

Design Reuse and Retargeting with Cynthesizer ESL Synthesis

Cynthesizer's SystemC-based design methodology is revolutionizing the way that electronic systems are created. Design teams have adopted the new methodology to quickly create complex ASICs, SoCs, and FPGAs for HDTV, digital cameras and copiers, optical and audio processing devices, and more. Their investment has yielded successful silicon as well as a proven methodology for their next generation of designs.

Cynthesizer includes features such as TLM Synthesis™, Platform Retargeting, and Design Exploration that provide design teams with the ability to quickly create IP blocks that can be reused throughout many generations of applications. Reuse of IP is one of the main strategies for dealing with the challenges of ever-increasing design size. Unfortunately, until now a designer's ability to effectively reuse previously-developed IP has been limited. Effective reuse has largely been limited to processors, memories, and other platform elements. Reuse of the core computation blocks that are key to consumer, multimedia, and gaming applications has been more elusive.

Each type of IP that uses a higher level of abstraction increases reusability. "Hard IP" is distributed at the physical level. Hard IP is specific to a particular process technology, process node, and foundry and is usable across a very limited range of clock speeds. The functionality of hard IP is strictly fixed. Most processor IP is distributed at this level.

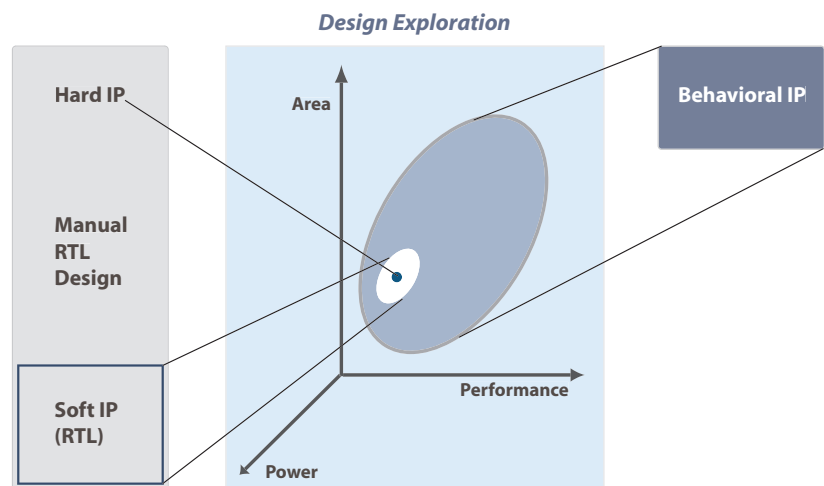
"Soft IP" is distributed at the RT level. This increased level of abstraction improves the reusability of soft IP compared to hard IP. Logic synthesis can retarget soft IP to different process nodes and foundries. Unfortunately, because the pipeline structure is hard coded in the RTL, soft IP is still quite limited in the range of clock speeds at which it can be reused. In addition, any modification to the functionality is subject to the large design effort associated with RTL.

"Behavioral IP" raises the level of abstraction and dramatically increases the range of reusability. Behavioral IP source code is written in high-level SystemC. Because SystemC is a C++ class library that adds hardware constructs to the C++ language, it has the semantic constructs needed to represent hierarchy, concurrency, protocol, and so on. The source code used for synthesis does not contain timing information that implies

a particular schedule or micro-architecture. Instead, constraints defined by the designer direct Cynthesizer to create a faster, smaller, or lower-power RTL implementation to achieve the optimal trade-off.

Behavioral descriptions allow the designer to explore the design space by creating many different RTL implementations from one behavioral source. A single behavioral description can be retargeted across a range of design characteristics such as clock speed (100 MHz vs. 500 MHz), target technology (ASIC vs. FPGA), and technology feature size (130nm vs. 90nm).

