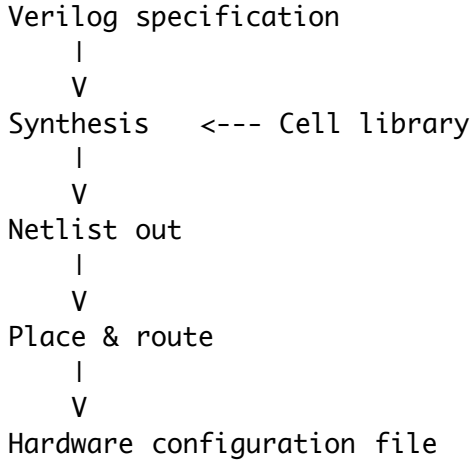
NOTES ON POSITIONING

William Bricken
May 2002

This is a collection of somewhat unrelated notes and comments about BTC technology and marketing, brought on by our latest technical design.

TOOL CHAIN INTEGRATION

Here's the current industry EDA methodology for FPGAs:



We do Synthesis and Place&Route with internal Pun tools. The Cell Library is Comesh cells.

To have some other EDA company do synthesis, we need to provide them with the Comesh libraries. You do not need Synopsis, etc. to do specification, all you need is a Verilog (VHDL) editor. In any event, we have to do place&route with our own tools.

Certainly we don't want to try to change engineers' habits, but I'm beginning to take a strong position that we want to restrict their design to what we need rather than what they do. We need a proper subset of what they do, so no new skills or behaviors other than doing less are required.

Someone once said that engineers will insist on verifying their designs by reintegrating them into the standard tool chain for timing analysis, etc. First: this appears to be a bad idea for us, since it forces a separation of BTC hardware and software, along with the attendant (huge) overhead of maintaining compatibility with every other EDA tool out there. Second (and very importantly!): reintegration checks timings etc. *only* for the competition's hardware. Comesh effectively removes the choice of timing, layout etc. from the engineers. So I see no gain in either using other

hardware's cell libraries (which we disassemble), or timing verification (which does not apply to our deterministic chip). Yes, we tell them how their designs work on Comesh, but this is all internal to BTC software.

So SUMMARY #1: it's a bad idea to put BTC software and libraries into other EDA companies' toolchains. At best we are investing resources to support our competition.

This does not mean we must be a EDA company as well. Certainly there are strategic choices such as spinning-off or buying a separate BTC-only EDA company.

OBSERVATION #1: Absolutely no one we have spoken to "gets it" wrt the power of the software when combined with the BTC hardware. The central point is that we can deliver what we hype: less design work. We should build our strategy to be consistent with this, i.e.: no attempt to integrate (degrade) into the former tool chain.

OBSERVATION #2: Engineers have little training in using FPGAs, much less in using them when they are well designed. We cannot give in to tool chain integration in order to foster behavior which negates our advantages.

OBSERVATION #3: Everything we said wrt our EDA capabilities 6 months ago is true.

Here's the new Comesh tool chain (same as the old BTC materials):

```
Verilog specification
    |
    V
Hardware configuration file
```

Although the internal steps may still be there, they are not only invisible, they are also not accessible. To be compatible with current design methods, we cannot make the internals "under design control", cause our internals are disruptively different. Hand-placement is not necessary if the software tools do as good a job as any human. Comesh effectively removes any advantage of hand-placement; hand-placement is necessary in the first place to correct weaknesses we do not have.

Also the EDA industry has been talking about some very fundamental problems for years. We have solutions to many of these problems (in the little domain of reconfigurable hardware), but degrading our solutions back into the existing problems to satisfy a miscomprehension about our tools seems to be rather undesirable. In particular,

-- *synthesis/place&route integration:* the old model of technology-independent synthesis does not work. Let's not buy into it.

-- *hardware/software codesign:* the two are not really separate any more. We do not treat them as separate, let's not force them apart.

