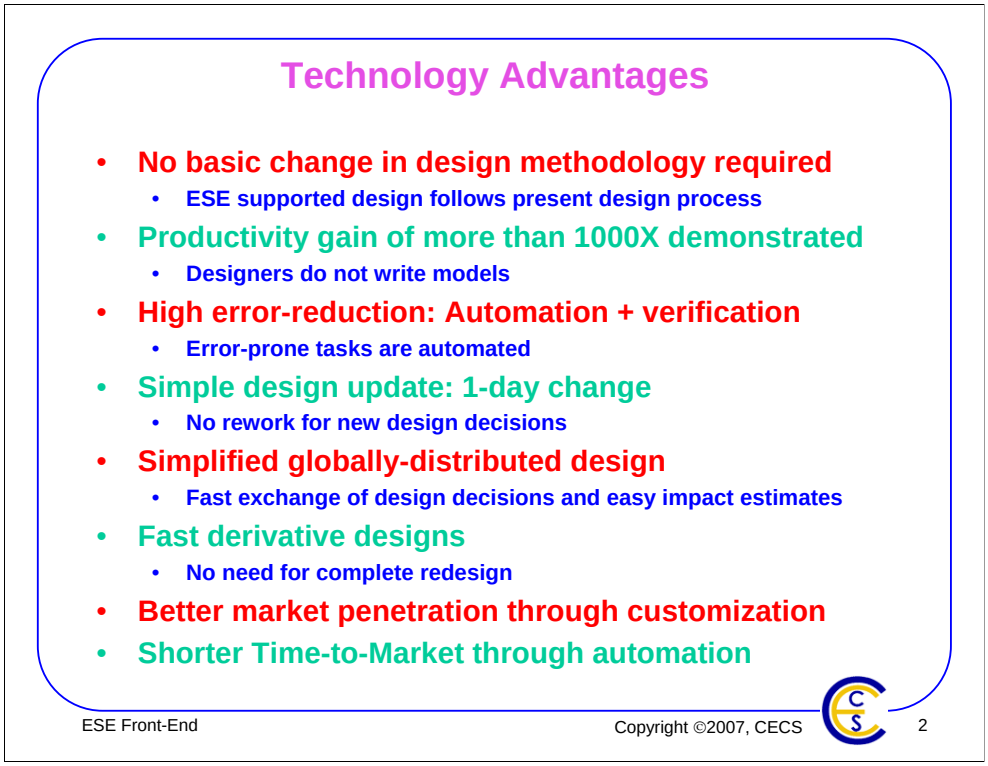
ESE Front End 2.0

D. Gajski, S. Abdi, Y. Hwang, L. Yu, H. Cho, I. Viskic
Center for Embedded Computer Systems
University of California, Irvine

http://www.cecs.uci.edu

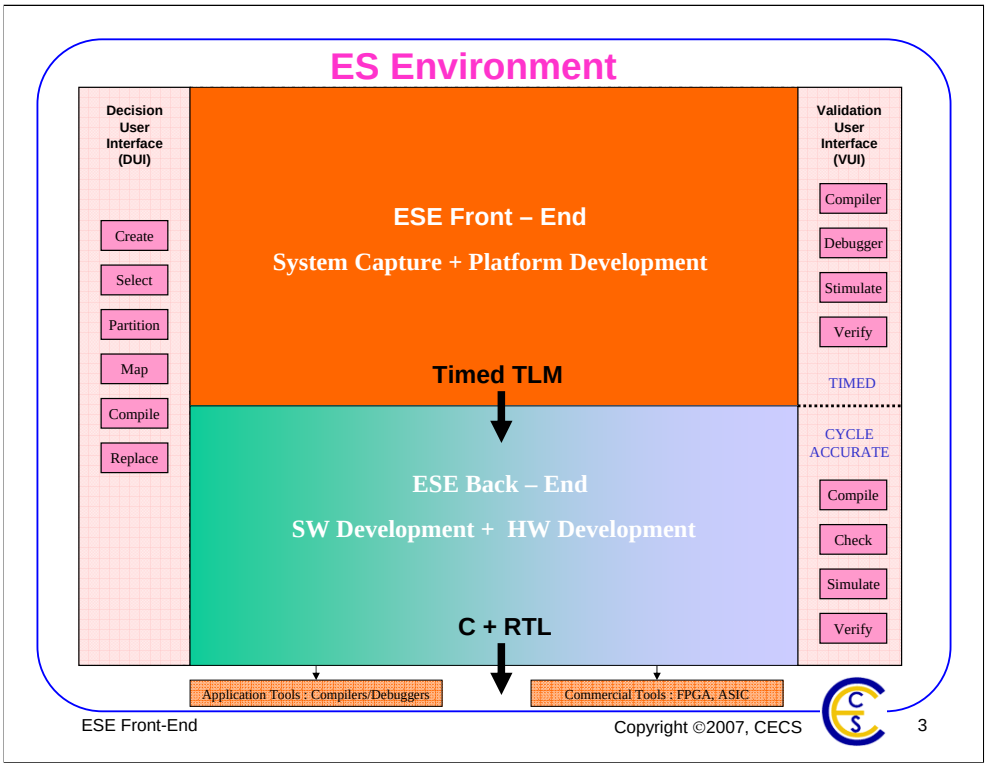


Technology Advantages

This new ESE methodology does not require any changes in the present corporate methodology and offers three orders of magnitude of productivity gain because of automatic model generation, synthesis and verification.. It reduces bugs since the mundane tasks of generating models and verifying them is automatic.

It also allows simple change management of few hours for small changes and few days for large changes. Since all the models and changes are made automatically it is easy to ship those models around the world for design, checking and upgrades.

However, the main advantage lies in the fact that every system or product can be easily upgraded with only few days of work. This type of customization allows better market penetration and shorter time-to-market.



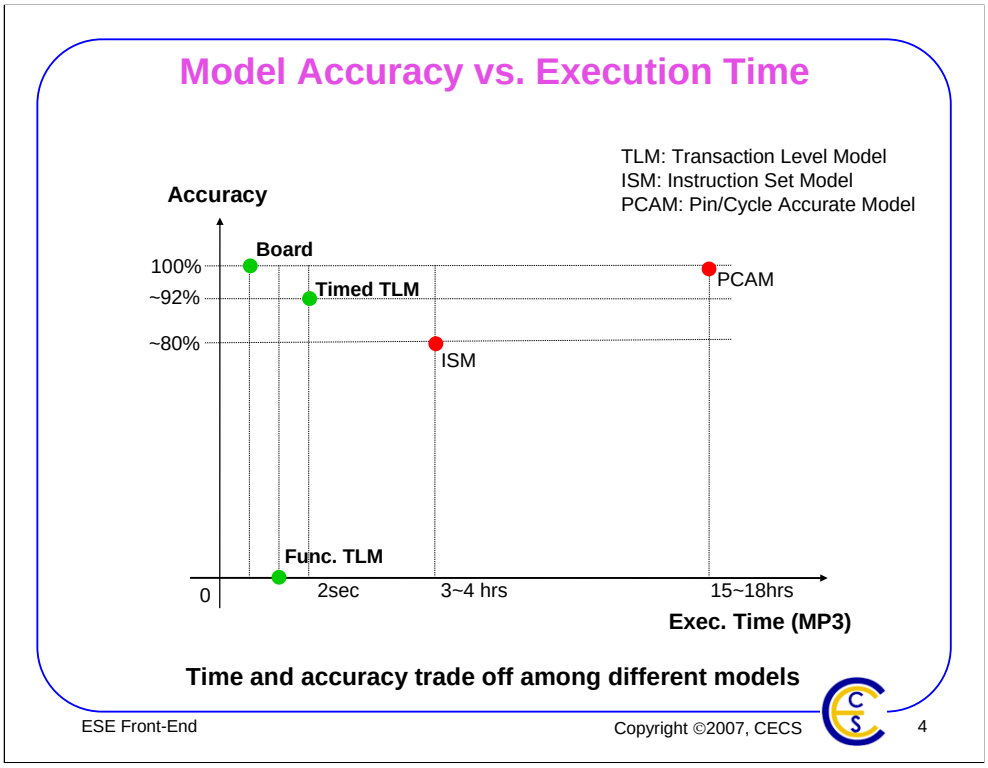
ES Environment (ESE)

The ESE consists of a **front-end** and a **back-end** supported by two interfaces.

