The Role of SystemC in the Evolution of Hardware Design

Zainalabedin Navabi
navabi@ece.wpi.edu

Worcester Polytechnic Institute

Noth America SystemC User's Group

Outline: Evolution, Principles, NP Example

- Transistor level to TLM
  - Design abstraction
  - System specification
  - Interfaces
- TLM Principles
  - What is TLM?
  - TLM advantages
  - Channels and mechanisms
- NP Example
- University Program
  - Research
  - Teaching

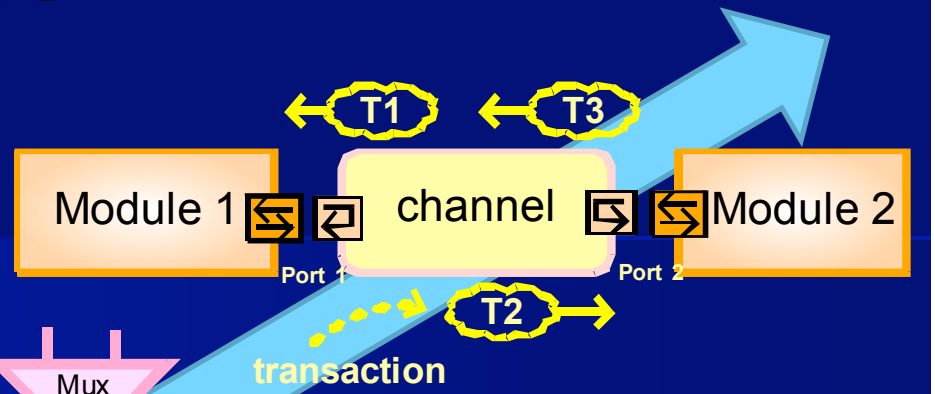
Outline: Evolution, Principles, NP Example

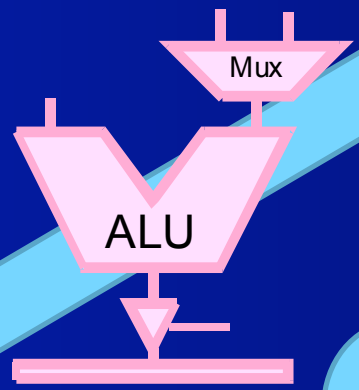
- Transistor level to TLM
  - Design abstraction
  - System specification
  - Interfaces
- TLM Principles
  - What is TLM?
  - TLM advantages
  - Channels and mechanisms
- NP Example
- University Program
  - Research
  - Teaching

