

Switching options

‘What if’ analysis allows a range of system architectures to be examined. By **Kai Liu**.

When designing a highly complex switch, the many architectural variables can greatly impact system level performance and cost. In the past, architectural comparison was performed by Excel spreadsheet analysis based on the experience of system architects. By moving to an automated process, a greater number of test cases can be evaluated with improved accuracy. Simulation can identify the combination of design variables which provides the best performance while minimising design expense, such as over designed components.

Siemens has used this approach in the design of a 100Mbit/s industrial Ethernet switch based upon IEEE802.3 standard. An Electronic System Level (ESL) design methodology was used to create an executable specification for the switch. This methodology and its corresponding toolset enabled macro architectural exploration.

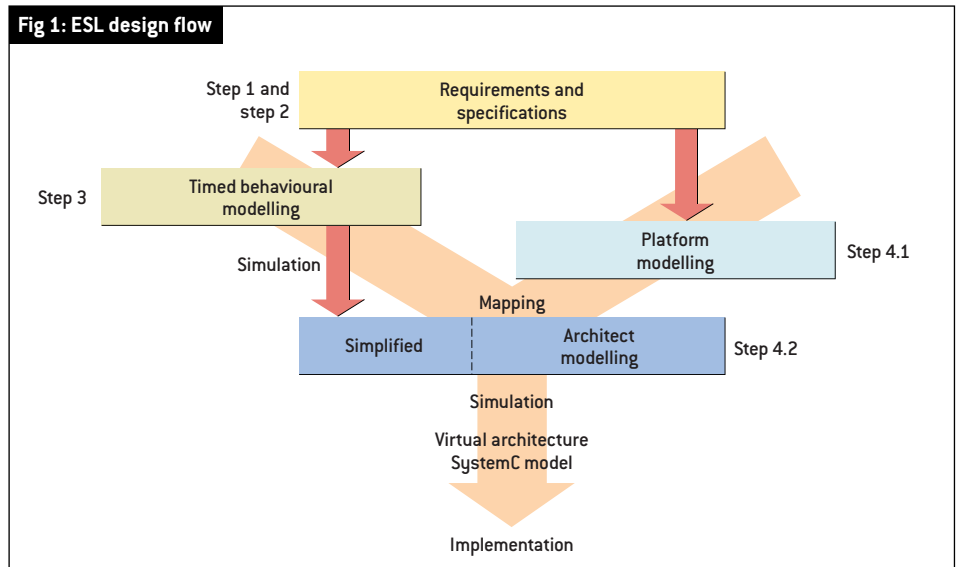
The basic design flow is shown in figure 1. The requirements and system level specification are defined first. Next, the timed behavioural modelling step defines the internal structure and signal transfer, explaining all necessary functions and couplings. Step 4.1 designs the platform, a ‘black box’ that contains the entire functionality of the system.



THE ESL TOOLSET ALLOWS THE DIFFERENT SOLUTIONS TO BE CONFIGURED ACCORDING TO

THE APPLICATION REQUIREMENTS.

Fig 1: ESL design flow



Architectural modelling in step 4.2 extends the meaning of the platform model by considering both the functional meaning and the platform meaning of a structure. Additional information is added to the behavioural model at the architectural design level. The values assigned to the interfaces between the functions and the platforms represent time duration. The performance of the entire system – such as bus utilisation, memory requirement, power consumption and cost – can be extracted through simulation.

Once the system environment is defined, its behaviour can then be studied. First, the system is described without refinements: this is a very abstract, algorithmic description. Benchmarking is based on this abstract system and can be reused later for the refined system.

After the abstract model is verified, the model is refined step by step. Various test cases are created, based on the traffic characteristics of industrial Ethernet communication. The test cases are built using an Excel file and the XML file

generated from the Excel file is read by the CoFluent Studio model testbench.

