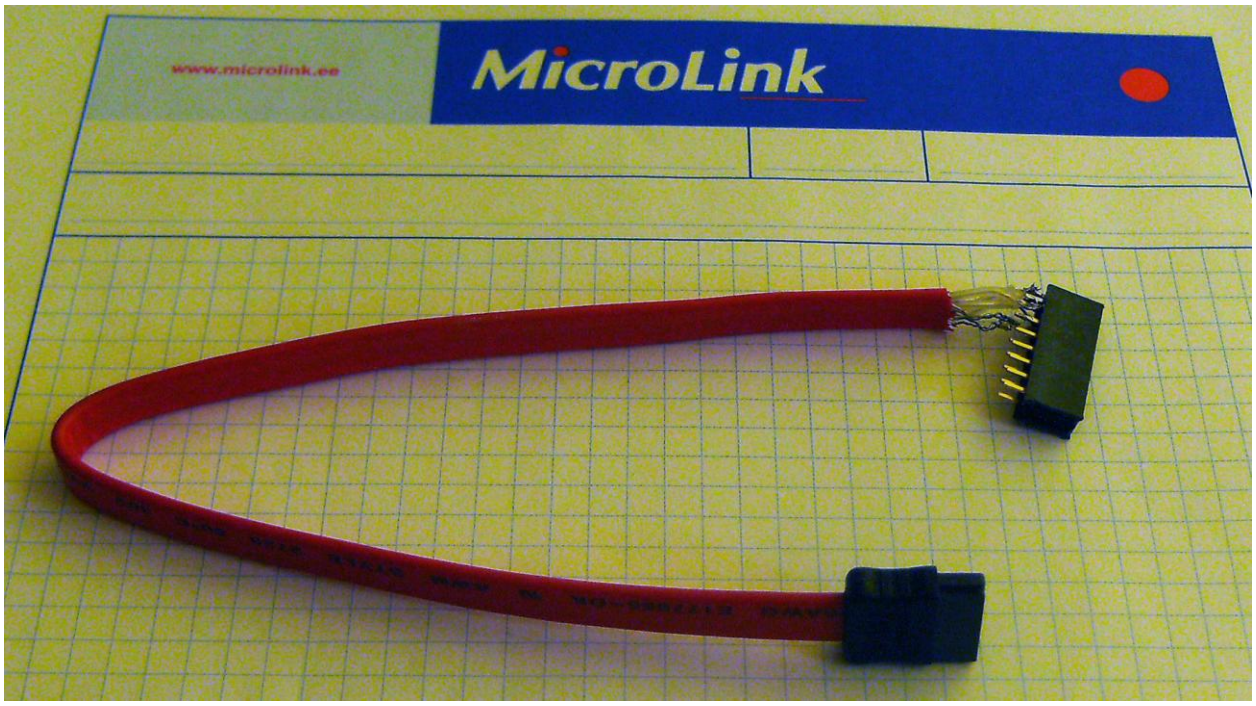
Antti-Brain

Issue 9

April 2009



Revised on April 27, 2009

Editorial

Why should the editorial story be technical?

Sitting in café, waiting for my son Andre to come from Karate lesson. On our way something happened what I cannot say what it was. I did see a well dressed man, dark grey suite in stripes, standing on wacky feet, with the help of eh don't know that thing disabled people use. He said instantly he did come to BEG for food, that he is only 2 months as using the leg prosthesis, and that his government aid pension is due on 5 Mai only. Well there was some big fuzz about those aids being delay lately in local press, but that was not it, what made me take out my wallet. He kept talking, with really shaky voice. I looked how he was apparently not able to stand steady, and did take out all the cash I had, wasn't much maybe 5 EUR in local currency, and one 20 EUR banknote. I looked at him once more, and gave that all, saying here, this is what I have, take it.

I know, I do have blue eyes. Chances that he was real are not so good. But that doesn't matter, he sounded real to me. The place was behind Estonian national opera, maybe he was an actor having a bet. Who knows. Funny, it was just about exactly 10 years ago, and about 100 meters away, when I entered a pub and said to the waiter, "I have no money, can you bring me something to eat?", he asked nothing, and accepted my order. It was not a big steak, but it was good and tasty meal. Then I was disabled(?) for a while, and well I never paid for that food. I did know that waiter by name, but had never actually talked to him before. Thank you, Andree, for that meal. At least I have said my thanks now.

