Tema 2

- Responsabili: Sergiu Dudă
- Deadline soft (fără penalizări): **08.12.2019** ora **23:59**; Deadline hard: **15.12.2017** ora **23:59**
- Data publicării: 24.11.2019
- Data ultimei actualizări: 27.11.2019, 20:42
- Q&A session: 30.11.2019 ora **12:00** sala **PR002**
- Istoric modificări:
  - 24.11.2019:
    - Initial: Publicat tema 2
  - 25.11.2019:
    - Update: Explicatii pentru SHL si SHR
  - 26.11.2019
    - Update: Definit comportamentul RTL-ului pentru nof_operands == 0
  - 26.11.2019
    - Update: Adaugat explicatii pentru “asertat” si “deasertat”
  - 27.11.2019
    - Update: Adaugat detalii (sala si ora) pentru sesiunea de Q&A

Obiectiv

Tema are ca scop exersarea lucrului cu noțiunile de Verilog folosite pentru proiectarea circuitelor secvențiale.

Sumar

Modulul va primi la intrare un pachet de date format din Header si Payload. Header-ul va contine informatii legate de ce operatie urmeaza sa se execute pe ALU, si numarul de cuvinte ce vor fi primite ca payload. Operanzii vor fi primiti fie in payload, fie vor fi cititi din memorie prin interfata AMM. Rezultatul va fi pus pe interfata de iesire conform structurii ce va fi descrisa in capitolele urmatoare.