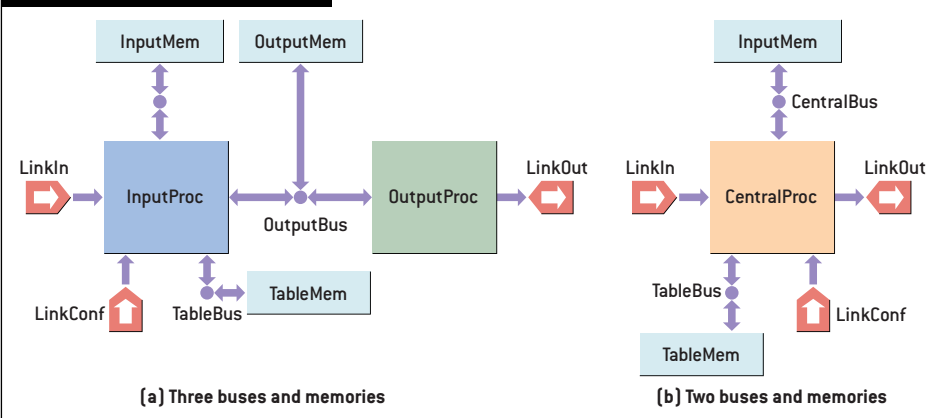
Once the functional diagram and algorithms for each block are defined, several configurations can be extracted. The ESL toolset allows the different solutions to be configured according to the application requirements. Seven different functional configurations were obtained for the application requirement.

Searching for the appropriate structure

The platform represents the physical solution for the project. Searching for the appropriate structure consists of designing a physical architecture. The platform is composed of processors and their relations, such as communication nodes and shared memory. A processor is a physical resource that is capable of running functions, either hardware or software.

The platform structure, representing physical components, is described. This abstract platform enables the first mapping from the functional model onto the platform model and ensures the

Fig 2: Platform model refinements



behaviour works properly after mapping.

The architectural model is created by mapping the functional model to platform model with the ESL toolset. It is a drag and drop operation. The mapping tool analyses the possible solutions automatically. The user selects how to map them onto the communication nodes on the platform and defines their corresponding attributes.

The bus width requirement varies for different architectures, from a single bus to multiple buses. Two refined platform solutions were developed for this design: one with three buses and memories, the other with two (see figure 2).

Tests were created to verify the impact of various design options, such as different scheduling algorithms, maximal hash times, priority encoding and the effect of breaking frames into smaller cells for transmission. Interleave versus non interleave mode was also considered.

In order to obtain the memory size requirement for this system, a test case using

burst traffic is selected. When the traffic burst ends, the memory requirement decreases. In this example, the burst duration was set to 1ms, the traffic period to 100ms and the load factor of the basic traffic to 12%. For load factors of more than 12%, the memory requirement increases continuously. The memory size requirement is approximately 60kbyte.

Using this test case, the memory size can be determined for different architectures. Table 1 lists the results for a transfer rate of 100Mbit/s.

If the width of InputBus is reduced in architectures B, D and E, the switch still works, but the memory size increases continuously as more memory is needed on the input side to store frames that cannot be forwarded immediately. From Table 1, we see that C2 requires smallest InputMem, since InputMem is only needed to store one longest frame for each port. All frames are forwarded immediately.

Siemens successfully used ESL design to generate and validate the candidate architectures

and to detect system level bottlenecks. Siemens found using an ESL methodology reduced the risk of redesign and accelerated the entire design process.

Ten architectural models were designed and compared. The toolset was used to construct the system quickly and enabled performance issues – such as memory size and bus width – to be studied efficiently. It took approximately two months to create five refined functional models and to verify their behaviour.

The functional models were mapped to the platform models and all of the performance data was extracted in two weeks. Automatic mapping and analysis tools enabled these goals to be reached quickly.

The performance analysis concluded that separating the address table and the VLAN table memory from the frame memory increased system performance, while the bus width between the frame memory and the processing units can be shortened. Memory requirements were similar in all of the architectures. The performance data was obtained from the simulation, and is more precise than static analysis methods.

Analysis determined that the critical point in the system is the buffer storing incoming frames and creating the memory became the focus throughout the switch design.

Bottlenecks were also found at the address table and the VLAN table. Creating different architectural solutions enabled the best architecture to be determined for the hardware implementation. Adjusting some critical parameters provides a suitable reference to the hardware designers.

The test cases were created in an Excel file. The XML file enabled the CoFluent Studio toolset and Excel to interoperate with the test cases. The switch model created in CoFluent Studio can be used as the executable specification of the further TLM or RTL implementation.

Author profile:

Kai Liu is a technical marketing engineer for TRIAS Mikroelektronik.

Fig 3: Memory size requirements at 100Mbit/s

Configuration	SharedMem	TableMem	InputMem	OutputMem	Total
A1	60				60
A2	27	33			60
B1			5	38	73
B2		33	2	28	63
C1			35	28	63
C2		33	1	28	62
D1			35	28	63
D2		33	2	28	63
E1			35	28	63
E2		33	2	28	63

All values in kbyte