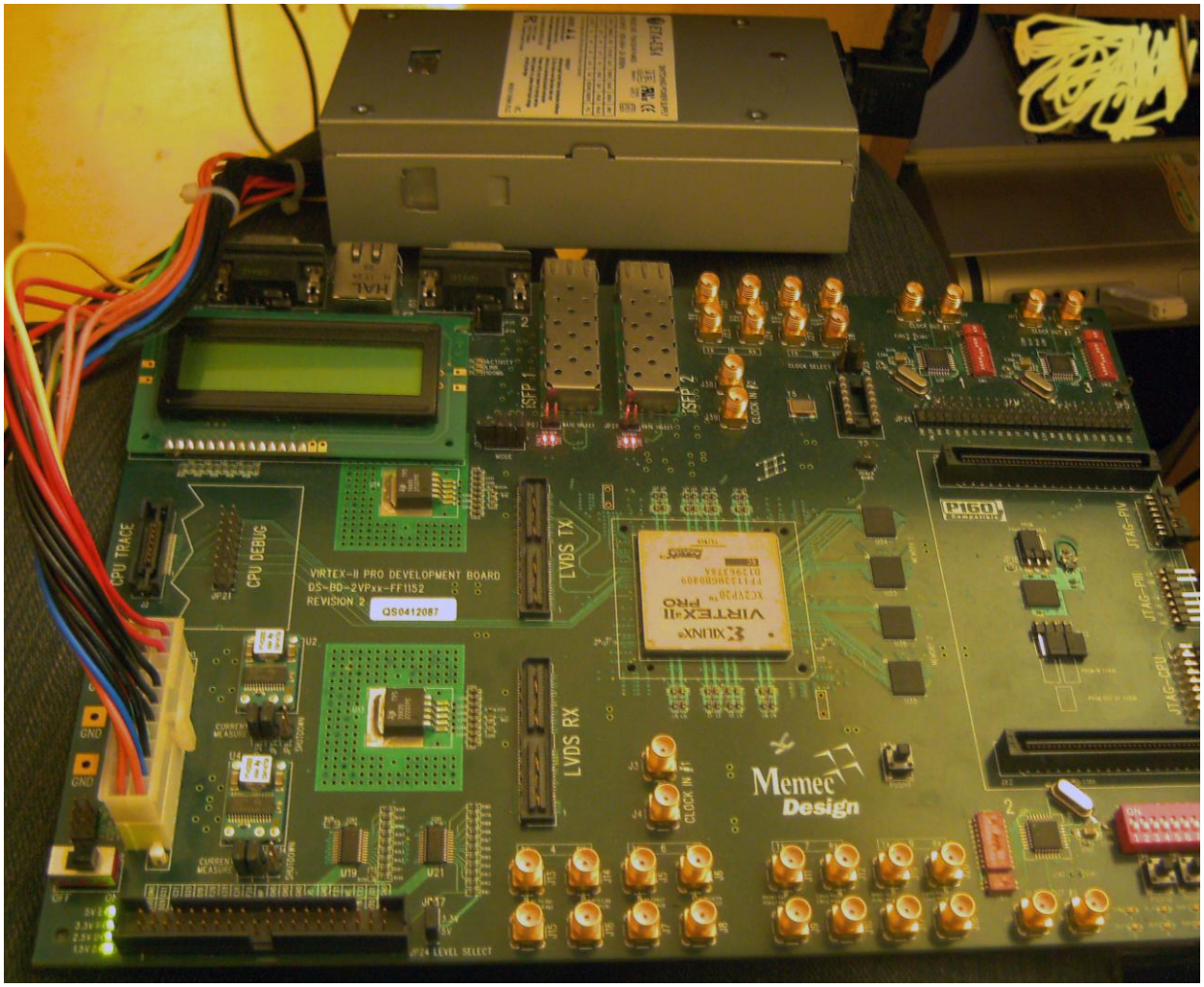
The story above should be small excuse that I am unable to write more today. Next month will be soon, and will bring new stories.

