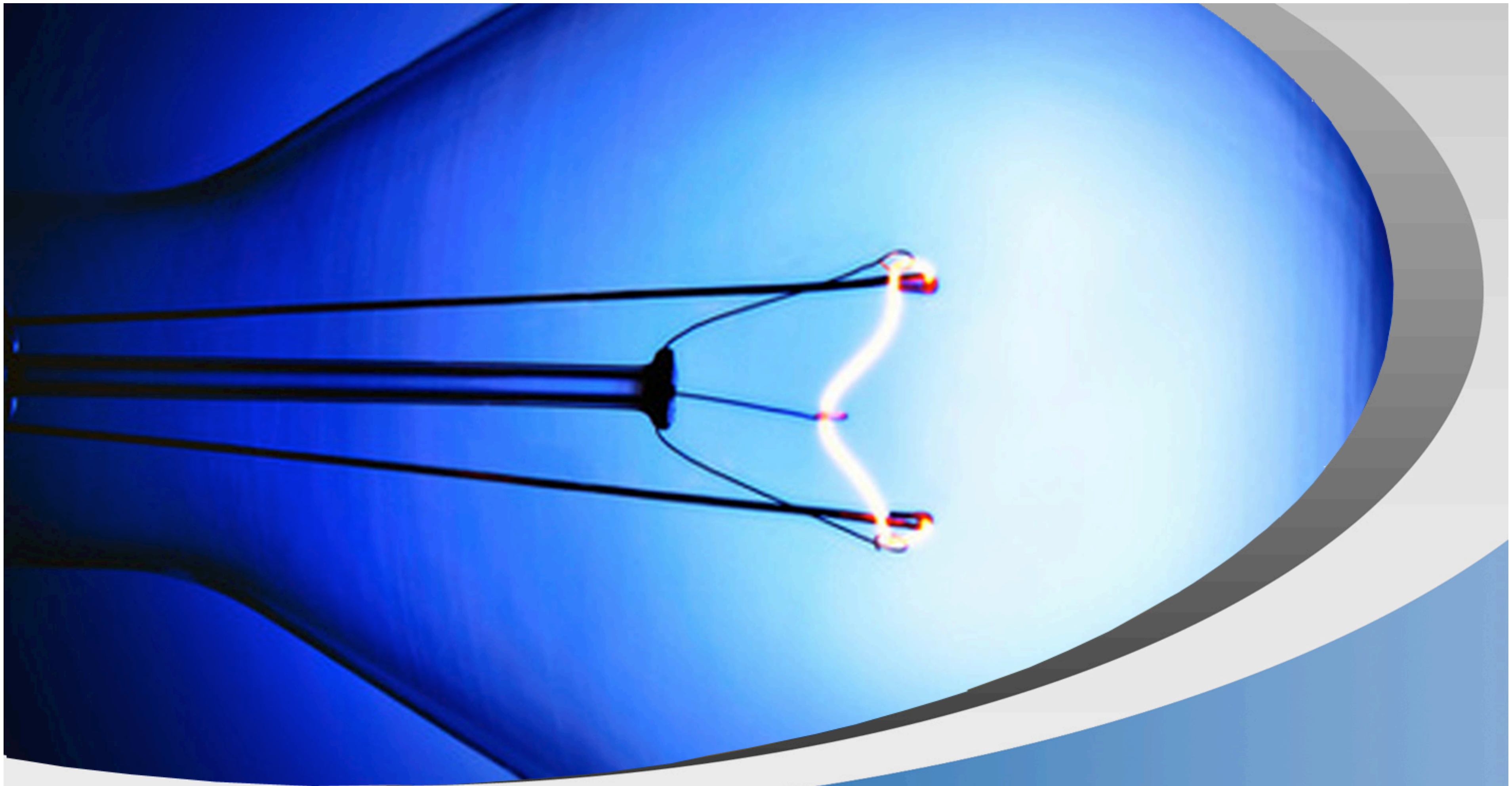


Deploying SystemC for Complex System Prototyping and Validation

DVCon 2013

Design & Verification Conference & Exhibition

SYSTEM C

accellera

SYSTEMS INITIATIVE

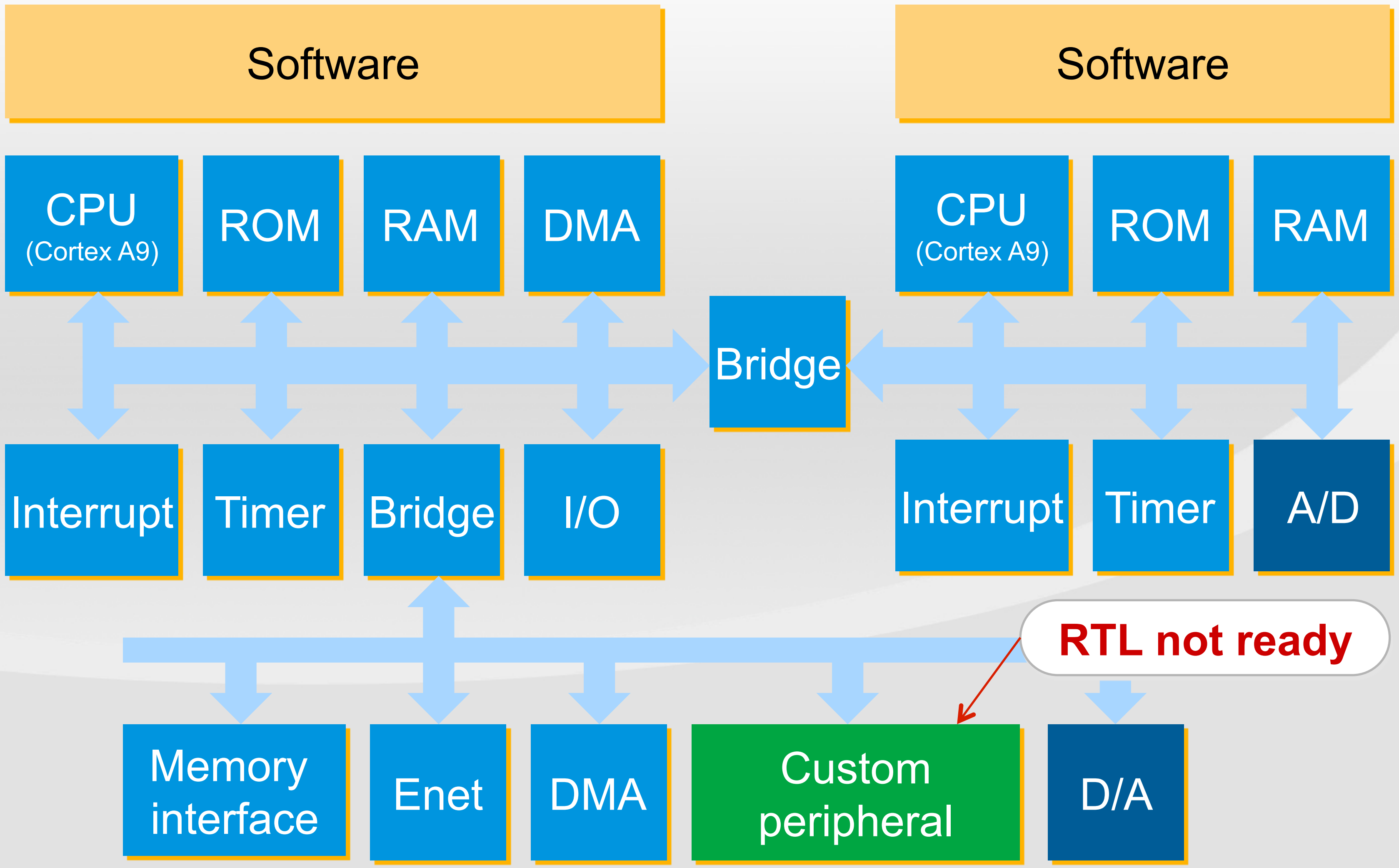
A SystemC Demonstrator

David C Black

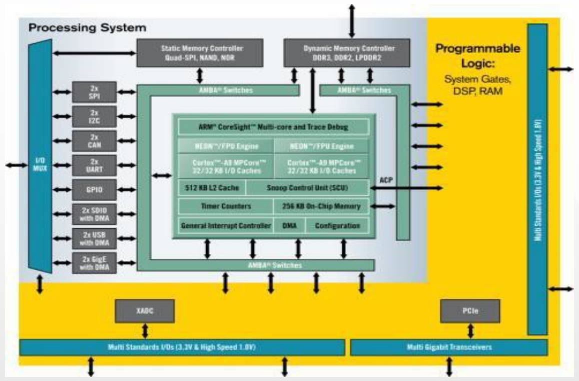
Senior Member of Technical Staff

Doulos Ltd

SoC

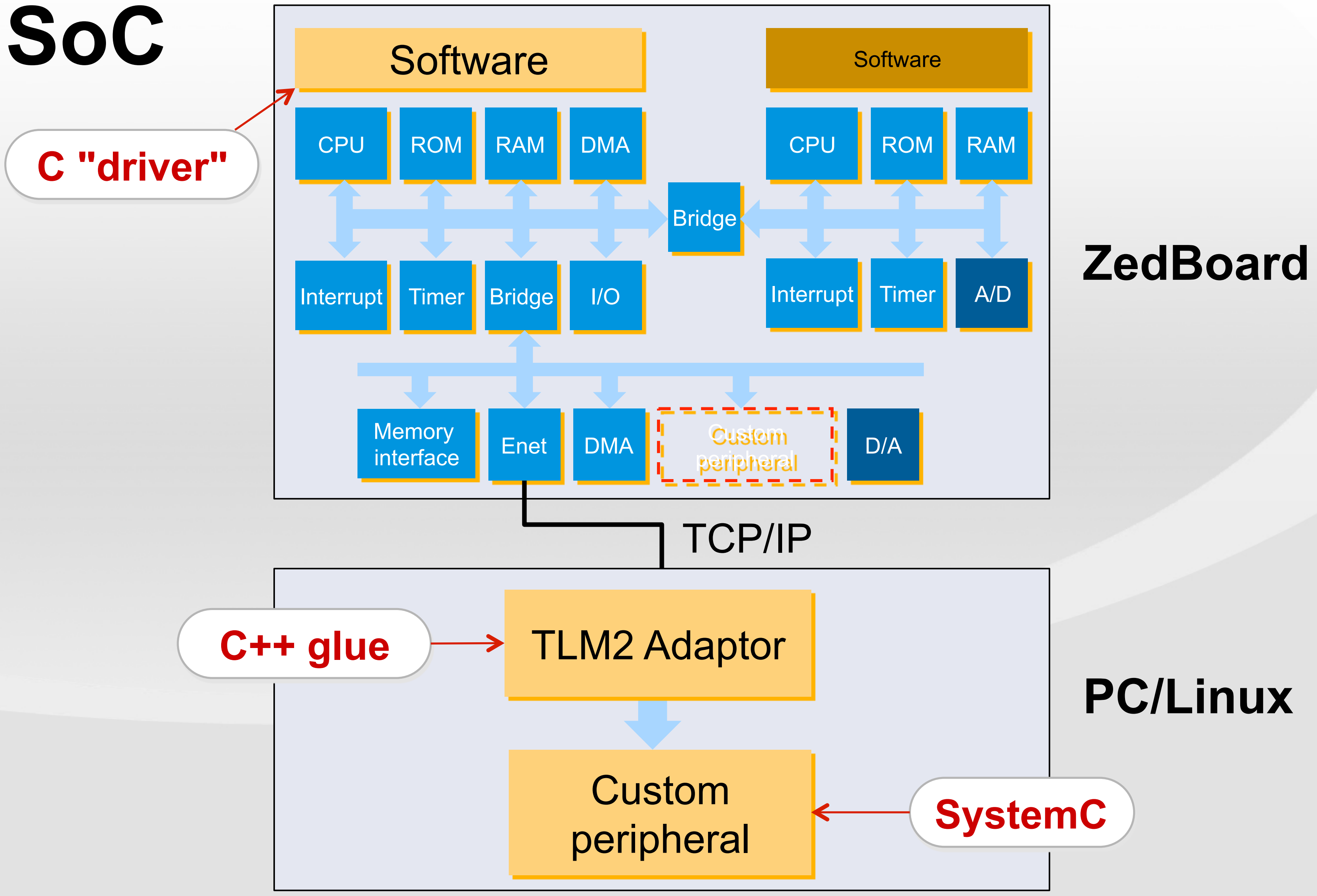

Software
Software
CPU (Cortex A9)
ROM
RAM
DMA
CPU (Cortex A9)
ROM
RAM
Bridge
Interrupt
Timer
Bridge
I/O
Interrupt
Timer
A/D
RTL not ready
Memory interface
Enet
DMA
Custom peripheral