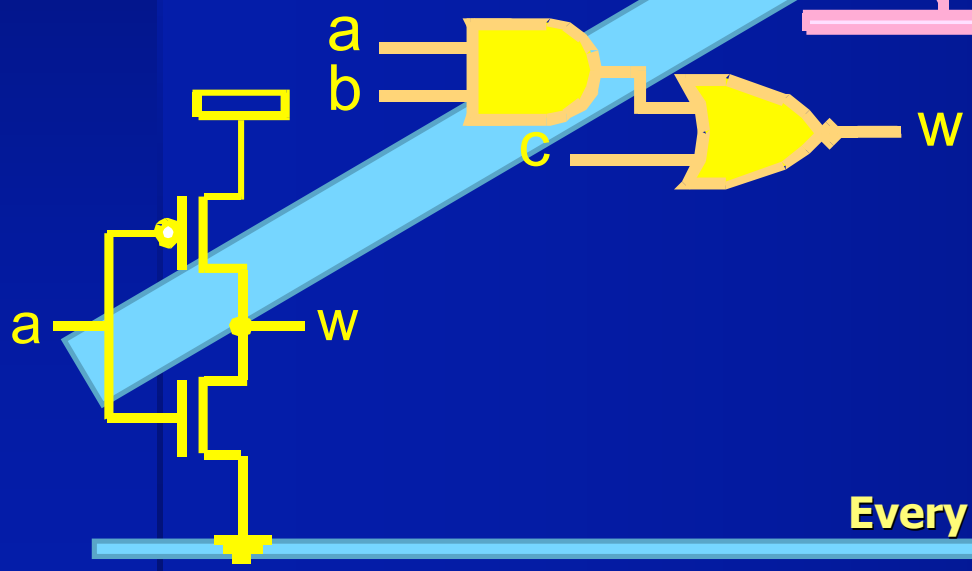
Design Abstraction

- Transistor level
- Gate Level
- RT Level
- Transaction Level

Transistor to TLM Evolution





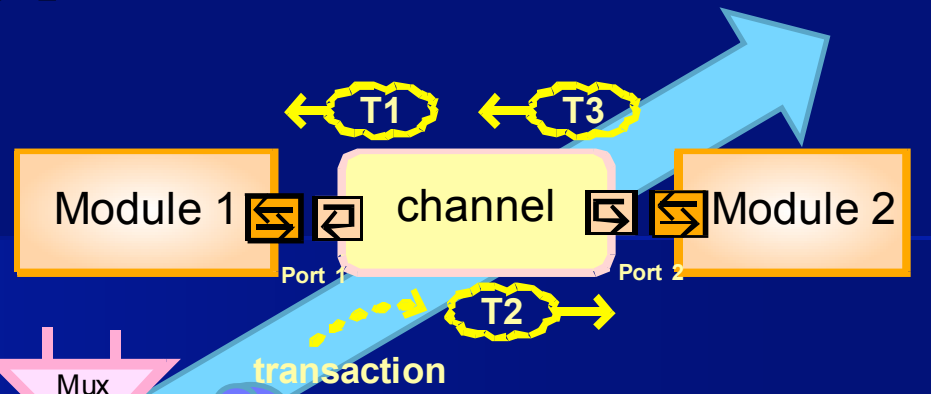


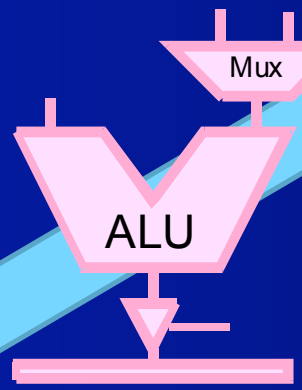
More complex components

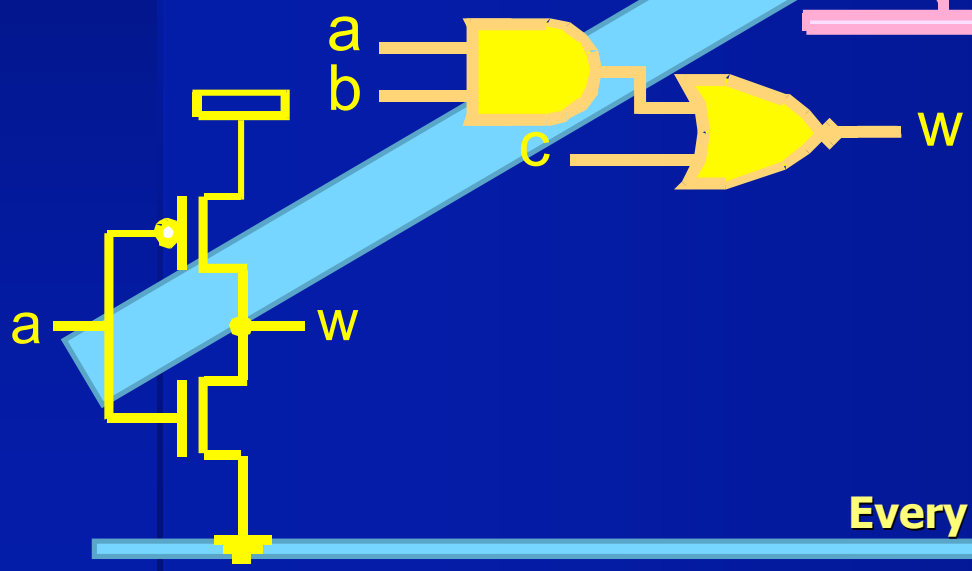
More complex communications

Every 15-18 years

Transistor to TLM Evolution







More Abstract Timings

More Data Being Transferred

Higher Level Descriptions

Every 15-18 years

System Description

- Transistor level
  - HSpice, Schematic
- Gate level
  - PALASM, TEGAS
- RT Level
  - HDLs
- Transaction Level
  - SystemC, C++, C, …

Transistor to TLM Evolution -Description

```
If (ready) {
     for(int i=0; i<31 ; i++){
    }
 }
```

```
always @( posedge clk )
  begin
    #4 q <= d;
  end
```

```
and #(2,4)
     ( im1, a, b ),
     ( im2, b, c );
or  #(3,5) ( y, im1, im2, im3 );
```

```
mn1 VSS IN OUT VSS nmos l=0.24u w=0.72u
mp1 VDD IN OUT VDD pmos l=0.24u w=0.72u
cLoad OUT VSS 50fF
```

Every 15-18 years

Transistor to TLM Evolution -Description

- **Any level of abstraction requires a way of describing its immediate lower level in a compatible language with the higher abstraction level**
  - Switches at the gate level
  - Gates at the RT level
  - RTL at TLM
    - TLM is a C/C++ language
    - RTL uses SystemC

SystemC

```
SC_MODULE (and2)
{
        sc_in   < sc_logic > a;
        sc_in   < sc_logic > b;
        sc_out < sc_logic > c;
        SC_CTOR(and2){
            SC_METHOD(anding);
            sensitive << a << b;
        }
        anding (){
            c = a & b;
        }
};
```

Outline: Evolution, Principles, NP Example

- Transistor level to TLM
  - Design abstraction
  - System specification
  - Interfaces
- TLM Principles
  - What is TLM?
  - TLM advantages
  - Channels and mechanisms
- NP Example
- University Program
  - Research
  - Teaching

What Is TLM?

- Means of Design at ESL
- ESL: Electronic System Design
- Defines Design Methodology
  - For implementing large digital systems
- A designer can easily focus on:
  - Task partitioning
  - System functionality
  - Developing testbenches in a more organized fashion

What Is TLM?

- TLM utilities include:
  - Ports, Interface, and Channels
  - For describing communications between modules

TLM is going to become the starting point in system level design

What is TLM: Timing

- **Un-timed**
  - No timing for computation and communication parts
  - Uses channels for communication
- **Approximately timed**
  - An approximate delay is computed for computation and communication parts
  - The computed delays are inserted to design with simple wait statements
- **Cycle Accurate**
  - The design must be simulated correctly in each clock cycle
  - A clock signal is generally inserted to the design