Descrierea functionala a modulului

Interfete

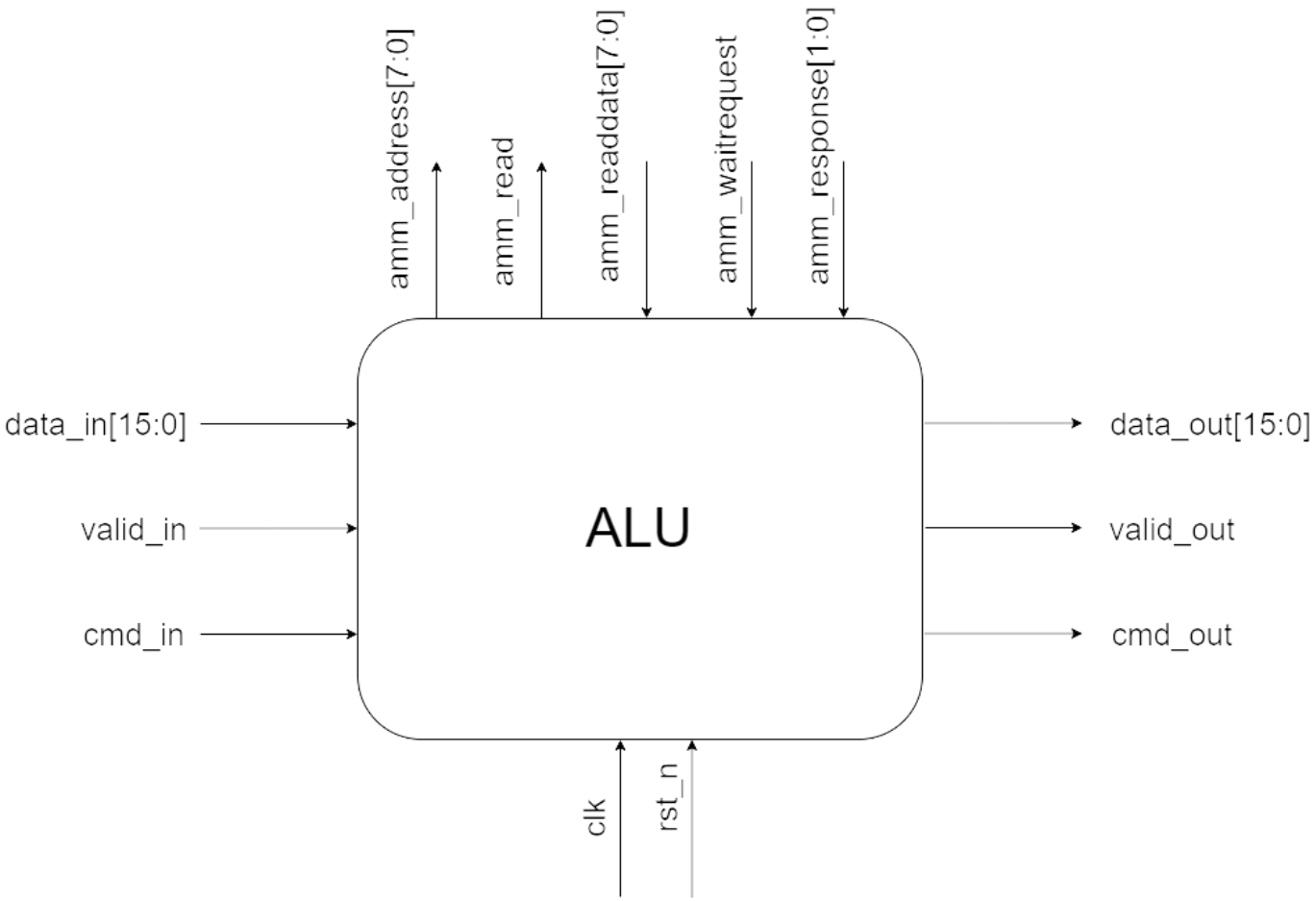


Fig. 1: ALU

Interfata de intrare

Port	Width (bits)	Directie	Valoare la reset	Descriere
data_in	16	in	0	Folosit pentru a primi datele de intrare
valid_in	1	in	0	Valideaza un cuvant de date
cmd_in	1	in	0	Marcheaza header-ul pachetului

Tab. 1: Interfata de Intrare

Interfata de iesire

Port	Width (bits)	Directie	Valoare la reset	Descriere
data_out	16	out	0	Folosit pentru a primi datele de iesire
valid_out	1	out	0	Valideaza un cuvant de date
cmd_out	1	out	0	Marcheaza header-ul pachetului

Tab. 2: Interfata de Iesire

Interfata de AMM (Avalon Memory-Mapped Interfaces)

Port	Width (bits)	Directie	Valoare la reset	Descriere
amm_address	8	out	0	Pe aceasta magistrala master-ul[1] va transmite adresa registrului care va fi citit din memorie.

amm_read	1	out	0	Acest semnal este asertat pentru a indica o operatie de citire a memoriei
amm_readdata	8	in	0	Pe acest bus se va transmite raspunsul la operatia de citire initiata de master
amm_waitrequest	1	in	0	Un slave[2] va tine semnalul amm_waitrequest assertat (activ / ridicat / 1) atunci cand nu poate sa accepte o operatie de citire. Un master va initia un transfer fara sa tina cont daca semnalul amm_waitrequest este asertat sau nu si va astepta pana cand semnalul amm_ waitrequest este deasertat (inactiv / coborat / 0).
amm_response	2	in	0	Contine statusul raspunsului primit de master de la slave

Tab. 2: Interfata de Iesire

[1] In cadrul temei master-ul va fi ALU
[2] In cadrul temei slave-ul va fi memoria (deja implementata in scheletul de cod)

