High-level component based models for functional and non-functional properties of SoCs

Yuliia Romenska

Supervisor: Florence Maraninchi

VERIMAG/Synchrone

SYNCHRON'14

Introduction

- Systems-on-chip (SoC) are **ubiquitous** and **complex**
- There is a need for a formalism for high-level models of individual components, subsystem or full systems
- It must be:
  - hierarchical
  - executable
  - suitable for both bottom-up and top-down design
- The formalism has to support:
  - functional, extra-functional requirements (power, temperature)
  - translation into existing open standards (ex. TLM/SystemC)

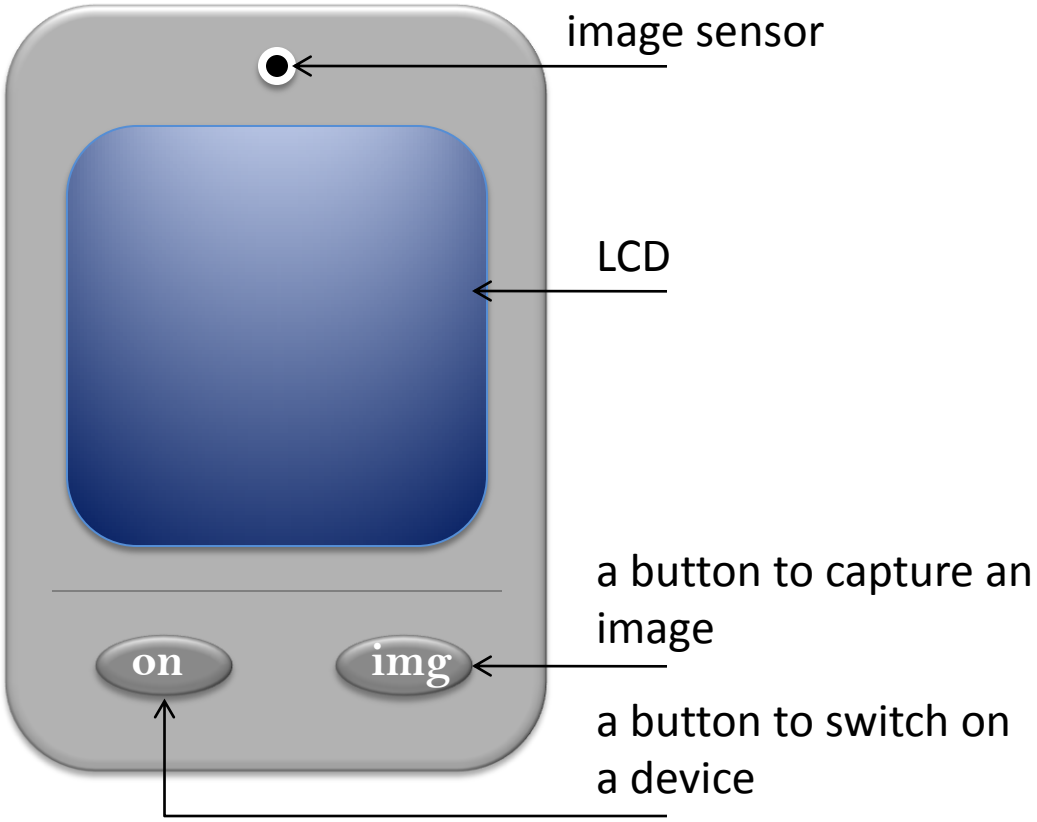
Outline

- Introduction
- The case-study
- Background
- Non-deterministic specification of a component
- Ongoing work
- Conclusions

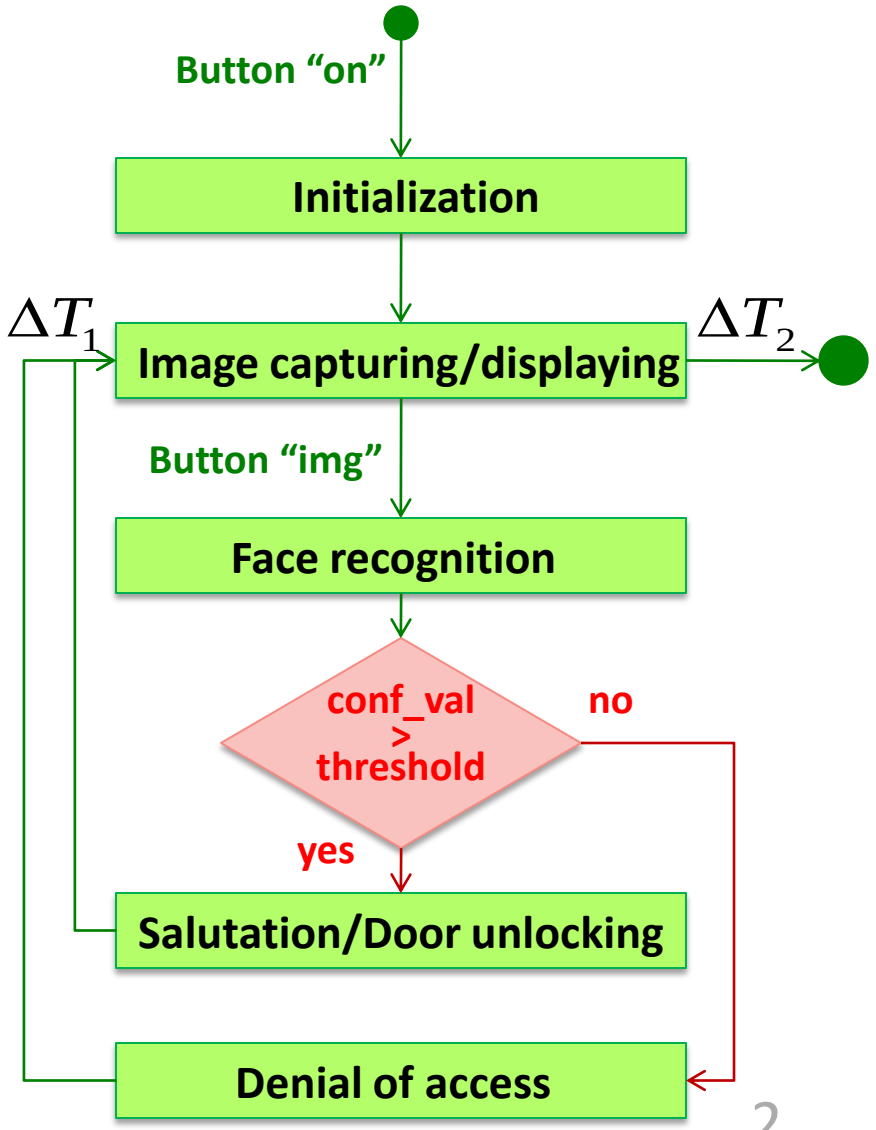
Outline

- Introduction
- The case-study
- Background
- Non-deterministic specification of a component
- Ongoing work
- Conclusions

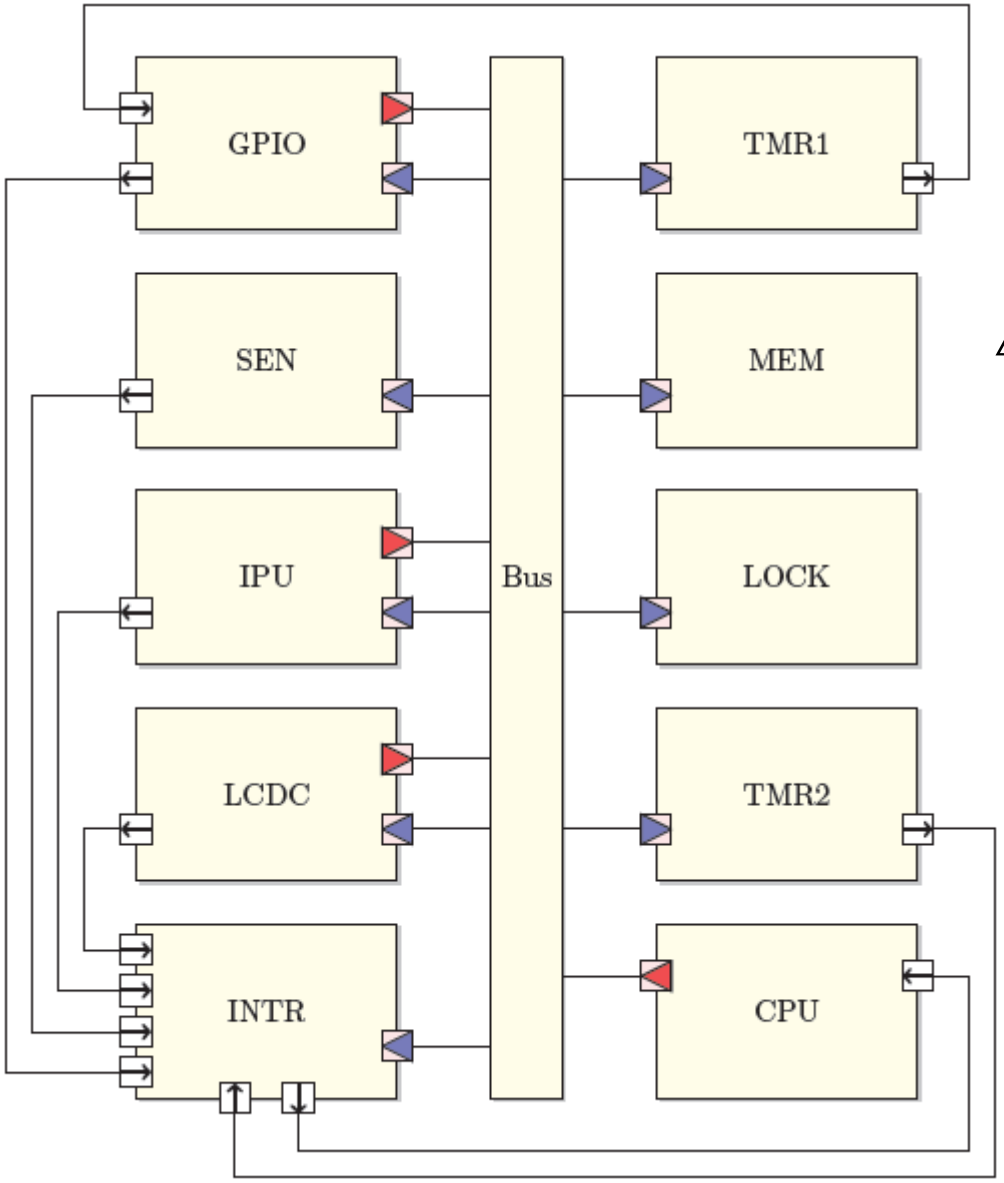
The Case-Study «A Smart Intercom»

