Sisteme cu F. P. G. A. și D. S. P.

– Implementarea unui numărător binar în mediul Matlab – Simulink –

– Prin intermediul generatorului automat de cod HDL Coder – [1]

I. INTRODUCERE:

Circuitele logice sau digitale cu rol de numărare, sunt utilizate în diverse aplicații precum: generarea semnalelor modulate în lățime, măsurarea frecvenței, măsurarea vitezei, numărarea obiectelor, contorizarea unor fenomene sau evenimente etc.

Implementarea unor astfel de circuite poate fi realizată în mod fizic (eng. hardware), cu ajutorul componentelor discrete (ex. porți logice, bi-stabile, memorii etc.), sau pe baza elementelor din componența unui nucleu FPGA.

În vederea implementării circuitelor logice (digitale) pe baza unui nucleu de tip FPGA (eng. Field Programmable Gate Array), este necesar să se cunoască cel puțin una dintre metodele de reconfigurare și programare fizică (hardware). Există deci trei astfel de procedee:
- programare sau reconfigurare pe bază cod Verilog sau VHDL;
- programare sau reconfigurare pe bază de diagrame bloc în format IP Integrator (Vivado);
- programare sau reconfigurare pe baza modelului Matlab – Simulink (System Generator);

În continuare, se va aborda metoda de programare sau reconfigurare pe baza modelului Matlab – Simulink utilizând generatorul automat de cod HDL Coder. Pentru a evidenția modul de funcționare, se va utiliza platforma de dezvoltare ZedBoard cu nucleu FPGA Zynq 7000 și componentele instalate pe placă (ex. indicatori LED și butoane).

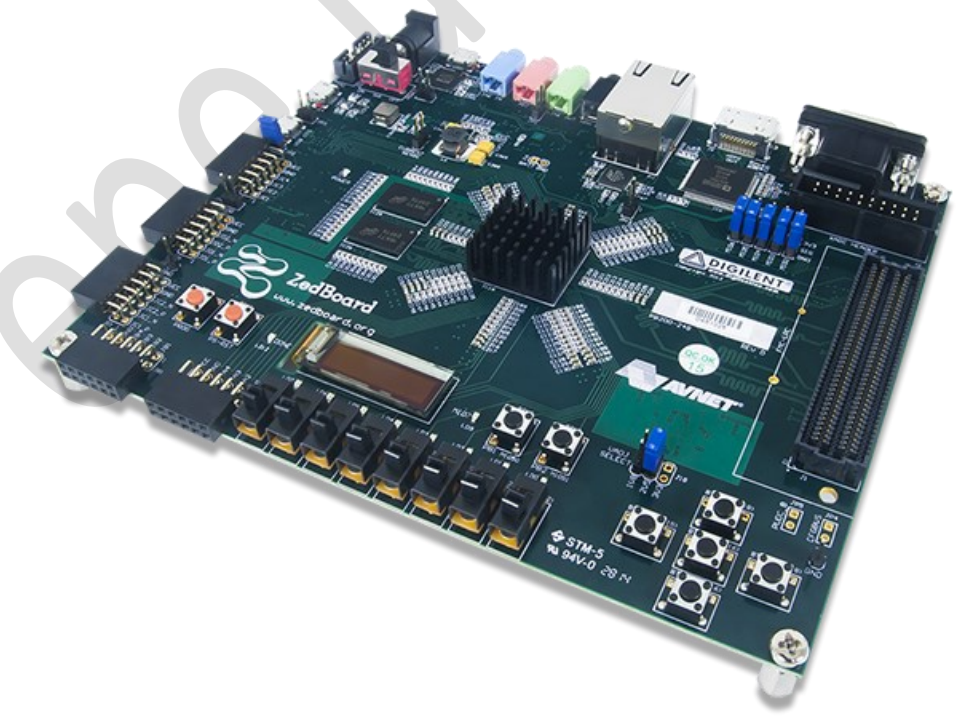


Fig. 1 - Platforma de dezvoltare cu FPGA ZedBoard – Zynq 7000 [2]

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

II. IMPLEMENTARE:

Pachetul de instrumente HDL Coder, se regăsește între componentele mediului Matlab – Simulink și conține o serie de sub-categorii de instrumente, utile în vederea dezvoltării aplicațiilor pentru sisteme de calcul bazate pe nuclee FPGA. Pachetul HDL Coder, reprezintă un mod de abordare universal, deci poate fi utilizat pentru orice tip de FPGA, indiferent de producător (ex. Xilinx, Digilent, Intel – Altera etc...).