The front-end consists of **System Capture**, which is a graphical user interface for capturing the definition of the platform architecture and product application code. **Platform Development** tool generates timed Transaction-Level Models (TLMs) of the platform architecture executing the product application defined by the capture tool. These timed TLMs provide reliable performance metrics and are used for early exploration of design choices.

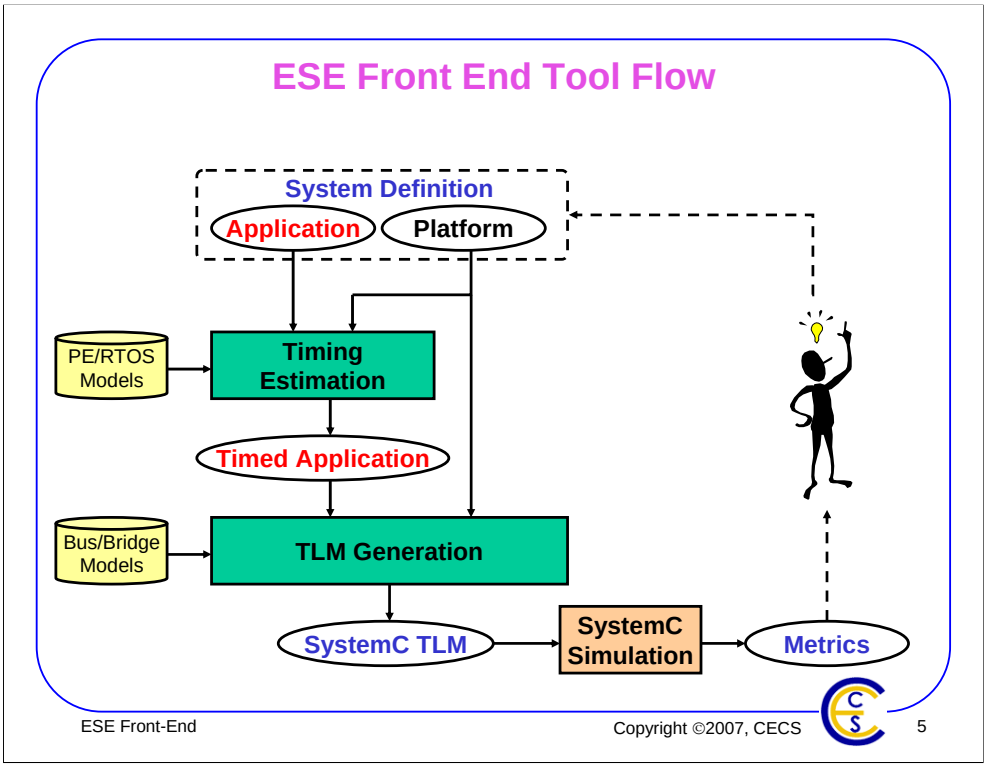
In the back-end, the **HW Development** component is used to generate cycle-accurate or RTL description of the HW components which can be further refined by commercially available tools for ASIC or FPGA manufacturing. **SW Development** generate firmware necessary to run communication and application SW on the platform.

Validation User Interface is used to debug and validate developed SW and HW. **Decision User Interface** is used by the designer, to estimate the quality metrics and make decisions such as component selection, task scheduling, mapping of SW functions to HW components and others.



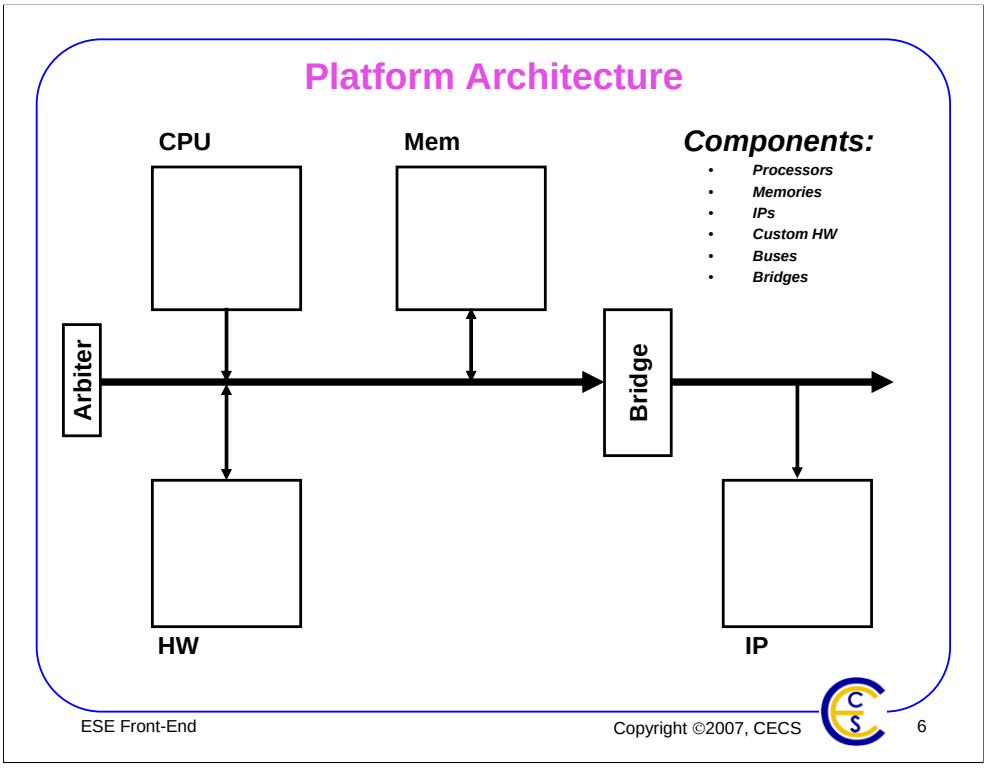
Model Accuracy vs. Execution Time

Different types of system models are used at different stages of design. Traditionally Pin/Cycle Accurate Models (PCAMs) and Instruction-Set Models (ISMs) have been used to validate designs at cycle accurate level. However, for a design like MP3 decoder, these models may take several hours to simulate. The timed TLMs generated by ESE provide accuracy in estimation within 8% of board prototype. TLMs are also applicable to any type or complexity of design as opposed to ISMs. Therefore using TLMs generated automatically by ESE, system designers can evaluate design decisions in a matter of seconds instead of waiting for hours or even days for a cycle accurate simulation to complete. Thus, the design modification and evaluation cycle is shortened, leading to significantly more opportunities for design optimization.