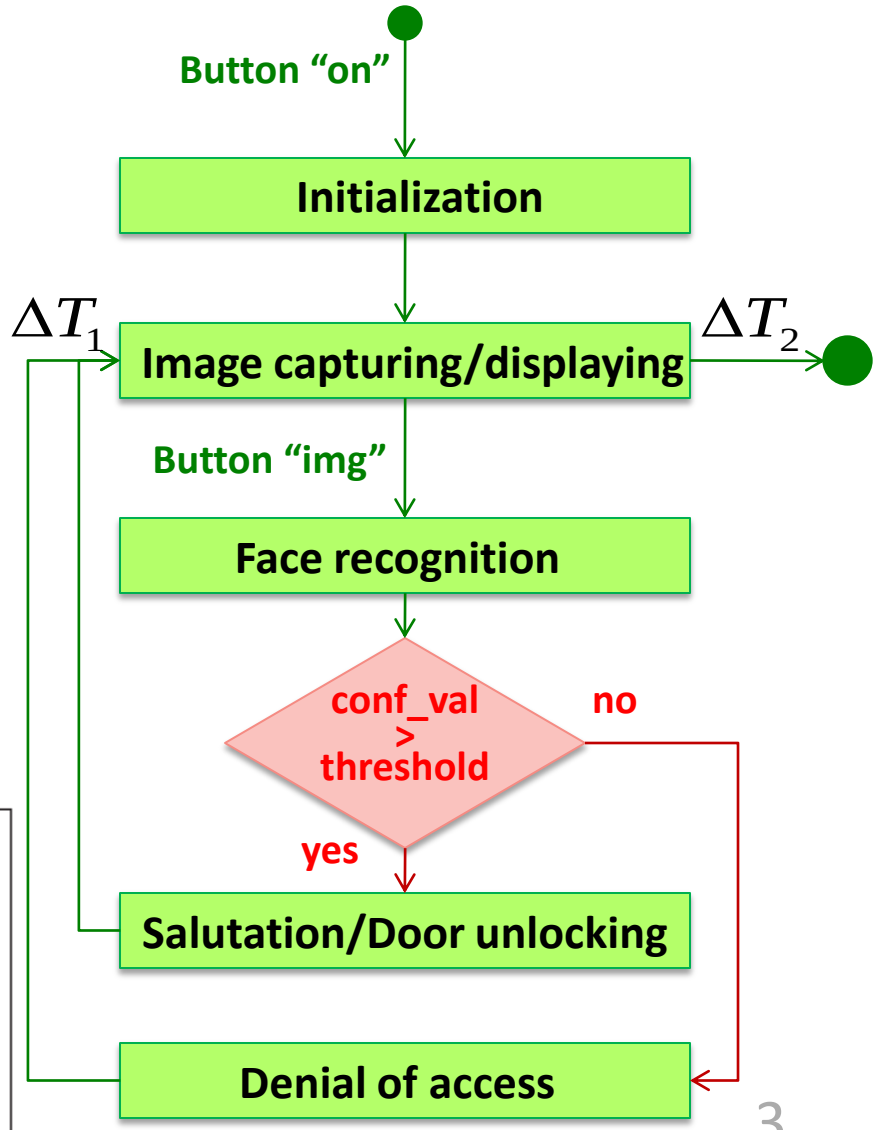


The Case-Study «A Smart Intercom»



The Case-Study «A Smart Intercom»





The Case-Study «A Smart Intercom»

```
                . . .
SC_MODULE(IPU)
{
        sc_core::sc_out<bool> irq_out;
        basic::initiator_socket<IPU> initiator_socket;
        basic::target_socket<IPU>     target_socket;

        IPU(sc_core::sc_module_name name);
        IPU(sc_core::sc_module_name name, int trac_mode);

        SC_HAS_PROCESS(IPU);

        void setMonitor(Monitor* m);

        tlm::tlm_response_status read(basic::addr_t addr, basic::data_t &d);
        tlm::tlm_response_status write(basic::addr_t addr, basic::data_t d);

private:
        sc_core::sc_event start_event;

                . . .
```

The Case-Study «A Smart Intercom»

. . .

```
void IPU::face_recognition_analysis()
{
        while(true){
                confidence_value = 0;
                recogn_img_addr = 0;

                /* initialize buffers */
                captured_image_buffer = new basic::data_t [image_size/sizeof(basic::data_t)];
                gallery_image_buffer  = new basic::data_t [image_size/sizeof(basic::data_t)];

                /* upload a captured image */
                upload_image(captured_image_addr, captured_image_buffer);

                basic::data_t temp_conf_value = 0;
                basic::addr_t gallery_image_addr;
                int i = 0;
                while ( i < gallery_size )
                {
                        gallery_image_addr = gallery_addr + i * image_size;
                        upload_image(gallery_image_addr, gallery_image_buffer);
                        temp_conf_value = compare_facial_features();
                        if(temp_conf_value > confidence_value)
                        {
                                confidence_value = temp_conf_value;
                                recogn_img_addr = gallery_image_addr;
                        }
                        i += 1;
                }
```

. . .

The Case-Study «A Smart Intercom»

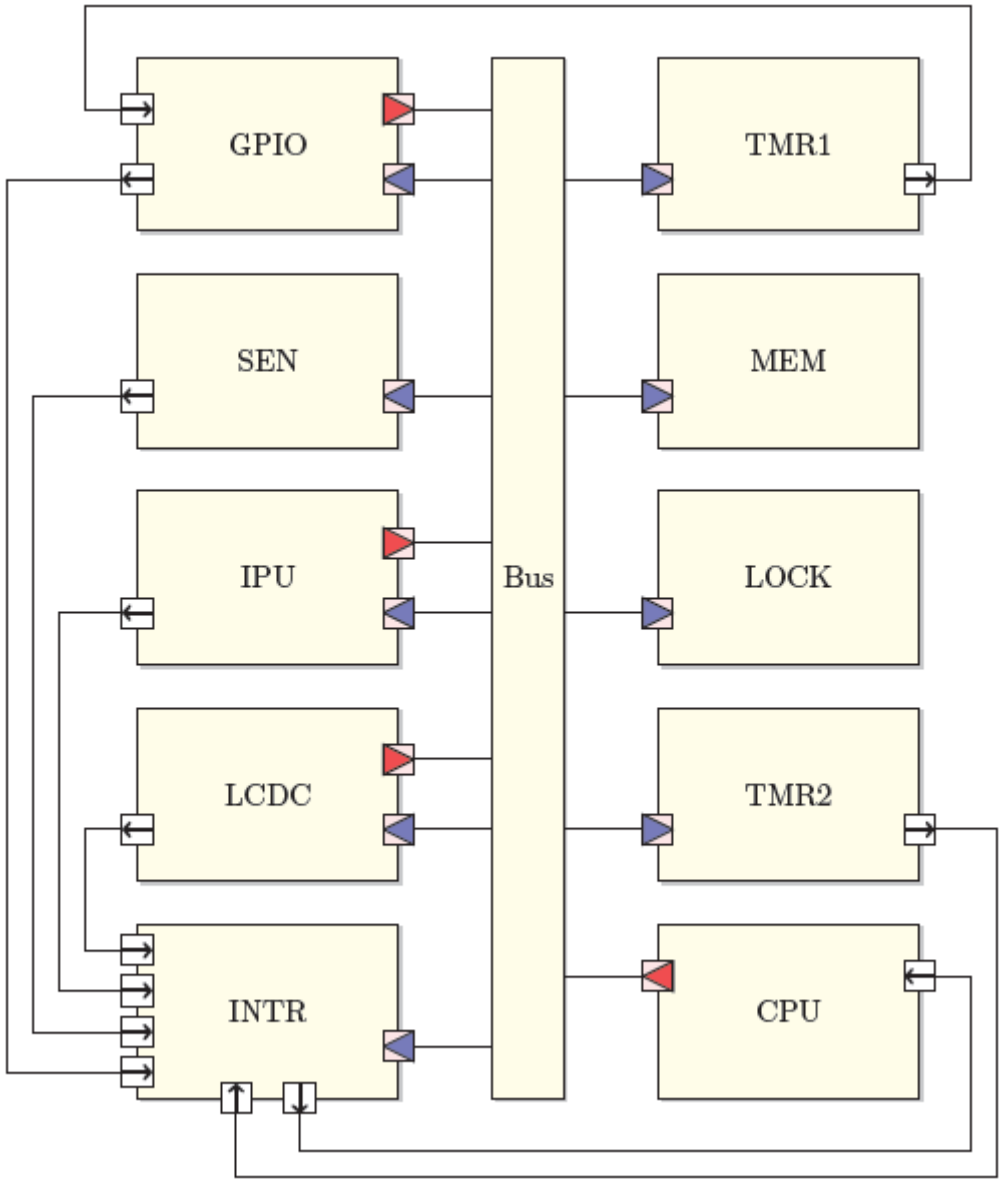
```
. . .
void IPU::face_recognition_analysis()
{
        while(true){
                confidence_value = 0;
                recogn_img_addr = 0;

                /* initialize buffers */
                captured_image_buffer = new basic::data_t [image_size/sizeof(basic::data_t)];
                gallery_image_buffer  = new basic::data_t [image_size/sizeof(basic::data_t)];

                /* upload a captured image */
                upload_image(captured_image_addr, captured_image_buffer);

                basic::data_t temp_conf_value = 0;
                basic::addr_t gallery_image_addr;
                int i = 0;
                while ( i < gallery_size )
                {
                        gallery_image_addr = gallery_addr + i * image_size;
                        upload_image(gallery_image_addr, gallery_image_buffer);
                        temp_conf_value = compare_facial_features();
                        recogn_img_addr = gallery_image_addr;
                }
                i += 1;
        }
. . .
```