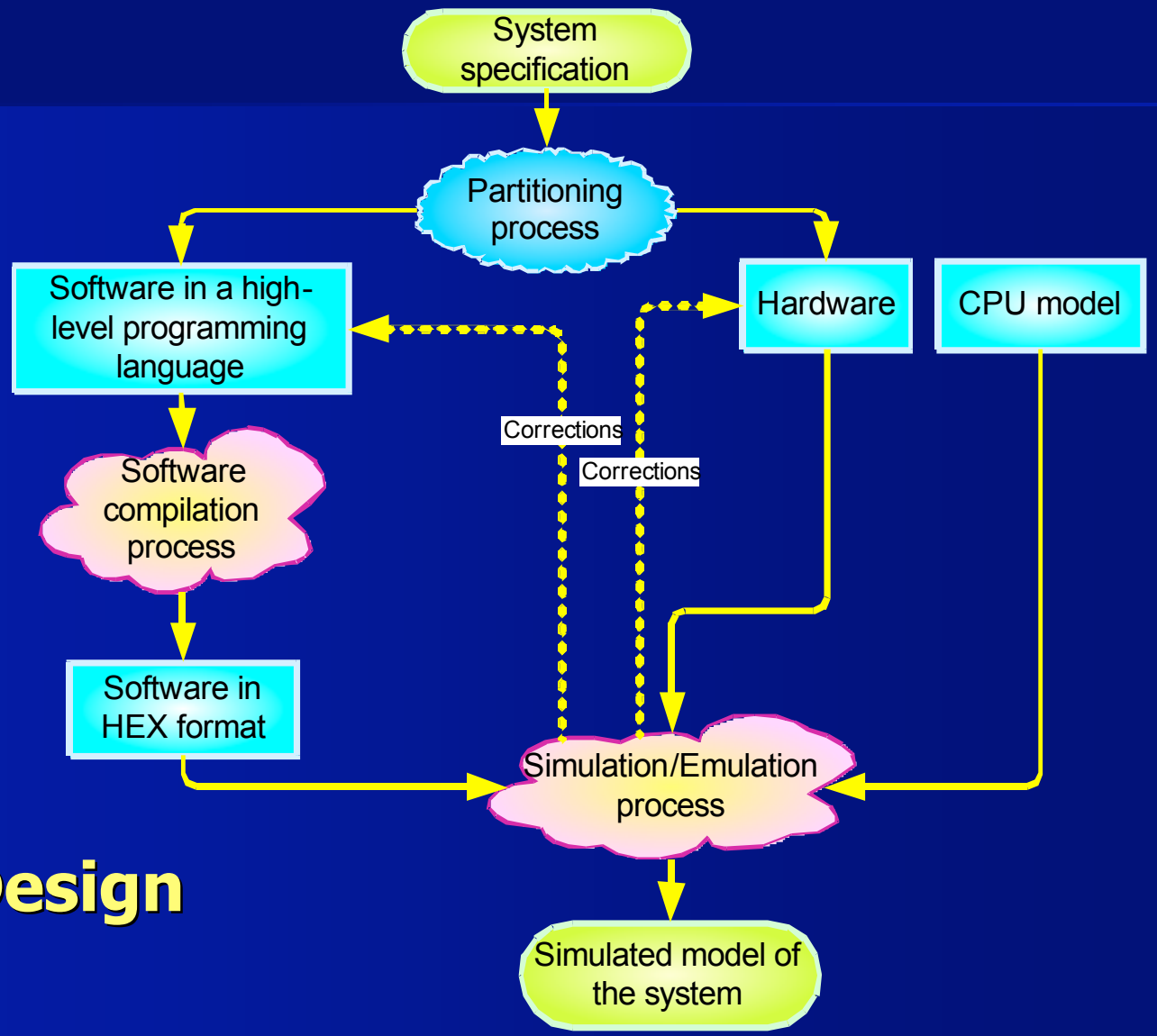
Advantages of TLM

- Fast Simulation
  - minimizing the number of events
- Modeling communications between modules are easy
- Reducing product development risk
- Rapid evaluation and customization of the design
  - Design Space Exploration
- Testbench migration

Advantages of TLM

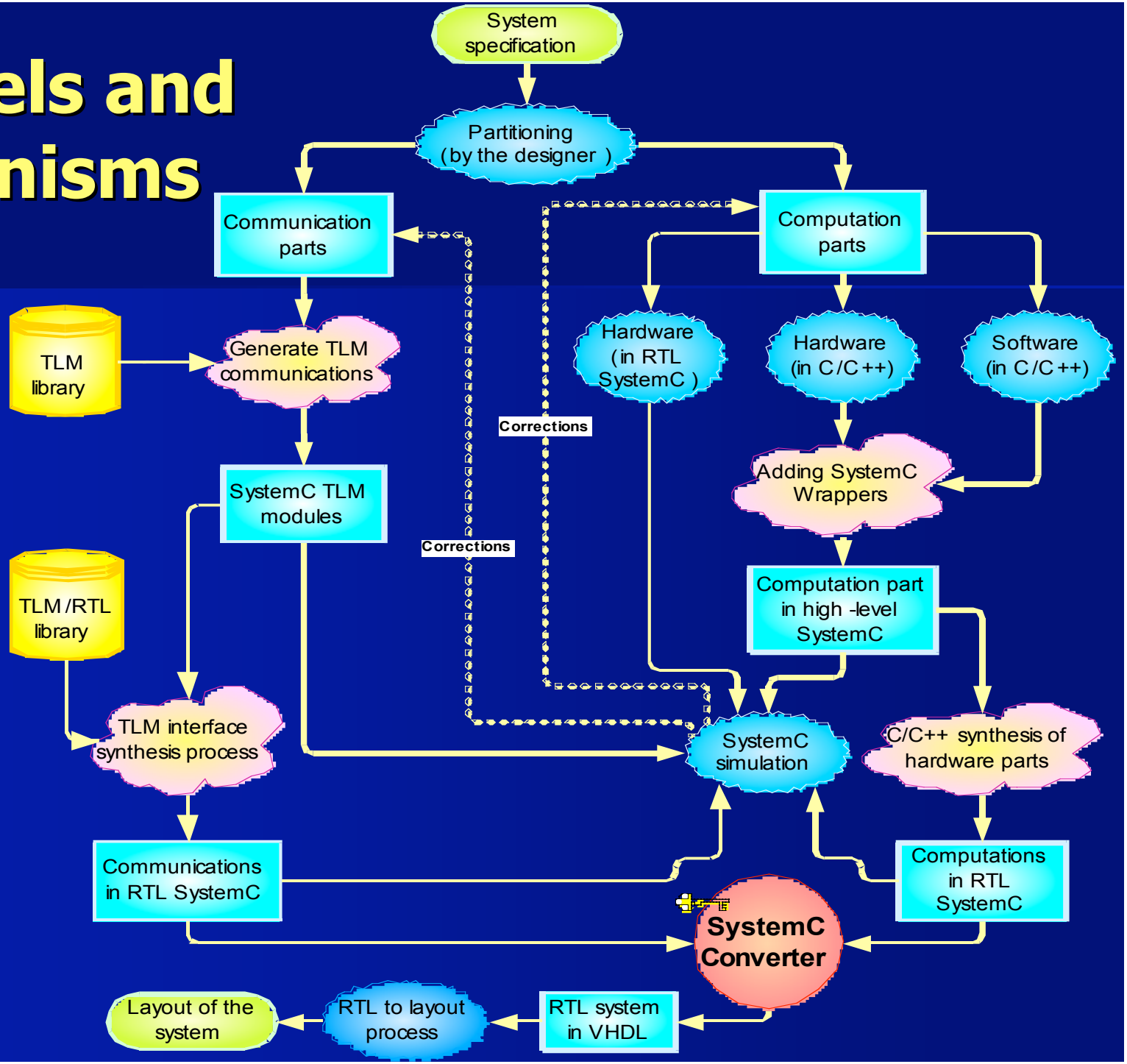
- **SW development delay**
  - SW team can begin SW developing or testing stage much sooner in ESL design methodology.
- **HW/SW communication**
  - HW parts can communicate with SW parts in this common environment.
  - Makes the SW debug easier.
- **Design space exploration**
  - Designers can decide on its partitioning (module and HW/SW partitioning) in the early stages of the design.
- **Simulation speed**
  - The number of events decreases.