AMM response poate lua urmatoarele valori

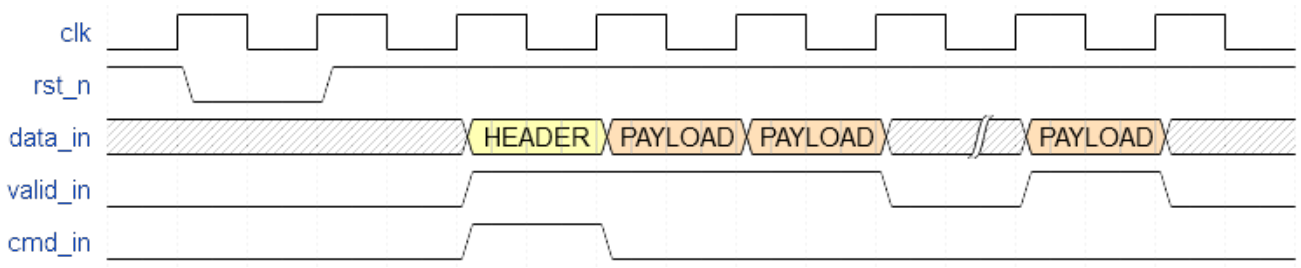
- 2'b00: OK - tranzactie incheiata cu success
- 2'b01: RESERVED - valoare nefolosita
- 2'b10: SLAVEERROR - indica o eroare intampinata de catre slave. Transactia este esuata.
- 2'b11: DECODEERROR - Addresa accesata nu exista

Protocol

Protocol de intrare

Nivel 1 (nivelul fizic)

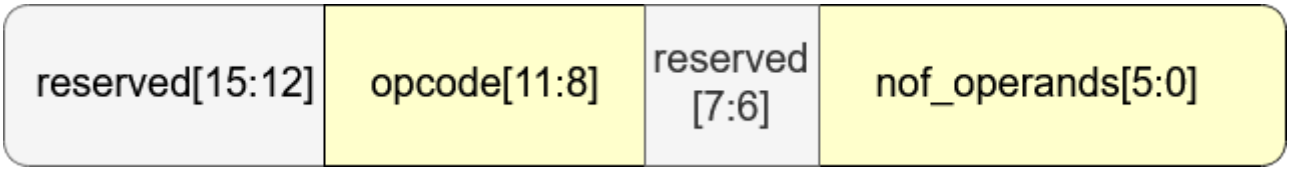
Datele for fi interpretate de modul doar daca sunt validate de semnalul valid_in. Primul cuvant dintr-un transfer este header-ul si trebuie marcat atat de valid_in cat si de cmd_in.



Semnalul valid_in poate fi deasertat intre doua cuvinte consecutive pentru un numar nedeterminat(dar finit) de cicli de ceas.

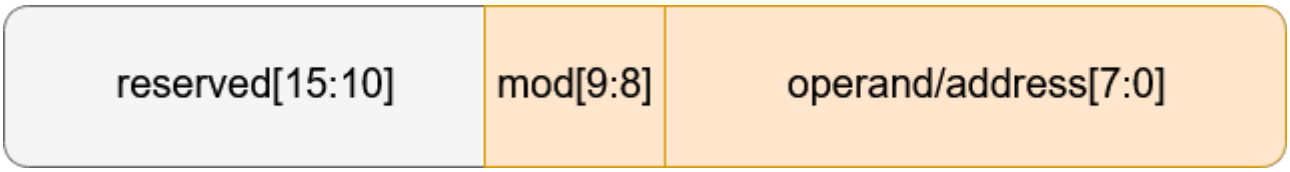
Nivel 2

Header



- Bitii 15:12 nu vor fi folositi in implementarea curenta.
- Bitii 11:8 reprezinta codul operatiei
- Bitii 7:6 nu vor fi folositi in implementarea curenta.
- Bitii 5:0 reprezinta numarul de operanzi ce vor folosi in operatia curenta (minim 1 - maxim 63; nof_operads == 0 va genera un pachet de eroare)

