HIGH-LEVEL SYNTHESIS FOR AUTONOMOUS DRIVE

ANOOP SAHA, MARKET DEVELOPMENT MANAGER,
DIGITAL DESIGN AND IMPLEMENTATION SOLUTIONS

W H I T E P A P E R

Mentor®
A Siemens Business

H I G H - L E V E L S Y N T H E S I S

www.mentor.com

Autonomous vehicles require a sophisticated framework of sensors to function. These sensors include LiDAR, Radar, video, etc. that continuously generate a high volume of data in real time about the environment surrounding the car. The sensors constantly send their output for analysis to powerful domain controllers connected to a processing unit. The discrete data from different sensors are then merged to create meaningful information related to the vehicle's location, speed, direction and surroundings. This process is known as **sensor fusion**.

Sensor Fusion is typically done using custom hardware – either FPGA or ASICs. The data is then processed to make decisions that impact the ADAS systems –such as turns, breaks or speed adaptation. The hardware incorporates algorithms which involve machine learning in sophisticated artificial intelligence (AI) applications to facilitate real-time processing of the sensor data.

COMPLEX ALGORITHMS FOR AUTONOMOUS SENSOR PROCESSING

However, implementing machine learning algorithms in hardware is a challenge in itself. For example, one common algorithm for object detection is based on CNNs (Convolutional Neural Networks), which aid in 'Adaptive Cruise Control' and in 'Forward/Rear Collision Warning Systems' – obviously crucial capabilities for achieving a fully autonomous vehicle. The CNN is made up of multiple layers, where each layer performs multiple sets of convolutions. The convolutional filters for each layer are "feature detectors" programmed to look for certain characteristics such as horizontal lines, vertical lines, etc.

It is common for the number of input and output channels to double as data progresses through the layers, resulting in an explosion in the number of convolutional filters as well as the filter weights (coefficients). Figure 1 shows that the first layer has 1 input channel and 16 output channels and it requires 16 different convolutional filter kernels. The second layer has 16 input channels and 36 output channels requiring 36x16 = 576 2-d convolutional filters. In many CNNs the last layer(s) consist of fully connected layers which are usually implemented using matrix multiplication.

