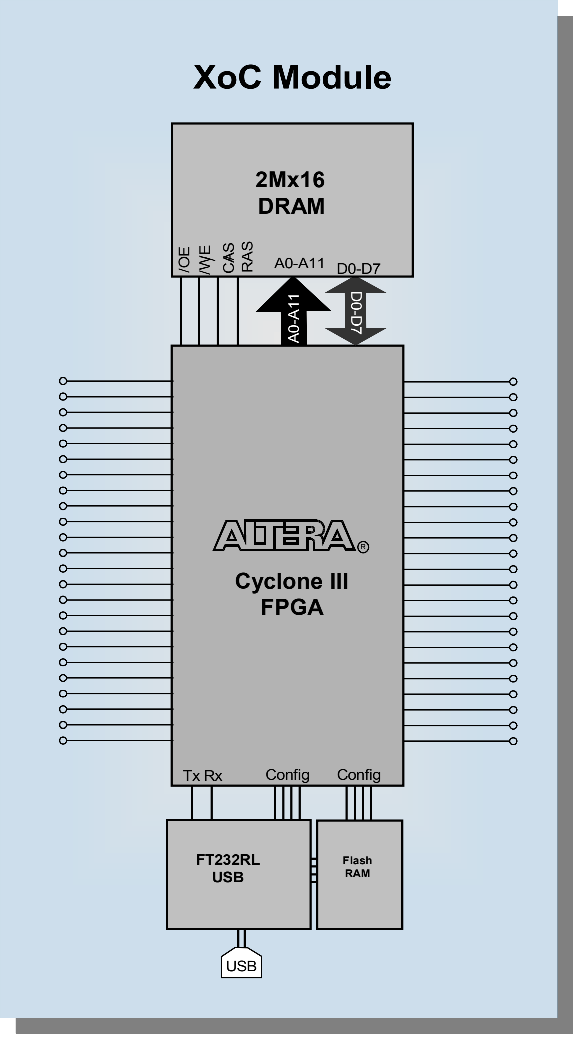
# The yPU System



***"Doing for Embedding processing what the Arduino did for Microprocessing"***

*If you have ever wished for "the perfect microprocessor Chip", this is it.*

*Imagine if you could custom configure a microprocessor to have the EXACT cpu and peripherals that you need for a particular application?*

*This becomes possible with the yPU. How?*

*By using an FPGA and embedding the exact CPU and components that you need.*

*But learning FPGA's is complicated... but not anymore, with the SoC construction kit you can configure **your MPU** (yPU) to your exact specifications.*

*When you design is ready for mass production, you can go from FPGA to ASIC easily using the FPGA Hardcopy Service.*

***The Ultimate Custom SoC Construction Kit***

*Donnaware International LLC*

# *yPU* is “Your” Microprocessor...

- Have you ever been working on a project and got to a point and just needed one more PWM, or two more DAC's, wish that you could delete the timers you don't need and replace them with some other interface?

- Have you ever wished you could get into FPGA's but the thought of learning Verilog or VHDL is just too daunting?

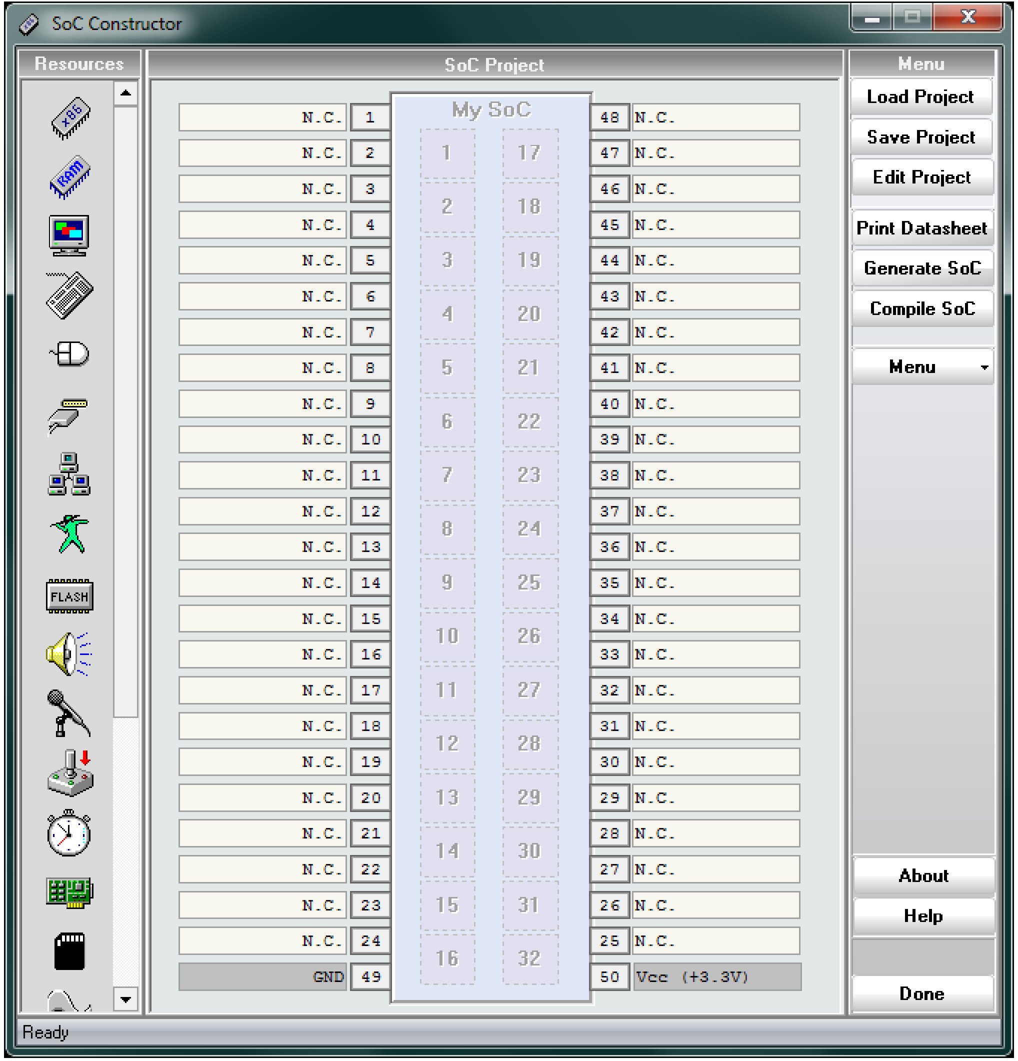
- Have you wished that you could get into embedding processors into an FPGA and perhaps even producing a custom ASIC chip for your product ?