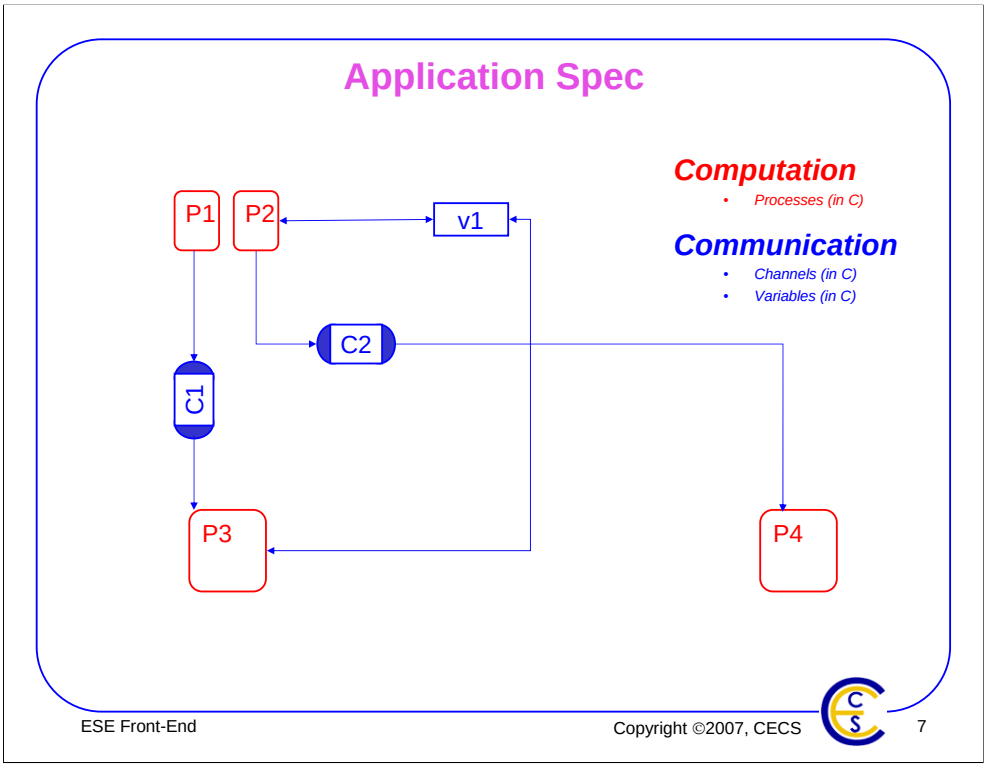
ESE Front End Tool Flow

The inputs to ESE front-end is the system definition consisting of a platform and application code. A library of processing elements, buses, bridges and RTOS is provided in ESE to develop such a platform. The retargetable timing estimation tool in ESE is used to annotate timing to the application code based on the mapping of application code on the platform components. The timed application and platform are input to the TLM generator tool that uses the bus and bridge models to generate a SystemC TLM. This SystemC TLM can be simulated by any commercial or freely available SystemC simulator to provide the performance metrics. The designer can use the metrics to make to application code and/or the platform in order to optimize the system for a particular metric.



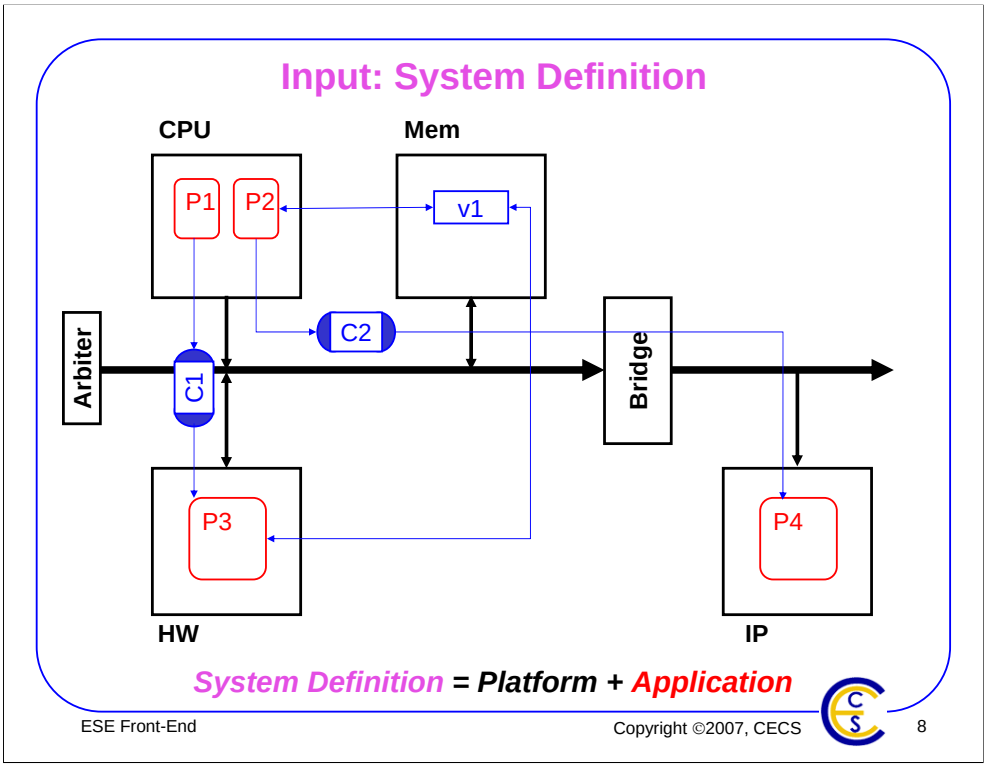
Platform Architecture

Platform architecture consists of set of components and set of connections selected from the library or defined by the user. The platform architecture can be completely or partially defined. More components and connections can be added at a later stage for optimization of some metrics. ESE will upgrade the TLMs automatically to satisfy the new upgrades. It must be noted that ESE does not put any restrictions on the number and type of components and connections used to define the platform.



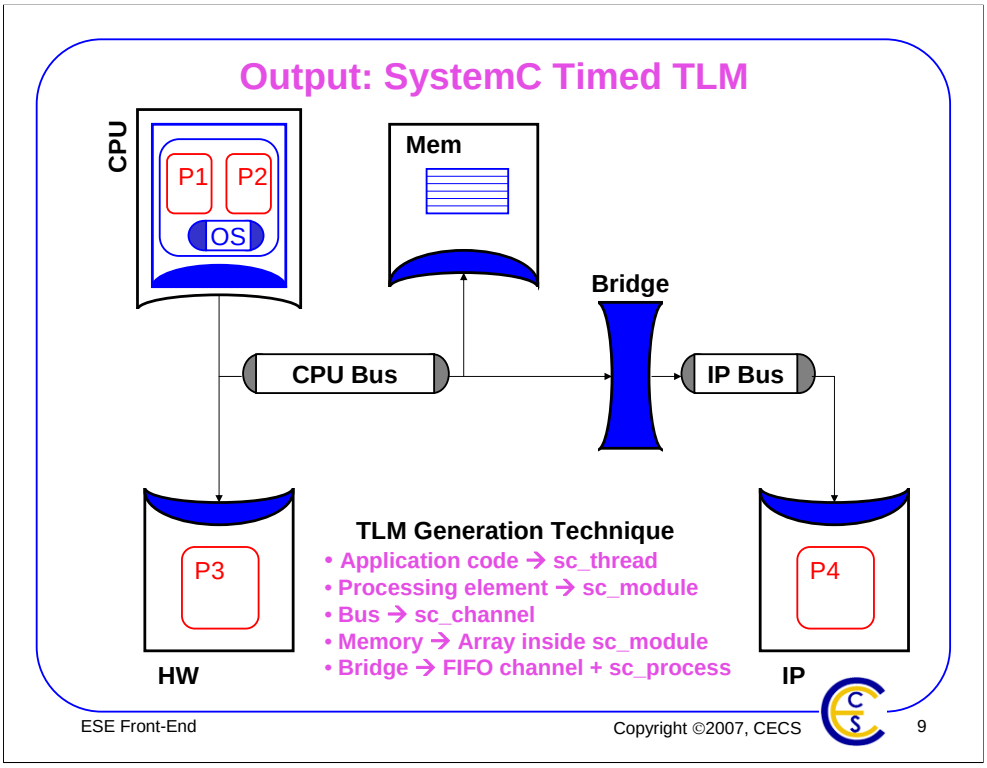
Application Spec

The application input to ESE is C/C++ processes communicating with channels or variables. The C code is part of processes and can be easily modified from the ESE GUI. Even legacy code (usually available in C) can be inserted inside processes. Processes use channels for synchronized communication and variables for unsynchronized communication. Since, the channels and variables can be added graphically, the application spec developer does not need to learn any new language, such as SystemC, for writing concurrent applications.



