Vérification

Transaction Level Modeling

Yann Thoma

Reconfigurable and Embedded Digital Systems Institute
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Février 2016

Introduction

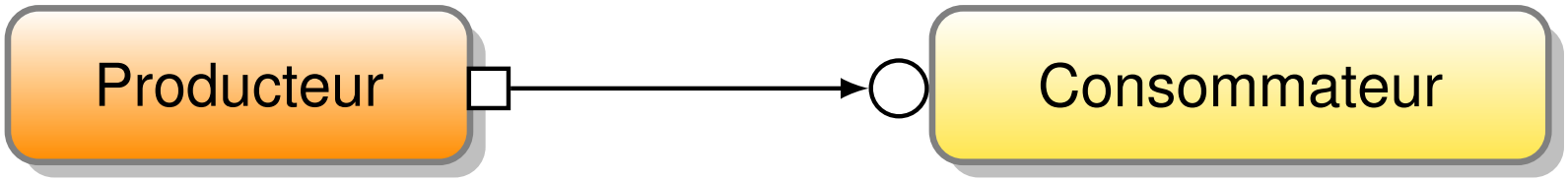
- Les terme *Transaction Level Modeling* a été introduit en 2000 par Synopsys
- But: Aider la modélisation des *System On Chip*
  - Développement rapide des premiers prototypes
  - Analyse de l'architecture
  - Vérification fonctionnelle facilitée
- Langage développé par Synopsys: *SystemC*

Eléments du TLM

- Les composants du système sont modélisés par des **Modules**
- La structure de communication est le **Canal**
- Les modules et les canaux sont liés grâce à des **Ports**
- Un ensemble de données est échangé au moyen d'une **Transaction**
- La synchronisation du système se fait grâce à des actions spécifiques entre modules

Producteur-consommateur

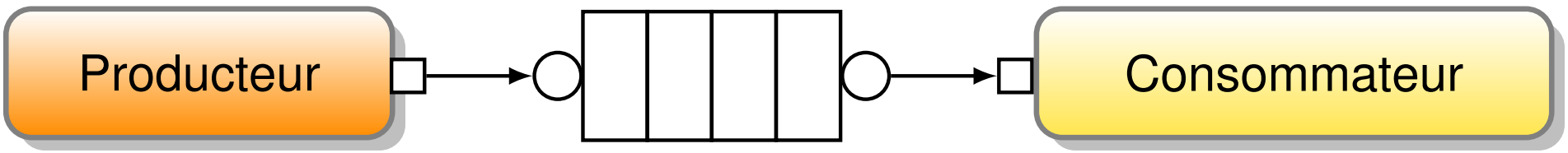
- Les modules possèdent des ports en tant que producteur ou consommateur



- Ils s'échangent des transactions
- Via des méthodes bloquantes ou non (`get`, `put`)