-- *formal verification:* not possible without both formal software and codesign. We have both. Let's not pretend that the engineer must engage in hand layout and verification to fix non-existent problems.

-- *timing is 95% of design:* yes when your hardware is not deterministic, not fast enough, and/or too expensive. We have fast and deterministic, but only if we do not permit engineers' to try to second guess our software/hardware integration by doing timing themselves. All we need to do is to meet price points, the rest is automated. If Comesh does not meet a designer's specifications and needs, let them move to a better technology (the only one of which is ASICs).

-- *modern designs are too large and complex to design automatically:* true if you are using last century's logic. Remember that all existing EDA design software and all reconfigurable hardware architectures have dysfunctionality (not incorrectness) built-in, when seen from a BTC perspective.

-- *engineers must see the internal workings:* fine, let them, so long as we show them what is actually there, rather than back-translating to their old concept systems.

-- *you must be compatible with the existing toolchain:* why? unless you are providing designs to other chip companies. We are black-box compatible. The new perspective is, why do we want to translate into netlists that other EDA and hardware companies can use?

MARKET SLOGAN: "It's time to stop timing". This is a disruptive idea, the engineer no longer needs to "make the hardware work". The analogy in the software industry was the development of the first high-level languages. Hardware design is currently like writing in assembly language: intricate, specialized, very-low-level instructions to particular machines. Our product is like the first high level language: write what you want abstractly and the automated compiler will assure you that all lower levels work. The automated optimizer will assure you that all lower levels work efficiently.

We are saying to management: you can reduce you development costs by 2/3. We are say to engineers: much of the work you currently have to do to get FPGAs to work is obsoleted. [Aside: as it turns out, jobs are usually not lost in this process. Rather, a bifurcation occurs: some engineers become legacy systems experts, others move to more productive higher-level design work with more powerful tools.]

ACTION ITEM #1: develop a business plan and financial analysis which excludes the use of "competing" EDA tools. Read "competing" as "dysfunctional to BTC".

APPRECIATE WHAT WE HAVE

Comesh blocks provide 150 logic gates at 300 MHz. That's enough for many FSMs. Superblocks (4x4 blocks) we believe can provide around 2000 logic gates as a single unit at 100 MHz. As we used to say, that's "100 million complete function evaluations per second". 2000 gates is enough for 80% of all applications [estimate]. I.e.: we can make almost all processes work very fast and *in synchronization*.

Here's the essential question: what does most of the market need? We tend to hear about the exotic 5% cause that's the edge and the failure point of current techniques. However:

-- if it has a plug, power doesn't matter.
-- if its not on a critical performance path, speed doesn't matter.
-- if the function fits for the price, logic density doesn't matter.
-- if it has a large market, price is the only thing that matters.

MARKET PLACEMENT

A good article by Tredennick and Shimamoto which supports some of these ideas is attached. Can't resist one great quote from them:

"PLDs [have] more overhead than the U.S. Government."

Tredennick's main point is that chip sales are driven by "lower cost and adequate performance". We have to deliver "better" in a "cheaper" manner. No big surprise. What this says about market placement is that a sweet spot is where customers need to step up from Spartan (low cost) or any CPLD to Virtex (high performance), cause the Spartan/CPLD performance level is insufficient. We need to avoid (as we have always been saying) using our better performance to enter the high price market.

Another of Tredennick's points is that the microprocessor is an antiquated solution with lots of inertia, still selling 5 billion embedded units a year. uPs provide programmability at the cost of efficiency, which says: 1) speed doesn't matter for large market segments, and 2) programmability does matter but current FPGAs have not provided dynamic programmable choices. We have both. We also have the ability to parse microprocessor instructions into Comesh hardware (see the next section).

Many FPGA folks are attacking DSPs as "where the uP cost does not pay in benefits". If we can beat DSPs, we can beat any uPs. There's a factor of 100 in favor of FPGAs, if they are dynamically (and easily!) programmable.