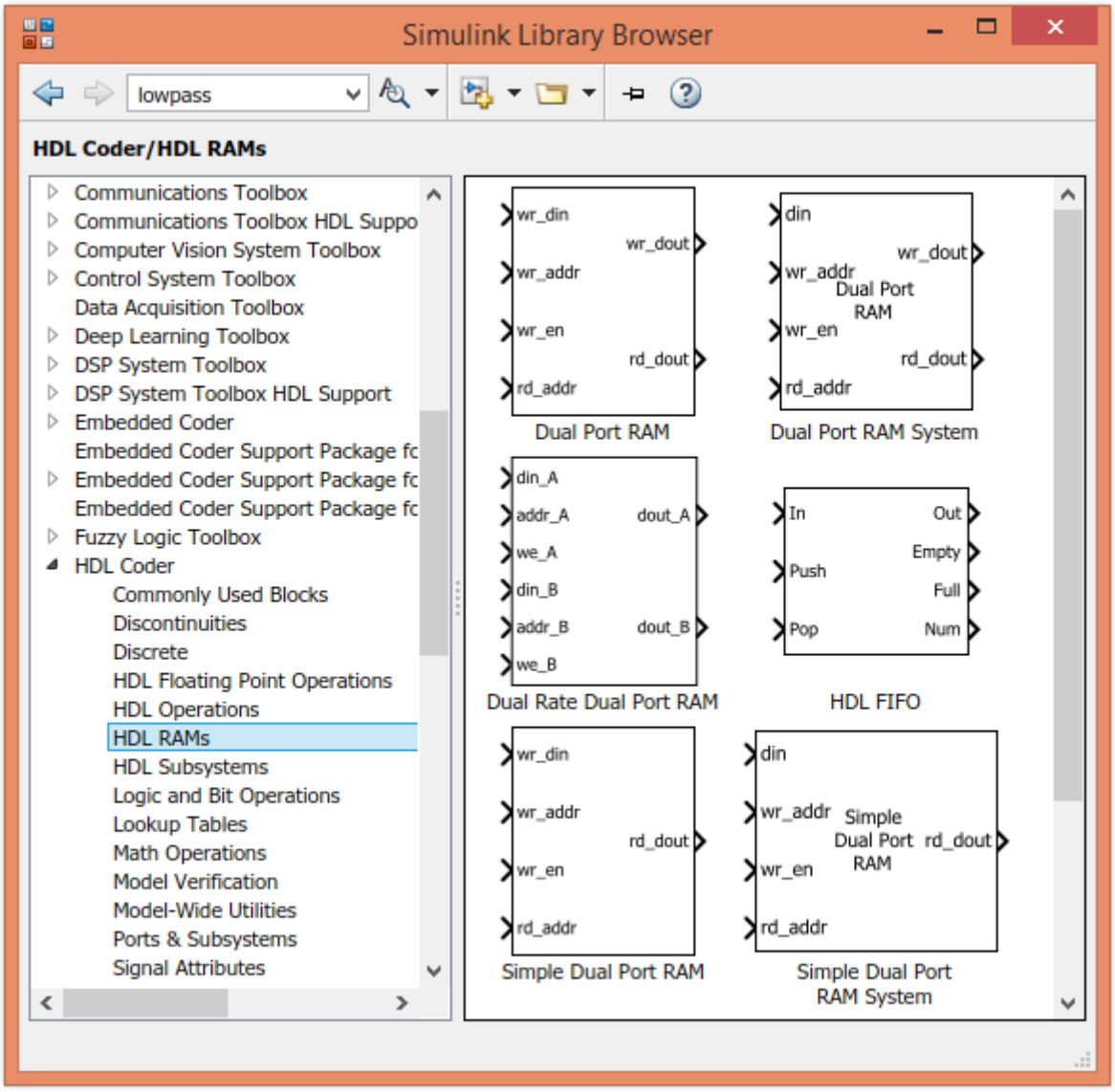


Fig. 2 – Paleta de instrumente specifică generatorului automat de cod „HDL Coder”

În cadrul paletei de instrumente „HDL Coder”, există o serie de blocuri, utile în procesul de dezvoltare al unei aplicații pentru un nucleu de tip FPGA. Spre exemplu, printre blocurile aferente categoriei „Logic and Bit Operations” sunt incluse și funcțiile logice primitive (ex. ȘI, SAU, NEGARE etc.). Utilizarea acestor funcții în cadrul modelului Matlab – Simulink, conduce la implementarea fizică a unor porți logice (grupări de porți), la nivelul capsulei nucleului FPGA.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Astfel, pe baza elementelor din paleta de instrumente „HDL Coder – Logic and Bit Operations”, se pot implementa circuite logice (digitale) la nivel de FPGA. Prin urmare, pe baza acestor elemente, se va implementa modelul de numărător binar la nivel de nucleu FPGA.