Producteur-consommateur

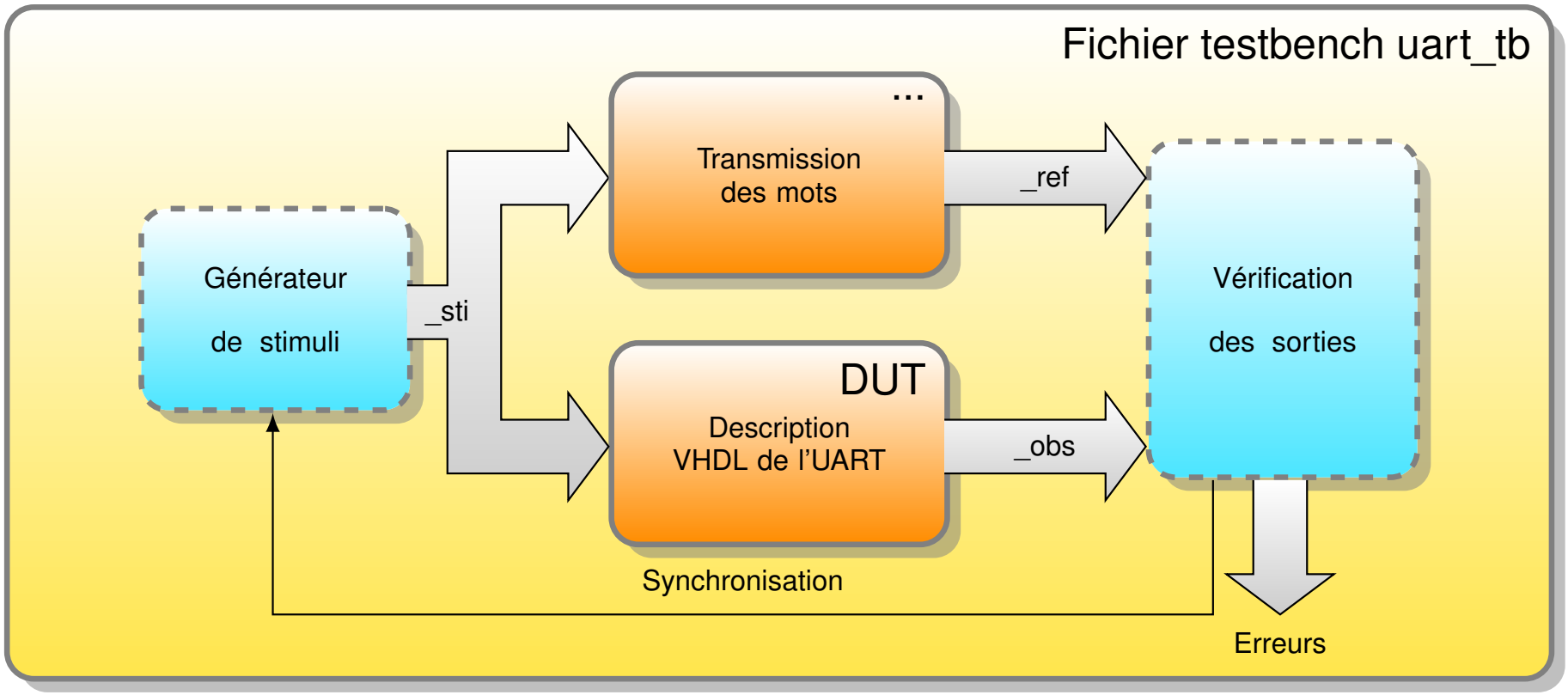
- Des FIFOs peuvent être placés entre les producteurs et les consommateurs



- Ils s'échangent des transactions
- Via des méthodes bloquantes ou non (`get`, `put`)

Transactions

- Contexte:
  - vérification d'un UART ayant un FIFO d'envoi d'une taille quelconque
  - Un processus génère des mots à envoyer
  - Un processus doit vérifier qu'ils ont été traités correctement



Transactions

- Problème: Comment transmettre les mots du processus de génération à celui de vérification?
- Solution: En utilisant un FIFO et y placer des *transactions*
- Utilisation du paquetage `tlm_fifo_pkg.vhd` ou `tlm_unbounded_fifo_pkg.vhd`
- Paquetage générique développé au REDS
  - Attention, nécessite l'usage de VHDL 2008

tlm_unbounded_fifo_pkg

```
package tlm_unbounded_fifo_pkg is

    generic (
        -- Type of the element stored into the FIFOs
        type              element_type;

        -- Maximum number of element stored.
        -- -1 means a virtually infinite number.
        nb_max_data     : integer := -1;

        -- raises objections when not empty
        use_objections : boolean := false;

        -- time between two trials in blocking methods
        retry_delta     : time := 1 ns
    );

    ...
```

tlm_unbounded_fifo_pkg

```
...

type tlm_fifo_type is protected
    -- Puts a data into the FIFO.
    procedure put(data: element_type;ok: out boolean);

    -- Gets a data from the FIFO.
    procedure get(data: out element_type;ok: out boolean);

    -- Indicates if the FIFO is empty
    impure function is_empty return boolean;

    -- Indicates if the FIFO is full
    impure function is_full return boolean;

    -- Returns the number of data available in the FIFO
    impure function nb_data_available return integer;
end protected tlm_fifo_type;

...
```

