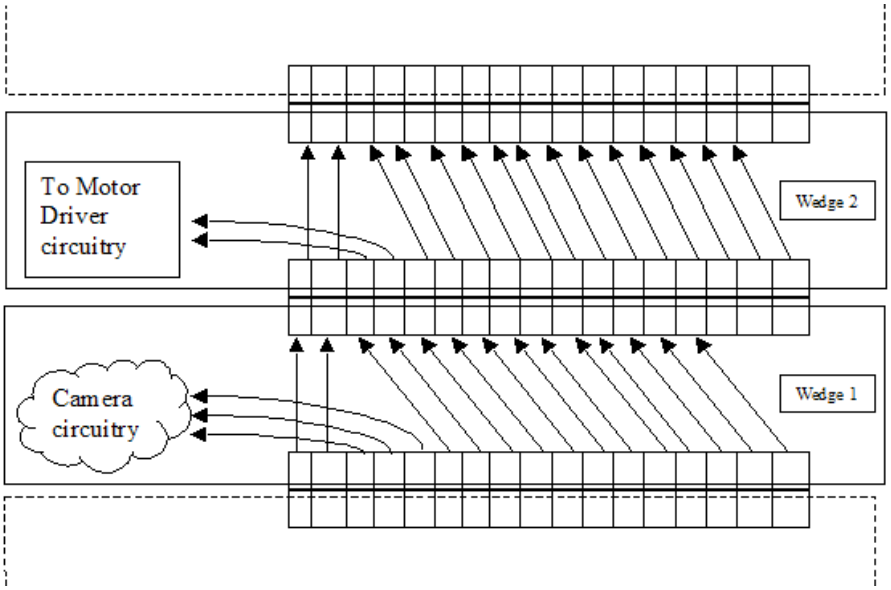



**Figure 4: Morphing bus signal routing [5]**

The connectors chosen for the morphing bus (discussed in Section 2.2) have a 0.3A per pin current rating. This is another important factor that needs to be considered when creating the individual wedges. They need to be designed such that the current draw is not enough that it will exceed the limit of the connectors. For typical wedges, this current limit should not be an issue as the current draw for each wedge should not exceed about 50mA. Assuming each wedge takes about 7 or 8 signals, there would be enough room to add 6 wedges onto the morphing bus. At 50mA each, that would just reach the current limit of the connector, but typically it is not expected that the wedges will require 50mA each (the camera, for instance, uses a significant amount of power, but the maximum current under "normal" operation is 30mA). If more current is needed for an individual wedge, the wedge will need to have its own power supplies; the control circuitry for the motors are one such wedge where part of it will need to be driven by an external power source due to the current requirements.

Applications using one or more camera interfaces were the target of much of the research of the bus standards. The camera interface was the main focus because it is one of the most likely to be used add-on cards and it has the potential to use many of the signals on the morphing bus. Different bus connection types were looked at for the camera, including serial connection using USB or I2C and parallel connection using either an 8-bit or a 16-bit data line.

Using a camera interface that has a parallel 16-bit data line in addition to the control signals would allow for only two cameras to be connected at once with little or no room for additional add-on cards. For this reason, the parallel 16-bit data line was changed to a parallel 8-bit data line (plus the control signals) to save on signals. Using a

parallel 8-bit data line would allow for two cameras to be connected with some signals left for other functions, but the camera interface would still take up well over half of the available signals on the morphing bus. Because of the high pin count of parallel camera interfaces, serial interfaces were looked at.