Implementation is not trivial


GPIO
TMR1
SEN
MEM
IPU
Bus
LOCK
LCDC
TMR2
INTR
CPU

Outline

- Introduction
- The case-study
- Background
- Non-deterministic specification of a component
- Ongoing work
- Conclusions

Outline

- Introduction
- The case-study
- Background
  - Non-deterministic Specifications
  - Contract-Based Design
- Non-deterministic specification of a component
- Ongoing work
- Conclusions

Non-deterministic Specifications

- Can be used
  - to model unpredictable, unknown environment
  - to represent implementation freedom

Non-deterministic Specifications

- Can be used
  - to model unpredictable, unknown environment
  - to represent implementation freedom

Ex. A beverage machine
The user may select any proposed good and a system must be able to response

Non-deterministic Specifications

- Can be used
  - to model unpredictable, unknown environment

Ex. A beverage machine
The user may select any proposed good and a system must be able to response.

  - to represent implementation freedom

Ex. An invitation to PhD defense

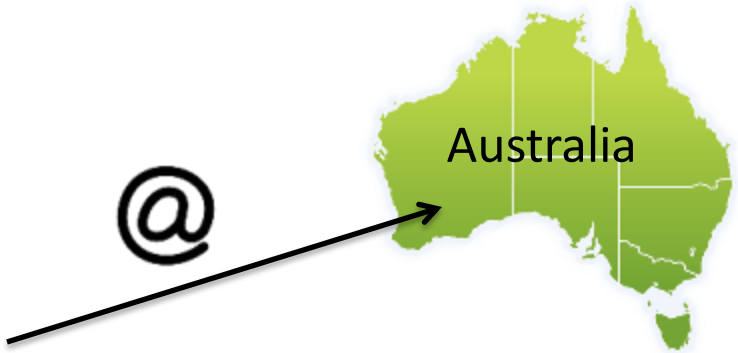
Non-deterministic Specifications

- Can be used
  - to model unpredictable, unknown environment

Ex. A beverage machine
The user may select any proposed good and a system must be able to response.

  - to represent implementation freedom

Ex. An invitation to PhD defense



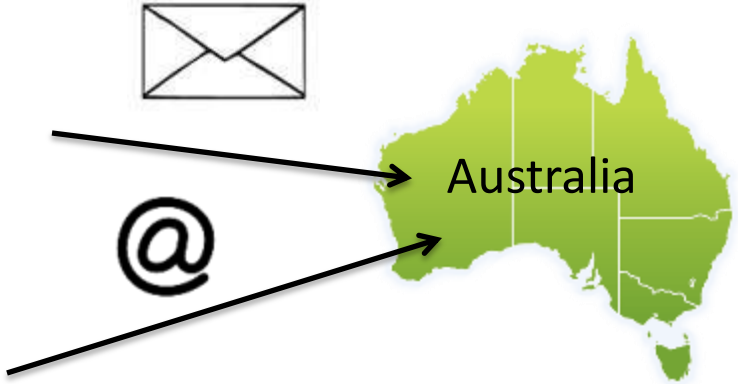
Non-deterministic Specifications

- Can be used
  - to model unpredictable, unknown environment

Ex. A beverage machine
The user may select any proposed good and a system must be able to response.

  - to represent implementation freedom

Ex. An invitation to PhD defense



Non-deterministic Specifications

- Can be used
  - to model unpredictable, unknown environment

Ex. A beverage machine
The user may select any proposed good and a system must be able to response.

  - to represent implementation freedom

Ex. An invitation to PhD defense

Non-deterministic Specifications

- Examples:
  - Modal specifications (Kim Larsen)
  - Contract computations (Benoit Caillaud and al.)
  - The Lutin language (Lurette tool)

Contract-based design

- A **contract** characterizes in a formal way
  - under which context the design is assumed to operate
  - what are obligations of the design
- A contract is a pair

$$C = (A, G) \quad \text{where}$$

A is the set of assumptions

G is the set of guarantees

Application of Contracts

- The Component-Based Software Engineering Community
  - syntactic contracts (Interface Description Languages)
  - behavioral contracts (Eiffel, iContracts, etc.)
  - synchronization contracts (Interface automata, JASS, etc.)
- The Hardware Community
  - don’t care conditions
  - executable specifications for Lustre (L. Morel)
  - modular verification (K. McMillan)

Outline

- ❑ Introduction
- ❑ The case-study
- ❑ Background
- ❑ Non-deterministic specification of a component
- ❑ Ongoing work
- ❑ Conclusions

An Example: IPU (Image Pocessing Unit)

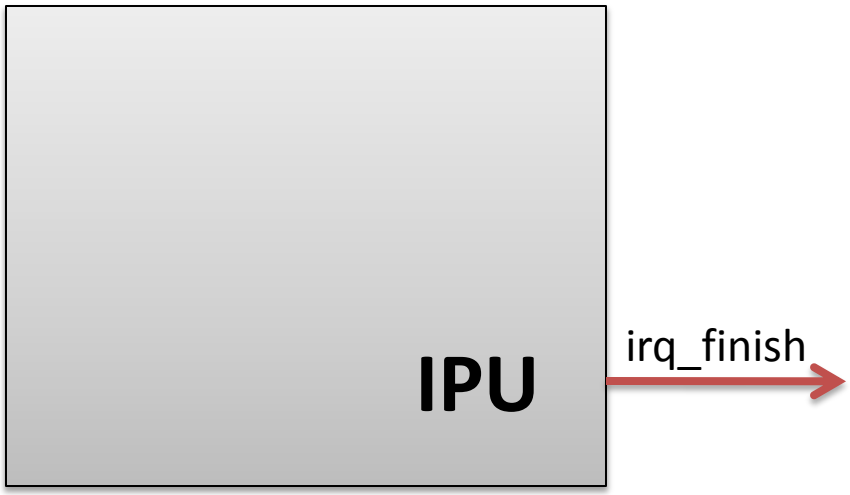
Performs face recognition analysis

Input Parameters:

- an image address
- an image size
- an address of an image gallery
- a size of a gallery

Supports:

- initialization phase
  - computes signatures of images
- face recognition phase
  - computes the address of an image from the gallery that corresponds to the best matching
  - computes a confidence value



An Example: IPU (Image Pocessing Unit)