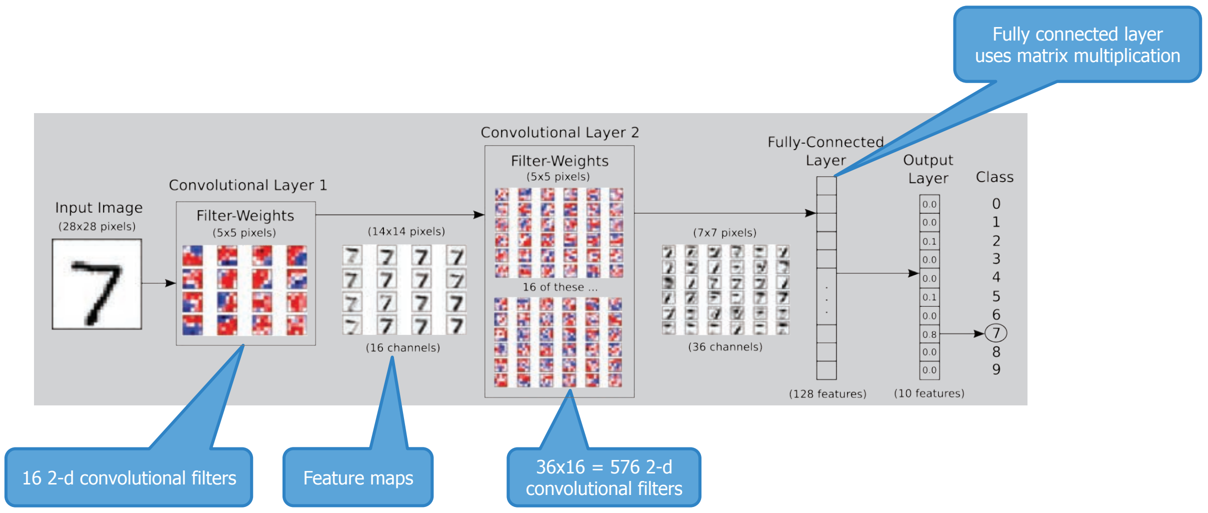


Figure 1: Simple 2-layer CNN for Character Recognition

Implementing a CNN for both performance and power-optimized hardware requires multiple man years of effort. Hardware design, by itself, is complex because of the nature of abstraction. For custom hardware for autonomous vehicles, the designer needs to explore the various architectures to ensure the best accuracy for inference.

To achieve the accuracy, an inference chip for autonomous vehicle needs to address the following additional challenges:

a. **Performance**: A single high definition camera can capture a 1920x1080 image at 60 frames per second. A car can have 10 or more such cameras. The inference engine must be able to process data coming at that rate to make meaningful decisions. As a result, the performance of the CNN becomes critical.

b. **Power**: Although not intuitive, power consumption is another important metric for an inference engine. AI inference could be a massively power intensive operation especially because the high volume of accesses to remote memories. For electric cars, it is important to reduce the power consumption in the ADAS system so that the battery power could be more efficiently used for physical operation of the car.

c. **Functional Safety**: Functional safety is another area which is becoming more important for autonomous vehicles. The algorithm should be able to detect functional safety issues that might creep in because of various faults in the hardware. The ability to ensure proper fault coverage through upfront verification is critical.

HLS DESIGN FLOW: FROM ARCHITECTURE TO VERIFIED RTL

The biggest challenge remains on the turnaround time for the traditional ASIC design flow. It takes somewhere from several months to a year to implement a new ASIC hardware. A majority of time is spent in the verification of the ASIC before it is sent for fabrication. Because of the high mask cost, as well as turnaround times for lower nodes, the design and verification teams would want to ensure high coverage in verification before tapeout. This becomes challenging for machine learning, as the algorithms are evolving rapidly. It becomes imperative for the hardware design to keep pace with the rapid evolution of software algorithms and frameworks – as shown in Figure 2.

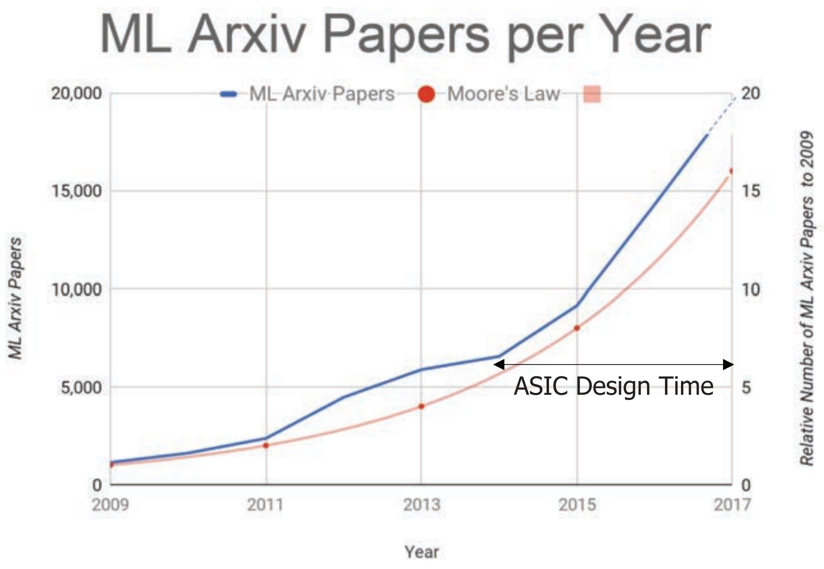


Figure 2: ASIC design time vs ML papers per year
(Image courtesy: Samba Nova; adapted from Jeff Dean, Scaled ML)

Initially, an autonomous system architect or designer relies on tools like TensorFlow, Caffe, MATLAB and theano to aid in capturing, collecting and categorically verifying data in a high-level abstract environment. These high level deep learning frameworks allow exploration of a multitude of parameters to explore, analyze and select the optimal solution for the algorithm.