Antti Lukats
Antti.Lukats@googlemail.com

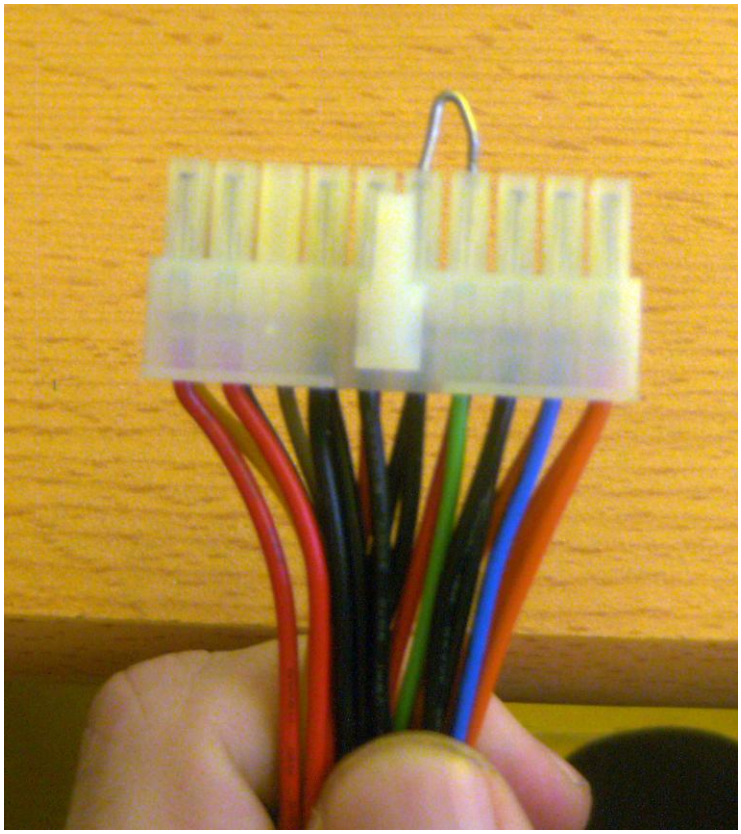
http://groups.google.com/group/antti-brain

Cover Story

What's on the cover? A special cable I used to make initial tests with Virtex-2Pro MGT's on ML300 board. I used a special IP core, that made a piezo speaker to buzz if connected directly to MGT outputs. Some while ago. Today? Well today April 27th Xilinx released ISE 11.1 and dropped all support for Virtex, Virtex-II, and Virtex-IIPro and also Spartan-II/E FPGA's. Is there still some use for Virtex2Pro devices? YES, there is! A real life example – a POWER SWITCH:



Using Virtex-2Pro VP20 as power on switch, the actual slide switch is located in the bottom left corner. The power is used for customer board with Virtex-4 (erased on the photo). Also visible the 16GB USB stick with the SVN repository.



A better power switch, the Virtex 2Pro FPGA has been replaced successfully!

My Own Processor I

Ever wanted to make one? Tried and failed? Or succeeded?

Here I have a Real-Life-Story: Soft Processor for A3P060 based FPGA-Stamp board. Initial design consideration was toward PIC165X like core, as it was "small enough" to fit the target device. A lot of work was done, then left on ice. Next candidate AVR-like something. Trying out AX8 out of the box code, and well it is larger than PIC core, but it would still fit. And it would allow me to re-use my AVR code, and use AVR Basic Compiler to write programs for the SoC.

Testing out of the box on Xilinx Spartan3A Starterkit, ISE, generate bitstream, download, works LED blinks! Ok now it's time to optimize the system.

Taking out AVR Ports, replacing with MicroFPGA ASIO style bit addressable I/O. The target board has only 30 I/O so I just allocate I/O addresses 0x00 to 0x1D to pins, so that Bit D0 only is used. Writes make the Pin output and update value, reads will convert the pin to input and read the pin state. Well there is small side effect, first read will return the pin value before it was made to input. But this design yields to smallest code both on hardware and software side.

On-Board LED gets allocated to I/O Address 0x1F, so there is still one lower half I/O Address free 0x1E.

Making new LED blink test software for this new hardware, testing works. Does it also work on the Actel board? Yes it does. Eh, it's getting bored when everything just works ☺

A3P060 has no Block RAM that can be initialized so I need to figure out how to bootstrap it. There is 128 byte FlashROM (only writeable over JTAG), so this one can be used, it would give 64 instruction small Bootloader.

Decision to make – UART Bootloader, should it use hardware UART, or is better (as of FPGA resources) to have it done as software only?

Ok, I still have the AVR 8-Bit Timer-Count in the system, so let's test if the interrupts works. And they do. However the timer takes rather large amount of versatiles and if I plan to use it mostly for timer TICK only then simpler one would do as well.