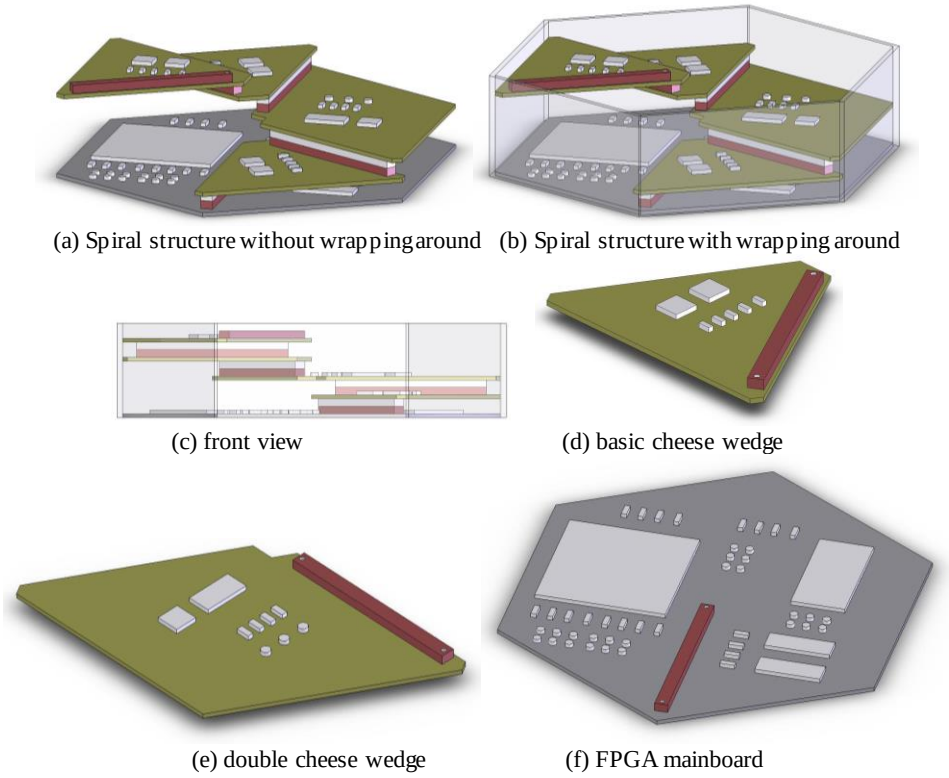
One serial interface that was considered was USB. This would cut the number of signals needed for the camera down to only two per camera (plus power and ground). If USB is implemented in a generic way, other USB devices could be used with the same add-on card. Using USB for the camera interface was decided against, however, due to routing concerns of the signals (need to be differential pairs) and because a suitable USB camera was not found. USB is still a potential add-on card, but it needs to be researched further.

Serial communication using the I2C protocol was then considered. A camera interface based on the I2C protocol would allow the camera to operate using only 8 or 9 signals on the morphing bus. This would allow for more add-on cards to be connected to the system at once. Since this interface used less signals than a parallel interface and a suitable I2C camera module was found, this was the design that was chosen to use for the camera interface.

## 2.2 Circuit Board Design

As mentioned above, the morphing bus will only have 50 signal lines available due to size constraints. One size constraint was due to the fact that the diameter of the TerminatorBot is only 69mm. The circuit board design (Figure 2) for the wedges

required a hole in the middle of the TerminatorBot for wires to pass through, so part of the "tip" of the wedge needed to be cut off to allow for this space. It was also required to leave a small gap between the circuit boards and the exterior wall of the TerminatorBot for extra airflow. So the rounded part of the wedge needed to be cut off. This was also done to allow easier manufacturing of the wedges. It is much easier to cut a straight edge than rounded edges of a PCB. This only left enough space for a 50 pin connector for connection between the wedges. Hirose's DF12, DF17, and FX8C series were the connectors that were considered. The DF12 series was selected due to its smaller footprint and the available stacking heights. Each connector is about 17mm x 6mm. The DF12 series provides stacking heights between 3.0mm to 5.0mm. Having multiple stacking heights allows for different wedge sizes (double-wedge, triple-wedge) because the stacking height can be increased to account for the extra width the multi-slice wedge takes up. The width is important because of the height constraints. The components on the wedges can only be a certain height or the wedge pieces will not fit together properly when connected together in the spiral design. The DF12 series connectors have a 0.5mm terminal pitch. This size is small enough so a large number of pins can be used, but still large enough to allow for routing of all the signals on a 4-layer board.