D/A

Use case



- **Situation**
  - Xilinx Zynq-7000™ All Programmable SoC (2×Cortex-A9 + Peripherals + Logic)
  - RTL design in progress
  - Software needs high performance platform to develop drivers and application
  - ESL models available, but wish to avoid ISS performance
- **Solution**
  - Use **Z**ync **E**valuation & **D**evelopment **B**oard <zedboard.org>
  - Configure Processing System
  - Use Ethernet TCP/IP to communicate with SystemC on Linux box
  - Use ESL for only the missing portions of model

SoC


Software
Software
C "driver"
CPU
ROM
RAM
DMA
Bridge
CPU
ROM
RAM
Interrupt
Timer
Bridge
I/O
Interrupt
Timer
A/D
Memory interface
Enet
DMA
Custom peripheral
D/A
ZedBoard
TCP/IP
C++ glue
TLM2 Adaptor
Custom peripheral
SystemC
PC/Linux

Components

- **Software driver(C)**
  - Driver interface implementation
    - Device interface (open, register put, register get, close)
    - Streaming socket client (TCP/IP)
  - TLM 2 style transactions
    - Send (command, address, data)
    - Receive (status, data)
- **TLM2 Adaptor**
  - Create thread to communicate
    - Streaming socket server (TCP/IP)
  - Thread-safe channel
    - Asynchronous request update
    - Queue payload by reference