System Definition

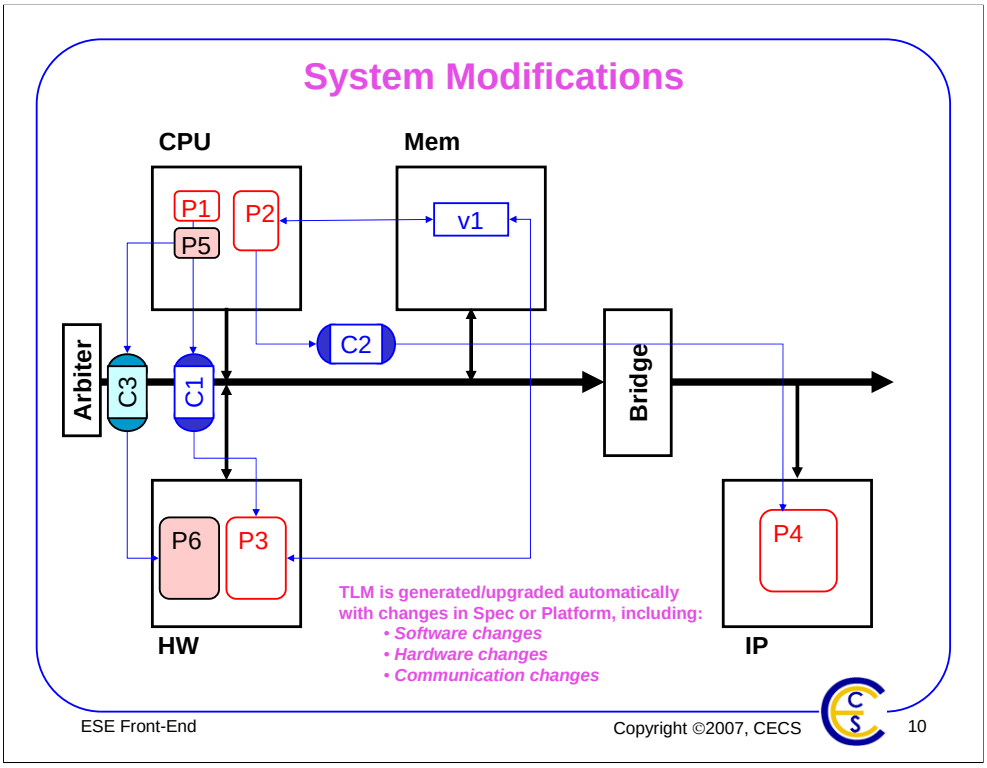
The input to ESE is essentially a mapping of given application to the platform. This mapping has well defined rules and can be easily performed in the ESE GUI. Processes map to processing elements (PEs) such as CPUs, HW components, and IPs. Variables map to memory components, either local to PEs or shared. Channels between processes are mapped to routes consisting of buses and bridges. A valid route must exist between the host PEs of the communicating processes for the channel to be implemented. ESE automatically provides a set of possible routes for each application channel. The designer can easily select the route for each channel in the GUI.



SystemC Timed TLM

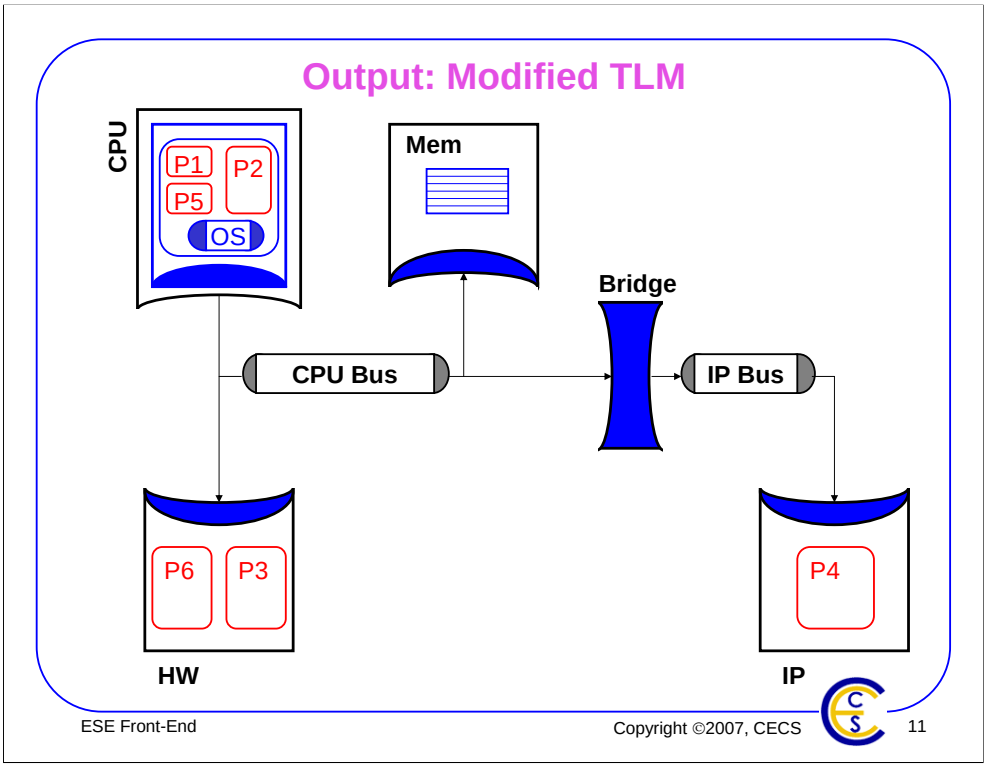
As mentioned earlier, a timed TLM in SystemC is generated automatically for the given system definition. The TLM captures the application, the platform architecture and the mapping of the application on the platform into a single executable model. Components in the model communicate through bus channels whose functionality will be moved to components during generation of PCAMs

The semantics of the ESE TLMs makes them easy to understand, debug and upgrade. Each application and platform object maps to a well defined SystemC construct as shown above.