(a) Spiral structure without wrapping around (b) Spiral structure with wrapping around

(c) front view (d) basic cheese wedge

(e) double cheese wedge (f) FPGA mainboard

**Figure 5: New Circuit Board design for the TerminatorBot [5]**

The size of the individual wedges was considered carefully before making a decision on how many pieces to divide the circle into. Four, six, and eight piece wedge slice designs were considered before deciding to use a six piece wedge design. A six piece wedge design was chosen because it offered the best trade-off between wasted space and usable space per wedge slice. Table 1 and Figure 6 shown below were used to make the decision. As the table shows, the four, six, and eight wedge designs offered 583mm^2, 505mm^2, and 412mm^2 of single-sided surface area, respectively. While each design "wasted" 340mm^2, 108mm^2, and 47mm^2, respectively, in the area that would be cut-off of the cone shaped wedge to make a triangular shaped wedge. Dividing the wedge surface area by the surface area of the circle cut-out gave the percent of area of

the circle that was actually used on the wedge; this came out to be 62.33% for the 4 wedge design, 80.96% for the 6 wedge design, and 88.14% for the 8 wedge design. From these figures, it is easy to see that a four wedge design would lose too much surface area due to the amount of PCB that would be cut off in making the triangular wedge shape. Therefore, the four wedge design was thrown out. The amount of wasted space was less for the eight wedge design than the six slice design, but the amount of surface area per wedge that was lost going from a six wedge design to an eight wedge design , about 100mm^2, was decided to be too much. There needed to be enough surface area per wedge to allow for an adequate number of components to be placed on the wedges. Therefore, the six wedge design was decided upon.

| Whole Circle Measurements | | | | | |
|---|---|---|---|---|---|
| Radius of Circle | 34.5 | Full Circle Area | 3739.3 | | |
| | | | | | |
| Number of wedges in a full circle | 4 | | 6 | | 8 |
| Central Angle | 90 | | 60 | | 45 |
| Segment Height ED | 10.105 | | 4.6221 | | 2.6262 |
| Apothem OE | 24.395 | | 29.878 | | 31.874 |
| Cord AB | 48.79 | | 34.5 | | 26.405 |
| Segment Area (cut-off part) | 339.7 | | 107.82 | | 46.593 |
| Triangle Area | 595.13 | | 515.39 | | 420.82 |
| Sector Area | 934.82 | | 623.21 | | 467.41 |
| Wedge Area compared to Sector Area | 63.66% | | 82.70% | | 90.03% |

| Center Cut-out Circle Measurements | | | | | |
|---|---|---|---|---|---|
| Radius of Circle | 5 | Full Cut-out Circle Area | 78.57 | | |
| Number of wedges in a full circle | 4 | | 6 | | 8 |
| Central Angle | 90 | | 60 | | 45 |
| | | | | | |
| Segment Height ED | 1.4645 | | 0.66987 | | 0.3806 |
| Apothem OE | 3.5355 | | 4.3301 | | 4.6194 |
| Cord AB | 7.0711 | | 5 | | 3.8268 |
| Segment Area | 7.135 | | 2.2647 | | 0.97864 |
| Triangle Area | 12.5 | | 10.825 | | 8.8388 |
| Sector Area | 19.635 | | 13.09 | | 9.8175 |
| Wedge Area compared to Sector Area | 63.66% | | 82.70% | | 90.03% |

| Final Circle Measurements | | | | | |
|---|---|---|---|---|---|
| Single Wedge area | 582.63 | | 504.565 | | 411.9812 |
| Wedge with hole Area compared to Sector Area | 62.33% | | 80.96% | | 88.14% |
| Single Hole area | 12.5 | | 10.825 | | 8.8388 |
| Full Circle Hole area | 50 | | 64.95 | | 70.7104 |