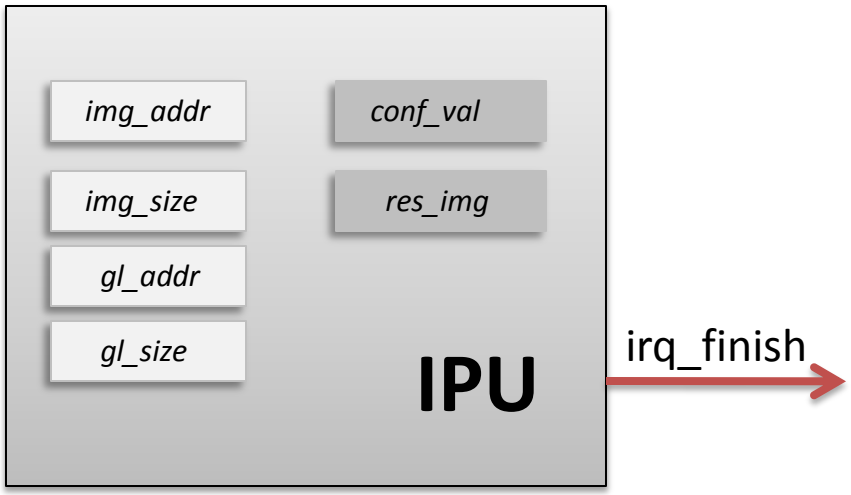
Performs face recognition analysis

Input Parameters:

- an image address
- an image size
- an address of an image gallery
- a size of a gallery

Supports:

- initialization phase
  - computes signatures of images
- face recognition phase
  - computes the address of an image from the gallery that corresponds to the best matching
  - computes a confidence value



An Example: IPU (Image Pocessing Unit)

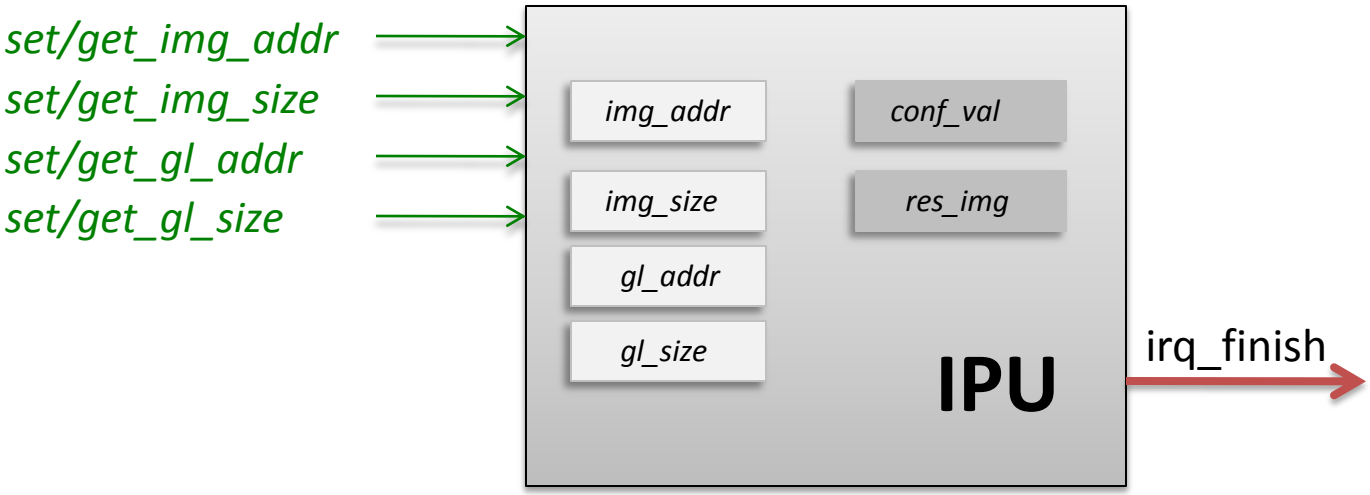
Performs face recognition analysis

Input Parameters:

- an image address
- an image size
- an address of an image gallery
- a size of a gallery

Supports:

- initialization phase
  - computes signatures of images
- face recognition phase
  - computes the address of an image from the gallery that corresponds to the best matching
  - computes a confidence value



An Example: IPU (Image Pocessing Unit)

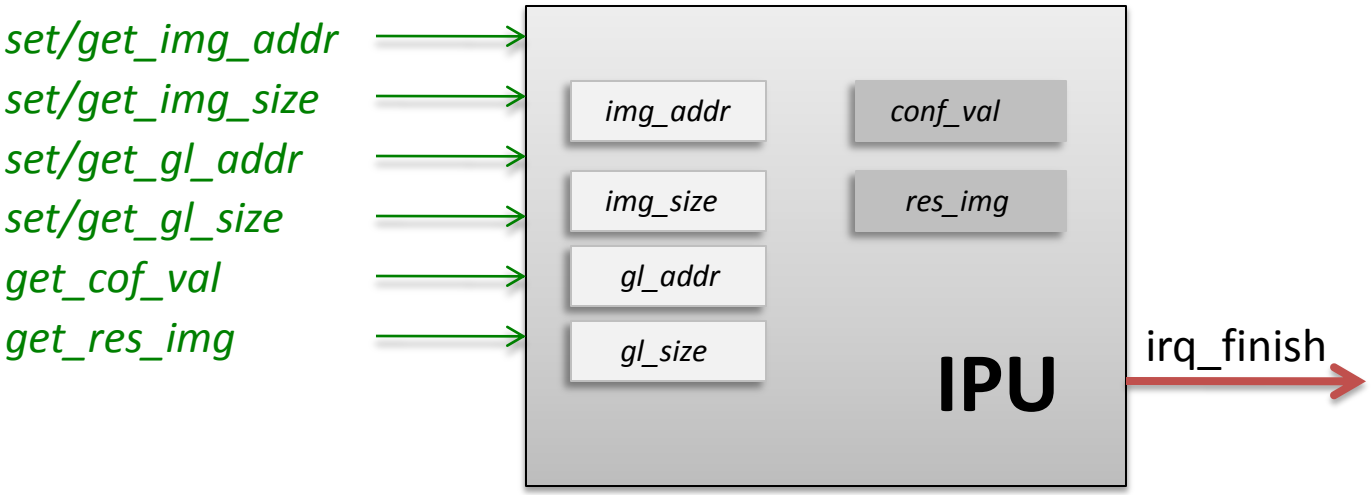
Performs face recognition analysis

Input Parameters:

- an image address
- an image size
- an address of an image gallery
- a size of a gallery

Supports:

- initialization phase
  - computes signatures of images
- face recognition phase
  - computes the address of an image from the gallery that corresponds to the best matching
  - computes a confidence value



An Example: IPU (Image Pocessing Unit)

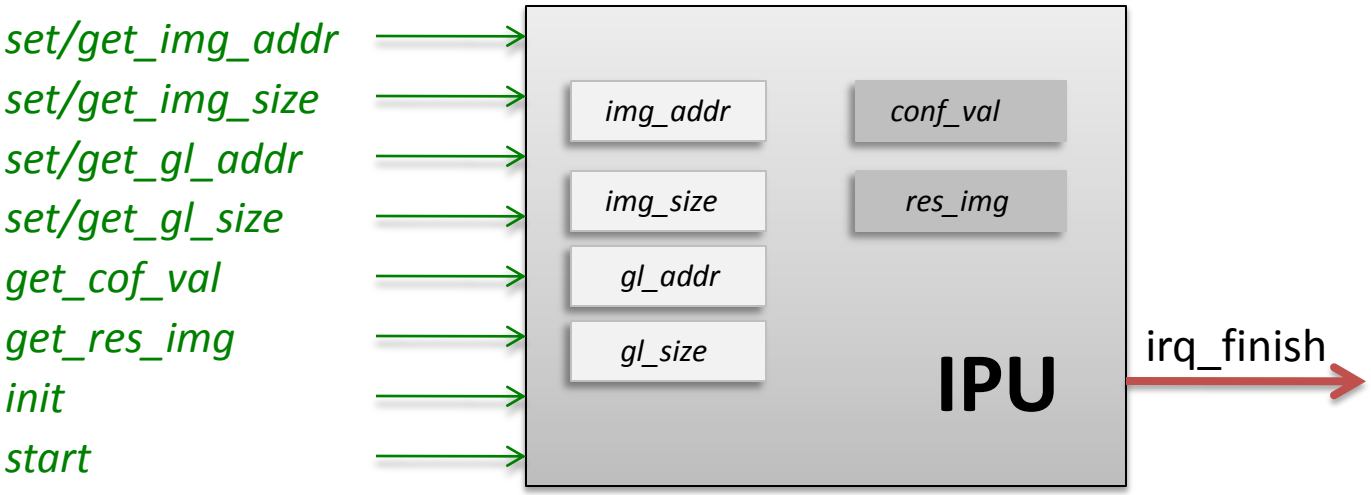
Performs face recognition analysis

Input Parameters:

- an image address
- an image size
- an address of an image gallery
- a size of a gallery

Supports:

- initialization phase
  - computes signatures of images
- face recognition phase
  - computes the address of an image from the gallery that corresponds to the best matching
  - computes a confidence value



An Example: IPU (Image Pocessing Unit)