- All these things and more are made easy with the yPU SoC construction Kit.

# The yPU Project

- **Design and Build your MPU using simple drag and drop user interface**
  - Use SoC constructor to drag and drop the desired components to your design
  - Drag components from the Resource palette and drop them on the you SoC
  - Up to 32 components can be populated on the current version of yPU
  - Then Generate and Compile your SoC MPU

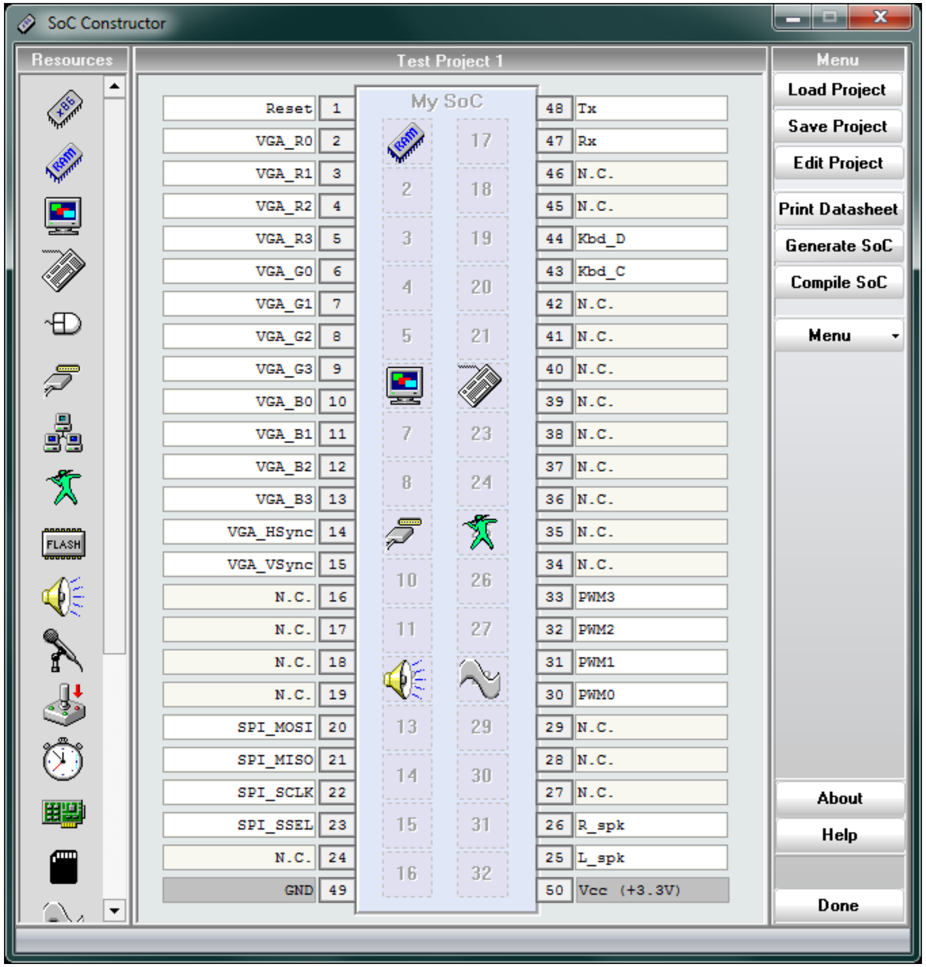
yPU SoC Constructor



|  |  |  |  |
|---|---|---|---|
| N.C. | 1 | 48 | N.C. |
| N.C. | 2 | 47 | N.C. |
| N.C. | 3 | 46 | N.C. |
| N.C. | 4 | 45 | N.C. |
| N.C. | 5 | 44 | N.C. |
| N.C. | 6 | 43 | N.C. |
| N.C. | 7 | 42 | N.C. |
| N.C. | 8 | 41 | N.C. |
| N.C. | 9 | 40 | N.C. |
| N.C. | 10 | 39 | N.C. |
| N.C. | 11 | 38 | N.C. |
| N.C. | 12 | 37 | N.C. |
| N.C. | 13 | 36 | N.C. |
| N.C. | 14 | 35 | N.C. |
| N.C. | 15 | 34 | N.C. |
| N.C. | 16 | 33 | N.C. |
| N.C. | 17 | 32 | N.C. |
| N.C. | 18 | 31 | N.C. |
| N.C. | 19 | 30 | N.C. |
| N.C. | 20 | 29 | N.C. |
| N.C. | 21 | 28 | N.C. |
| N.C. | 22 | 27 | N.C. |
| N.C. | 23 | 26 | N.C. |
| N.C. | 24 | 25 | N.C. |
| GND | 49 | 50 | Vcc (+3.3V) |

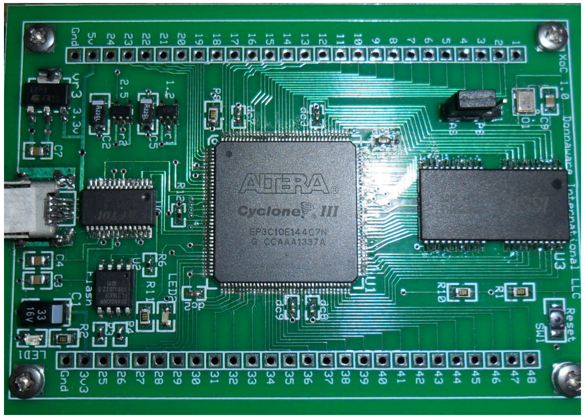
# The yPU Project

- Once your MPU Has been Generated, download the configurationinto the yPU development module and it assumes the characteristics you have selected.