*all units are in mm and mm^2

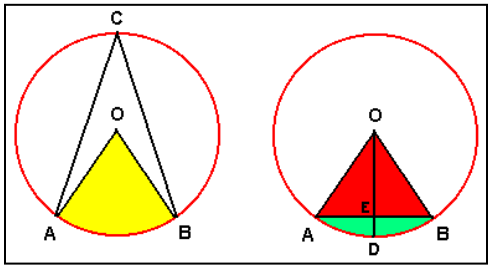



**Figure 6: Figure used to calculate wedge size**

The final wedge shape and dimensions were then created based on having the two connectors centered about a six wedge design. That meant that the actual PCB wedge size would be larger than the dimensions listed in the Table 1. The corners also needed to be cut-off to allow for the wider wedge. The final design can be seen in the figure below. The area for this design is 717.75mm^2 per surface of the wedge. Each wedge has the two standard pass-through connectors which take up about 102mm^2 each, so the total area for routing on each surface of the wedge is about 616mm^2.

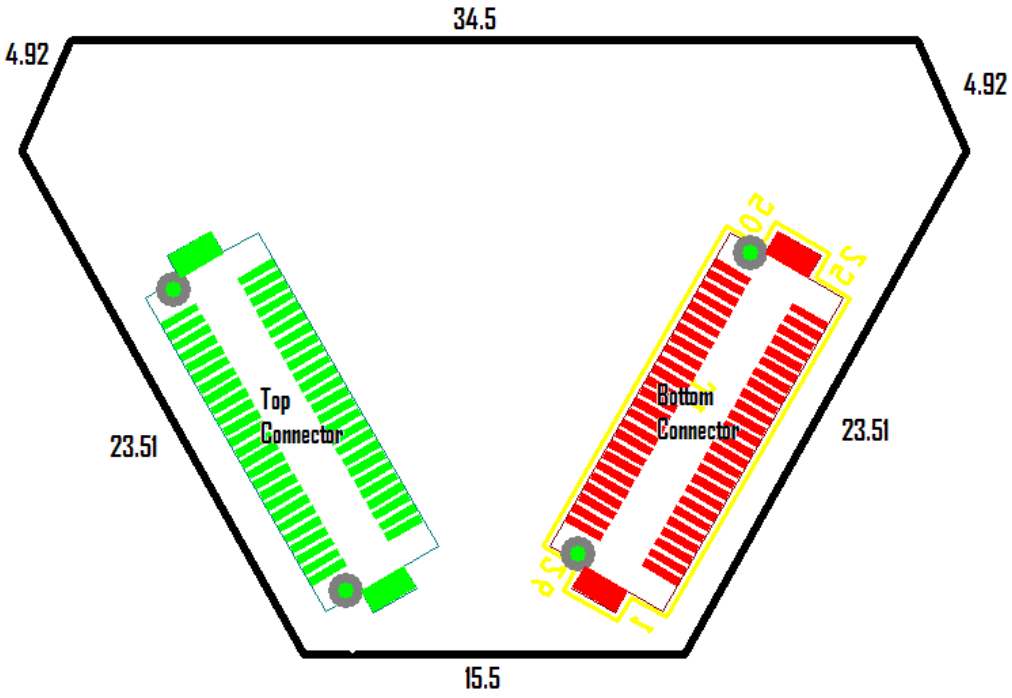



**Figure 7: Final wedge design (J1 and J2 pin numbering is incorrect.)**

The schematic (Figure 8) for the camera interface was created and the signals were imported into layout for routing onto the wedge. The camera wedge was then routed on a four-layer PCB. The routed camera wedge can be seen in the Figure 9 below. The routing for the camera wedge had to be done by hand due to the small size constraints and repetitive pattern for routing the signals. The layout program was not able to auto route the signals because it could not identify the pattern to follow due to the signal pinout. This pattern was relatively easy to follow (by hand) once it was discovered, and should be relatively easy to recreate for other wedge designs. Each wedge that uses a different number of signals will need to be re-routed because the signals that are passed straight through will be shifted by the difference between the number of signals used on the wedges.