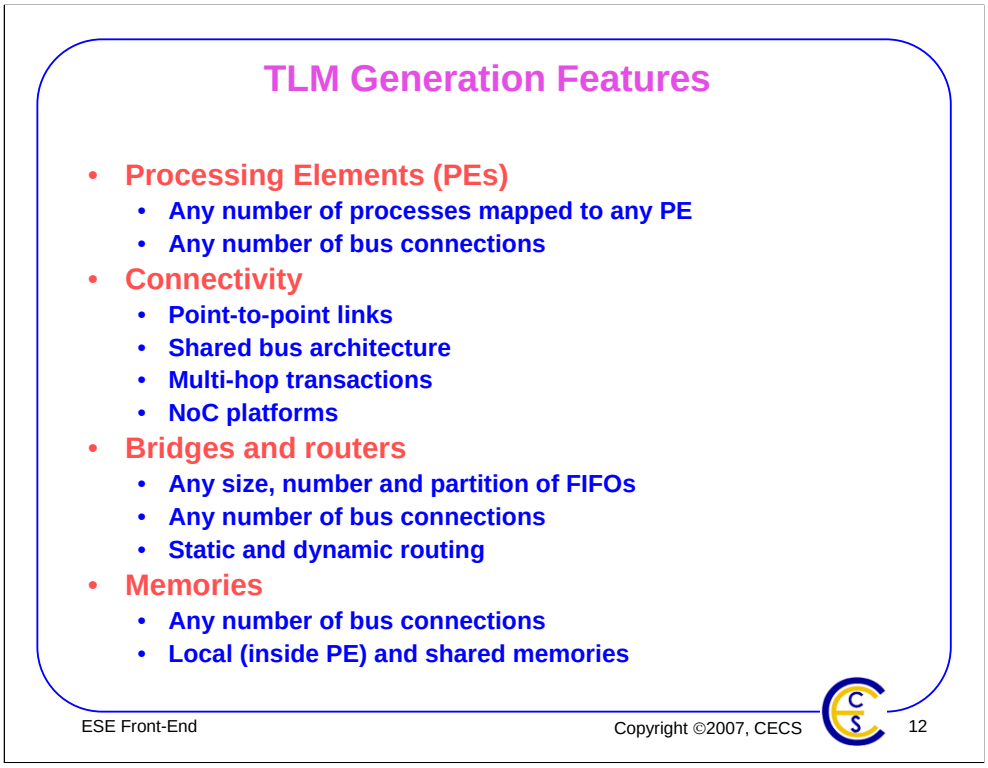
System Modifications

The graphical input in ESE makes it very simple to perform changes to the platform as well as the application. In the above example, new processes P5 and P6 are added to the system design as part of an upgrade. A new channel, C3, is introduced between P5 and P6. Since the application is clearly distinguished from the platform and is entered graphically, such an upgrade is very straightforward in the GUI. A new TLM with added upgrades is generated automatically by ESE. A similar change would be quite difficult to make directly in a SystemC model and more so in a cycle accurate model.



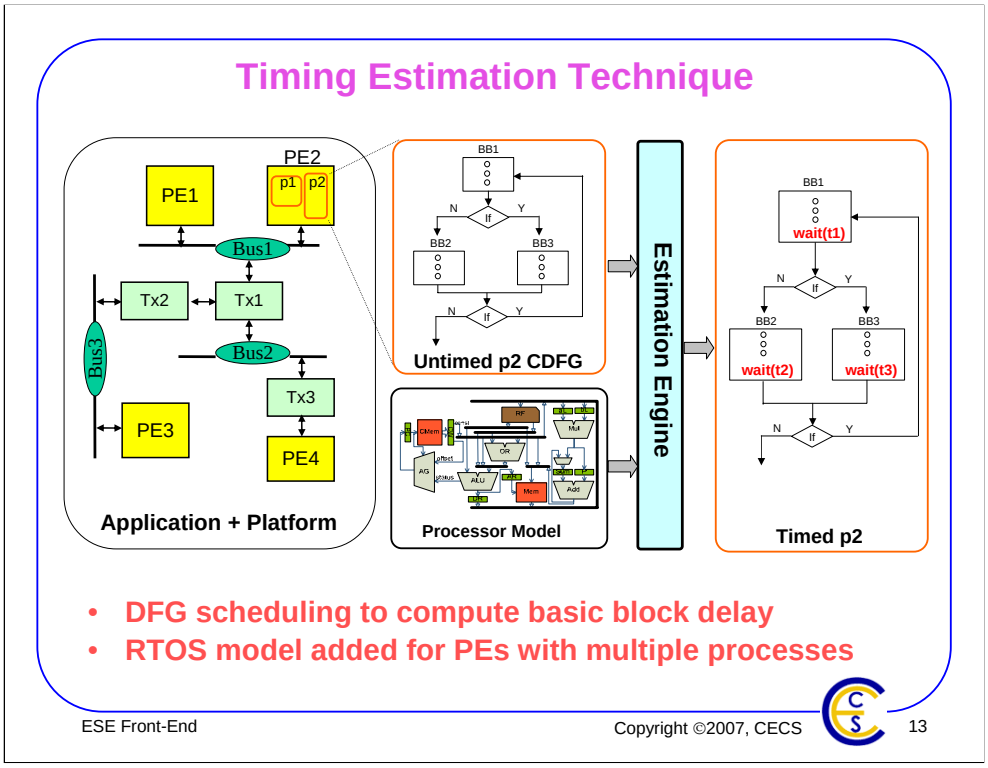
Modified TLM

The automatic TLM generation tool in ESE along with the simplified system upgrade capability provided by the GUI makes TLM upgrade effortless. Hence, even with several design modification cycles, ESE enables designers to keep their TLMs consistent with the latest system definition.



TLM Generation Features

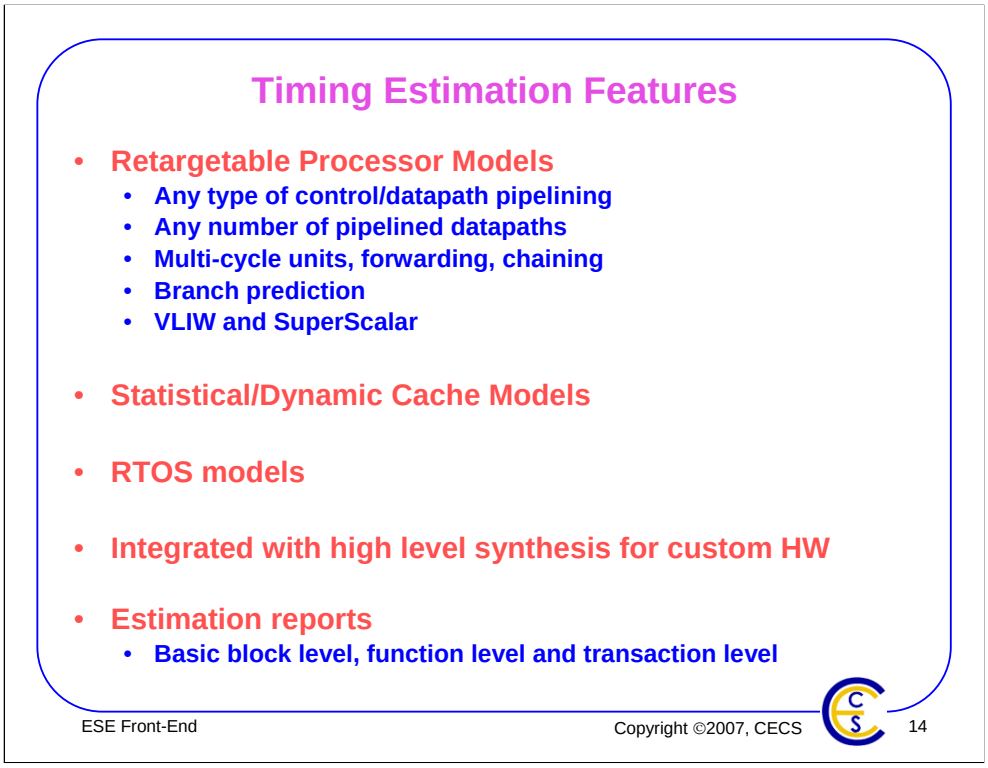
As mentioned earlier, ESE provides great flexibility in platform and application choices. There is no restriction on the number of processes mapped to PEs. If more than once process is mapped to a processor, an RTOS is added from the ESE library to the processes. If the PE is not a third part IP, then any number of connections may be made from PE to system buses. The bus object in ESE is highly flexible as it can be used to denote a point-to-point link, a shared bus or a network link. Also, ESE provides automatic generation of transducers that can act as shared memories, bridges or routers. Therefore, TLMs can be generated for practically any type of platform ranging from single processor systems to complex architectures such as networks on chip.