| | | | |
|---|---|---|---|
| Reset | 1 | 48 | Tx |
| VGA_R0 | 2 | 47 | Rx |
| VGA_R1 | 3 | 46 | N.C. |
| VGA_R2 | 4 | 45 | N.C. |
| VGA_R3 | 5 | 44 | Kbd_D |
| VGA_G0 | 6 | 43 | Kbd_C |
| VGA_G1 | 7 | 42 | N.C. |
| VGA_G2 | 8 | 41 | N.C. |
| VGA_G3 | 9 | 40 | N.C. |
| VGA_B0 | 10 | 39 | N.C. |
| VGA_B1 | 11 | 38 | N.C. |
| VGA_B2 | 12 | 37 | N.C. |
| VGA_B3 | 13 | 36 | N.C. |
| VGA_HSync | 14 | 35 | N.C. |
| VGA_VSync | 15 | 34 | N.C. |
| N.C. | 16 | 33 | PWM3 |
| N.C. | 17 | 32 | PWM2 |
| N.C. | 18 | 31 | PWM1 |
| N.C. | 19 | 30 | PWM0 |
| SPI_MOSI | 20 | 29 | N.C. |
| SPI_MISO | 21 | 28 | N.C. |
| SPI_SCLK | 22 | 27 | N.C. |
| SPI_SSEL | 23 | 26 | R_spk |
| N.C. | 24 | 25 | L_spk |
| GND | 49 | 50 | Vcc (+3.3V) |

yPU Development Module



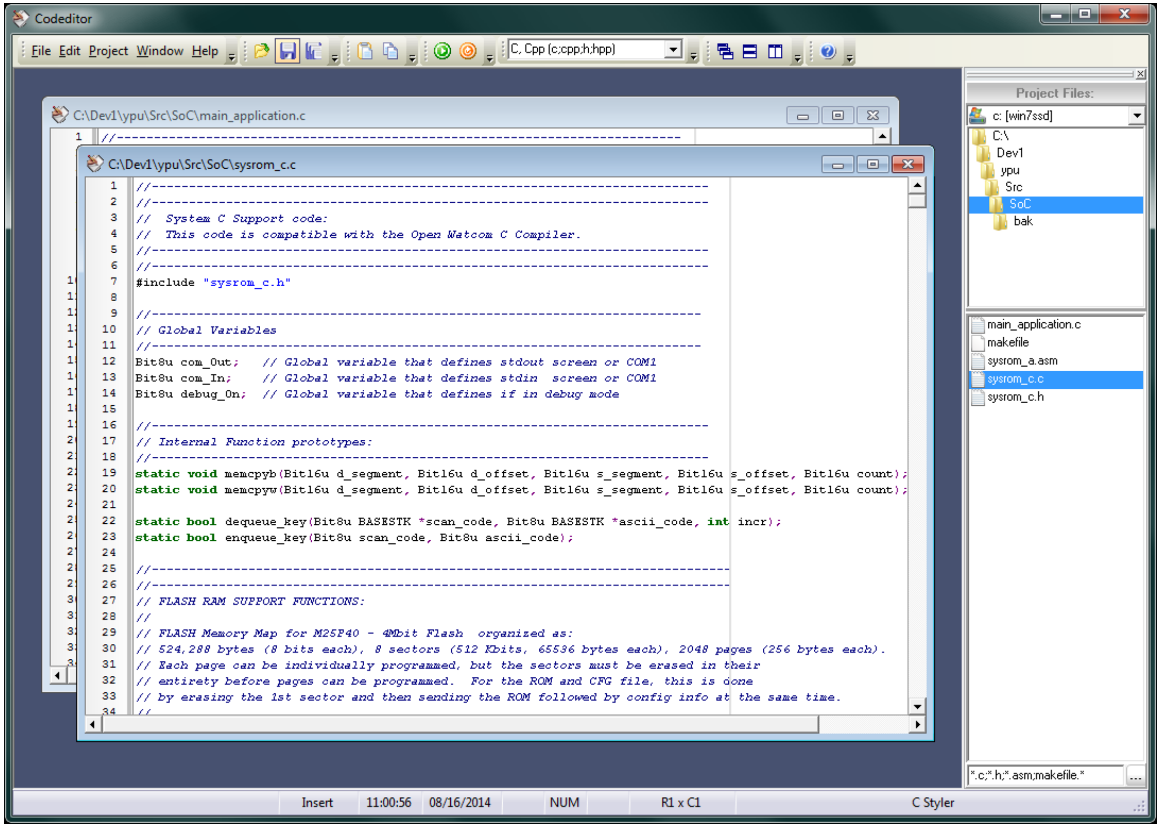
*The Ultimate Custom SoC Construction Kit*

*Donnaware International LLC*

# The yPU Project

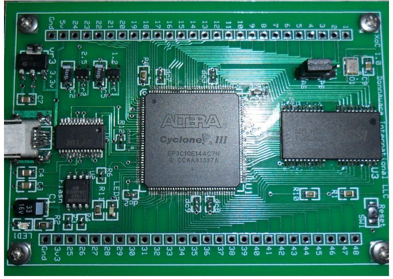
- **Now that your MPU is configured the way you desire, use the yPU IDE to develop your code using C or Arduino Sketch.**

yPU Integrated Development Environment



```
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//  System C Support code:
//  This code is compatible with the Open Watcom C Compiler.
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
#include "sysrom_c.h"

//--------------------------------------------------------------------------
// Global Variables
//--------------------------------------------------------------------------
Bit8u com_Out;   // Global variable that defines stdout screen or COM1
Bit8u com_In;    // Global variable that defines stdin  screen or COM1
Bit8u debug_On;  // Global variable that defines if in debug mode

//---------------------------------------------------------------------------
// Internal Function prototypes:
//---------------------------------------------------------------------------
static void memcpyb(Bit16u d_segment, Bit16u d_offset, Bit16u s_segment, Bit16u s_offset, Bit16u count);
static void memcpyw(Bit16u d_segment, Bit16u d_offset, Bit16u s_segment, Bit16u s_offset, Bit16u count);

static bool dequeue_key(Bit8u BASESTK *scan_code, Bit8u BASESTK *ascii_code, int incr);
static bool enqueue_key(Bit8u scan_code, Bit8u ascii_code);

//----------------------------------------------------------------------------
//----------------------------------------------------------------------------
// FLASH RAM SUPPORT FUNCTIONS:
//
// FLASH Memory Map for M25P40 - 4Mbit Flash  organized as:
// 524,288 bytes (8 bits each), 8 sectors (512 Kbits, 65536 bytes each), 2048 pages (256 bytes each).
// Each page can be individually programmed, but the sectors must be erased in their
// entirety before pages can be programmed.  For the ROM and CFG file, this is done
// by erasing the 1st sector and then sending the ROM followed by config info at the same time.
//
```