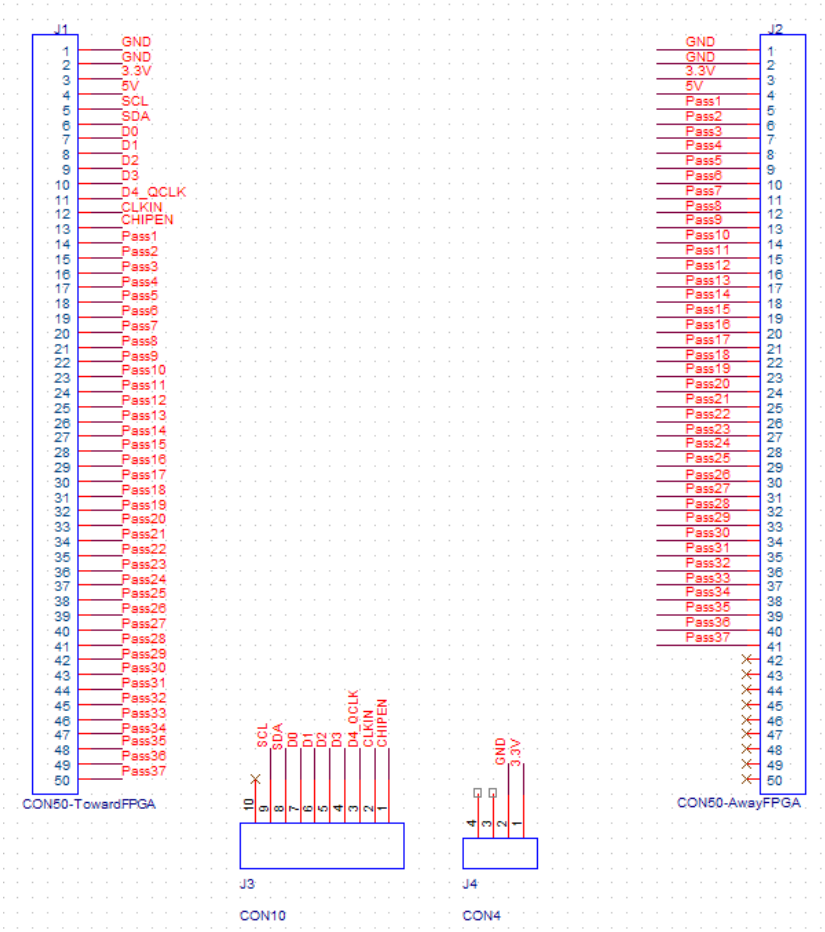



**Figure 8: Camera interface schematic**

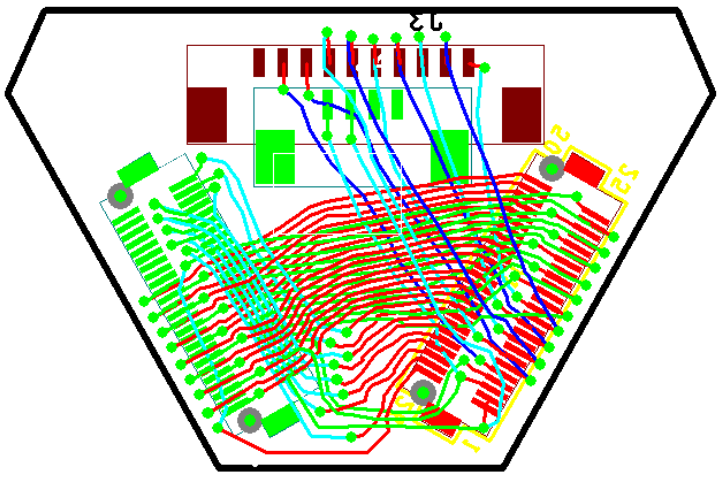


**Figure 9: Camera interface wedge (J1 and J2 pin numbering is incorrect.)**

# 3 Dynamic Reconfigurable Computing

Research was done for applications of dynamic reconfigurable computing. The research was concentrated on robotics applications. One idea was to use reconfiguration for video sensing. For example, there are many different techniques that can be used for robots to keep track of their environment and where they are in that environment. One of these methods is to use an object of known size, another robot for instance, to determine the size of everything else in the environment. This type of environment recognition could require less data processing because there is a known data set for sizing. The data can also be stored in the hardware for faster access and can be optimized better than if it were stored in software. On the other hand, if a known reference point is not in the viewing area, the robot will need to do a different type of sizing technique, possibly relative sizing. This technique could require more data processing because every object needs to be compared to all the other objects in the viewing area. In this case, more hardware could be dedicated to the processing of the objects in the viewing area by "borrowing" hard ware from regular tasks like motion. Once the robot has the basics of its environment down, the hardware could be reconfigured back to "normal" operation by restoring the hardware to how it was and resume activities. This would allow the robot to learn about its environment faster than it could if it had limited hardware resources.

Another application for dynamic reconfigurable computing is to conserve power in battery operated devices. Power can be conserved by reconfiguring the hardware when power hungry applications/algorithms are not needed [6]. For example, this idea could be applied to video capture. When conditions are bad (poor lighting, weather conditions, etc.), higher quality video can be collected to allow for better image processing.

Retrieving high quality video uses a lot of power, and for battery operated devices power is a big factor. If the conditions are good for capturing video data, the capturing process can be scaled down to collect lower quality video. This will require less power and conserve precious battery life.

# 4 Future work

Design and layout of the main CPU/FPGA baseboard still needs to be completed. Once it is completed, the whole system can be tested. Until then, the wedges are not able to be tested.