Channels and mechanisms



Traditional Design

Channels and mechanisms

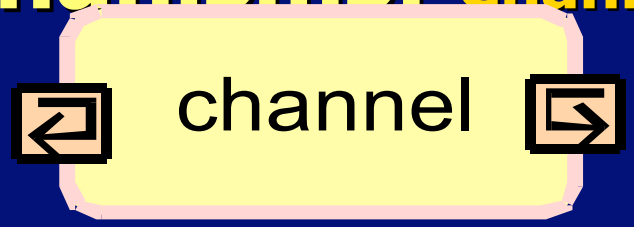


TLM Mechanisms

Modules: Described in C/C++/SystemC

Channels: Are TLM Protocols

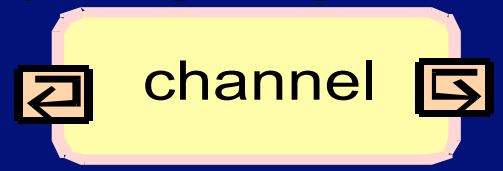
TLM Mechanisms: Channels




```
template <class T>class tlm_fifo :
  public virtual tlm_fifo_get_if<T>,
  public virtual  tlm_fifo_put_if<T>,
  public sc_prim_channel {
public:
     T get( tlm_tag<T> * ) {
      while( is_empty() ) { wait( m_data_written_event );  }
        ...
        return buffer->read();
     }
circular_buffer<T> *buffer;
 }
```

Unidirectional

TLM Mechanisms: Channels



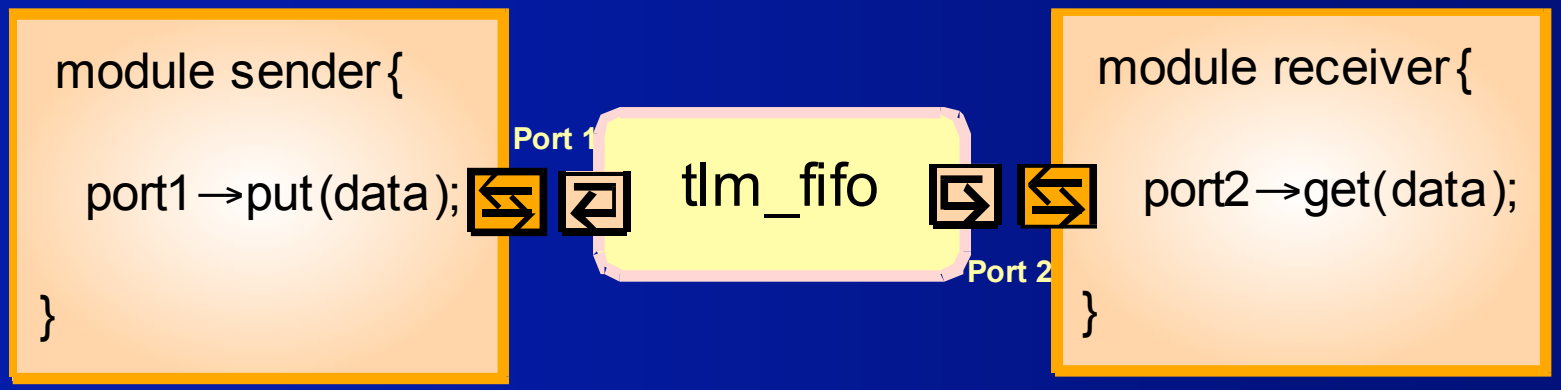
Bidirectional
FIFO Based

```
template < REQ , RSP >
  class tlm_req_rsp_channel : public sc_module{
public:
  // uni-directional slave interface
  sc_export< tlm_fifo_get_if< REQ > > get_request_export;
  sc_export< tlm_fifo_put_if< RSP > > put_response_export;
  // uni-directional master interface
  sc_export< tlm_fifo_put_if< REQ > > put_request_export;
  sc_export< tlm_fifo_get_if< RSP > > get_response_export;

  tlm_fifo<REQ> request_fifo;
  tlm_fifo<RSP> response_fifo;
};
```