yPU Development Module



Download

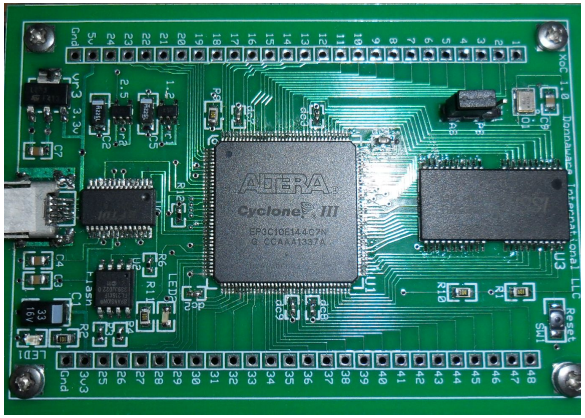
*The Ultimate Custom SoC Construction Kit*

*Donnaware International LLC*

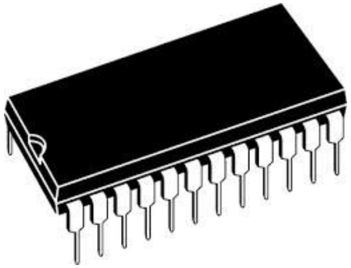
# The yPU Project

- Once you are happy with your MPU Design, if you want to take it to that next level and go into large scale production, that is possible using the FPGA Hardcopy service

yPU Development Module

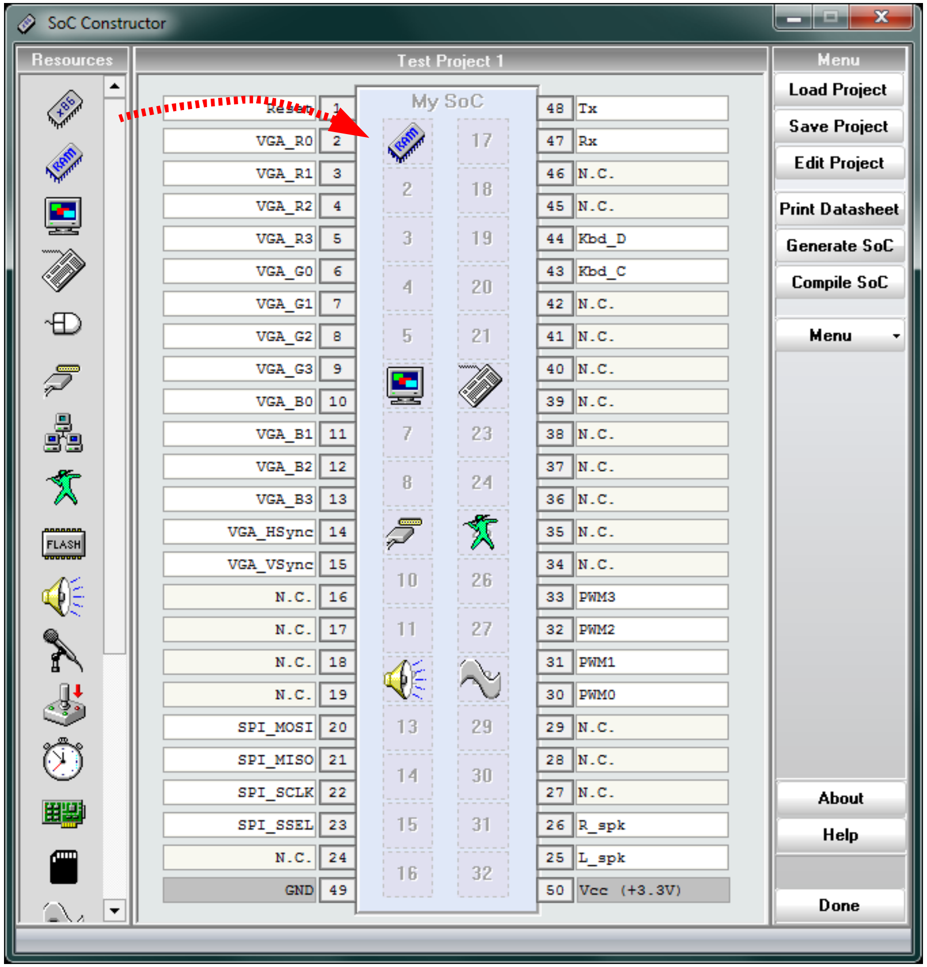


yPU ASIC “Hardcopy”



# *With yPU SoC Constructor...*

- There is NO need to know any Verilog or VHDL...
- Simply pick the desired resource modules from the resource pallete...
- Once you have just the right set of components for your application...
- Then just click on the “Generate SoC” button and your Custom SoC will be generated in just a couple of minutes...