Right now, the morphing bus is only statically reconfigurable. Because it was designed so that each wedge uses the least amount of signals as possible, no identification signals were built into the bus interface. Standard bus protocols typically assign device IDs or use some other type of identification so that the master device knows what and how many devices are attached to it. Research can be done for a method to do this using the morphing bus idea. As it is, the CPU needs to know the order in which the wedges are connected. If the CPU was able to auto-detect the devices as they are connected, it would be able to dynamically reconfigure as the wedges are changed.

# 5 References


[1] "Reconfigurable computing," September 25, 2005, http://en.wikipedia.org/wiki/Reconfigurable_computing.

[2] R. Wilson (rwilson@cmp.com), "The constantly shifting promise of reconfigurability," September 11, 2002, http://www.eet.com/reshaping/reconfig/OEG20020911S0066.

[3] R. M. Voyles, "TerminatorBot - Locomotion and Manipulation from a Single Mechanism," accessed September 29, 2005, http://www-users.cs.umn.edu/~voyles/.

[4] Xilinx Corporation, "Spartan-3 FPGAs: Overview," accessed September 29, 2005, http://www.xilinx.com/products/silicon_solutions/fpgas/spartan_series/spartan3_fpgas/overview.htm.

[5] B.H. Kim, C. D'Souza, R. M. Voyles, J. Hesch, S. Roumeliotis, "A Reconfigurable Computing Platform for Plume Detection with Mobile Sensor Networks".

[6] J. Liang, R. Tessier, D. Goeckel, "A Dynamically-Reconfigurable, Power-Efficient Turbo Decoder," accessed May 3, 2006, http://www.ecs.umass.edu/ece/tessier/turbo-fccm04.pdf.

# 6 Appendix A: Components and Connector Height

The volume for components above and below the wedges is limited due to the spiral nature of the morphing bus (wedges can spiral back over themselves) and the allowance of multiple spirals from the same baseboard. The Hirose DF12 connectors allow for different connectors heights (section 2.2) and the Morphing Bus specification allows for different wedge sizes (Figure 5). All wedges are nominally made of 0.062" thick FR-4 PCB material.

The hexagonal design dictates the fundamental unit of measure for the morphing bus is the 60-degree single wedge shown in Figure 10. The double wedge covers 120 degrees. A triple wedge was considered, but is not feasible with the available connector stacking heights of the Hirose DF12. The board edge is 4.5 mm from the connector centerline.