Timing Estimation Technique

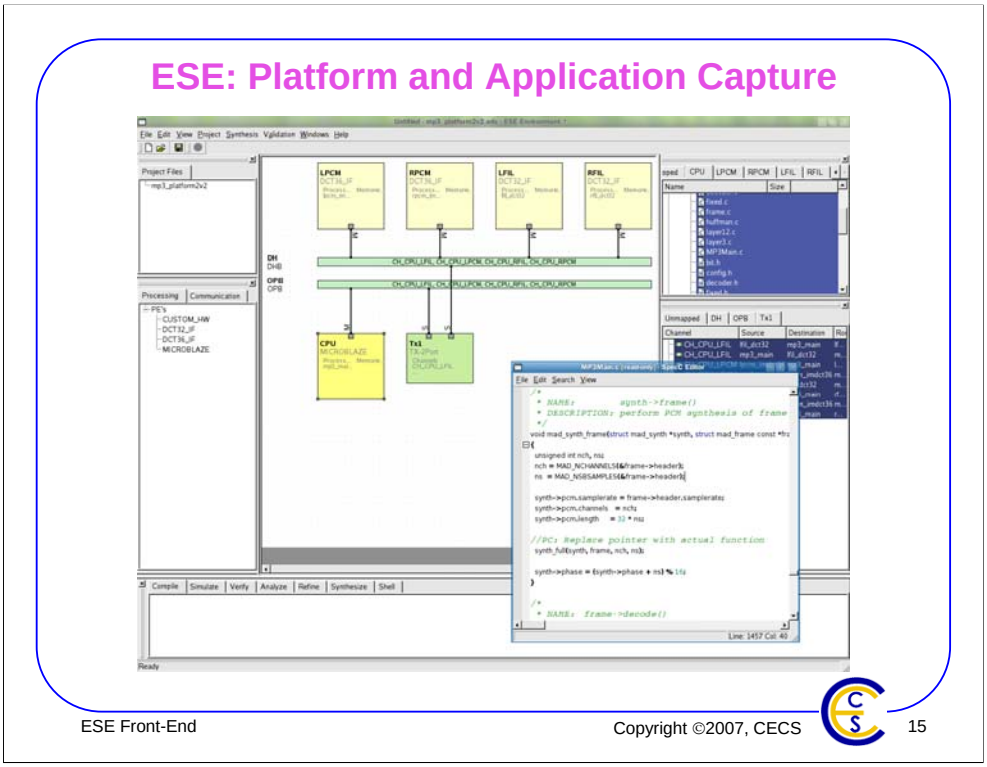
Automatic generation of timed TLMs is the key feature of ESE. This consists of adding timing information to the application code based on its mapping to a given PE. The application code in each process is converted into a CDFG representation as shown above. Then, a retargetable PE model, provided in the ESE database, is used to analyze the execution of each basic block of process code on the given PE. This analysis provides estimated delay for each basic block in the process. The basic blocks are then annotated with the estimated delay to produce a timed process model. This compiled estimation technique, unique to ESE, can be applied to any application code mapped to any type of PE.

Therefore, ESE estimation is fast, retargettable and provides better accuracy at TLM level than even ISMs.



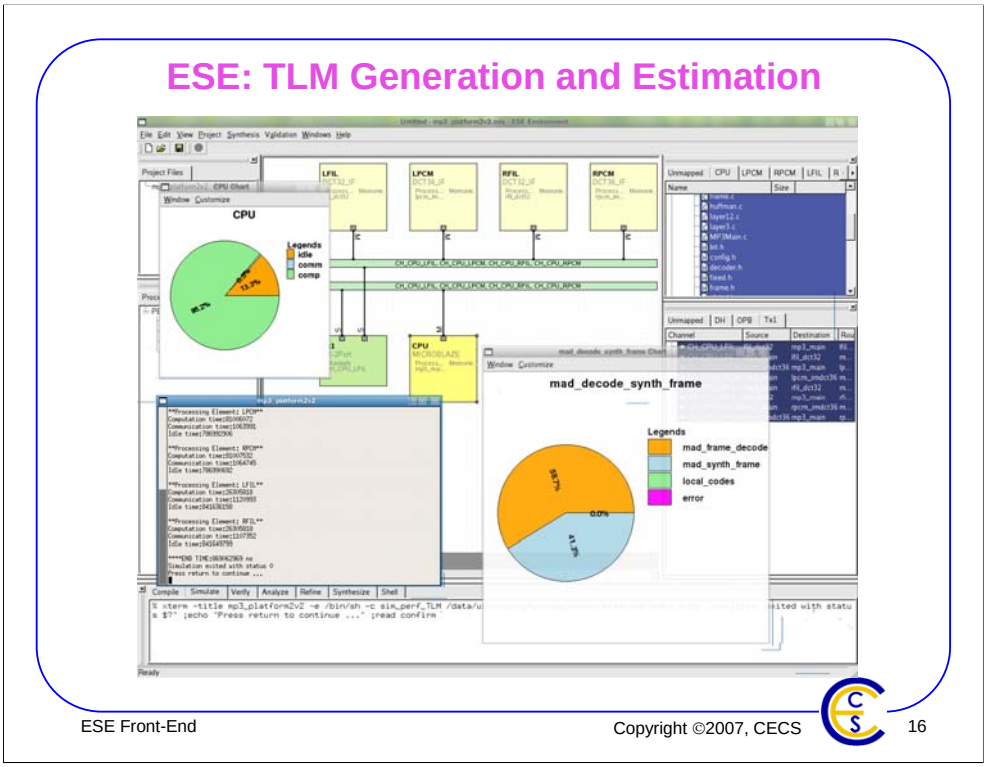
Timing Estimation Features

ESE supports several different types of processors in the platform for heterogeneous system design. The estimation technique in ESE is applicable to processors with various architectural features such as pipelining, branch prediction and various memory hierarchies. It also supports multi-threaded and multi-process applications using RTOS models. Designers can choose different granularity of estimation reports for appropriate analysis of system performance.



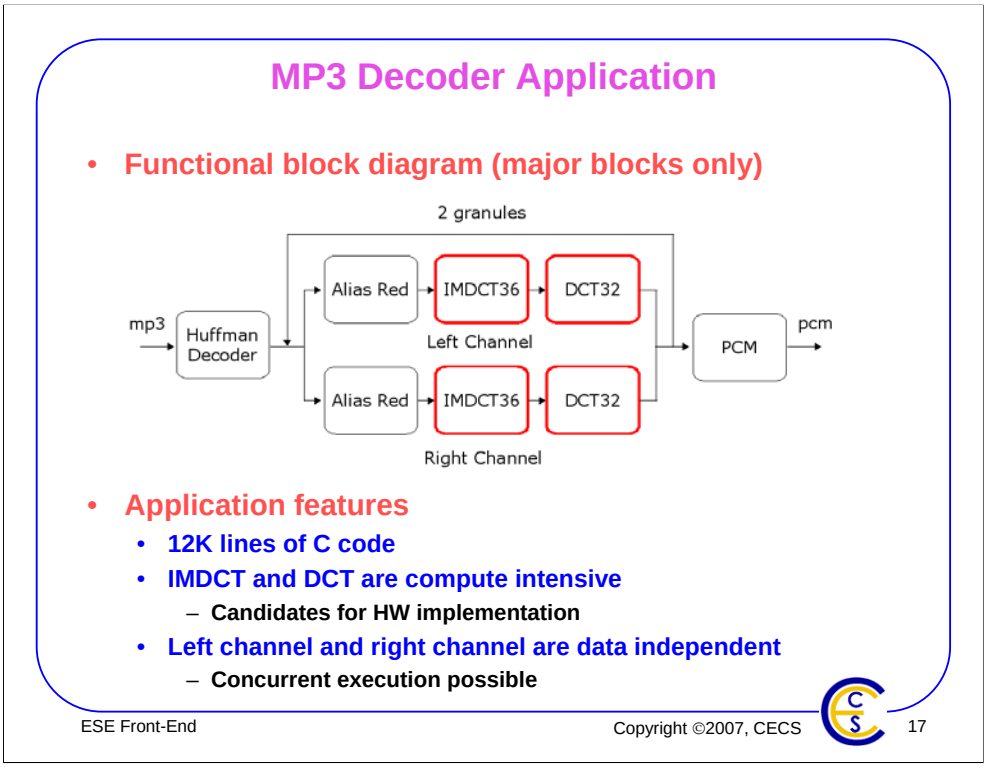
Platform and Application Capture

The central canvas of the ESE GUI shows the platform, which has been assembled from the component database shown on lower left corner. The application processes are organized on the upper right corner, which carries the mapping of the processes to platform PEs. The communication channels are shown in the lower right corner. Process C code can be edited using the editor as shown.