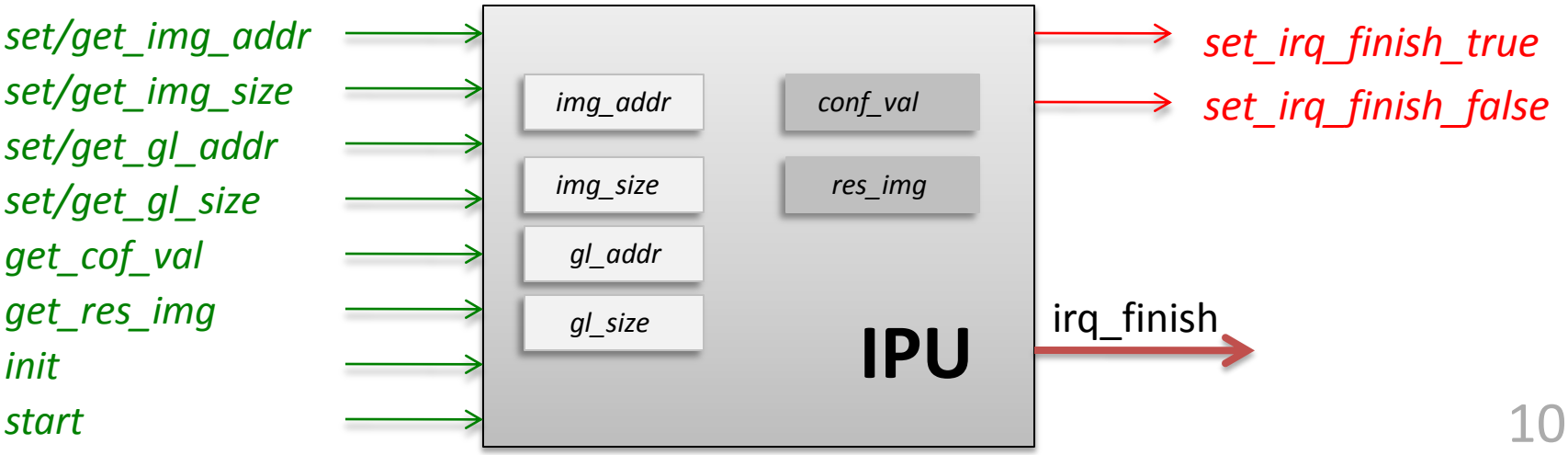
Performs face recognition analysis

Input Parameters:

- an image address
- an image size
- an address of an image gallery
- a size of a gallery

Supports:

- initialization phase
  - computes signatures of images
- face recognition phase
  - computes the address of an image from the gallery that corresponds to the best matching
  - computes a confidence value



An Example: IPU (Image Pocessing Unit)

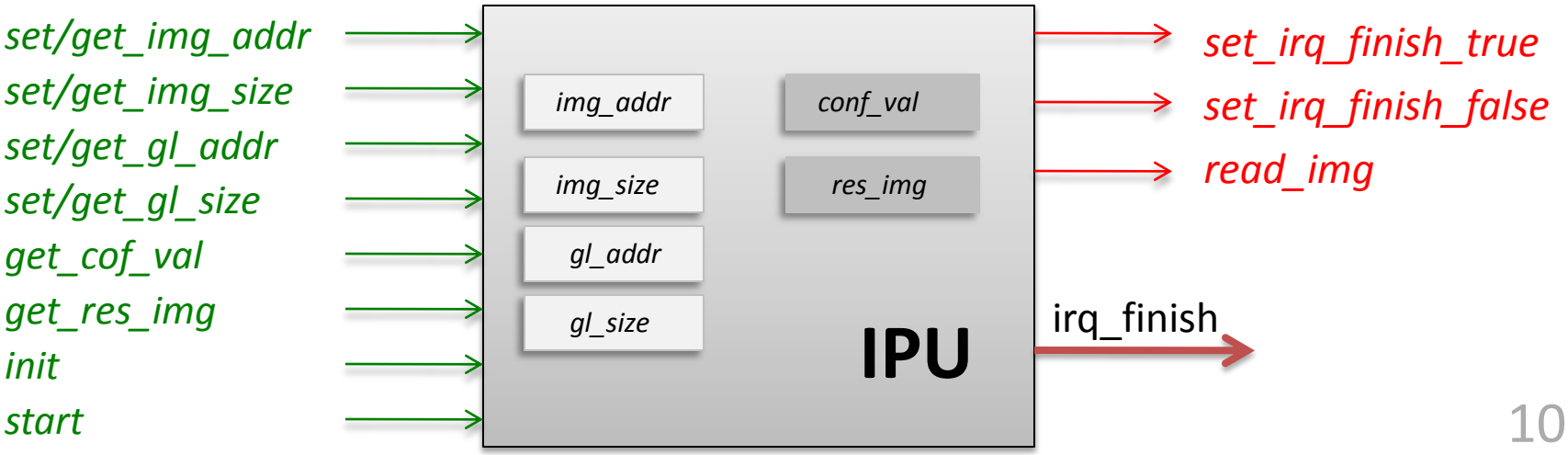
Performs face recognition analysis

Input Parameters:

- an image address
- an image size
- an address of an image gallery
- a size of a gallery

Supports:

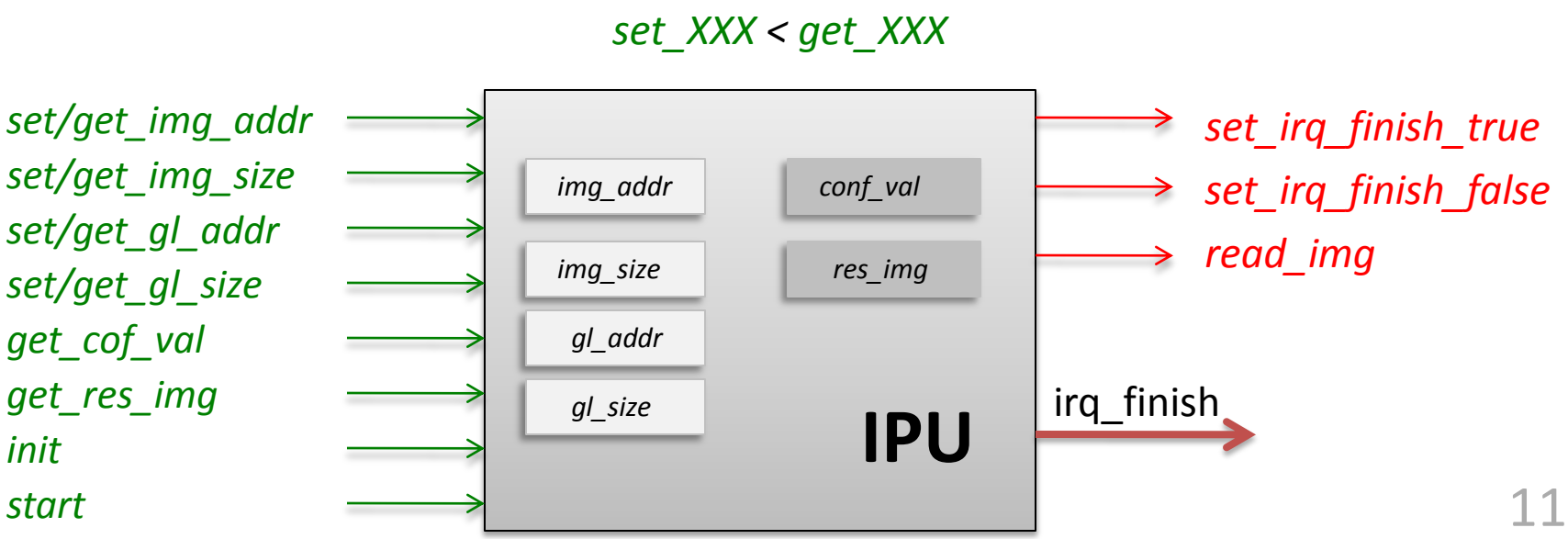
- initialization phase
  - computes signatures of images
- face recognition phase
  - computes the address of an image from the gallery that corresponds to the best matching
  - computes a confidence value



An Example: IPU (Image Pocessing Unit)

Specification:

1) Values of “read/write “ registers can be read only after being set



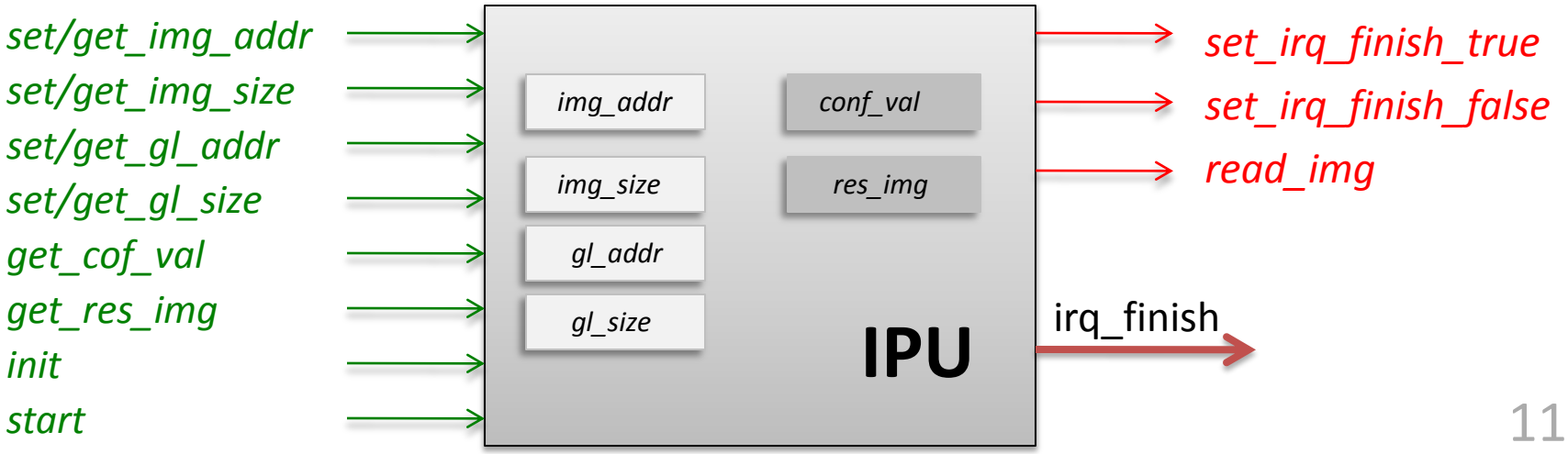
An Example: IPU (Image Pocessing Unit)