There is another concern, the FlashROM on A3P only works max 10MHz clock, so if the systems has to be initialized from FlashROM there is a need for a low clock. Possibilities are first setting main system clock to 10MHz or below, using second slow clock for the FlashROM (requires clock mux) or then using same approach as I already use in some MicroFPGA designs, a NCO generated clock that is always starting at slow clock on power-on so the processor can speed itself up, if desired.

The NCO phase accumulator could be made longer than needed, that is taking the system clock from the middle of the accumulator. Then the higher bits could be used as TICK source. And also as free running counter for PWM ☺

Timer interrupts can be simplified as well, only enable-disable bit is needed and prescaler select bits. No need to read or reset flags. Ok, removed the AVR Timer, added the tick counter, modified the interrupt logic. Resource utilization (Xilinx) was before the modification 516/811/237 (slice/lut/ff), after modification: 481/765/209! Ok, the modified version has fixed prescaler so we just can't compare directly. Synplify resource report for the TC8 timer block was 78 Actel versatiles what isn't that much, but as there are only 1536 versatiles total, there would be better use of the resources. Maybe I could even used the fixed prescaler? Need calculate the NCO Value and possible clock frequencies, and make a table of UART baud rates that are obtainable by changing NCO (main clock) only and fixed tick timer prescaler.

But does the new tick timer work also? Downloading bitfile, and yes the LED blinks again!

Well the target FPGA has still some resources free, so need fill them out also, things planned are: JTAG communication support (internal TAP access to the processor), PLL with dynamic control, and then the fun begins, filling out the rest, there are possible some more resources available to add more I/O multiplexing, maybe some long counter that would help frequency measurement. Ah yes I also wanted to add 4 bit PWM to control the on-board LED.

Oh, but before that I should finalize the shadow bios to RAM copy engine, and test it out in hardware too. That is a must, all the other goodies are now pretty much optional.

Ha! This interesting.. I do from time to check out "what's new" on Atmel, and now it did display ATtiny10, and well, I had just a this morning decided to map the ROM into RAM space, and voila, the ATtiny10 does it as well, mapping the ROM to RAM space at some offset. To my best knowledge all (or most other at least) AVR before only access ROM using LPM/SPM special instruction. But RAM mapping is much more easier to implement in FPGA so hence I had had selected the direct mapping.

Hm.. ATtiny10 must be really dirt cheap, SOT-23-6 package, debugWire is taken out, no IAP and flash programming only at 5V.

Doing some more modification, and oh no, it not working any more. I am backing up, trying to take older sources, troubleshooting like mad-dog… and gosh, I had connected reset input to slide switch what was in left in RESET position, no wonder the application didn't start. But does the new version also work in Actel? Yes, it still does. Ok, still some code to add.

Adding the ROM/Flash BIOS Shadow copy to RAM functionality, testing, and working. Still works in Actel too? No ☹

Hum, maybe just maybe the problem is found, my target board has no reset circuit, so it depends on proper Reset-On-Configuration. As one of last changes I added a I/O register to selects the core frequency (input to NCO), this register should be initialized to minimal value at reset, what is generated internally. The blink LED however was ON with >50% intensity. But on some power on attempts the blinker blinked correctly in sync with the same design running in Xilinx FPGA.

So I need to enhance the power on behavior! Hope it fixes the issues why the design didn't work. Well Actel FPGA's are live at power up so there should be no need to force some long reset to settle something. Ha, Actel had forgotten the pull-up attribute on the reset pin, so the design was all working!

```
Compile report:
===============

    CORE                      Used:   1332  Total:   1536   (86.72%)
    IO (W/ clocks)            Used:     33  Total:     80   (41.25%)
    GLOBAL (Chip+Quadrant)    Used:      3  Total:     18   (16.67%)
    PLL                       Used:      0  Total:      1   (0.00%)
    RAM/FIFO                  Used:      4  Total:      4   (100.00%)
    Low Static ICC            Used:      0  Total:      1   (0.00%)
    FlashROM                  Used:      1  Total:      1   (100.00%)
    User JTAG                 Used:      1  Total:      1   (100.00%)
```

It looks I can soon stop adding more features, the chip is mostly full already ☺

Things planned to-do still: Connecting the PLL (with dynamic control port) and some multiplex of the input and output of the PLL, adding selector for the timer tick.

If those are done, and it still works, then maybe run a few rounds to optimize it, and look what else can fit.

Interesting, first time trying the shadow BIOS copy on Actel, does it? Yes it works! Cool.

But blink rate is different than on the Xilinx parallel test?