tlm_unbounded_fifo_pkg

```
    ...

    procedure blocking_put(fifo: inout tlm_fifo_type;
                           data : element_type);

    procedure blocking_get(fifo : inout tlm_fifo_type ;
                           data : out element_type);

    procedure blocking_timeout_put(fifo           : inout tlm_fifo_type;
                                   data           : element_type;
                                   timeout        : time;
                                   ok             : out boolean);

    procedure blocking_timeout_get(fifo           : inout tlm_fifo_type;
                                   data           : out element_type;
                                   timeout        : time;
                                   ok             : out boolean);
end tlm_unbounded_fifo_pkg;
```

Transactions

Définition d'un type à utiliser

```
library ieee;
use ieee.std_logic_1164.all;

package user_pkg is

    type user_type is
    record
        data: std_logic_vector(31 downto 0);
        addr: std_logic_vector(7 downto 0);
    end record user_type;

end user_pkg;
```

Transactions

Instanciation du paquetage

```
library ieee;
use ieee.std_logic_1164.all;
use work.user_pkg.all;
use work.tlm_unbounded_fifo_pkg;

-- Create a specialized package that will offer unbounded FIFOs with the
-- specific user type of data in it
package unbounded_user_tlm_pkg is new tlm_unbounded_fifo_pkg
  generic map (element_type => user_type);
```

Transactions

Variable partagée

```
architecture testbench of tlm_example_tb is

    shared variable unbounded_fifo_user:
                            work.unbounded_user_tlm_pkg.tlm_fifo_type;
```

Transactions

Processus d'écriture

```
write_proc3: process is
    variable sample : user_type;
begin
    for i in 1 to 30 loop
        cycle;
        while (unbounded_fifo_user.is_full) loop
            cycle;
        end loop;
        sample.data := std_logic_vector(to_unsigned(i,32));
        sample.addr := std_logic_vector(to_unsigned(i,8));
        GenerateStimuli(sample);
        blocking_put(unbounded_fifo_user,sample);
    end loop;
    wait;
end process;
```

Transactions

Processus de lecture

```
read_proc3: process is
    variable sample : user_type;
begin
    loop
        cycle;
        while (unbounded_fifo_user.is_empty) loop
            cycle;
        end loop;
        blocking_get(unbounded_fifo_user,sample);
        report "read_proc3. Read a sample. Data: " &
               integer'image(to_integer(unsigned(sample.data)))&
               ". Addr: " &
               integer'image(to_integer(unsigned(sample.addr)));
    end loop;
end process;
```