Positioning of the connectors is as specified in Figure 10. The preferred reference for locating the connector on a wedge is the mounting hole (9.45 mm from the center point). The distance to pin 1 is also reliable for placement. The distance to the center of the mechanical solder pad is unreliable as it varies according to the width of the pad. (For reference, this dimension is approximately 8.65 mm.)

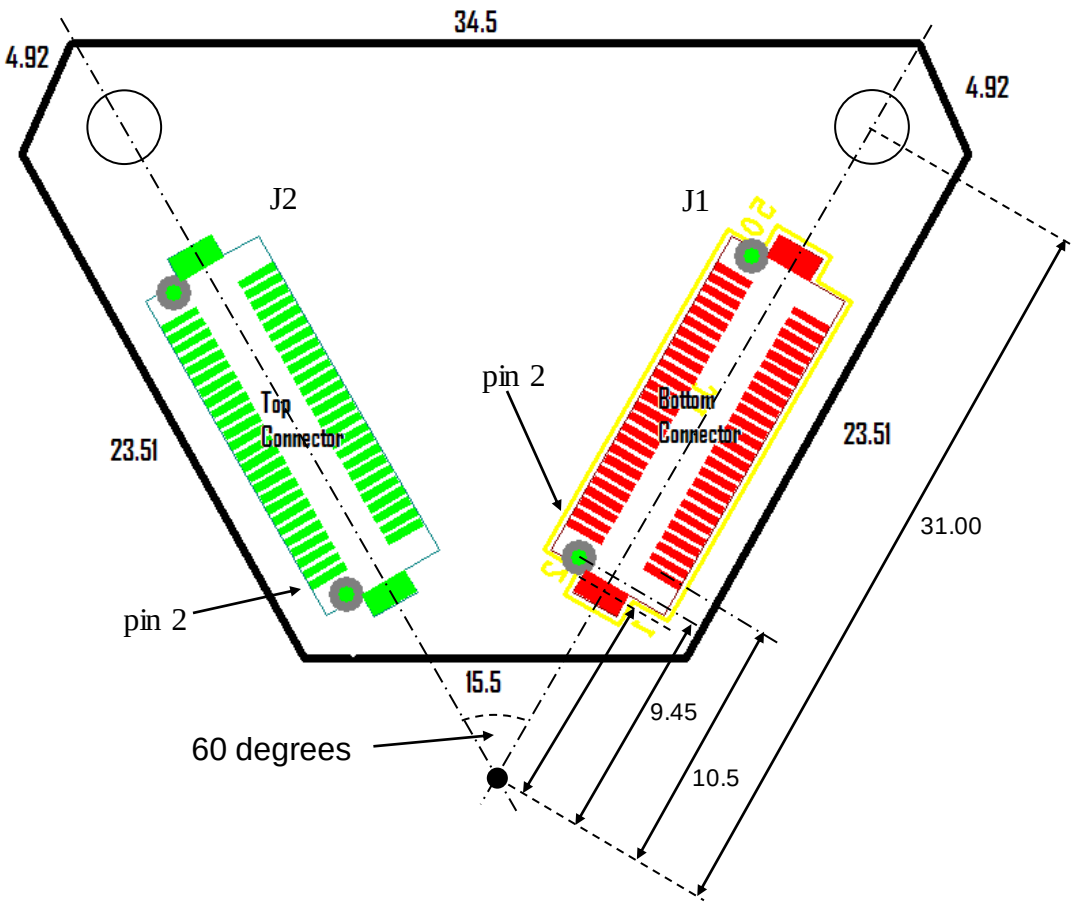



**Figure 10: Positioning of connectors and mounting holes.**

Nominally, the bottom connector (to the baseboard) is designated J1 and is a DF12 female header with 3.5 mm stacking height for single wedges (P/N DF12 (3.5)-50DP-0.5V(86)) and 5.0 mm stacking height for double wedges (P/N DF12 (5.0)-50DP-0.5V(86)). These connector configurations allow component heights on the bottom of the board that do not exceed 2 mm (Figure 11). In extreme cases, the single wedge can employ the 5.0 mm female header (shown in Figure 12) normally reserved for the double wedge, to allow up to 3.5 mm component heights on the bottom of the single wedge. The 5.0 mm stacking height connector allows the low-profile Molex PicoBlade 1.25mm right angle connector series (P/N 53261 series) to be installed on the bottom of the wedge with a current rating of 1 amp per pin (shown in Figure 13).