Specification:

2) The initialization can be started only if values of *img_size*, *gl_addr*, *gl_size* registers are defined. As soon as the component is started it reads external memory several times and sends an interrupt

(set_img_size, set_gl_addr, set_gl_size) < init => (read_img)^{N} < set_irq_finish_true < set_irq_finish_false

set_XXX < *get_XXX*



An Example: IPU (Image Pocessing Unit)

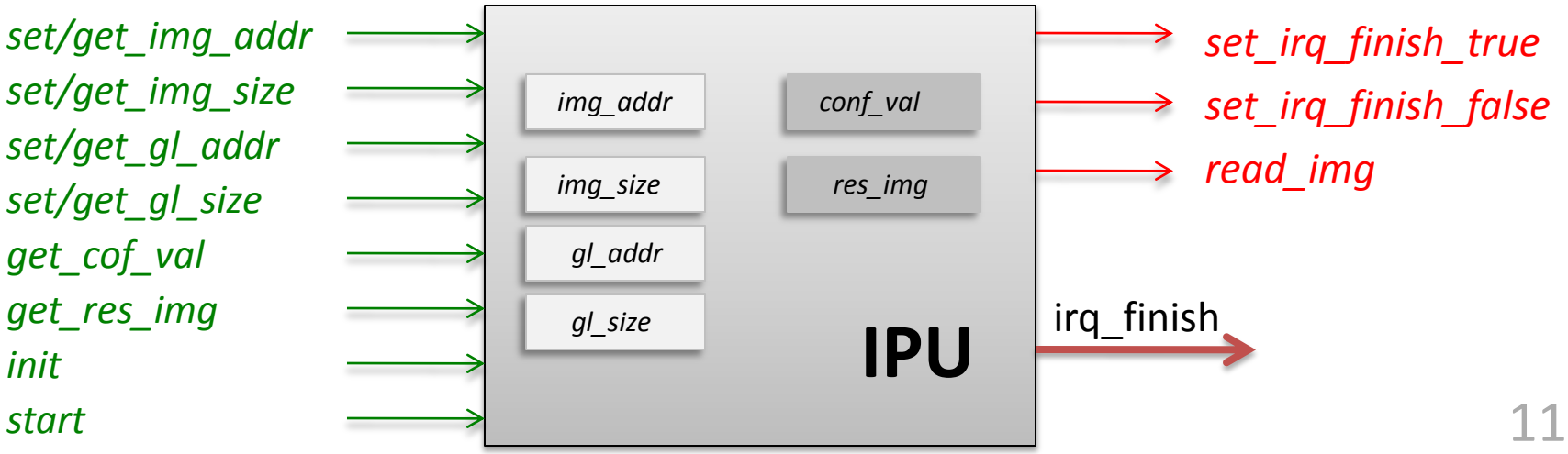
Specification:

3) Face recognition can be started only if the *img_addr* register is set and the component has been initialized. As soon as the component is started it reads external memory several times and sends an interrupt

(set_img_addr, set_irq_finish_true) < start => (read_img)^N < set_irq_finish_true < set_irq_finish_false

(set_img_size, set_gl_addr, set_gl_size) < init => (read_img)^N < set_irq_finish_true < set_irq_finish_false

set_XXX < *get_XXX*



An Example: IPU (Image Pocessing Unit)

Specification:

4) The "read only" registers can be read only after being computed at least once

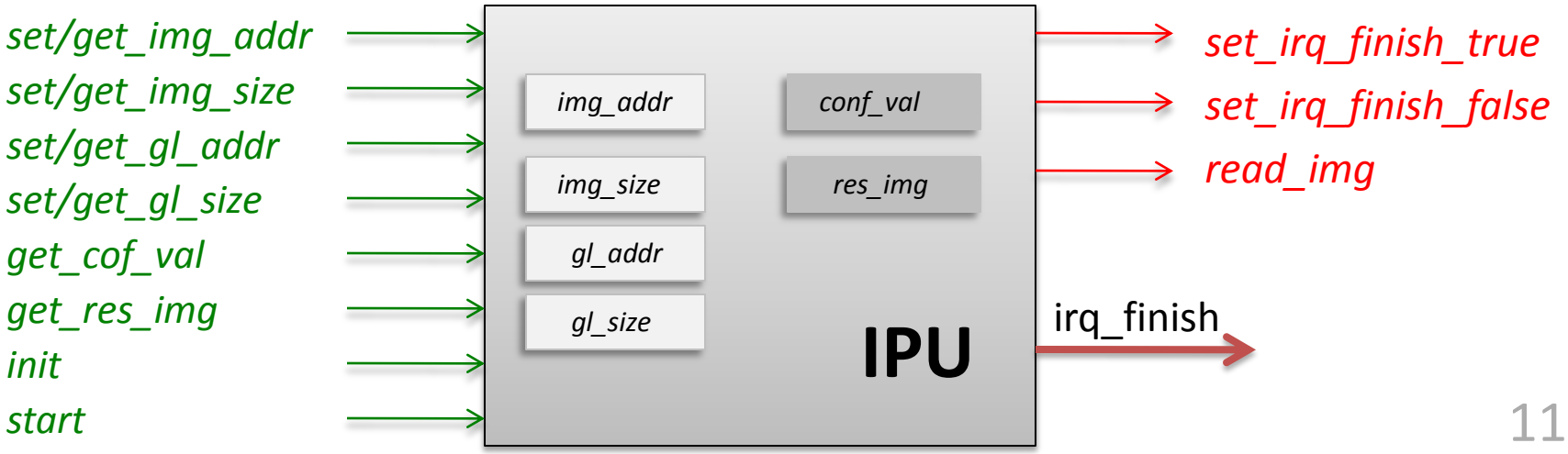
(start < set_irq_true) < get_res_img

(start < set_irq_true) < get_conf_val

(set_img_addr, set_irq_finish_true) < start => (read_img)^{N} < set_irq_finish_true < set_irq_finish_false

(set_img_size, set_gl_addr, set_gl_size) < init => (read_img)^{N} < set_irq_finish_true < set_irq_finish_false

set_XXX < *get_XXX*



An Example: IPU (Image Pocessing Unit)

Specification:

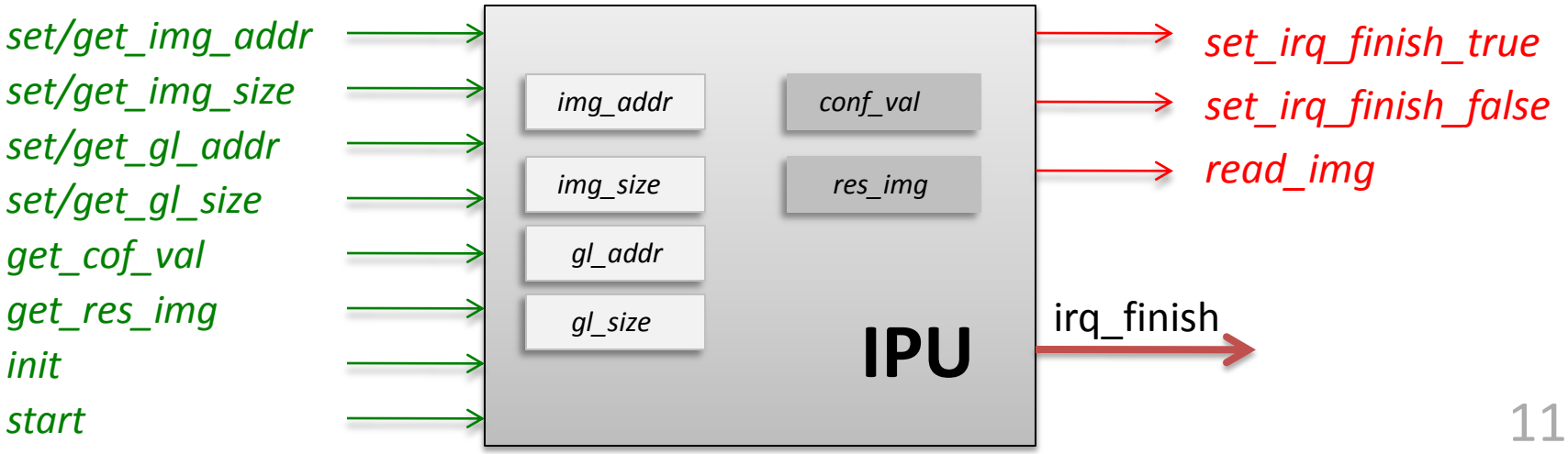
(start < set_irq_true) < get_res_img

(start < set_irq_true) < get_conf_val

(set_img_addr, set_irq_finish_true) < start => (read_img)^{N} < set_irq_finish_true < set_irq_finish_false