TLM Mechanisms: Channels

```
class tlm_transport_channel : public sc_module{
public:
  // master transport interface
  sc_export< tlm_transport_if< REQ , RSP > > target_export;
  // slave interfaces
  sc_export< tlm_fifo_get_if< REQ > > get_request_export;
  sc_export< tlm_fifo_put_if< RSP > > put_response_export;
  ...
  RSP transport( const REQ &req ) {
    request_fifo.put( req );
    response_fifo.get( rsp );
  }
  tlm_req_rsp_channel< REQ , RSP > req_rsp;
};
```

TLM Example


module sender{
port1→put(data);
}
Port 1
tlm_fifo
Port 2
module receiver{
port2→get(data);
}

TLM Details: MP3 Example

Initial Module -- Input Stream (fopen, fread ...) -- Output Stream (fwrite, fclose ...)

- → **setwavehead** (fseek, fread, …)
- → **get_audio_info_module** (….)
  - → **decode_header_module** (….)

Decode Module

- → **process_header** (mpa_decode_header, check_header, decode_header)
- → **process_frame**
  - → **decode_layer1** (l1_unscale)
  - → **decode_layer2** (l1_unscale, l2_select_table, l2_unscale_group)
  - → **synth_filter_module** (synth_filter (dct32, round_sample, MULS))
  - → **decode_layer3**
    - → **decode_side_info** (code segment until seek_to_main_data function)
    - → **decode_scale_factors** (seek_to_main_data, lsf_sf_expand, exponents_from_sclae_factors)
    - → **huffman_decode_module** (huffman_decode)
    - → **decode_stereo** (compute_stereo)
    - → **reorder_module** (reorder_block)
    - → **antialias_module** (compute_antialias_integer, compute_antialias_float)
    - → **IMDCT_module** (MULL, imdct36, imdct12)

TLM Details: MP3 Example

```
#include <stdio.h>
#include "systemc.h"

#include "initialModule.h"
#include "decodeModule.h"

using tlm::tlm_req_rsp_channel;

int sc_main(int argc, char* argv[])  {

     initial initial_inst("initial_inst");
     decode decode_inst("decode_inst");

     tlm_req_rsp_channel< mpa_data_buf_type, bool > mpa_buf_ch;
     tlm_req_rsp_channel< data_buf_type, bool > data_buf_ch;
     tlm_req_rsp_channel< int, bool > nFrameSize_ch;
     tlm_req_rsp_channel< int, bool > nDataLen_ch;
     tlm_req_rsp_channel< AVCodecContext , bool > avctx_ch;

. . .

     initial_inst.data_buf_req(data_buf_ch.get_request_export);
     initial_inst.data_buf_rsp(data_buf_ch.put_response_export);

     decode_inst.data_buf_req(data_buf_ch.put_request_export);
     decode_inst.data_buf_rsp(data_buf_ch.get_response_export);

. . .

// repeat for the other 4

       sc_start(-1);

       return 0;
```

TLM Details: MP3 Example

```
// Data Types

#ifndef __TLMTYPES
#define __TLMTYPES

#include "typedef.h"
#include "mp3dec.h"

struct mpa_data_buf_type
{		unsigned char mpa_data[1792];

};

struct data_buf_type
{		signed short data[PCM_FRAME_SIZE];
};

struct head_buf_type
{		UINT8 head[44];
};

struct granule
{		GranuleDef granules[2][2];
};

struct exponent
{		int16_t exponents[576];
};

#endif
```

TLM Details: MP3 Example

```
#ifndef __INITIALMODULE
#define __INITIALMODULE

using tlm::tlm_blocking_get_if;
using tlm::tlm_blocking_put_if;
using tlm::tlm_req_rsp_channel;

SC_MODULE(initial)
{
. . .
        sc_port< tlm_blocking_get_if< data_buf_type > > data_buf_req;
        sc_port< tlm_blocking_put_if< bool > > data_buf_rsp;
. . .
        // repeat for the other 4
. . .
        // The rest of the initial block header
}
#endif
```

TLM Details: MP3 Example

```
#ifndef __DECODEMODULE
#define __DECODEMODULE

using tlm::tlm_blocking_get_if;
using tlm::tlm_blocking_put_if;
using tlm::tlm_req_rsp_channel;

SC_MODULE(decode)      {

. . .
        sc_port< tlm_blocking_put_if< data_buf_type > > data_buf_req;
        sc_port< tlm_blocking_get_if< bool > > data_buf_rsp;
. . .
        // repeat for the other 4

. . .
        // The rest of the decode block header
{

#endif
```

TLM Details: MP3 Example

```
#include "initialModule.h"
#include "mp3dectab.h"

void initial::initial_main_thread ()
{
   bool rsp;
   int nRead = 0;
   int i;
   int FrameCounter = 0;
   . . .
   while ((nRead = fread (mpa_data_type.mpa_data+4, 1, . . .
            {
                    . . .
                    data_buf_req -> get (data_type);
                    data_buf_rsp -> put (rsp);
                    . . .
            }
};
```