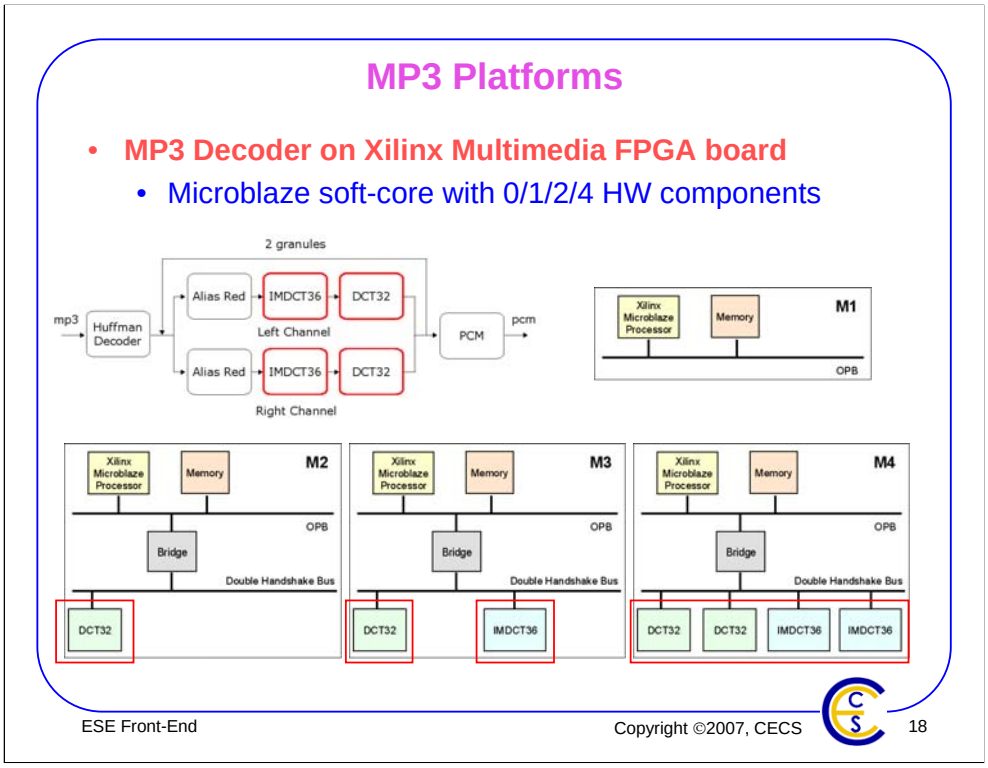
TLM Generation and Estimation

The results after TLM generation and estimation are shown graphically. After automatic TLM generation, the simulation launches a SystemC simulator terminal. At the end of simulation, the estimated metrics can be analyzed graphically using pie charts.



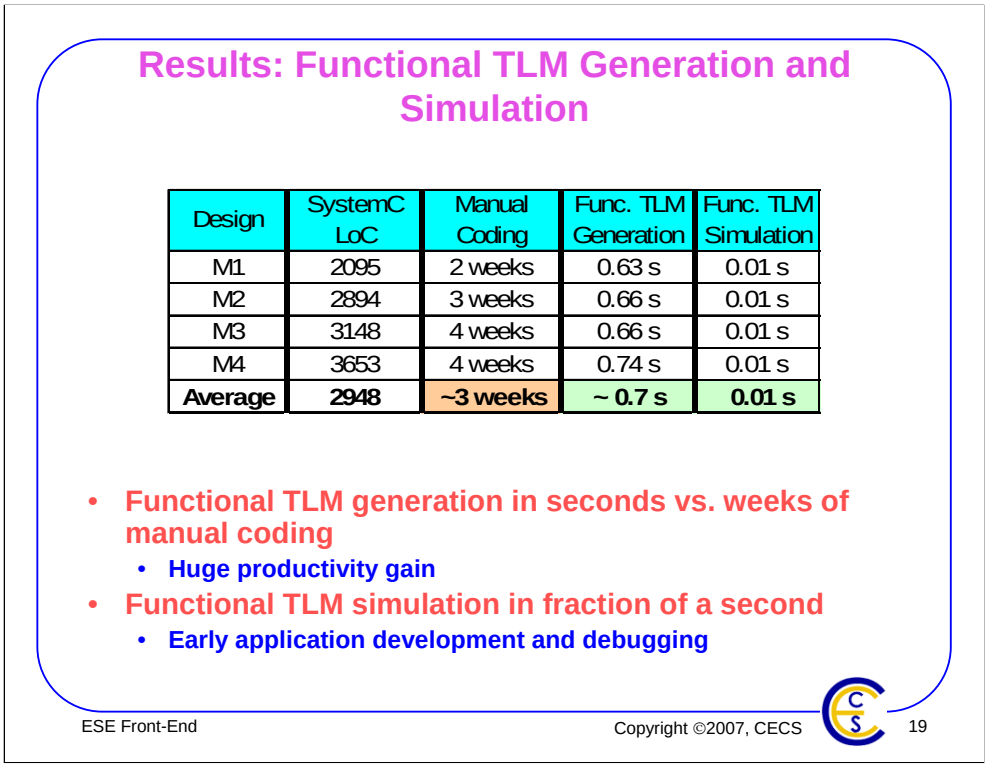
MP3 Decoder Application

To demonstrate the usefulness of ESE, an MP3 decoder application was chosen. The block diagram above shows the IMDCT and DCT transforms that are applied during the stereo decoding on the left and right channels of the MP3 input. These function blocks are the most time consuming part of the decoding and are hence ideal for implementation using custom HW for faster decoding. The C model is also used to create test benches with golden PCM output files. These test benches are used later to verify the ESE generated TLMs.