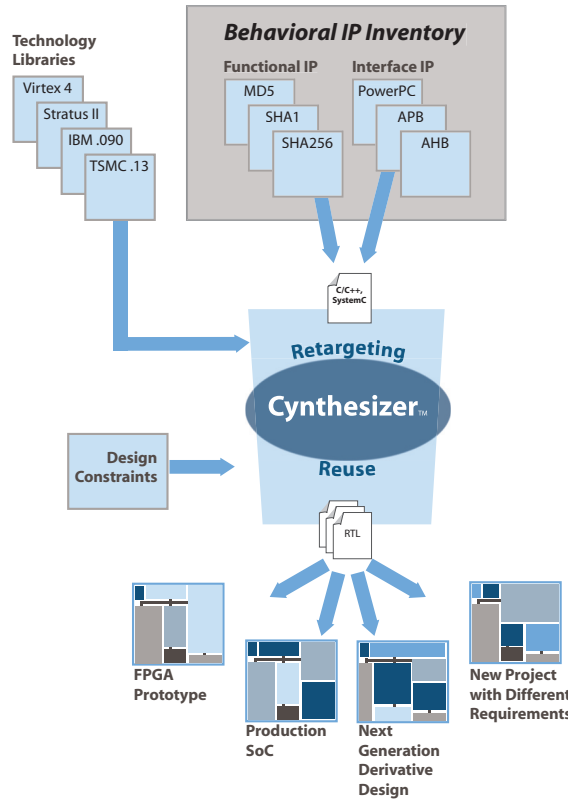
By using a behavioral design that is independent of the RTL micro-architecture, behavioral IP can be easily reused for different applications. Cynthesizer users recognize that by building an inventory of behavioral IP they improve their reuse opportunities for future projects and maximize the ROI of their current design efforts.



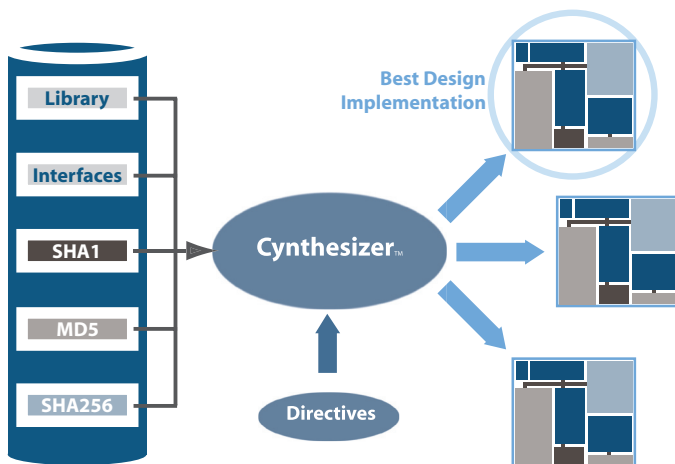
	Abstraction	Language	Process Tech	Speed	Function
Behavioral IP	Behavioral	SystemC	Variable	Variable	Parameterizable
Soft IP	RTL	Verilog VHDL	Variable	Limited Variability	Minimal Parameterization
Hard IP	Physical	GDSII	Fixed	Fixed	Fixed

In addition to the reuse benefits of creating IP at the behavioral level, there are substantial productivity benefits. Designing with Cynthesizer and SystemC can reduce time-to-verified-RTL by months with fewer resources needed. One Cynthesizer user (Oki) recently reported that they reduced their design effort by two-thirds. The result is more verified gates in less time while freeing up designers to work on additional hardware blocks.

Using Cynthesizer's TLM Synthesis™ capability, designs can be quickly retargeted to different bus architectures or interfaces. A design using TLM Synthesis™ is written with transaction-level function calls for its I/O operations. This transaction-level modeling style (TLM) raises the design and verification abstraction level by allowing the designer to ignore these details during the early stages of the design and verification process. This reduces errors in coding and significantly improves simulation performance. The TLM is then used directly for synthesis. Cynthesizer automatically creates RTL implementations with the appropriate pin-level interface included. This process is much faster and simpler and reduces the risk of errors. In addition, changing design interfaces is as simple as changing a declaration.



Design	Blocks	Process (K)	Speed	RTL (K)
SHA256+512	2	0.09	83 300mhz	134
MD5+SHA1	2	0.09	22 300mhz	28
MD5+SHA1+SHA256+512	4	0.09	107 300mhz	148



Why is TLM important at the block level?

While TLM is widely used by system designers, block designers using Cynthesizer will greatly benefit in using TLM Synthesis. It facilitates high-speed TLM simulations for block-level verification, which reduces verification time or allows greater functional coverage in the time allocated for simulation.

Using transaction-level interfaces results in functional simulation 5X - 10X faster than pin-level behavioral simulation or 50X to 200X faster than RTL simulation. For 60% - 80% of all block-level behavioral simulation runs, TLM simulation may be used. The remainder requires the use of pin-level interfaces either to validate pin-level behavior or to simulate with other blocks using pin-level interfaces.

FORTE DESIGN SYSTEMS

Forte Design Systems headquarters:
100 Century Center Court
Suite 100
San Jose, CA 95112 USA
tel: 408.432.9430
fax: 408.432.9433
email: sales@ForteDS.com

Forte Design Systems European office:
tel: +33 468 940 808
fax: +33 680 130 909
email: fconstant@forteds.com

www.ForteDS.com