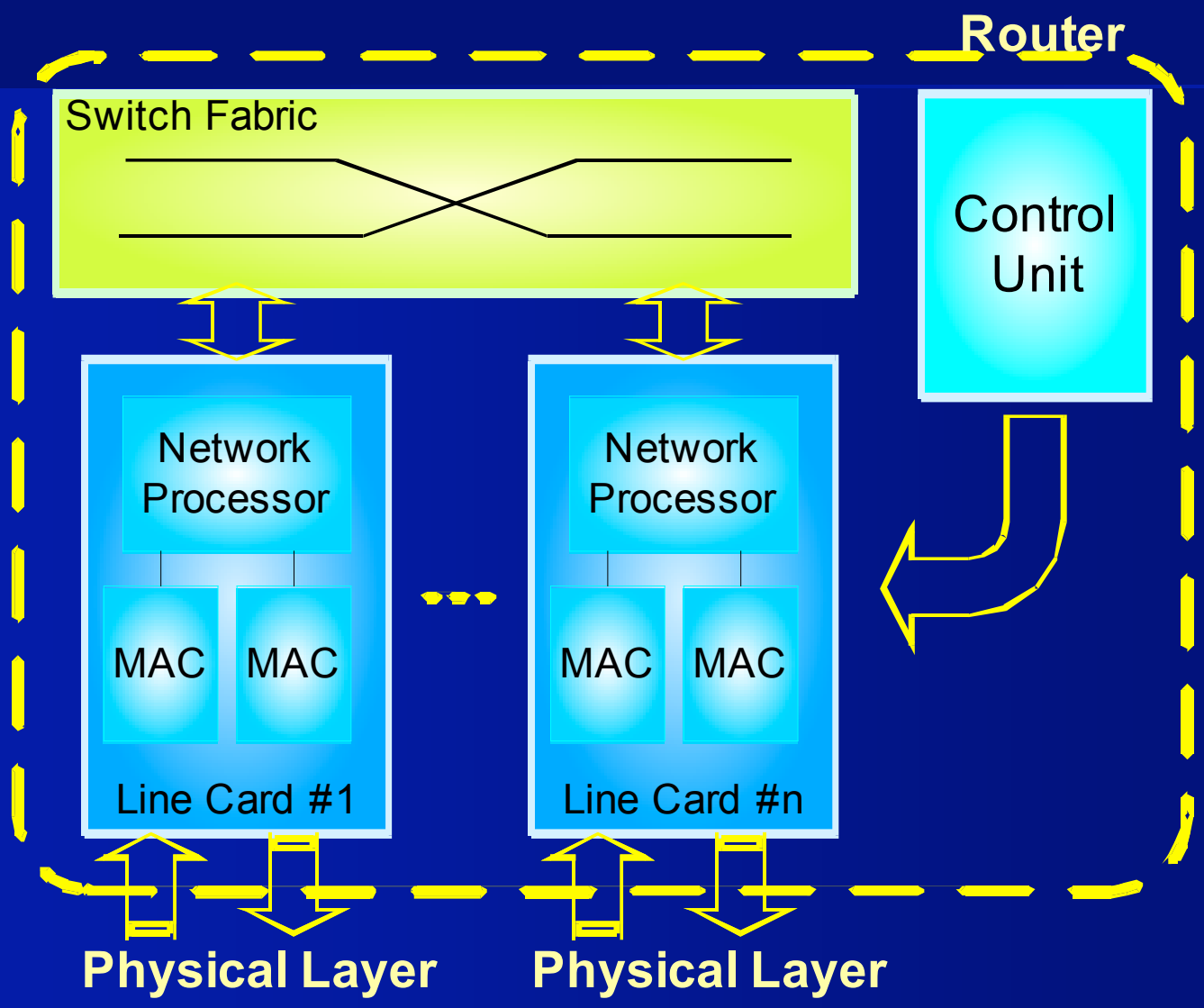
TLM Details: MP3 Example

```
#include "decodeModule.h"
#include <conio.h>

void decode::decode_main_thread()
{
        bool rsp;
        int nDLen;

        while (1) {

                if(nDLen>0)
                {
                        data_buf_req-> put (data_type);

                        data_buf_rsp-> get (rsp);

. . .
                }

        }
}
```

Outline: Evolution, Principles, NP Example

- Transistor level to TLM
  - Design abstraction
  - System specification
  - Interfaces
- TLM Principles
  - What is TLM?
  - TLM advantages
  - Channels and mechanisms
- NP Example
- University Program
  - Teaching
  - Research