Payload

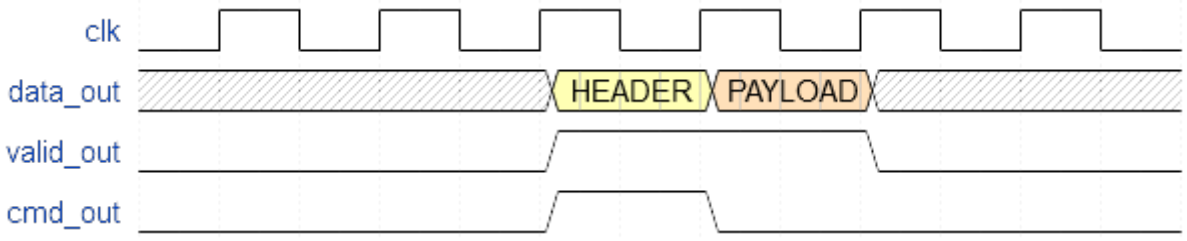


- Bitii 15:10 nu vor fi folositi in implementarea curenta.
- Bitii 9:8 reprezinta modul de addresare
  - MOD 00 - adresare imediata
  - MOD 01 - adresare indirecta
- Bitii 7:0 in fuctie de modul de adresare, pot sa contina operandul (adresare imediata) sau o adresa de memorie (adresare indirecta)

Protocol de iesire

Nivel 1 (nivelul fizic)

Datele for fi interpretate de modul doar daca sunt validate de semnalul valid_out. Primul cuvant dintr-un transfer este header-ul si trebuie marcat atat de valid_out cat si de cmd_out.



Nivel 2

Header

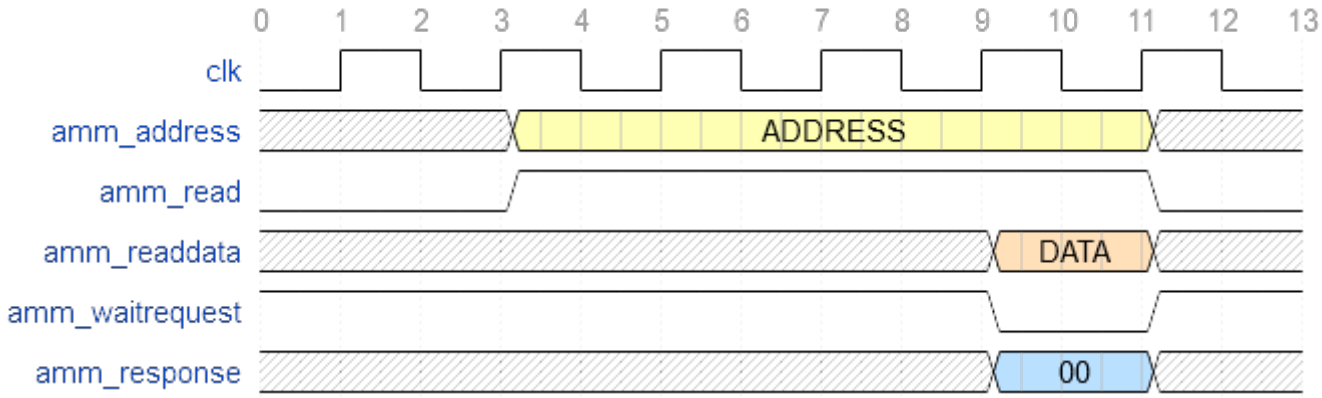
- Bitii 15:5 nu vor fi folositi in implementarea curenta.
- Bitul 4 este setat doar in cazul unei erori
- Bitii 3:0 reprezinta codul operatiei

Payload

- Bitii 15:12 nu vor fi folositi in implementarea curenta.
- Bitii 11:0 vor contine rezultatul acestei operatii.

Dimensiunea campului result a fost aleasa a.i. sa nu fie posibil sa existe overflow la operatia de adunare

Protocol AMM



Sincronizarea semnalelor din diagrama de mai sus este urmatoarea:

- Pe frontul pozitiv de ceas (3) masterul aserteaza amm_read si pune adresa pe magistrala pentru a initia tranzactia
- Atat amm_read, cat si amm_address trebuie tinute constante cat timp amm_waitrequest este assertat (minim de 3 cicli de ceas pentru a realiza un handshake cu memoria), altfel tranzactia va fi ignorata.
- Pe frontul pozitiv de ceas (9) slave-ul deaserteaza waitrequest si seteaza pe magistrala amm_readdata si amm_response
- Pe frontul pozitiv de ceas (11) master-ul detecteaza ca waitrequest-ul s-a deasertat, preia amm_readdata si amm_response si apoi de-aserteaza amm_read , incheind astfel transferul.