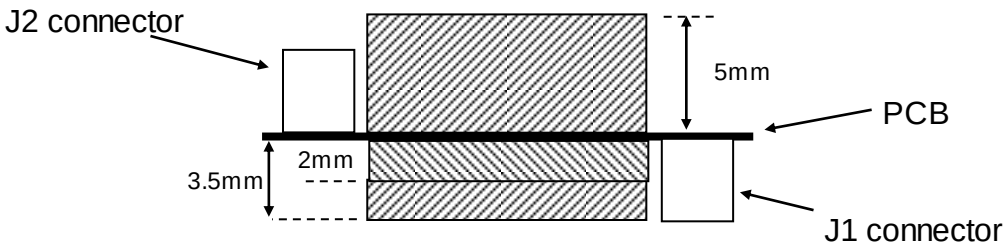



**Figure 11: Allowable component heights for single and double wedges.**

On double wedges, since the stacking height of the DF12 series is not a ratio of 2:1, it is preferable to place tall components on the bottom side of the wedge near the J1 connector. This will minimize potential interference with a second morphing bus spiral. The top side of all wedges provides more room for components. Regardless of size (single or double) the top side components can extend up to 5 mm above the top side of the wedge printed circuit board (PCB), as shown in Figure 11.

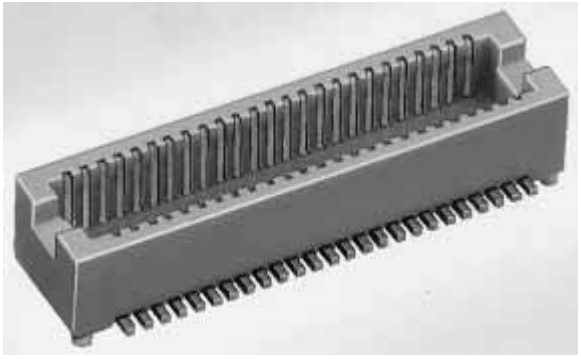


**Figure 12: Hirose DF12 Female Header for use as J1.**

Nominally, the top connector is labeled J2. This connector daisy-chains the Morphing Bus, allowing additional wedges to be stacked in a spiral (Figure 5a). The

Hirose DF12-50DS-0.5V(86) connector is used for J2, which is the same as the baseboard connectors.

The mechanical integrity of the Morphing Bus stack is assured by the connectors and mounting holes that allow for rigid standoffs to be inserted. These holes are #4 through holes (0.1285" or 3.26 mm) and are positioned as shown in Figure 10 at every point of the hexagon. Sufficient space should be left around every mounting hole for the shoulder of a nylon standoff.

Pin numbering on the Morphing Bus connectors alternates like a double-row header. In other words, all odd pins are on one side and all even pins are on the other. For the 50-pin connectors used for the Morphing Bus, the four corner pins are numbered 1, 2, 49, and 50. Pins 1, 2, 3, and 4 are always ground, while pins 5 and 6 are 3.3 V, and pins 7 and 8 are 5.0 V. Each contact is rated at 300 mA, so the maximum current draw of the entire Morphing Bus stack for both 3.3 V and 5.0 V supplies is 600 mA each.

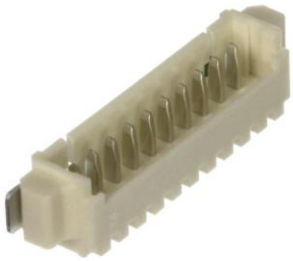


**Figure 13: Molex PicoBlade connector**