Network Processor

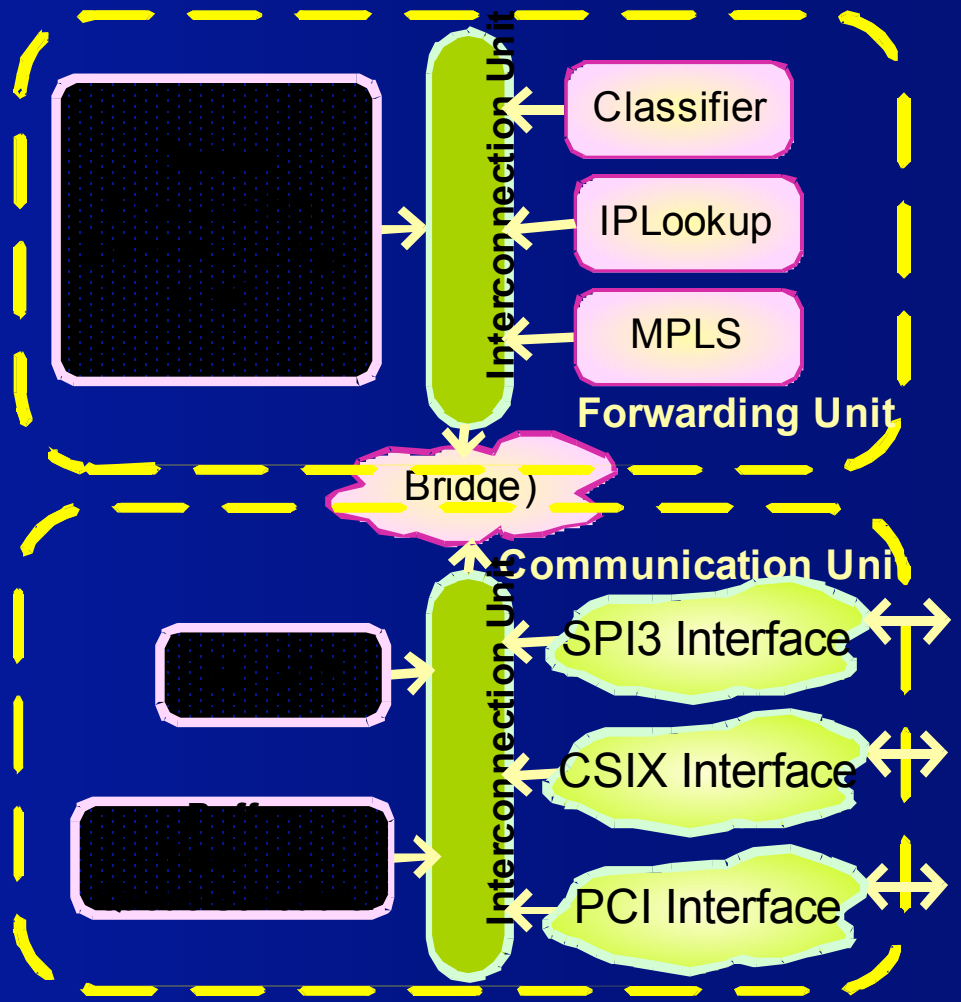


Network Processor - Specification

- Fast path unit tasks:
  - Parsing the header of a packet
  - Classifying a packet (e.g. giving proper priorities to every packet)
  - Looking up a packet destination address (IP address) and retrieving the proper output port number of the destination line card
  - Scheduling the output packets based on their priority

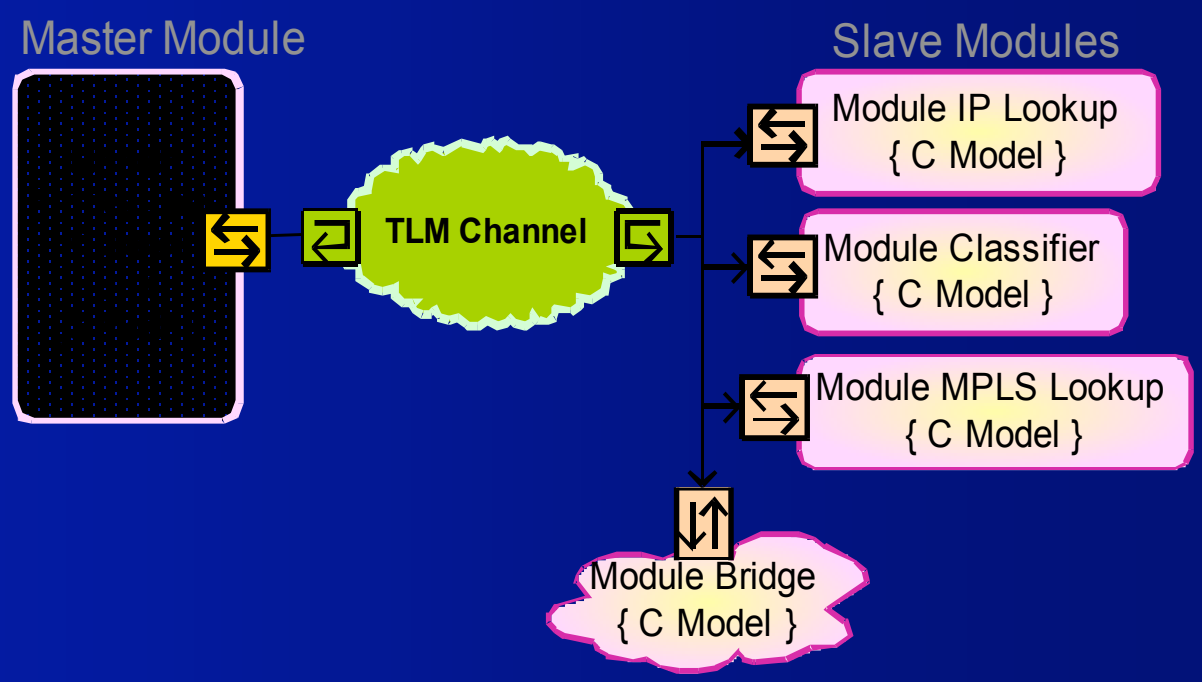
Final Architecture

- Forwarding Unit
  - Packet processing
- Communication Unit
  - Queue management
  - NP interfaces