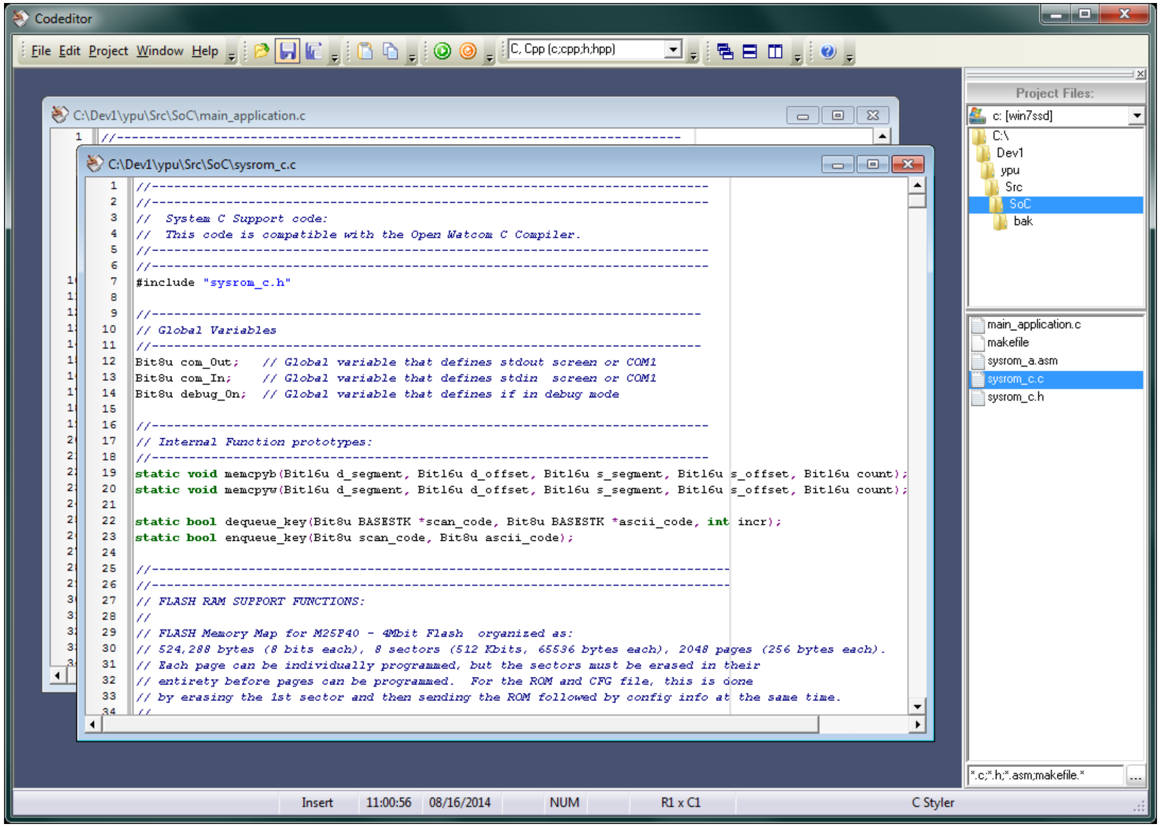
| | | | |
|---|---|---|---|
| Reset | 1 | 48 | Tx |
| VGA_R0 | 2 | 47 | Rx |
| VGA_R1 | 3 | 46 | N.C. |
| VGA_R2 | 4 | 45 | N.C. |
| VGA_R3 | 5 | 44 | Kbd_D |
| VGA_G0 | 6 | 43 | Kbd_C |
| VGA_G1 | 7 | 42 | N.C. |
| VGA_G2 | 8 | 41 | N.C. |
| VGA_G3 | 9 | 40 | N.C. |
| VGA_B0 | 10 | 39 | N.C. |
| VGA_B1 | 11 | 38 | N.C. |
| VGA_B2 | 12 | 37 | N.C. |
| VGA_B3 | 13 | 36 | N.C. |
| VGA_HSync | 14 | 35 | N.C. |
| VGA_VSync | 15 | 34 | N.C. |
| N.C. | 16 | 33 | PWM3 |
| N.C. | 17 | 32 | PWM2 |
| N.C. | 18 | 31 | PWM1 |
| N.C. | 19 | 30 | PWM0 |
| SPI_MOSI | 20 | 29 | N.C. |
| SPI_MISO | 21 | 28 | N.C. |
| SPI_SCLK | 22 | 27 | N.C. |
| SPI_SSEL | 23 | 26 | R_spk |
| N.C. | 24 | 25 | L_spk |
| GND | 49 | 50 | Vcc (+3.3V) |

# *yPU* Integrated Development Environment

The SoC Constructor will automatically generate a skeleton C Code Project with the appropriate drivers and stub code to get you started, the code is based on the Arduino Sketch type structure...

The yPU IDE (Integrated Development Environment) allows you to perfect and compile your code...

Once your code is ready to test, just click on the Project Menu and pick the Programmer or Special Fterminal to upload your FPGA configuration and your program...



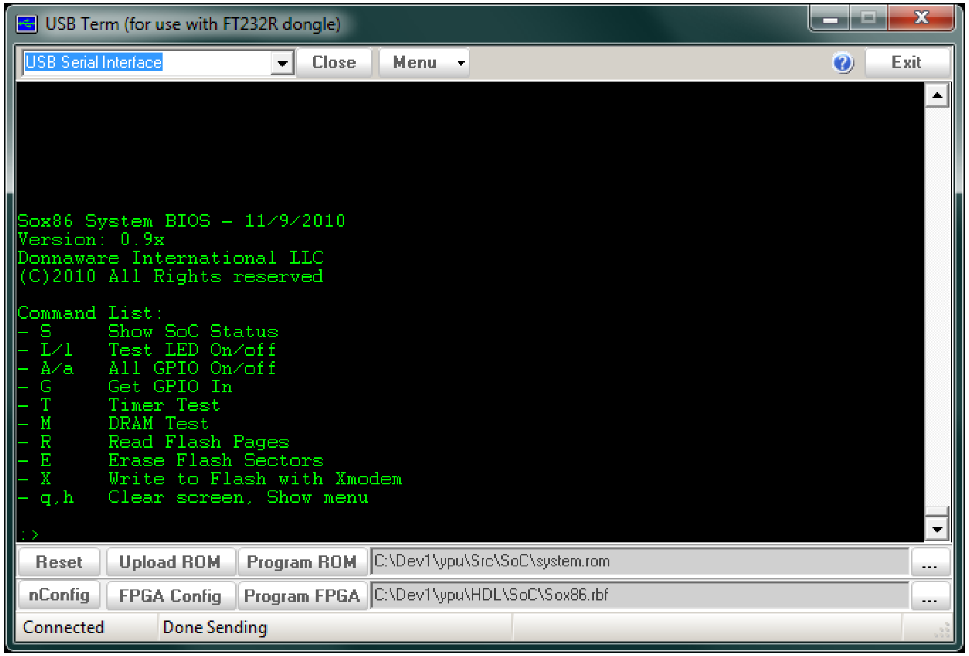
```
//------------------------------------------------------------------------------
//------------------------------------------------------------------------------
//  System C Support code:
//  This code is compatible with the Open Watcom C Compiler.
//------------------------------------------------------------------------------
//------------------------------------------------------------------------------
#include "sysrom_c.h"

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------
Bit8u com_Out;   // Global variable that defines stdout screen or COM1
Bit8u com_In;    // Global variable that defines stdin  screen or COM1
Bit8u debug_On;  // Global variable that defines if in debug mode

//------------------------------------------------------------------------------
// Internal Function prototypes:
//------------------------------------------------------------------------------
static void memcpyb(Bit16u d_segment, Bit16u d_offset, Bit16u s_segment, Bit16u s_offset, Bit16u count);
static void memcpyw(Bit16u d_segment, Bit16u d_offset, Bit16u s_segment, Bit16u s_offset, Bit16u count);

static bool dequeue_key(Bit8u BASESTK *scan_code, Bit8u BASESTK *ascii_code, int incr);
static bool enqueue_key(Bit8u scan_code, Bit8u ascii_code);

//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
// FLASH RAM SUPPORT FUNCTIONS:
//
// FLASH Memory Map for M25P40 - 4Mbit Flash  organized as:
// 524,288 bytes (8 bits each), 8 sectors (512 Kbits, 65536 bytes each), 2048 pages (256 bytes each).
// Each page can be individually programmed, but the sectors must be erased in their
// entirety before pages can be programmed.  For the ROM and CFG file, this is done
// by erasing the 1st sector and then sending the ROM followed by config info at the same time.
//
```

```
//------------------------------------------------------------------------------
```