Operatii

Modulul ALU este capabil sa execute 10 operatii:

Operatie	Codul Operatiei (hex)	Exemplu de header(hex)	Nota
ADD	0x0	0x0005	Se va realiza suma a 5 numere naturale.
AND	0x1	0x0102	Se va realiza & logic a 2 numere naturale.
OR	0x2	0x0203	Se va realiza | logic a 3 numere naturale.
XOR	0x3	0x0301	Se va realiza ^logic pentru 1 numar natural ⇒ acelasi numar va fi trimis la iesire (repetor).
NOT	0x4	0x0401	Se neaga toti bitii operandului. Daca nof_operands != 1 se va genera un pachet de eroare
INC	0x5	0x0501	Se incrementeaza operandul. Daca nof_operands != 1 se va genera un pachet de eroare
DEC	0x6	0x0601	Se decrementeaza operandul. Daca nof_operands != 1 se va genera un pachet de eroare
NEG	0x7	0x0701	Se neaga operandul. Daca nof_operands != 1 se va genera un pachet de eroare
SHR	0x8	0x0802	In cazul “»”, primul operand este numarul care va fi shiftat, iar al doilea operand reprezinta numarul de biti care vor fi shiftati. Daca nof_operands != 2 se va genera un pachet de eroare

SHL	0x9	0x0902	In cazul “«”, primul operand este numarul care va fi shiftat, iar al doilea operand reprezinta numarul de biti care vor fi shiftati. Daca nof_operands != 2 se va genera un pachet de eroare

Campul **nof_operands** are valori valide in intervalul [1:63]. In cazul in care **nof_operands == 0**, se va genera un pachet de eroare.

Memoria

Modulul de memorie este deja implementat in scheletul de cod. Datele din memorie se vor accesa prin intermediul interfetei AMM. Acesta are un spatiu de adrese pe 8 biti (255 de adrese disponbile), insa intern modulul nu va folosi toate aceste adrese, ci va fi impartit in 3 blocuri de registri.

Register block 1	Start address: 0x00	End address: 0x0F
Register block 2	Start address: 0x30	End address: 0x7F
Register block 3	Start address: 0xA0	End address: 0xFF

Orice acces in intervalele de mai sus se va termina cu success. Un access in afara intervalelor definite, va genera o eroare (amm_response = DECODEERROR)

In cadrul testelor, se vor accesa adrese atat din spatiul definit anterior, cat si din afara acestora.

Cazuri de eroare

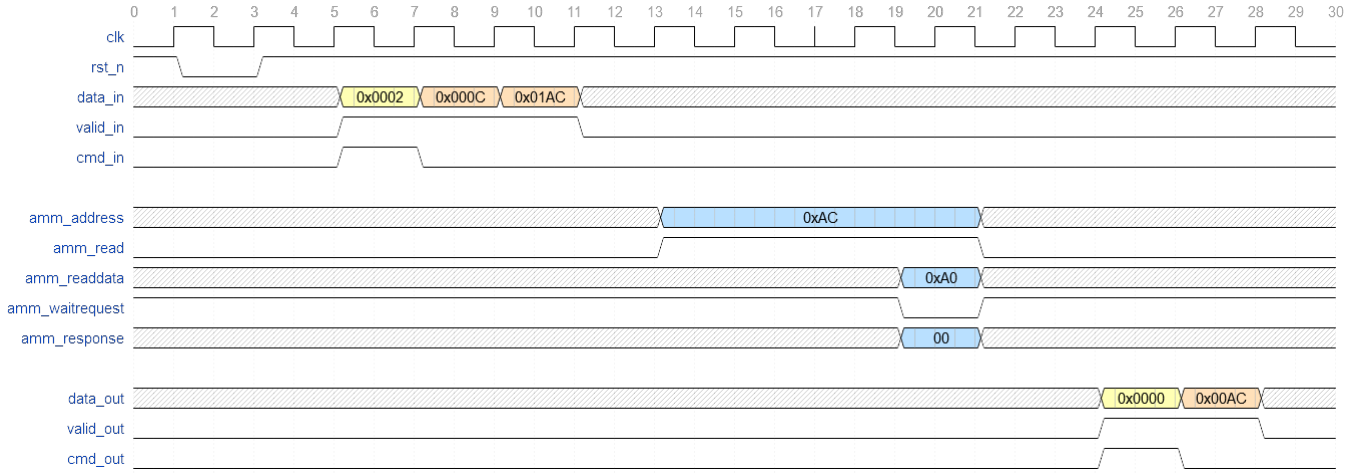
Toate cazurile de eroare au fost mentionate deja in sectiunile precedente. ALU va seta bitul de eroare in doua cazuri:

- nof_operands nu are o valoare valida pentru operatia ce urmeaza sa fie executata.
- amm_response != 2’b00

! Important ! In caz de eroare result primeste valoarea 0xBAD

Exemplu

Exemplu 1: Adunarea a doua numere, primul operand adresare imediata, al doilea adresare indirecta (adresa valida).



Decodificare pachet de intrare:

- Header
  - reserved = 0
  - opcode = 0 (operatia de adunare)
  - nof_operands = 2 (vor fi doi operanzi)
- Payload 1:
  - reserved = 0
  - mod = 00 (adresare imedita)
  - operand = 0x0C
- Payload 2:
  - reserved = 0
  - mod = 01 (adresare indirecta)
  - adresa = 0xAC

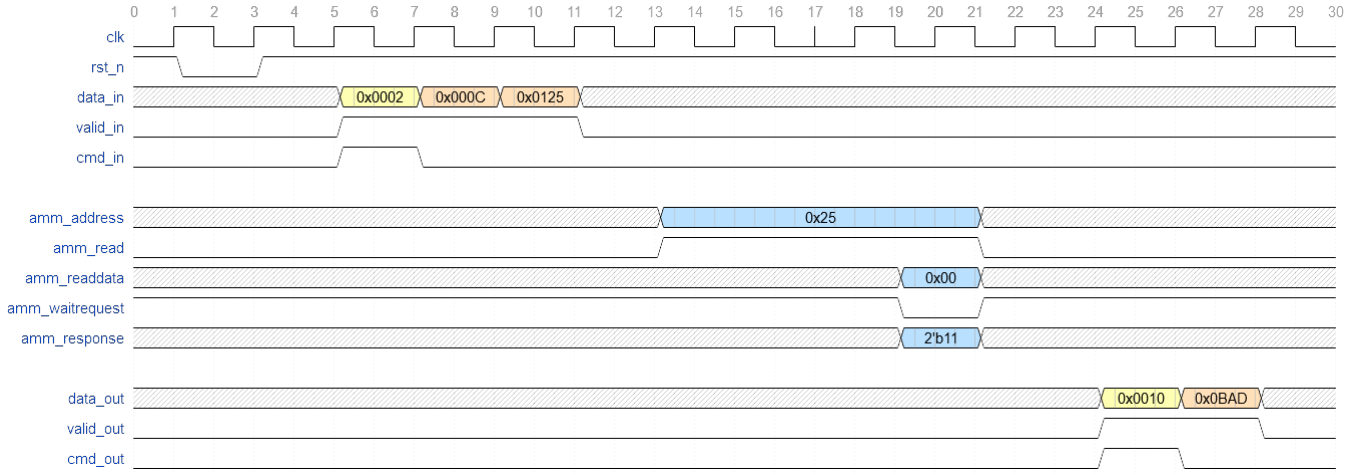
Deoarece exista un operand cu mod de adresare indirecta, o tranzactie de AMM este inceputa la adresa 0xAC. Dupa 3 cicli de ceas memoria returneaza amm_readdata = 0xA0 si amm_response = 2'b00 indicand ca accesul a fost incheiat cu success.