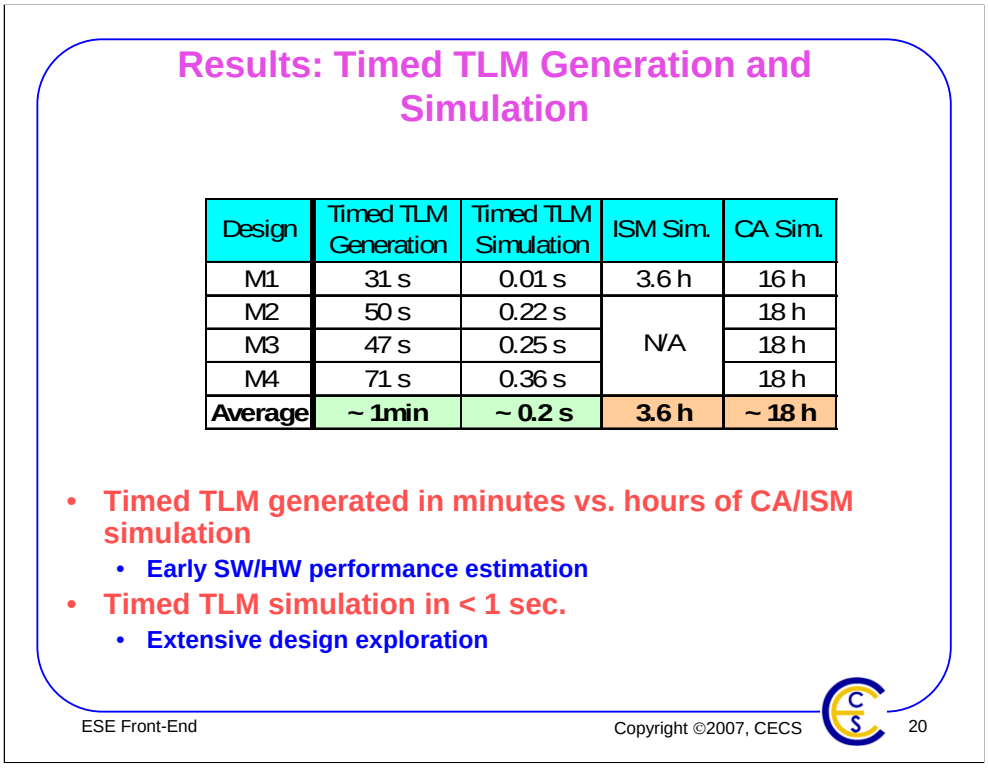
MP3 Platforms

To demonstrate the ESE support for heterogeneous multi-processor platforms, 4 designs were selected as shown above. These platforms consist of 2 different buses, OPB (on-chip peripheral bus) and DH (double-handshake). There are two different types of PEs, a Microblaze processor from Xilinx and custom HW IPs for DCTs and IMDCTs. For platforms consisting of two buses, a transducer is used to allow communication between PEs on the different buses.



Functional TLM Generation and Simulation Results

Besides the timed TLM, ESE can also generate a high speed untimed TLM that can be used for functional verification and application development. Functional TLMs for the selected platforms consist of over 2000 lines of SystemC code that took weeks to write manually. The same code was generated in less than a second by ESE. Since there was no timing added to the application code, the functional TLMs simulated in a fraction of a second. These TLMs are ideal for application development and debugging when the timing estimation for every code change is not of concern.



Timed TLM Generation and Simulation Results

ESE also generates timed TLMs in few minutes even for complex platforms with up to 5 different components. The simulation time for these TLMs was less than a second. In contrast ISMs took 3.6 hours to simulate while PCAM simulation was in the order of 16 to 18 hours. This does not take into account the coding, modification and debugging of ISMs and PCAMs. Design changes at pin/cycle accurate level are significantly more error prone and time consuming which results in days or even weeks for each design change and evaluation cycle. Using ESE, this cycle can be reduced to an order of minutes. Therefore, designers can spend greater time in innovative design optimizations and almost no time in mundane pin/cycle accurate model development and simulation.

Results: Estimation Quality

Error %= (1 - Estimated cycles/ Board Cycles)*100

ISM Error

Cache size	M1	
	Board	ISM Error
0K/0K	27215K	39.48%
2K/2K	8914K	18.38%
8K/4K	5828K	3.55%
16K/16K	4413K	-16.32%
32K/16K	4384K	-16.60%
Average	N/A	**18.86%**

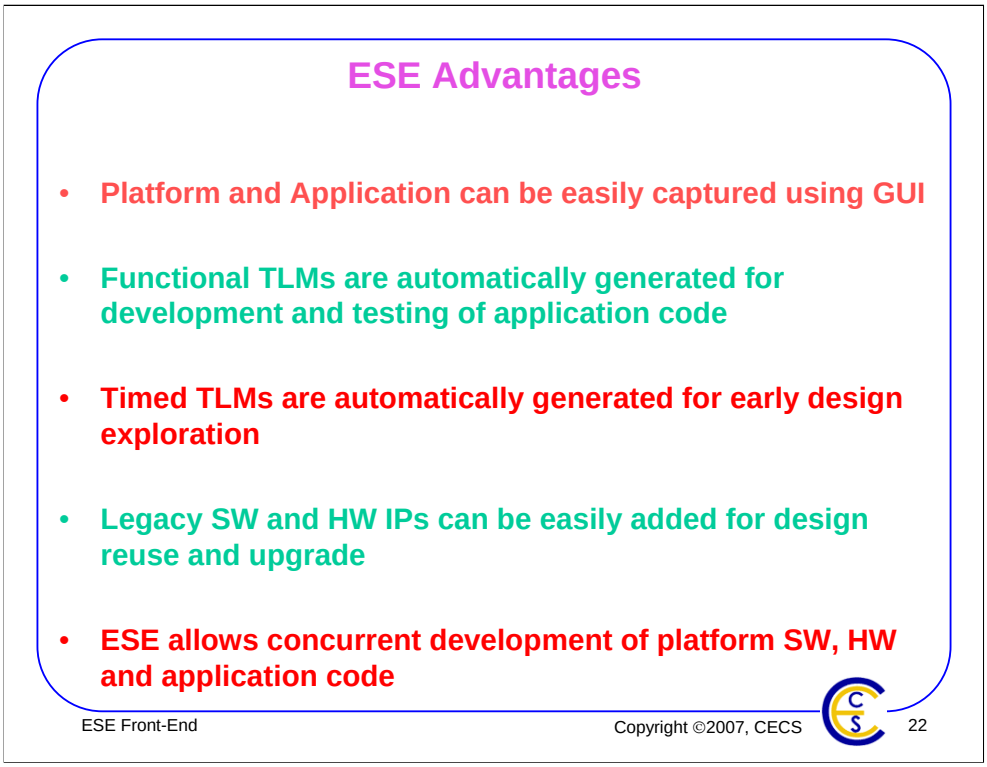
Timed TLM Error

Cache Size	M1	M2	M3	M4
0K/0K	6.27%	9.00%	18.18%	18.61%
2K/2K	6.68%	-7.16%	-15.79%	-9.35%
8K/4K	4.74%	9.13%	-1.66%	-0.18%
16K/16K	-13.83%	4.66%	2.63%	3.65%
32K/16K	-13.89%	-8.29%	1.57%	2.29%
Average	**9.08%**	**7.65%**	**7.97%**	**6.82%**

- **TLM estimation applicable to all designs**
  - **ISM only available for SW**
- **TLM estimation error < ½ of ISM error**
  - **Reliable design exploration with timed TLMs**

Estimation Quality Results

ESE can automatically generate timed TLMs for heterogeneous multi-processor platform. In contrast, ISMs can only provide estimation for single processors. (That is why ISM is available only for M1 platform.) The above results compare the ESE TLM estimation with actual board measurements obtained using a timer. It can be seen that timed TLMs are more accurate than ISMs in most cases and less than half as erroneous as ISMs on average. Therefore, designers can use ESE for reliable estimation of their design at an early stage even before the HW RTL is ready.



ESE Advantages

There are numerous advantages of using ESE. The product specification and implementation is easily captured with proprietary GUI. All models are generated automatically after design decisions are made by the users. This saves enormous amount of time in learning modeling languages and writing and interfacing appropriate models. Reliable timing estimation in ESE allows for rapid and early design space exploration.

Product upgrades are simplified because ESE allows convenient reuse of legacy application code and design decisions. ESE allows parallel development of SW, HW and application code and their integration. In other words, it allows early testing of each thereby allowing faster and globally distributed development. Similarly, the upgrades can be easily developed any time and anywhere.

Acknowledgments

- **We would like to acknowledge the previous R&D teams who contributed many concepts and methods used in ESE 2.0**
  - SpecCharts/SpecSyn ('92): F. Vahid, S. Narayan, J. Gong, S. Bakshi
  - SpecC/SCE ('00) team: R. Doemer, J. Zhu, A. Gerstlauer, J. Peng, D. Shin, L. Cai, H. Yu
- **We also want to thank P. Chandraiah for MP3 reference code, Q.V Dang for developing the GUI, and A. Gerstlauer for many very useful discussions.**