(set_img_size, set_gl_addr, set_gl_size) < init => (read_img)^{N} < set_irq_finish_true < set_irq_finish_false

set_XXX < *get_XXX*



An Example: Defensive Programming for Contracts

set_img_addr < *get_img_addr*

```
void step(int trans_name)
{
        /*transition system encoding*/
        switch(trans_name)
        {
        case IPU_SET_IMG_ADDR:
                set_img_addr_flag = true;
                printTrans(trans_name);
                break;
         case IPU_GET_IMG_ADDR:
                if (set_img_addr_flag)
                        printTrans(trans_name);
                else
                        exception(trans_name, "an attempt
                        to get a captured image address before it is set");
                break;
                 ...
```

An Example: Defensive Programming for Contracts

set_img_addr < *get_img_addr*

```
void step(int trans_name)
{
        /*transition system encoding*/
        switch(trans_name)
        {
        case IPU_SET_IMG_ADDR:
                set_img_addr_flag = true;
                printTrans(trans_name);
                break;
         case IPU_GET_IMG_ADDR:
                if (set_img_addr_flag)
                        printTrans(trans_name);
                else
                        exception(trans_name, "an attempt
                        to get a captured image address before it is set");
                break;
                 ...
```

An Example: Defensive Programming for Contracts

set_img_addr < *get_img_addr*

```
void step(int trans_name)
{
        /*transition system encoding*/
        switch(trans_name)
        {
        case IPU_SET_IMG_ADDR:
                set_img_addr_flag = true;
                printTrans(trans_name);
                break;
        case IPU_GET_IMG_ADDR:
                if (set_img_addr_flag)
                        printTrans(trans_name);
                else
                        exception(trans_name, "an attempt
                        to get a captured image address before it is set");
                break;
                 ...
```

An Example: Defensive Programming for Contracts

set_img_addr < *get_img_addr*

```
void step(int trans_name)
{
        /*transition system encoding*/
        switch(trans_name)
        {
        case IPU_SET_IMG_ADDR:
                set_img_addr_flag = true;
                printTrans(trans_name);
                break;
         case IPU_GET_IMG_ADDR:
                if (set_img_addr_flag)
                        printTrans(trans_name);
                else
                        exception(trans_name, "an attempt
                        to get a captured image address before it is set");
                break;
                 ...
```

An Example: Defensive Programming for Contracts

set_img_addr < *get_img_addr*

```
void step(int trans_name)
{
        /*transition system encoding*/
        switch(trans_name)
        {
        case IPU_SET_IMG_ADDR:
                set_img_addr_flag = true;
                printTrans(trans_name);
                break;
         case IPU_GET_IMG_ADDR:
                if (set_img_addr_flag)
                        printTrans(trans_name);
                else
                        exception(trans_name, "an attempt
                        to get a captured image address before it is set");
                break;
                 ...
```

```
tlm::tlm_response_status IPU::write(basic::addr_t addr, basic::data_t d)
{
        switch(addr)
        {
        case IPU_CFG_IMAGE_OFFSET:
                tracking_mode == SPEC_MODE ? monitor->step(name(), IPU_SET_IMG_ADDR);
                captured_image_addr = d;
                break;
```

An Example: Defensive Programming for Contracts

set_img_addr < *get_img_addr*

```
void step(int trans_name)
{
        /*transition system encoding*/
        switch(trans_name)
        {
        case IPU_SET_IMG_ADDR:
                set_img_addr_flag = true;
                printTrans(trans_name);
                break;
         case IPU_GET_IMG_ADDR:
                if (set_img_addr_flag)
                        printTrans(trans_name);
                else
                        exception(trans_name, "an attempt
                        to get a captured image address before it is set");
                break;
                 ...
```

```
tlm::tlm_response_status IPU::write(basic::addr_t addr, basic::data_t d)
{
        switch(addr)
        {
        case IPU_CFG_IMAGE_OFFSET:
                tracking_mode == SPEC_MODE ? monitor->step(name(), IPU_SET_IMG_ADDR);
                captured_image_addr = d;
                break;
```

Non-determinism of Implementation

- Why?
  - To avoid over specification of a system's design
  - To explore more behaviors during the execution
- How?
  - …

Towards Partial Implementation

```
void IPU::face_recognition_analysis()
{
   while(true){
      confidence_value = 0;
      recogn_img_addr = 0;

      /* initialize buffers */
      captured_image_buffer = new basic::data_t [image_size/sizeof(basic::data_t)];
      gallery_image_buffer  = new basic::data_t [image_size/sizeof(basic::data_t)];

      /* upload a captured image */
      upload_image(captured_image_addr, captured_image_buffer);

      basic::data_t temp_conf_value = 0;
      basic::addr_t gallery_image_addr;
      int i = 0;
      while ( i < gallery_size )
      {
              gallery_image_addr = gallery_addr + i * image_size;
              upload_image(gallery_image_addr, gallery_image_buffer);
              temp_conf_value = compare_facial_features();
              if(temp_conf_value > confidence_value)
              {
                      confidence_value = temp_conf_value;
                      recogn_img_addr = gallery_image_addr;
              }
              i += 1;
      }
            ...
```

$start \Rightarrow (read\_img)^{N} < set\_irq\_finish\_true < set\_irq\_finish\_false$

Towards Partial Implementation

```
void IPU::face_recognition_analysis()
{
   while(true){
      confidence_value = 0;
      recogn_img_addr = 0;

      /* initialize buffers */
      captured_image_buffer = new basic::data_t [image_size/sizeof(basic::data_t)];
      gallery_image_buffer  = new basic::data_t [image_size/sizeof(basic::data_t)];

      /* upload a captured image */
      upload_image(captured_image_addr, captured_image_buffer);

      basic::data_t temp_conf_value = 0;
      basic::addr_t gallery_image_addr;
      int i = 0;
      while ( i < gallery_size )
      {
              gallery_image_addr = gallery_addr + i * image_size;
              upload_image(gallery_image_addr, gallery_image_buffer);
              temp_conf_value = compare_facial_features();
              if(temp_conf_value > confidence_value)
              {
                      confidence_value = temp_conf_value;
                      recogn_img_addr = gallery_image_addr;
              }
              i += 1;
      }
            ...
```

start => (*read_img*)^{N} < *set_irq_finish_true* < *set_irq_finish_false*

Towards Partial Implementation

```
void IPU::face_recognition_analysis()
{
  while(true){
```

$start \Rightarrow (read\_img)^{N} < set\_irq\_finish\_true < set\_irq\_finish\_false$

```
    /* upload a captured image */
    upload_image(captured_image_addr, captured_image_buffer);

    basic::data_t temp_conf_value = 0;
    basic::addr_t gallery_image_addr;
    int i = 0;
    while ( i < gallery_size )
    {
            gallery_image_addr = gallery_addr + i * image_size;
            upload_image(gallery_image_addr, gallery_image_buffer);
            temp_conf_value = compare_facial_features();
            if(temp_conf_value > confidence_value)
            {
                    confidence_value = temp_conf_value;
                    recogn_img_addr = gallery_image_addr;
            }
            i += 1;
    }
          ...
```

Towards Partial Implementation

```
void IPU::face_recognition_analysis()
{
  while(true){
```

$start \Rightarrow (read\_img)^{N} < set\_irq\_finish\_true < set\_irq\_finish\_false$

```
    /* upload a captured image */
    upload_image(captured_image_addr, captured_image_buffer);

    basic::data_t temp_conf_value = 0;
    basic::addr_t gallery_image_addr;
    int i = 0;
    while ( i < gallery_size )
    {
            gallery_image_addr = gallery_addr + i * image_size;
            upload_image(gallery_image_addr, gallery_image_buffer);
            temp_conf_value = compare_facial_features();
            if(temp_conf_value > confidence_value)
            {
                    confidence_value = temp_conf_value;
                    recogn_img_addr = gallery_image_addr;
            }
            i += 1;
    }
          ...
```

Towards Partial Implementation

```
void IPU::face_recognition_analysis()
{
   while(true){
```

start => (read_img)^{N} < set_irq_finish_true < set_irq_finish_false

```
    int i = 0;
    while ( i < gallery_size )
    {
            gallery_image_addr = gallery_addr + i * image_size;
            upload_image(gallery_image_addr, gallery_image_buffer);
            temp_conf_value = compare_facial_features();
            if(temp_conf_value > confidence_value)
            {
                    confidence_value = temp_conf_value;
                    recogn_img_addr = gallery_image_addr;
            }
            i += 1;
    }
          ...
```

Towards Partial Implementation

```
void IPU::face_recognition_analysis()
{
   while(true){
```

start => (*read_img*)^{N} < *set_irq_finish_true* < *set_irq_finish_false*

```
int i = 0;
while ( i < gallery_size )
{
        gallery_image_addr
        upload_imag                                  image_size;
        temp_c                                       gallery_image_buffer);
                                                     racial_features();
                                                     confidence_value)

                confidence_value = temp_conf_value;
                recogn_img_addr = gallery_image_addr;
        }
        i += 1;
}
          ...
```

can be simplified

Towards Partial Implementation

```
void IPU::face_recognition_analysis()
{
   while(true){
```

start => (read_img)^{N} < set_irq_finish_true < set_irq_finish_false

```
while(true){
   srand(time(NULL));
   int repeat_count = 2 + (std::rand() % ( MEM_SIZE/sizeof(basic::data_t) - 2));
   while(repeat_count > 0)
   {
        initiator_socket.read(gallery_addr, data);
        repeat_count--;
    }
```

...