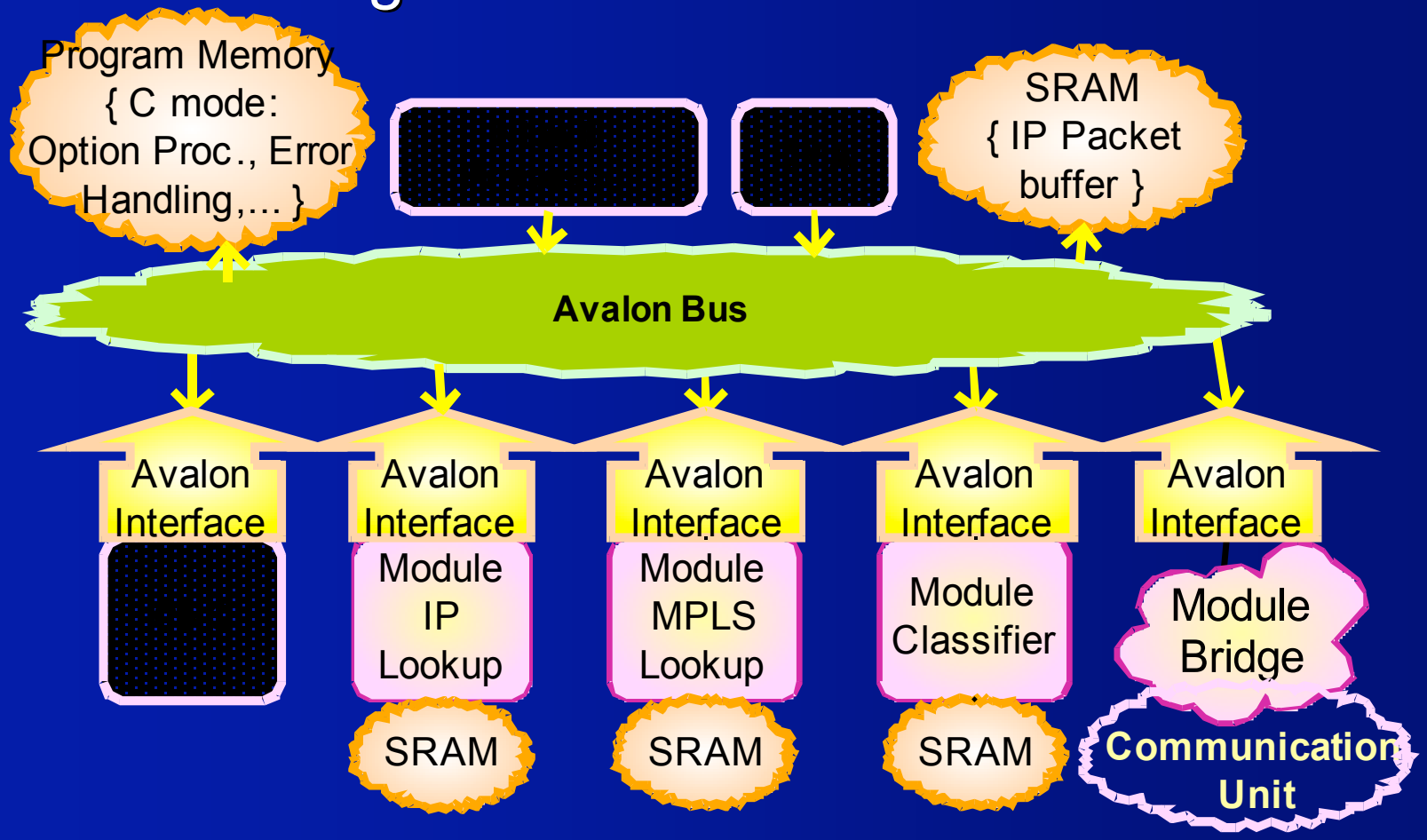
Network Processor - Implementation Methodology

- TLM Channel
  - OSCI channel
  - OSCI Interface
- Functional unit
  - SystemC
  - High level model
    - C Function
  - SC_MODULE
    - SC_PORT
    - SC_THREAD



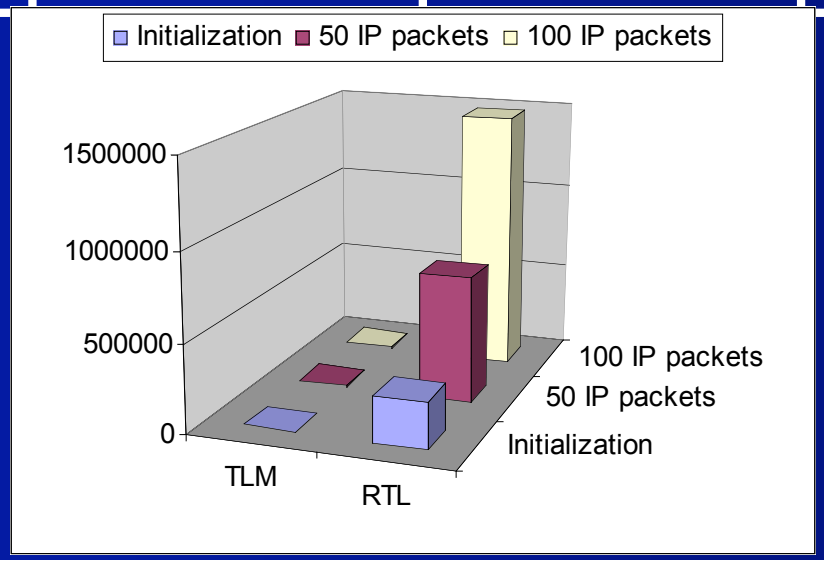
RTL Architecture

- Forwarding Unit



Network Processor Simulation Result (run time)

Testbench part	TLM run-time (ms)	RTL run-time (ms)	TLM/RTL performance
Initialization	715	261,313	365.4
50 IP packets	4469	723,188	161.8
100 IP packets	9187	1,486,422	161.8



Outline: Evolution, Principles, NP Example

- Transistor level to TLM
  - Design abstraction
  - System specification
  - Interfaces
- TLM Principles
  - What is TLM?
  - TLM advantages
  - Channels and mechanisms
- NP Example
- University Program
  - Teaching
  - Research

University Program - Research

- TLM Methodology
- TLM Languages: SystemC, Ada, SystemVerilog, . . .
- TLM Based Architectures
  - Design
  - Test
  - Configuration
- TLM Simulation and Debugging
- TLM Synthesis
- TLM Verification
- TLM Testing
  - DFT
  - BIST
  - ATG, FS
- Testbench

University Program - Teaching

- Teaching modern design methodologies
  - Make sure RTL is covered
  - Cover all aspects of RTL
  - RTL: Test, Simulation, Verification, Synthesis
- Cover C/C++
- SystemC Course
- Cover Hardware Languages
- Teach Bus Structures, e.g., AMBA, Avalon
- Embedded System Design
- TLM Based Design

Conclusions

- SystemC is affecting hardware design methodology
- Hardware design must be taught the new way
- Hardware research must consider this new wave
- Other alternatives must be studied and considered