Rezultatul este obtinut adunand ambii operanzi: 0xA0 + 0x0C = 0xAC.

Decodificare pachet de iesire:

- Header
  - reserved = 0
  - error = 0
  - opcode = 0 (operatia de adunare)
- Payload:
  - reserved = 0
  - result = 0xAC

Exemplu 2: Adunarea a doua numere, primul operand adresare imediata, al doilea adresare indirecta (adresa invalida).



Decodificare pachet de intrare:

- Header
  - reserved = 0
  - opcode = 0 (operatia de adunare)
  - nof_operands = 2 (vor fi doi operanzi)
- Payload 1:
  - reserved = 0
  - mod = 00 (adresare imedita)
  - operand = 0x0C
- Payload 2:
  - reserved = 0
  - mod = 01 (adresare indirecta)
  - adresa = 0x25

Deoarece exista un operand cu mod de adresare indirecta, o tranzactie de AMM este inceputa la adresa 0x25.Deoarece este o adresa invalida, dupa 3 cicli de ceas, memoria va seta amm_response = 2’b11. Valoarea de pe amm_readdata este irelevanta.

Decodificare pachet de iesire:

- Header
  - reserved = 0
  - error = 1
  - opcode = 0 (operatia de adunare)
- Payload:
  - reserved = 0
  - result = 0xBAD

Precizări

- Arhiva temei (de tip **zip**) trebuie să cuprindă în rădăcina sa (**fără alte directoare**) doar:
  - fișierele sursă (extensia .v)
  - fișierul README
- Arhiva nu trebuie să conțină fișiere de test, fișiere specifice proiectelor etc.
- Fișierului README va conține minim:

    - numele și grupa
    - prezentarea generală a soluției alese (ex: diagrama automatului implementat)
    - explicarea porțiunilor complexe ale implementării (poate fi făcută și în comentarii)
    - alte detalii relevante
- Vmchecker ne permite să revenim la orice soluție încărcată de voi; cereți revenirea la cea mai convenabilă soluție trimisă (punctaj teste automate + depunctare întârziere) printr-un mail responsabilului de temă
- Nu se oferă suport pe forum/mail pentru teme care nu abordează automatul secvențial și combinațional și nu folosesc atribuirile conform recomandărilor. Este obligatoriu la sesiuni de consultații să aveți diagrama de stări a automatului implementat.
- Tema trebuie realizată individual; folosirea de porțiuni de cod de la alți colegi sau de pe Internet (cu excepția site-ului de curs) poate fi considerată copiere și va fi penalizată conform regulamentului.

Notare

- 10 pct: corectitudine
- -10 pct: folosirea construcțiilor nesintetizabile din Verilog (while, repeat, for cu număr variabil de iterații etc.)
- -1 pct: lipsa fișierului README.
- -0.5 pct: pentru fiecare zi de întârziere; tema poate fi trimisă cu maxim 7 zile întârziere față de termenul specificat în enunț (față de deadline-ul soft).
- -0.2 pct: folosirea incorectă a atribuirilor continue (`assign`), blocante (=) și non-blocante (<=).
- -0.2 pct: indentare haotică
- -0.2 pct: lipsa comentariilor **utile**
- -0.1 pct: comentarii inutile (ex. `wire x; // semnalul x`)
- -0.2 pct: diverse alte probleme constatate în implementare (per problemă)

Dacă tema primește 0 pe *vmchecker*, se pot acorda maxim 2 puncte pe ideea implementării, la latitudinea asistentului. Ideea și motivele pentru care nu funcționează trebuie documentate temeinic în README și/sau comentarii. Temele care au erori de compilare vor fi notate cu 0 puncte.

Resurse

- Schelet
- **Tester offline (Coming Soon!)**
- PDF temă
- Avalon® Interface Specifications
- Ghidul studentului la AC
- Utilizarea vmchecker
- Debugging folosind Xilinx ISE