# *yPU* Development

- Simply connect the yPU development module to you PC and run FTerminal from the IDE.
- The FPGA Configuration file can be temporarily uploaded to the yPU development module, this turns it into your custom SoC.
- It is now ready to receive your software, click on upload ROM and your program is uploaded into the yPU and executed. In both cases everything is running from within Static RAM.
- Once everything is perfected, you can program your SoC config and your software into the Flash RAM at which time the module is stand alone and will automatically boot up to your project.

FTerminal USB Terminal and Programmer



```
Sox86 System BIOS - 11/9/2010
Version: 0.9x
Donnaware International LLC
(C)2010 All Rights reserved

Command List:
- S     Show SoC Status
- L/l   Test LED On/off
- A/a   All GPIO On/off
- G     Get GPIO In
- T     Timer Test
- M     DRAM Test
- R     Read Flash Pages
- E     Erase Flash Sectors
- X     Write to Flash with Xmodem
- q,h   Clear screen, Show menu

:>
```

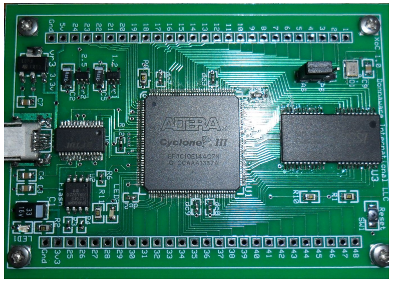
C:\Dev1\ypu\Src\SoC\system.rom

C:\Dev1\ypu\HDL\SoC\Sox86.rbf

Connected | Done Sending

USB Cable

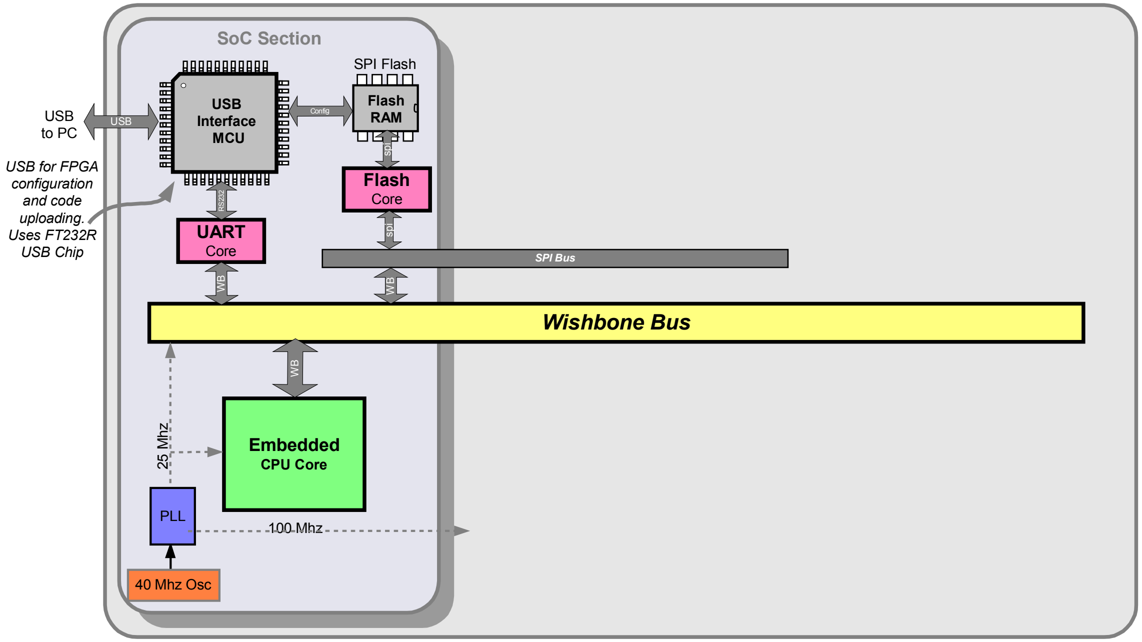
yPU Development Module



# *yPU* Architecture...

- Start out with the basic SoC when you drag and drop the CPU module, this gives you the basic CPU, Flash RAM and USB Comm port to get started.