Once the algorithm is determined, the designer then captures the flow in C++ or SystemC. The advantage of C++ is that it is a general programming language that is already in wide use for modeling algorithms/behavior at the high level for both hardware and software. SystemC on the other hand allows users to refine the design by adding more implementation detail – which is crucial for the hardware implementation of the sensor processing unit.

The next step is to start designing the actual hardware algorithmic block for autonomous applications. Here the designer is faced with a choice between hand-coded RTL or using High-Level Synthesis (HLS) to generate RTL from C++ or SystemC. Although RTL has traditionally been the starting point for digital design, it is becoming too expensive and time consuming (Fig 2). Large teams are required to generate hundreds of thousands, if not a million of lines of Verilog or VHDL code. Small changes can cause big re-designs and it is hard to find skilled engineers for these teams. Verification times and costs are escalating out of control because RTL simulations run 106 times slower than C++ simulations.

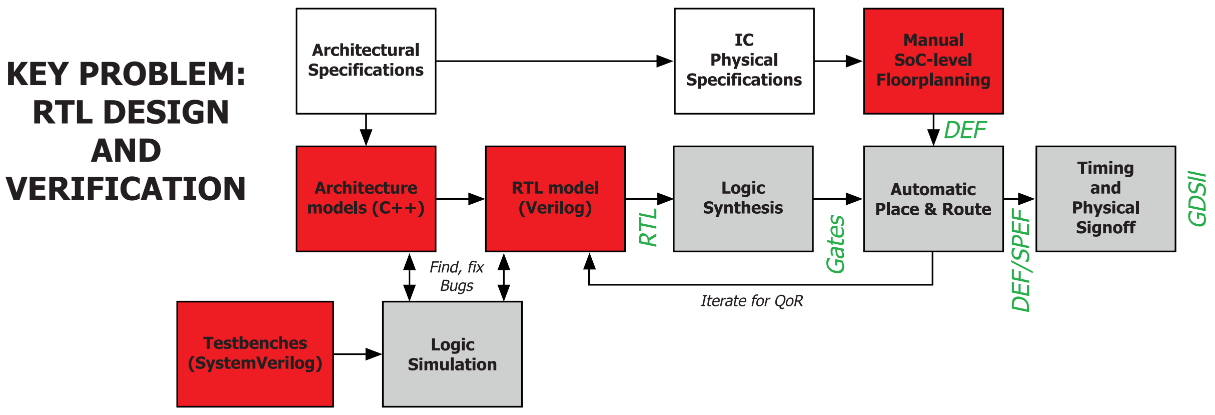
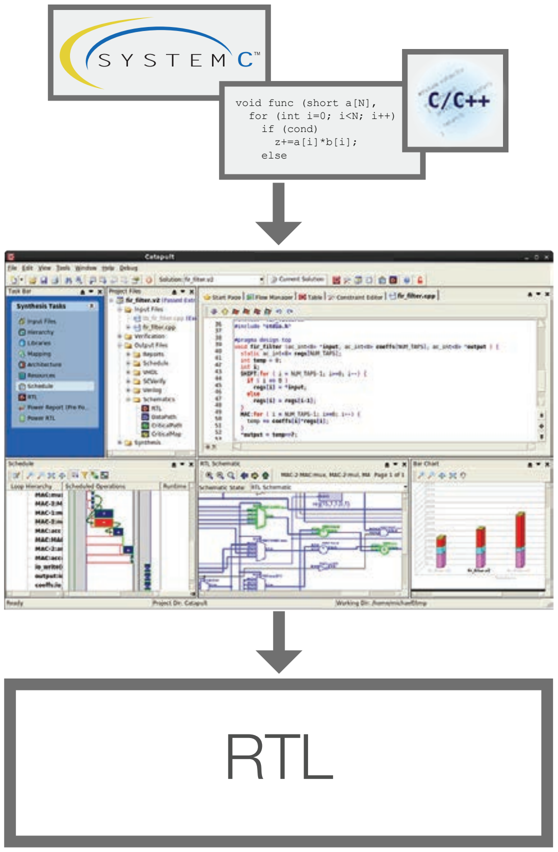

