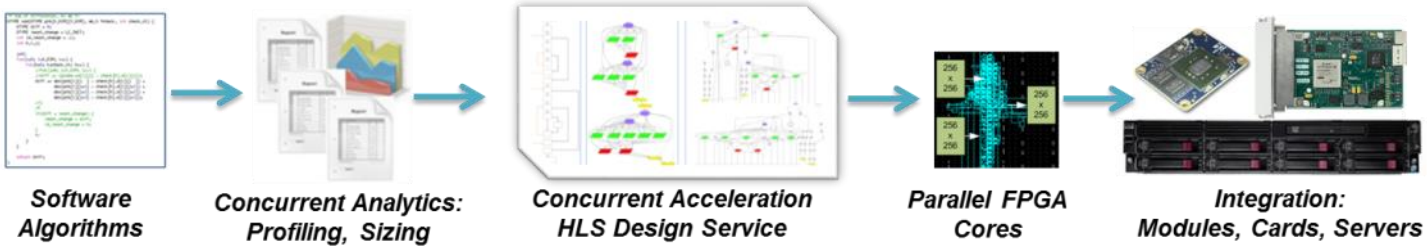
Concurrent Design Services Faster Designs, Shorter Time



CONCURRENT ACCELERATION™ converts your software algorithm into a high-performance FPGA design.

ABSTRACT

Concurrent EDA's technology enables rapid synthesis of a software algorithms into FPGA hardware. This datasheet provides an overview of this service. The input is your software algorithm and the output is a completed FPGA design that produces bit-for-bit identicle results. Performance improvement is typically 10-30 times faster than a processor. Other design services are also available.

INPUT

This service starts with your software compiled into an executable. Multiple languages are supported but not all languages generate the same quality of executable. This is true for embedded programming and for FPGA synthesis. Since Concurrent EDA performs x86 synthesis, various languages are possible.

- **C** is recommended as it generates a tight binary and is able to be optimized at a low level.
- **C++** can be used as long as dynamic memory allocation isn't used since FPGAs have fixed memory sizes.
- **Fortran, Ada** and other compiled languages are possible.
- **Matlab** can be used with its Embedded Coder as this generates C code.
- **Matlab M code, Java, Python, Perl** and **C#** can be used as a specification and Concurrent EDA can translate these to C code for you.

ANALYSIS

The first step is to analyze the software to determine which portions should be accelerated in hardware and which should be kept in software. Next, the hardware portion is analyzed and sized for an FPGA. Finally, the data structures are analyzed so they can be optimally partitioned.

CONCURRENT ANALYTICS™ performs the following set of analyses to enable synthesis:

- **Performance Analysis**
- **Hardware/Software Partition**
- **Complexity Analysis**
- **Data Analysis**
- **FPGA Device Analysis**

See the CONCURRENT ANALYTICS™ Data Sheet for more information.

SOFTWARE OPTIMIZATION

High performance FPGA designs start with optimized software.

- **Precision optimization**. Optimizing the bit-precision of operations increase performance and reduces FPGA area.
- **Library optimization.** Some software libraries are not efficiently written and optimizing them can improve area and performance.
- **Data structure optimization.** FPGAs have hundreds of internal memories that can operate in parallel.
- **Data optimization.** Loop-level data dependencies limit parallelism and can frequently be removed.

EXTRACT PARALLELISM

Concurrent EDA has world-class synthesis tools that automatically transform software into FPGA hardware. Customers gain the benefits of these tools through our turn-key design service. The end result is a high-performance FPGA design that is functionally identical but at a higher performance.

Concurrent EDA's tools are driven by experienced engineers who know how to extract various levels of parallelism from sequential software. For example:

- **Instruction-Level Parallelism.** Software executes one instruction at a time. FPGA hardware can execute hundreds to thousands of operations in parallel.
- **Loop-Level Parallelism.** Loops can usually be transformed into pipelined logic that generates one result per clock cycle.
- **Function-Level Parallelism**. Sequences of functions can be executed in parallel if they operate independently on different data.

HDL CODE GENERATION

Pipelining

After the parallelism has been extracted, the design is represented as a graph of computations. At the lowest level, each node in the graph is an operation. During synthesis, static timing

Concurrent Design Services Faster Designs, Shorter Time

analysis is performed to determine the delay through the graph. Registers are then inserted to ensure that the design hits its target frequency. This is a time consuming process when performed by hand using VHDL and Verilog. From our experience, each line of C code generates 10 to 100 lines of VHDL. Thus, a small 100 line algorithm will generate 1,000 to 10,000 lines of VHDL – prohibitively expensive to do manually. Concurrent EDA's tools can insert these registers automatically using the device-specific timing.