Découpage du banc de test

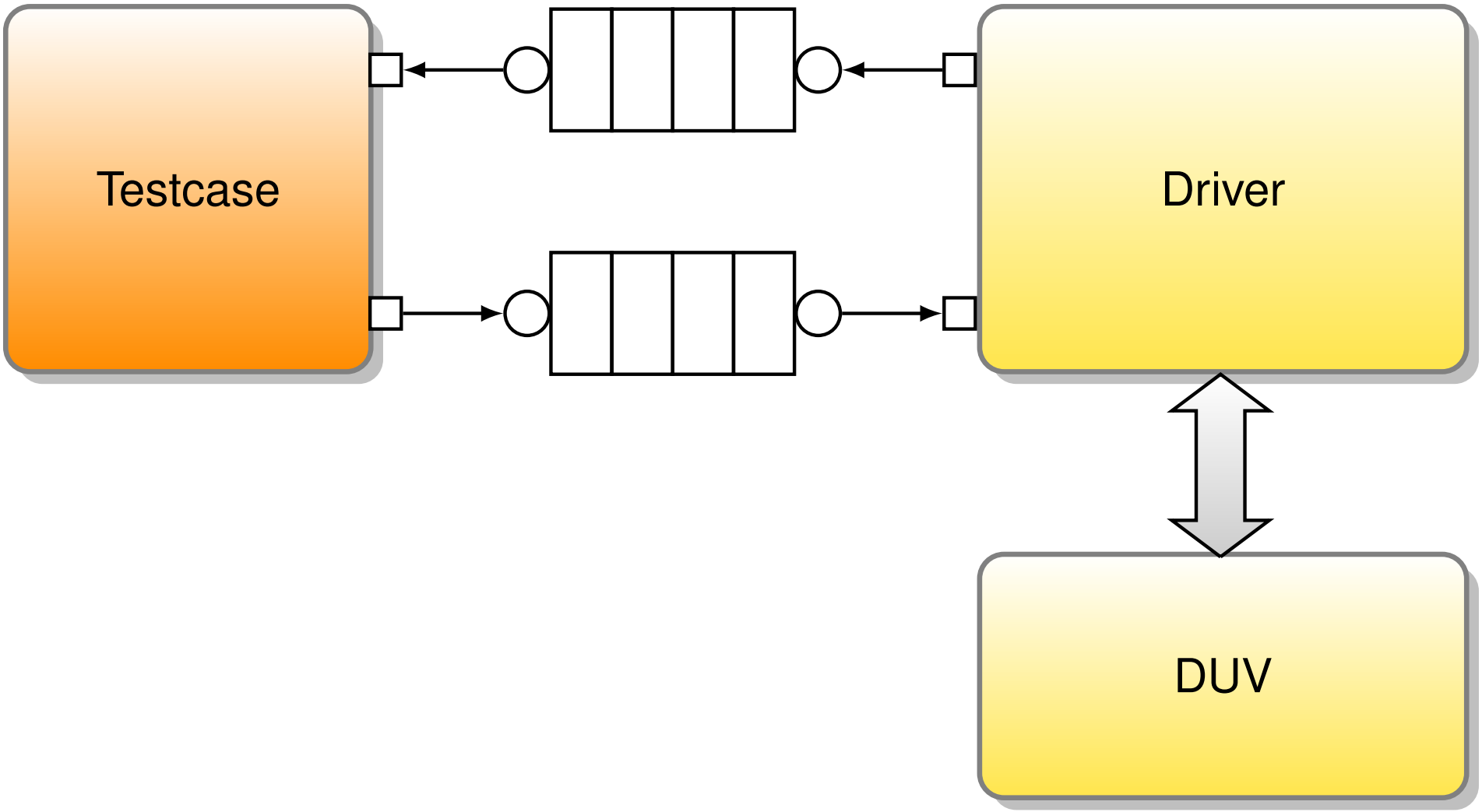
- Le TLM permet de découper le banc de test en une partie abstraite, au niveau transactions, et une *concrète*, au niveau RTL
- Les deux niveaux échangent des informations grâce aux transactions
- Le niveau RTL est rythmé par l'horloge
- Le niveau transactionnel est rythmé par les synchronisations des transactions

Découpage du banc de test

- Exemple pour une mémoire:
  - Un processus qui attend des transactions et interragit avec la mémoire (niveau RTL) : le *driver*
  - Un processus qui génère les transactions
    - Peut être fait dans une procédure
    - $\Rightarrow$ Une procédure par test case
- Intéressant: On peut interchanger des drivers pour tester différentes mémoires avec les mêmes scénarios

$\Rightarrow$ Meilleure réutilisabilité du code

Découpage du banc de test



Transaction: à quel niveau?

- Les transactions représentent une donnée exploitable et entière
- Exemple:
  - Paquet PCIe, paquet Ethernet
  - Transaction mémoire (read ou write)
    - Un burst, éventuellement
  - Une image, une ligne, ou un pixel
  - Un ensemble de données nécessaires à un calcul

Transactions - Conclusions

- Les transactions sont largement exploitées en SystemVerilog/UVM
- Permettent de rester à un niveau d'abstraction plus élevé
- Permettent de s'affranchir d'un temps de traitement dans le DUV
- VHDL-2008 permet de s'approcher de cette manière de faire