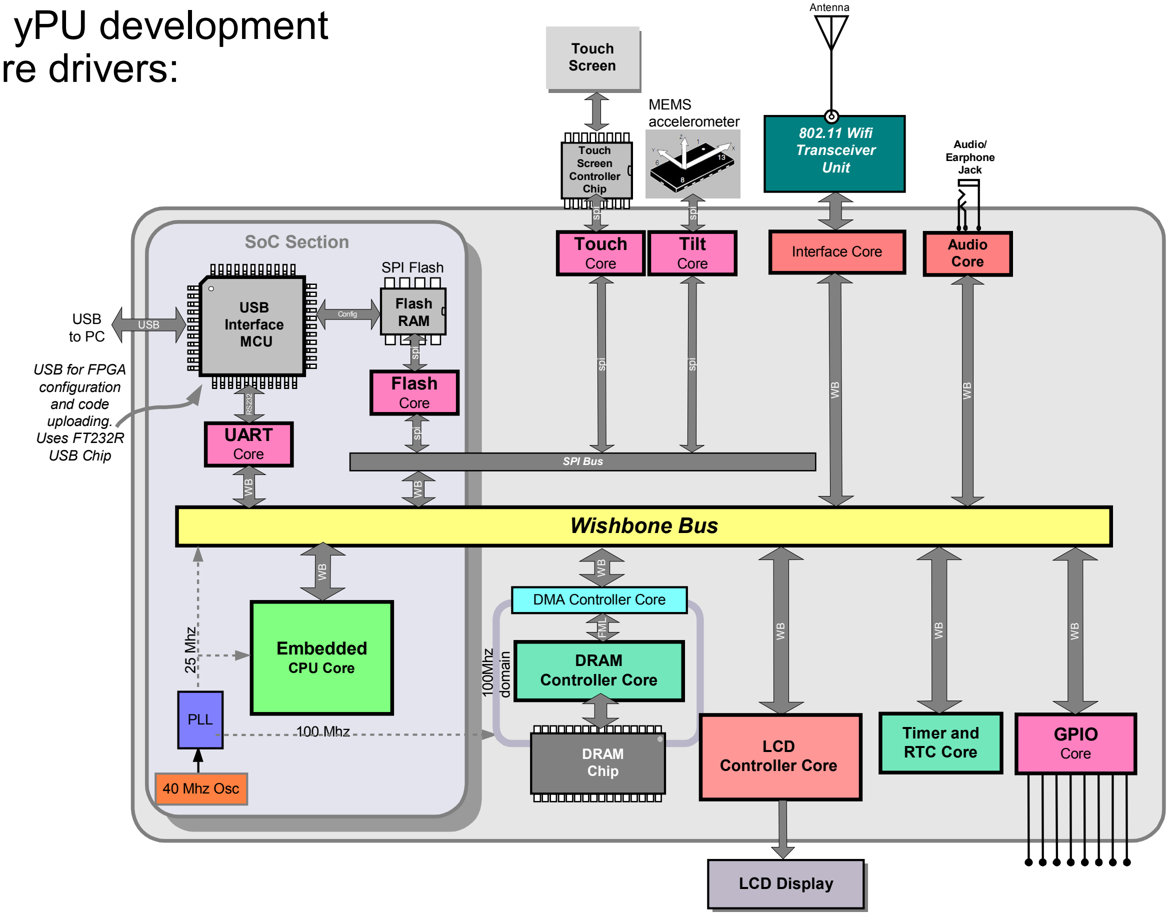
- From there, the Wishbone bus provides the framework to build upon using standardized open source IP (Intellectual Property) cores available from the opencores.org web site or from the yPU web site.
- Or add custom cores of your own.



# *yPU* Architecture

Plenty of cores come with the yPU development system that include the software drivers:

- CPU's
- VGA
- Graphic LCD
- Keyboard & Mouse
- RS232
- Ethernet
- SPI
- Sound
- Timer
- GPIO
- PWM
- And many more
- ….



# yPU Summary of Advantages...

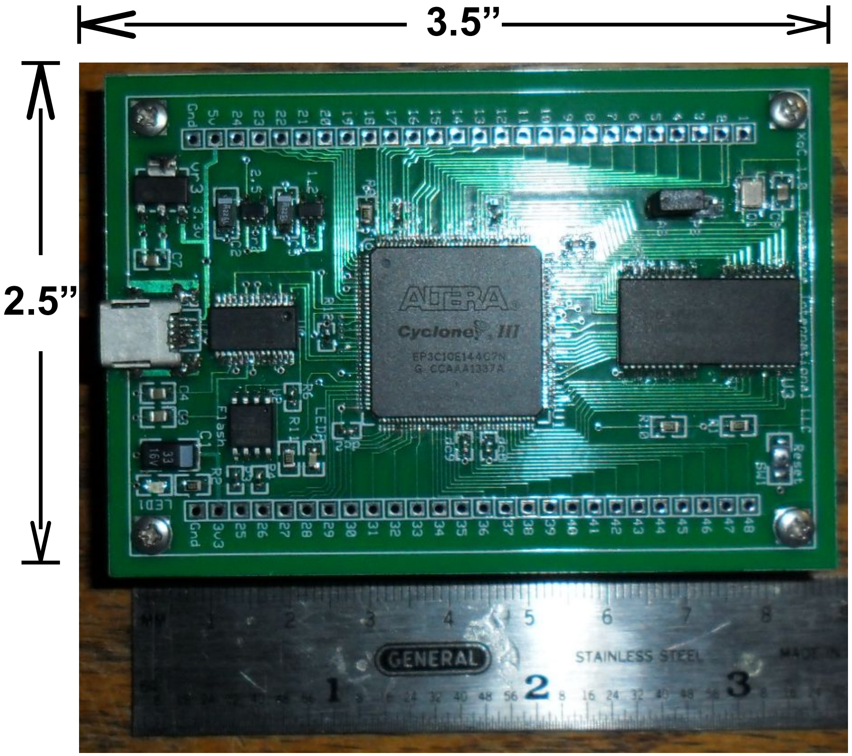
- Costs and part counts can be reduced because you can instantiate all the interfaces you need directly onto the FPGA without external chips or interfaces
- The GUI SoC Constructor Kit minimizes the effort involved with developing the FPGA embedded processor and interface cores.
- Get all the benefits of FPGA development without the effort and frustration and high development costs.
- Path to “Hardcopy” ASIC for mass production later means you can go big FAST.
- Or, re-use the Development module endlessly using the reprogrammable FLASH Configuration RAM, perfect for the inventor who does not want to stock piles of chips and parts.

# yPU Extensibility

- Future development may include additional CPU's, possibly even multiple CPU cores
- Since the cores are based on cores available from http://opencores.org/ and are based on the Wishbone Bus, the user can develop their own custom cores and link them into the SoC Constructor.
- Using the Altera Quartus software, additional cores can be developed using Schematic input, the user can draw a schematic using standard TTL gates and import that into their project using the SoC Constructor.