Loops to Pipelines

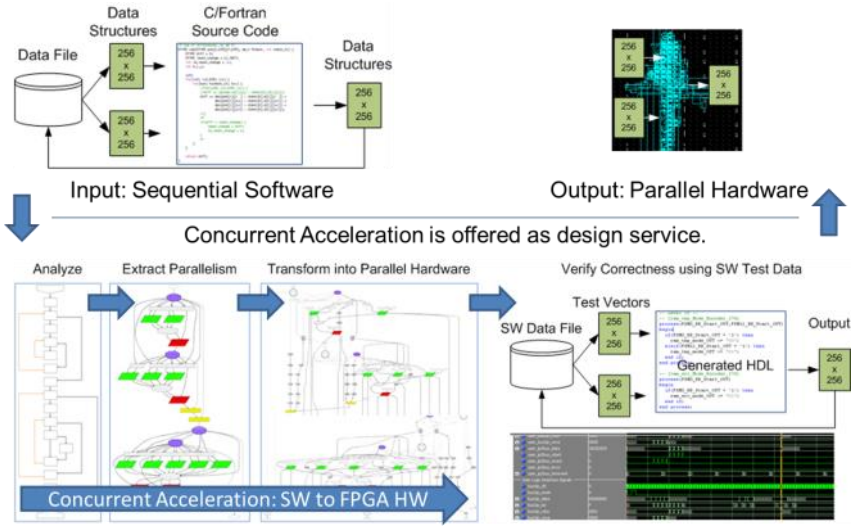
Algorithms utilize loop structures to perform iterative operations. If the loops can be executed in parallel, then the loop body can be transformed into a pipeline of computations and output one result per cycle. Concurrent EDA's tools do this quite well and typically generate FPGA logic that is 20 times faster than a PC. Nested loops are supported and can be arbitrarily deep. Conditional loops and branching within loops are also fully supported.

Function-level Parallelism

Sequences of functions can be executed in parallel if they operate independently on different data sets. Parallel execution is common for image, signal and data processing applications where the application takes in raw data, performs a sequence of operations and outputs transformed data.

SYNTHESIS AND PLACE-AND-ROUTE

The output of Concurrent EDA's tools is VHDL. Vendor-specific RAMs, FIFOs and multi-cycle math cells are utilized to achieve maximum performance and area efficiency. The vendor synthesis tools are then used to transform the HDL into a netlist. Place-and-route is then performed to get a final FPGA bitstream. Concurrent EDA has expertise in vendor tools and a cluster of computers to handle simultaneous place-and-route runs.



VERIFICATION

Verification is the process of confirming that the HDL generates the correct results, and can consume half of the design time in a manual design flow. Test data is generated from the software algorithm and then used to test the HDL. Keep in mind, however, the output of the software is really generated by the compiled executable. Since different compilers and compiler flags can change the order of operations and thus the round-off from operations can be different, the golden specification is really the executable. This distinction seems minor until the verification stage when the results are different than expected. Concurrent EDA translates directly from the binary and therefore the HDL has the same behavior as the software. The result is fast verification and bit-for-bit accurate results from the HDL.

INTEGRATION

After the FPGA design has been fully verified and turned into an FPGA core, it must be integrated into a hardware board or design. For algorithm acceleration, the objective is not just the FPGA portion but also the remaining software and the hardware/software interface. Concurrent has experience with various interfaces, including ARM x86, AXI4 Streams, external memory, video DMA, PCIe, VME, I2C, LVDS and others. This experience ensures a smooth integration process.

Application Example	Description	Acceleration over 3 GHz PC
Image Processing	30K lines of C → 500K lines of VHDL 10 images sensor through a single FPGA	20x
Ortho-rectification	24K lines of C → 133K lines of VHDL Double Precision Floating Point Intensive	15x
Computational Searching	3.8K lines of C →179K lines of VHDL 605 lines of C → 36K lines of VHDL	15x 25x
Cryptography	Gb/s 3DES, AES, SHA1, SHA2, Deduplication	17x
Text Searching	Gb/s data, regular expressions	10x
Video & Image Processing	Filters, motion detection, compression, graphics overlay, edge enhancement	15-30x

www.ConcurrentEDA.com

Concurrent EDA Measurably Faster

info@ConcurrentEDA.com
412-687-8800