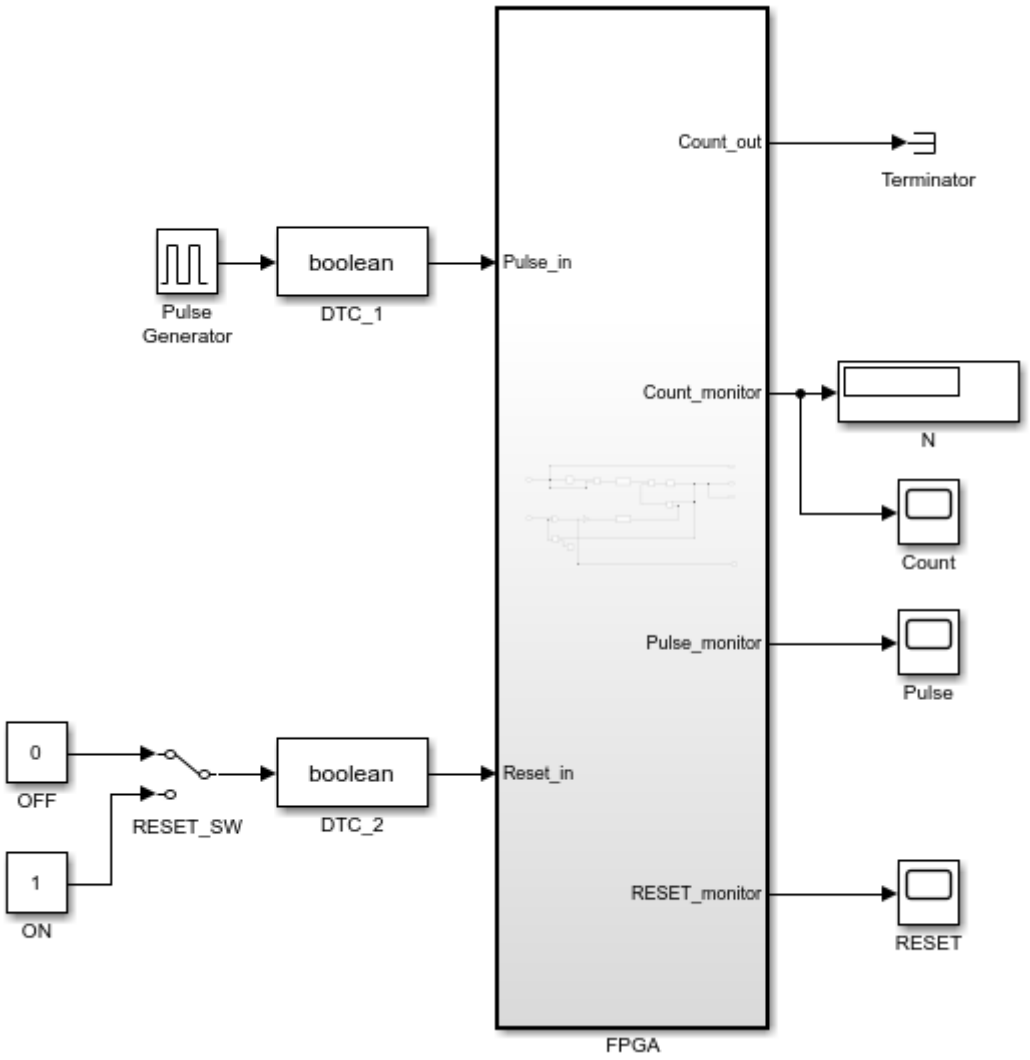


Fig. 3 – Modelul Matlab – Simulink – HDL Coder al numărătorului

În cadrul modelului propus, se va introduce în mod provizoriu, pentru testare, un generator de impulsuri, care va imita comportamentul unui buton fizic cu apăsare și revenire. De asemenea, se va creea o structură de comutator bipozițional, cu ajutorul a două constante („OFF” - 0 și „ON” - 1) și a unui bloc comutator acționat în mod manual. Rolul acestei structuri este de a permite re-inițializarea operației de incrementare (adică funcția RESET). La ieșirea blocului – sub-sistem „FPGA”, se vor atașa instrumente indicatoare precum osciloscoape virtuale și afișaje numerice. Pentru ieșirile care nu au un corespondent numeric, se va introduce blocul de încheiere de capăt „Terminator”. Ieșirile care nu vehiculează o mărime numerică în model, sunt alocate, de obicei, înspre resursele fizice ale platformei de dezvoltare (ex. indicatori luminoși de tip LED). Sub-sistemul „FPGA” conține logica de comandă. Blocurile care compun structura internă a sub-sistemului, sunt specifice paletei de instrumente „HDL Coder”, astfel după modelul sub-sistemului se va re-configura structura capsulei FPGA.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