KEY PROBLEM: RTL DESIGN AND VERIFICATION
Architectural Specifications
IC Physical Specifications
Manual SoC-level Floorplanning
DEF
Architecture models (C++)
RTL model (Verilog)
RTL
Logic Synthesis
Gates
Automatic Place & Route
DEF/SPEF
Timing and Physical Signoff
GDSII
Find, fix Bugs
Iterate for QoR
Testbenches (SystemVerilog)
Logic Simulation

Figure 3: Key problem: RTL Design and Verification (Image Source: NVIDIA)

Unlike RTL, HLS separates functionality from implementation with powerful capabilities for targeting and implementation at any time (Fig 3). As a result, HLS accelerates algorithmic design time with a higher level of abstraction resulting in 50x less code than RTL. That means smaller design teams, shorter development time and faster verification. Algorithmic design and verification can be reduced from a year to just a few months and new features can be added in days instead of weeks. Since algorithms can be easily modified and regenerated, late stage functional changes can be made without impacting tapeout schedule. Because of a higher level of abstraction, simulation is extremely fast enabling rapid exploration of architectural options to achieve optimal power, performance and area.



```
void func (short a[N],
  for (int i=0; i<N; i++)
    if (cond)
      z+=a[i]*b[i];
    else
```

Figure 4: High-Level Synthesis platform

Once the design team is satisfied with the results, HLS generates high quality RTL from C/C++ and/or SystemC descriptions. Verification productivity also requires a functional coverage methodology similar to what is done in RTL today, but will benefit from the speed up in simulation runtime. This includes formal property checking and linting as part of this flow to ensure that the source code is "clean" for both synthesis and simulation. Along with this, tools are required that can measure code coverage, including line, branch, and expression coverage. Moving to a higher level design requires strong debug and visualization tools including IDE and waveform views plus the ability to visualize optimization bottlenecks along with the control and data flow. The goal is to achieve RTL that is correct by construction by precise consistency of representation and simulation results between C++ algorithm and synthesized RTL.

At this point, the algorithmic designer needs to decide what type of hardware platform for the autonomous vehicle system should be used. The designer could express the algorithms as software in a CPU, DSP or GPU but these are expensive and have additional drawbacks. The CPU is not fast or efficient enough, while the DSP is good at image processing it lacks enough performance for Deep AI. Although a GPU is good at training, it is too power hungry for an in-vehicle solution. The other choice is to instantiate the algorithm in a hardware platform in a custom design in either an FPGA, ASIC and dedicated form. Although it requires more upfront design, FPGA or ASIC deliver optimal power/performance results. That is why more and more design teams are choosing to implement their algorithmic designs in hardware.

With HLS it is very easy to implement in either ASIC or FPGA. One scenario is to use a FPGA and take advantage of the simpler design cycle that eliminates complex and time consuming routing, placement and floor planning methodology. ASICs on the other hand, can handle mass volumes of chip production but are much more labor intensive and expensive. Some designers use an FPGA for initial prototypes to analyze both software and hardware design. Once they are satisfied and are ready for production, they then take the output from HLS and retarget it quickly to the ASIC technology of choice.

The Catapult® HLS Platform and PowerPro® solutions from Mentor, A Siemens Business is the industry's leading HLS platform with proven quality of results. Catapult empowers designers to use industry-standard ANSI C++ and SystemC to describe functional intent and move up to a more productive abstraction level. The Catapult Platform provides a powerful combination of high-level synthesis paired with PowerPro for measurement, exploration, analysis, and optimization of RTL power and a verification infrastructure for seamless verification of C++ and RTL (Fig 4).