Software Driver

- **Application interface**
- **Hide implementation**
  - Real or simulated
- **Return 0 on success**
- **Non-zero => error**

C

```
typedef uint64_t       addr_t;
typedef unsigned char  data_t;
typedef uint16_t       dlen_t;
void dev_open(void);
int dev_put
( addr_t  address
, data_t* data_ptr
, dlen_t  datalen
, bool    debug=false
);
int dev_get
( addr_t  address
, data_t* data_ptr
, dlen_t  datalen
, bool    debug=false
);
void dev_close(void);
```

TCP packet structure

- **Maps to TLM 2.0**
  - Generic Payload
- **Little endian**
  - TCP/IP
  - Intel & ARM
- **Extended commands**
  - To control interface

C/C++

```
enum command_t : uint8_t {
  EXIT, READ, WRITE,
  DEBUG_READ, DEBUG_WRITE };
enum status_t : uint8_t {
  UNK, OK, ADR, BUS, GEN };
struct tlmx_packet {
  int32_t     id;
  command_t   command;
  status_t    status;
  uint16_t    datalen;
  uint64_t    address;
  uint8_t     data[*];
};
char* pack_tlmx(tlmx_packet);
tlmx_packet unpack_tlmx(char*);
```

SystemC Adaptor (async server)

- **Create server thread**
  - C++11 #include <thread>
  - Processes TCP/IP
- **Server**
  - Sets up connection
  - Accesses thread-safe channel (async_chan)
  - Closes channel when done

```
SC_MODULE(Adaptor) {
 tlm_initiator_socket<> to_bus;
 thread m_pthread;
 tlmx_channel async_chan;
 SC_CTOR(Adaptor):to_bus("tb")
 , async_chan("async_chan")
 , m_pthread(server,this,
 ...
 void server(async_chan&) {
  // TCPIP socket setup
  for(;;) {
   tlmx=receive_tcpip()
   async_chan->push(tlmx);
   async_chan->wait_for_put();
   async_chan->pull(tlmx)
   format_and_send(tlmx);
  }
```

SystemC/C++11

SystemC Initiator

- **Create server thread**
  - C++11 #include <thread>
  - Processes TCP/IP
- **Server**
  - Sets up connection
  - Accesses thread-safe channel (async_chan)
  - Closes channel when done

```
SC_MODULE(IP_Top) {
 ...
 SC_THREAD(initiator_thread);
};
void initiator_thread(void) {
  // TCPIP socket setup
  for(;;) {
    wait(async_chan->
                 put_evt());
    async_chan->get(tlmx);
    trans=to_tlm2(tlmx)
    to_bus->
        b_transport(trans,…);
    tlmx=to_tlmx(trans);
    async_chan->put(tlmx);
  }
}
```

Thread safe channel – part 1 of 3

- **Primitive channel**
- **Protect data**
  - Lock guard + Mutex
  - C++11 easy
- **Thread safe call**
  - New for 1666-2011

```
struct tlmx_channel
: sc_prim_channel, •••
{
private:
  list<tlmx_packet*> recv_queue;
  mutable mutex      recv_mutex;
  sc_event           put_event;
  •••
public:
  void push(tlmx_packet* pl){
    lock_guard<mutex>
                protect(recv_mutex);
    recv_queue.push_front(pl);
    did_push = true;
    async_request_update();
  }
  •••
```

Thread safe channel – part 2 of 3

- **Protect data**
  - Read/modify/write
- **Notify SystemC**
  - sc_event
- **Both push & pull**

```
…
void update(void) {
  { // Handle push
    lock_guard<mutex>
           protect(recv_mutex);
    if (did_push == true) {
      put_event.notify(0);
      did_push = false;
    }
  }
  { // Handle pull
  …
}
…
```

Thread safe channel – part 3 of 3

- **Protect data**
  - Remove from queue
- **Notify async**
  - OS thread-level

```
  ...
  void get(tlmx_packet* pl) {
    lock_guard<mutex>
                 protect(recv_mutex);
    pl = recv_queue.back();
    recv_queue.pop_back();
    // Let async server know
    wait_get_mutex.unlock();
    wait_get_mutex.lock();
  }
}
  ...
};
```

Miscellaneous notes

- **Challenges**
  - C-style pthreads harder than C++11
  - Debug
  - Synchronize startup
- **Things to learn**
  - TCP/IP streaming sockets
  - Pthreads (C-side)
  - C++11 threads & mutexes
  - async_request_update/update mechanism
  - ZedBoard tool chain (Xilinx SDK + Vivado HLS + Xilinx ISE)
- **Code available on Doulos website (soon)**

Thank You