Towards Partial Implementation

```
void IPU::face_recognition_analysis()
{
   while(true){
```

start => (read_img)^N < set_irq_finish_true < set_irq_finish_false

```
while(true){
   srand(time(NULL));
   int repeat_count = 2 + (std::rand() % ( MEM_SIZE/sizeof(basic::data_t) - 2));
   while(repeat_count > 0)
   {
        initiator_socket.read(gallery_addr, data);
        repeat_count--;
    }
```

...

Towards Partial Implementation

```
void IPU::face_recognition_analysis()
{
   while(true){
```

start => (read_img)^N < set_irq_finish_true < set_irq_finish_false

```
while(true){
   srand(time(NULL));
   int repeat_count = 2 + (std::rand() % ( MEM_SIZE/sizeof(basic::data_t) - 2));
   while(repeat_count > 0)
   {
        initiator_socket.read(gallery_addr, data);
        repeat_count--;
    }
```

...

Towards Partial Implementation

```
void IPU::face_recognition_analysis()
{
   while(true){
```

start => (read_img)^ < set_irq_finish_true < set_irq_finish_false

```
while(true){
   srand(time(NULL));
   int repeat_count = 2 + (std::rand() % ( MEM_SIZE/sizeof(basic::data_t) - 2));
   while(repeat_count > 0)
   {
        initiator_socket.read(gallery_addr, data);
        repeat_count--;
    }
```

...

Towards Partial Implementation

```
void IPU::face_recognition_analysis()
{
   while(true){
```

start => (read_img)^ < set_irq_finish_true < set_irq_finish_false

```
while(true){
   srand(time(NULL));
   int repeat_count = 2 + (std::rand() % ( MEM_SIZE/sizeof(basic::data_t) - 2));
   while(repeat_count > 0)
   {
        initiator_socket.read(gallery_addr, data);
        repeat_count--;
    }

   irq_out.write(true);
   wait(5, sc_core::SC_NS);
   irq_out.write(false);
```

...

Towards Partial Implementation

```
void IPU::face_recognition_analysis()
{
   while(true){
```

start => (read_img)^{N} < set_irq_finish_true < set_irq_finish_false

```
while(true){
   srand(time(NULL));
   int repeat_count = 2 + (std::rand() % ( MEM_SIZE/sizeof(basic::data_t) - 2));
   while(repeat_count > 0)
   {
        initiator_socket.read(gallery_addr, data);
        repeat_count--;
   }

   irq_out.write(true);
   wait(5, sc_core::SC_NS);
   irq_out.write(false);
```

...

Outline

- ❑ Introduction
- ❑ The case-study
- ❑ Background
- ❑ Non-deterministic specification of a component
- ❑ Ongoing work
- ❑ Conclusions

Ongoing work

- Formalism:
  - Executable
  - Hierarchical
  - Interoperable
  - Suitable for expressing
    - Functional properties
    - Non-functional properties

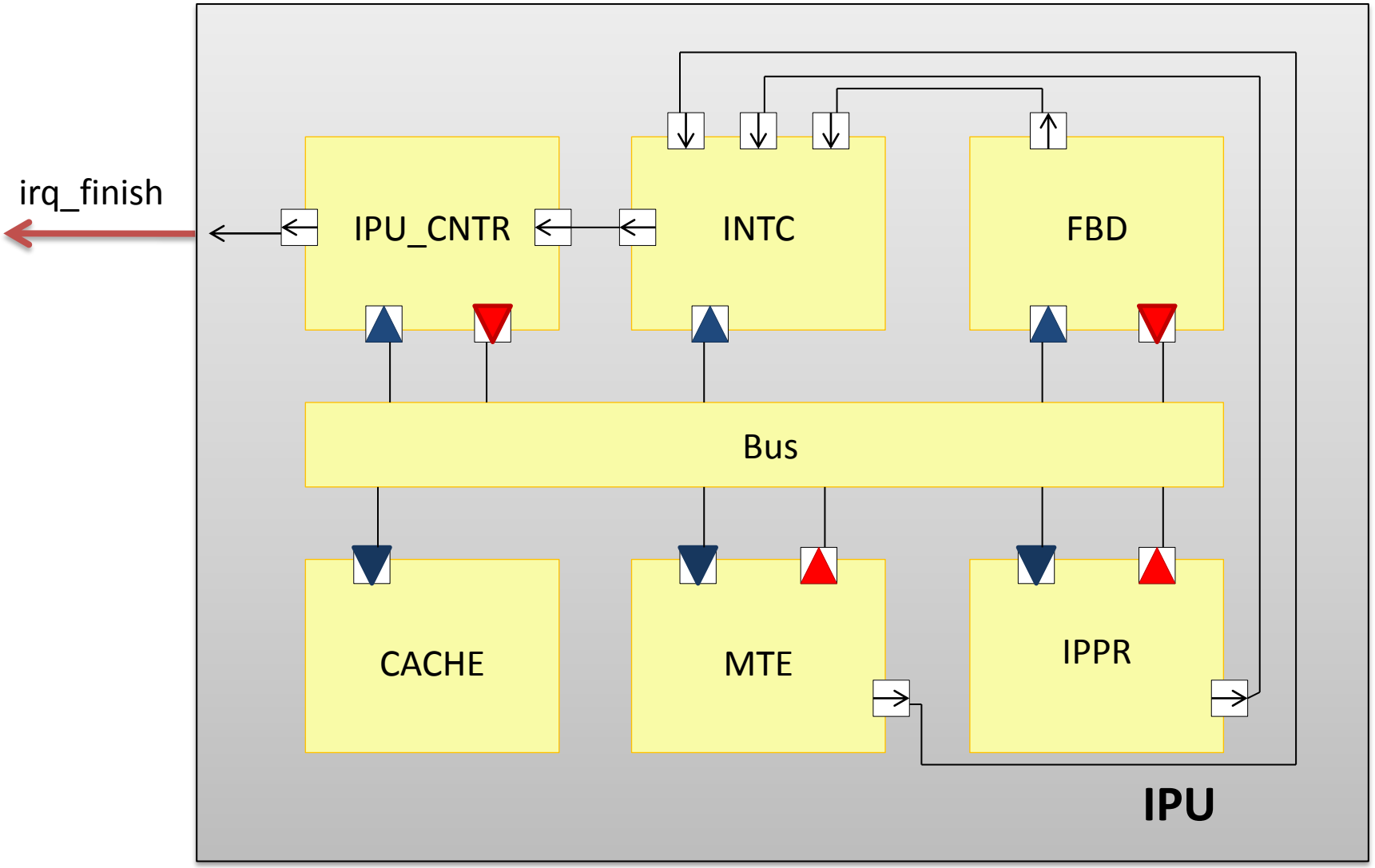
Ongoing work

- Formalism:
  - ✓ Executable
  - Hierarchical
  - Interoperable
  - Suitable for expressing
    - ✓ Functional properties
    - Non-functional properties

Ongoing work

- Formalism:
  - ✓ Executable
  - Hierarchical
  - Interoperable
  - Suitable for expressing
    - ✓ Functional properties
    - Non-functional properties


irq_finish
IPU


irq_finish
IPU_CNTR
INTC
FBD
Bus
CACHE
MTE
IPPR
IPU

Ongoing work

- Formalism:
  - ✓ Executable
  - Hierarchical
  - Interoperable
  - Suitable for expressing
    - ✓ Functional properties
    - Non-functional properties

Ongoing work

- Formalism:
  - ✓ Executable
  - Hierarchical
  - Interoperable
  - Suitable for expressing
    - ✓ Functional properties
    - Non-functional properties

How to instantiate specifications into existing standard (SystemC/TLM)?

Ongoing work

- Formalism:
  - ✓ Executable
  - Hierarchical
  - Interoperable
  - Suitable for expressing
    - ✓ Functional properties
    - Non-functional properties

Ongoing work

- Formalism:
  - ✓ Executable
  - Hierarchical
  - Interoperable
  - Suitable for expressing
    - ✓ Functional properties
    - Non-functional properties

    (reuse of HELP project results)

How to write non-deterministic contracts for power/temperature properties?

Conclusions

- A lot of work to be done…
- See you next year!

Thank you!

Yuliia Romenska
Yuliia.Romenska@imag.fr