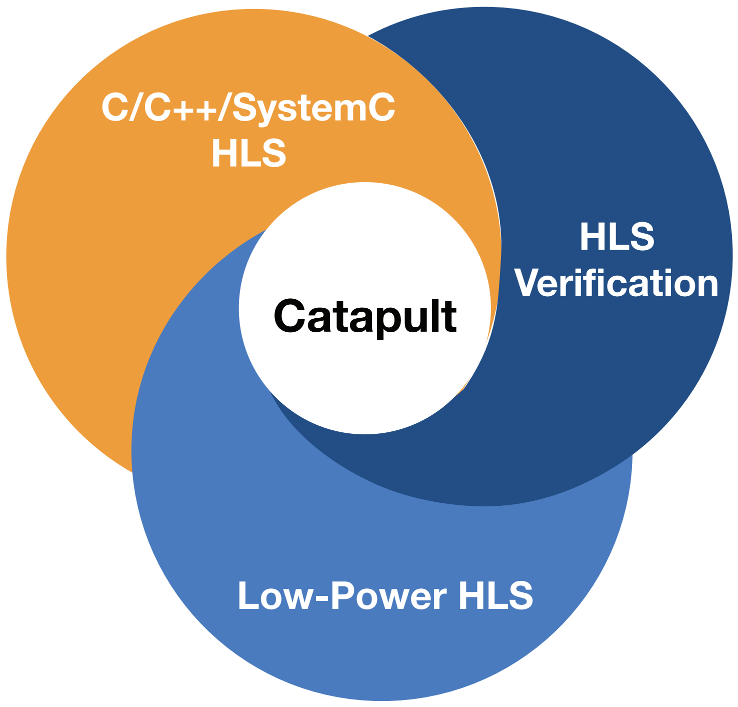


Figure 5: The Catapult HLS platform combines HLS with power analysis and optimization of RTL power and verification infrastructure

AUTONOMOUS ASIC EXAMPLES

Major semiconductor suppliers for the autonomous vehicles are already successfully implementing algorithmic-intensive designs using Catapult It is helpful to briefly touch on some of their success stories.

Bosch realized that to keep their number one position in the automotive space, they needed to gradually shift to autonomous solutions. The BOSCH Visiontec team develops state-of-the-art IP and ICs containing high-performance processors that implement algorithms to recognize images from cameras in automobiles. The stakes were high, as they needed to implement three image processing designs in less than a year. There was no way that the team could manually code RTL and verify these designs in this ever-changing environment. So, they decided to move up to the C++ level and employ the Catapult flow.

With HLS flow that included power analysis and optimization, BOSCH Visiontec team was able to successfully deliver the new designs ahead of schedule in seven months even though the specifications evolved over the design cycle. Using Catapult's micro-architectural exploration, the team was able to quickly produce higher quality designs and an overall reduction in power of 30 percent through continuous refinement and by using the low-power flow. Because of the team's success, this flow will be used for future new designs and for updated designs that target new standards or process technologies.

Chips&Media is a leading provider of high-performance video IP for a variety of markets including automotive and autonomous sensor processing blocks. They employed an HLS design flow for computer vision IP that incorporates deep learning, inference object detection for processing input video at 4K resolution and 30fps. Since it was the first time using an HLS approach, they decided to develop the IP in two parallel tracks. One team would use the conventional approach of developing the IP using hand-coded Verilog and the other team would start with C and use HLS to synthesize the code into Verilog. They would then compare the results of each flow.

Compared to 5 months for the RTL coding, it only took 2.5 months to learn the HLS flow and Catapult, write the C code, synthesize to RTL, verify, and integrate the IP into a FPGA board to successfully demonstrate the video IP product. The HLS team explored many architectures with little effort, which would have been very difficult to do in the traditional RTL flow.

STMicroelectronics' imaging division uses Catapult HLS for a range of signal processing applications including automotive sensors (Figure 6). One of the major benefits of HLS is the increase in IP value without additional engineering resources. Since it was easier to work in SystemC, the team could efficiently add new features and explore different architectures to achieve optimal performance, power and area.