# yPU proof of concept Prototype

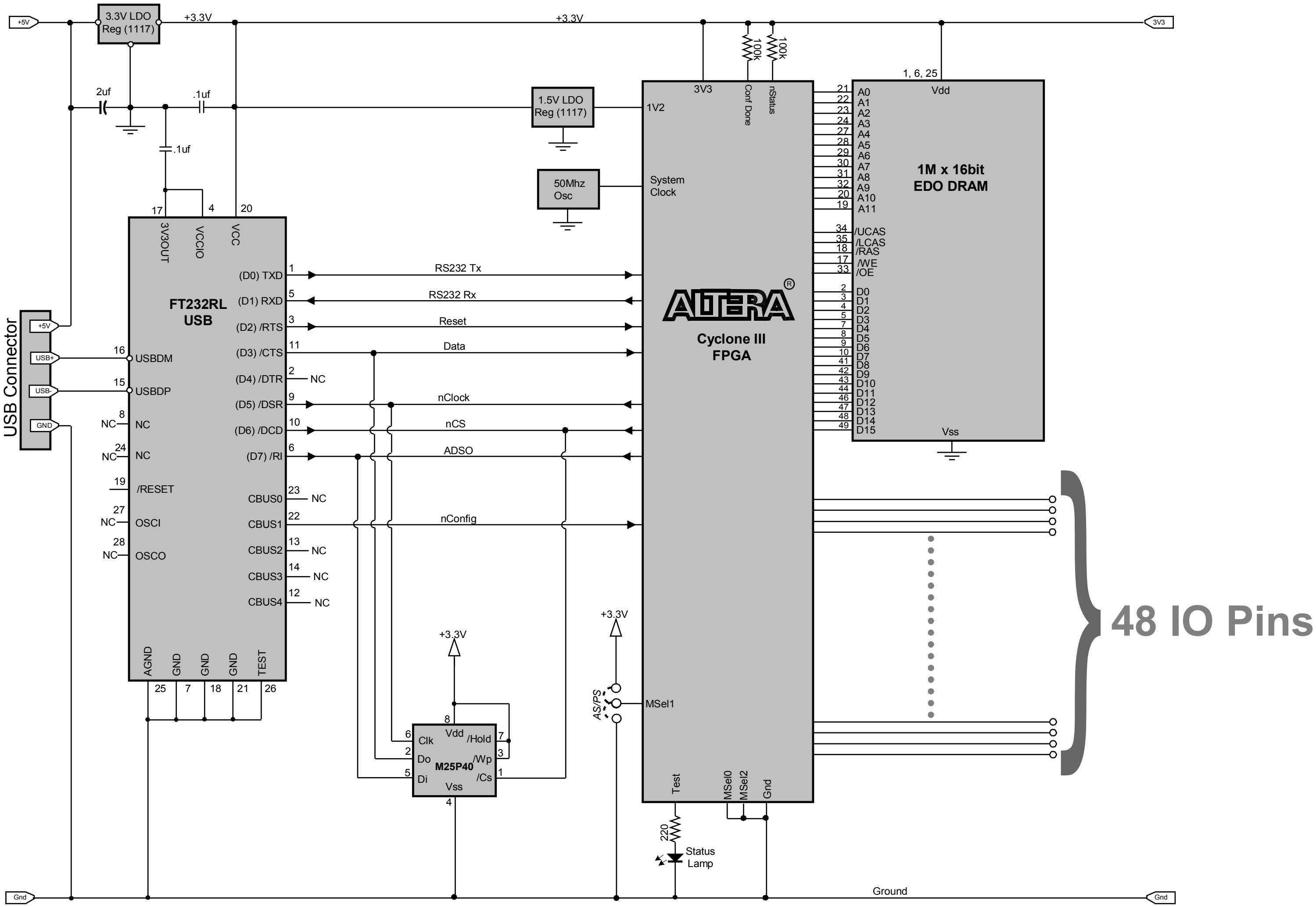
*Photo to the right shows a working “Proof of Concept” prototype that has been developed using a low cost Altera Cyclone III FPGA.*



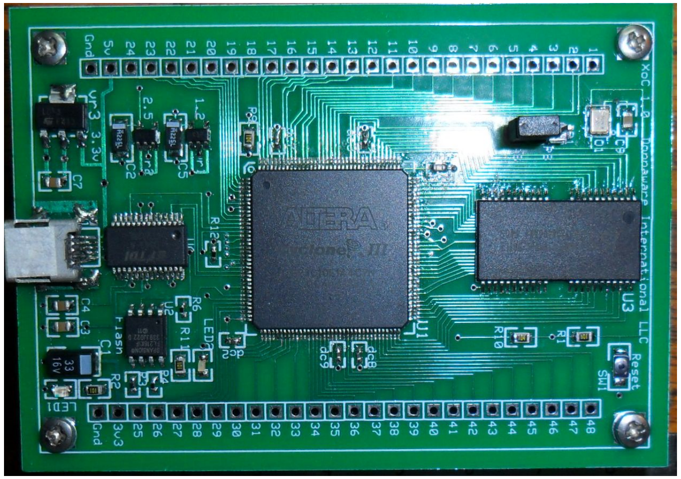
*The prototype unit is 2.5 x 3.5 x .25 inches.*

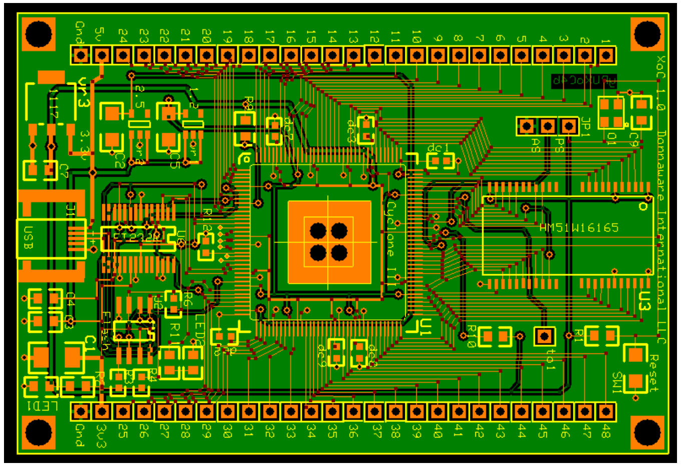
*The goal of the project is to shrink it to a module that is the same form factor as a standard 48 pin DIP IC with .1” pin spacing.*

# yPU Detailed Schematic Diagram



# yPU PCB and Photo's





Thank You for your attention.