Ok, rewriting test without interrupts, testing again, blinking is in sync. But now I have too long test cycle on Xilinx as I don't have data2mem initialization for the "BIOS" memories. This would speed up development of the firmware. Yes, done working now it is easy to compile and update the bitstream, and test same SoC system as it targeted for the Actel board. Well not fully, I have 2 small memories, each 128 bytes, but EDK BMM file doesn't seem to allow this kind of memories to be combined into one block. So I can only initialize one of them. Still better than running Xilinx implementation flow on each soft change. Reading data2mem manual, eh looks it is possible to define any size of blocks, so trying out new BMM setting. O yeah, Xilinx documentation, the latest data2mem document is from 2007, and ADDRESS_SPACE keyword is plain missing in the formal syntax. And it doesn't work as described in the manual either. So the minimum block size is fixed to 2Kbyte, and I cannot set it to 128 byte using MEMORY keyword as the manual suggest. After some experiments ISE doesn't complain anymore. But doesn't do what I want either, the MEMORY keyword specifies user memory type, and the BRAM location do not get located and are unusable. I must define both memories as 2KB size and adjust the MEM file to write both block to right addresses (so telling a lie to data2mem to force proper alignment).

Now this is funny, the design still fully works on Xilinx, but Synplify for Actel does optimize 95% of the design away? Not yet clear why. Ok, taking some older version (without some optimizations) of the AVR core, now Actel also works with test program, so I need step by step investigate where Synplify goes too aggressive with the optimizations. Ok, reverted back, and have redone most optimization of dead code,

Actel version still working too. Added the Dynamic PLL, ha tricky, it is feed by the output of the NCO that generates the processor clock, so it is possible to select many PLL output frequencies without reprogramming it. Can I use the multiplexers also in the PLL tile? I wanted to have the processor core to run from non-PLL generated clock (leaving PLL for the user to generate any output clock desired, including in frequency range unsuitable as core clock). Seems I can still use the PLL tile clock multiplexers, by feeding the NCO output to all PLL tile clock inputs, and selecting CLKC pass-through to GLC at power up, that saves me the trouble of adding own clock multiplexer and control circuitry. So the system clock is coming via PLL block, but does not depend on the PLL being locked or having valid frequency output.

For the core dead code optimization I am doing now a Excel table with all AVR instruction bits, trying to look where I can reduce the decode widths. Another optimization, the Actel version only pushes output register to I/O for the LED output, but the other flip-flops are not pushed into the I/O registers. That could save another 60 tiles. Maybe some Actel tool option? No, it was more than that for the I/O packing the data and tristate registers must be controlled by same net for clock enable for Actel. So that was preventing the register packing.

Hey, now the registers are packed to I/O's on Actel too, but what's this, half of the pads are un-assigned? Seems some weird restriction prevents some of the I/O's to use registers packing so only 17 from the 30 I/O's can be placed to needed located when selecting register packing. Ok, it still some saving in tile count.

So now moving on to the BIOS, or factory Bootloader programming. Writing code (in basic using my own compiler!), testing in simulations. Looks good, using Excel table to calculate NCO setting and delay value for 9600 baud. Assigning some I/O's to UART pins on Xilinx Spartan3A Starterkit board – error during map? Ah they used a input only pin for RXD and my design has all I/O's bidirectional. Ok I need little modify the top-level for the FGPA test on that board.

And after some tweaking of the code, we have UART echo back working. Software UART transmit and receive routines take 24 instructions total, if we think that UART functions for hardware UART would take 6 instructions, then the software implementation takes 18 instructions only. This is pretty nice, I wasn't sure if to implement the UART in hardware or not, as both the FPGA and ROM resources are low in Actel FPGA.

Well, the first Bootloader is ready, it is really tiny, 45 instructions total, this includes the software UART receive and transmit functions!

Following commands are defined:

- @..F : X = X << 4 | (rxchar & 0x0F);
- $: Z = Z << 8 | X;
- !: RAM[Z] = X;
- #:RAM[Z++]=X; increment low order byte of Z only (saves one instruction!)
- s: start user application, or call user loaded code

This is pretty minimal, but it allows to set I/O state of all pins, to initialize the NCO, initialize the PLL, fill the instruction ROM and execute code. Well the auto increment command is optional of course, could also be removed, but it would only save 2 instructions. I wonder if I could squeeze UART slave and SPI master mode code both into 64 instructions? Sounds doable, then I would probably drop the ROM BIOS, and keep only the 128 Byte Flash BIOS in the system. It would save some FPGA resources that could be used to add more peripheral features.