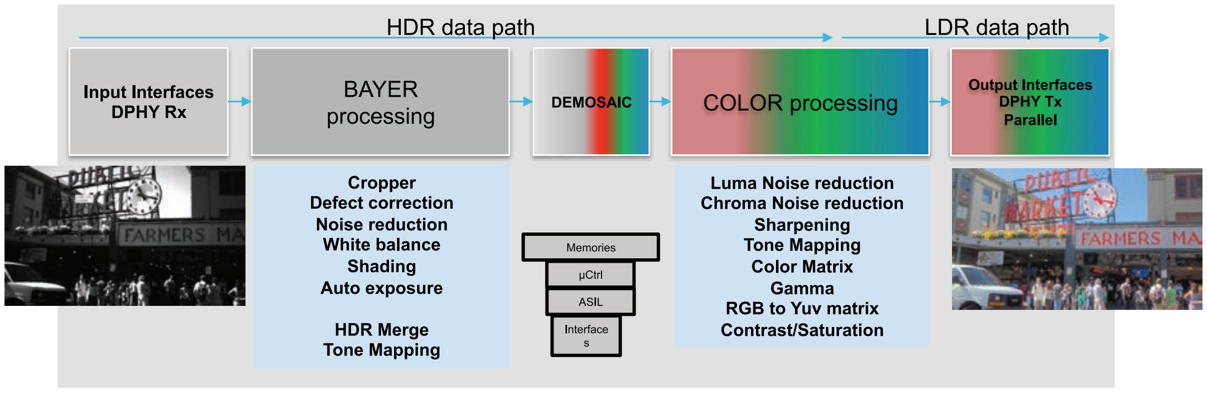


Figure 6: Complex, high quality image signal processing for automotive applications

Verification time and effort was also reduced. They were able to run thousands of tests in hours versus weeks that it would take for RTL. The verification team found that automatic functional clock gating saved considerable recoding and debugging which reduced the power by 10%. Also, the quality of results of the generated RTL was excellent for these complex designs. As a result of using HLS, the team created more than 50 image signal processing designs over two years, ranging in size from 10K gates to 2 million gates.

CONCLUSION

Autonomous designers are grappling with new silicon architectures optimized for neural computing and computer vision to make autonomous vehicle solutions better and faster to market than ever before.

To have the most efficient implementation of the silicon for ADAS, they need to move up the abstraction layer for higher design and verification productivity.

Existing RTL-based design methodologies cannot scale to meet this demand, since SoC designs for autonomous vehicles are too complex to design efficiently by hand-coding RTL. Additionally, verification times are escalating out of control, making it necessary to verify designs as early as possible. HLS is proven to deliver higher quality algorithmic-based designs faster and more efficiently than RTL for autonomous drive systems.

Corporate Headquarters
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, OR 97070-7777
Phone: 503.685.7000
Fax: 503.685.1204

Sales and Product Information
Phone: 800.547.3000
sales_info@mentor.com

Silicon Valley
Mentor Graphics Corporation
46871 Bayside Parkway
Fremont, CA 94538 USA
Phone: 510.354.7400
Fax: 510.354.7467

North American Support Center
Phone: 800.547.4303

Europe
Mentor Graphics
Deutschland GmbH
Arnulfstrasse 201
80634 Munich
Germany
Phone: +49.89.57096.0
Fax: +49.89.57096.400

Pacific Rim
Mentor Graphics (Taiwan)
11F, No. 120, Section 2,
Gongdao 5th Road
HsinChu City 300,
Taiwan, ROC
Phone: 886.3.513.1000
Fax: 886.3.573.4734

Japan
Mentor Graphics Japan Co., Ltd.
Gotenyama Trust Tower
7-35, Kita-Shinagawa 4-chome
Shinagawa-Ku, Tokyo 140-0001
Japan
Phone: +81.3.5488.3033
Fax: +81.3.5488.3004

Mentor® A Siemens Business

MGC 06-19 TECH18260-w