Another round of the Bootloader optimization, I better drop the write no increment, and keep the post increment write only. And also drop the start application command, as it can be done by overwriting the loader executable code as it is RAM copy of the flash BIOS. What else? Actually if badly needed then the UART transmit can be dropped also, the bootstrap could as first thing load the transmit routine, and extend the command parser, if desired. This would give extra space. Eh fitting a dual mode Bootloader into 64 instruction ROM isn't that easy!

Ah, the Bootloader design isn't using the timer tick interrupts at all, so if needed all the interrupts could be taken out. At the moment the fixed prescaler timer tick interrupt is still present.

Another round of testing, eh the AX8 core is much buggy, the LD command decode instruction was wrong as example. Had to fix it to be able to read write the ROM mapped to RAM space.

Testing the FlashROM BIOS on A3P060, after some struggle it work, small test program is loaded from the FlashROM and it blinks the LED. I did manually enter the HEX values from the listing into the UFC file, what is simple XML file. Need to make a converter for this. Resource use went a little down being at 1249 tiles (from 1536 total) so I have some more room for peripheral and special features.

Hum.. this story is already too long, isn't it? I better stop here. There is still some resources left, what can be filled in. The product should soon be available so all the rest will be revealed then too.

Organize my Life™

Going slow, but going on. One 300GB HDD is transferred to the big junk 1TB drive. I stupidly started the copy in FAR so it did take overnight and half a day also.

What now? I have introduced version control to several companies where I have been before, but despite some attempts never used it for my own projects. Tried a few times but never long enough for it to pay off. But it really is a must, also if you are just one person, well I do have some co-workers too, so it even more important.

Choice to make – what is simplest and best way to use a version control system/repository?

I could setup SVN on my own leased server, done that once, but then I would need a internet connectivity always to access the files. Same for any other web hosted service.

So I do get a 16GB USB memory stick (32GB cost more than twice the 16GB price) and dedicate it for SVN repository only. Creating folder named SVN, and a SVN repository there. Now on the PC side, on all PC's I use for development I create folders named C:\prj\SVN\ and place all projects that are under version control there. Before doing any SVN commands on the PC's I use

SUBST S: X:\

Where X is the drive letter of the USB stick. This is possible not the most elegant way of doing it, but it works nicely. Drive S is now always the "SVN" repository drive.

Adding first projects to the repository, eh this feels good. Really. And making a backup is of the projects is now easier too, I just need backup the memory stick completely, even if it comes almost full it still be only 16GB. Or better, I could just do full SVN update regularly on different PC's, that would create redundant backup as well, with less file copy overhead.

Ha, due to notebook crash the CAD tool had managed to make the one PCB document corrupt, restored from SVN ☺

Shorts

Just short!

Xilinx ISE 11.1

First I was very excited – sure I expected that Spartan-6/Virtex-6 support will be included, but then it wasn't! This means that Xilinx must be seeing delays with 6 silicon, and the support comes later than expected.

Time to first "Portability" error, 3 hours. Time to first fatal error, 3.5 hours. This is so much better than previous major releases where overage time to first crash has been less than 30 minutes.

Good/bad and/or new things

Good: Auto-BMM support, simply specify MEM file in VHDL, and BMM file is auto-generated. Nice!

Good: Spartan-6 BSCAN is now more similar to Virtex one, has 4 instances/chains.

Bad: "Module level utilization" does not longer show the BRAM usage?

Good: V-6 has new high speed partial reconfiguration primitive PPR_FRAME.

Good: free running clock is finally available from the STARTUP_SPARTAN6 primitive.

Good: S-6 suspend requests can be delay and synchronized by user logic.

Good: S-6 Slave SPI primitive! I guess that Xilinx finally added SPI slave configuration mode, Lattice and Silicon Blue have this mode already.

Good: S-6 I/O Serdes primitives.

Good: PLL in S-6.

Good(bad?): Hardwired memory controller in S-6, this means that S-6 cannot achieve performance for memory IP without hard primitive?

Bad: S-6/V-6 support is not included, comes in 11.2 only.

BAD: Spartan II and Virtex/II support is no longer offered.

References

- http://www.trioflex.com

Instead of adding the URL links at the end of each issue, I will be adding them to the Trioflex online link collection, so they can be updated more frequently.