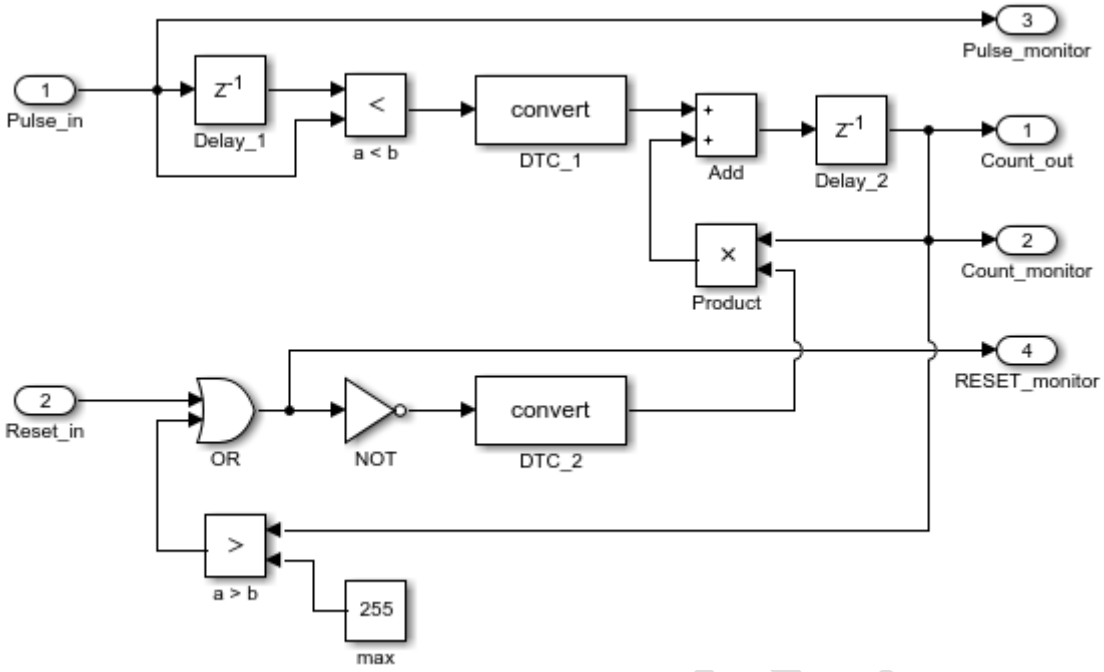
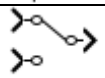


Fig. 4 – Structura internă a blocului „FPGA” din model

În cadrul modelului întocmit se regăsesc următoarele blocuri:

Simbol bloc	Denumire bloc	Categorie / Subcategorie
Pulse Generator	Pulse Generator	Simulink – Sources
Delay	Delay	HDL Coder – Discrete
Relational Operator	Relational Operator	HDL Coder – Logic and Bit Operations
convert Data Type Conversion	Data Type Conversion	HDL Coder – Commonly Used Blocks
Add	Add	HDL Coder – Math Operations
Product	Product	HDL Coder – Math Operations
Display	Display	HDL Coder – Sinks
Scope	Scope	HDL Coder – Sinks

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

	Logical Operator	HDL Coder – Logic and Bit Operations
	Manual Switch	Simulink – Signal Routing
	Constant	HDL Coder – Commonly Used Blocks

Principiul de funcționare al modelului implementat se bazează pe următoarele etape:
- achiziționare semnalului de la o intrare digitală furnizat de un buton sau un senzor digital;
- determinarea frontului crescător al semnalului digital (eng. rising edge detection);
- acumularea prin incrementare a numărului de impulsuri și stocarea acestuia în memorie;
- revenirea la starea inițială în momentul atingerii pragului maxim, sau acționării unui buton;

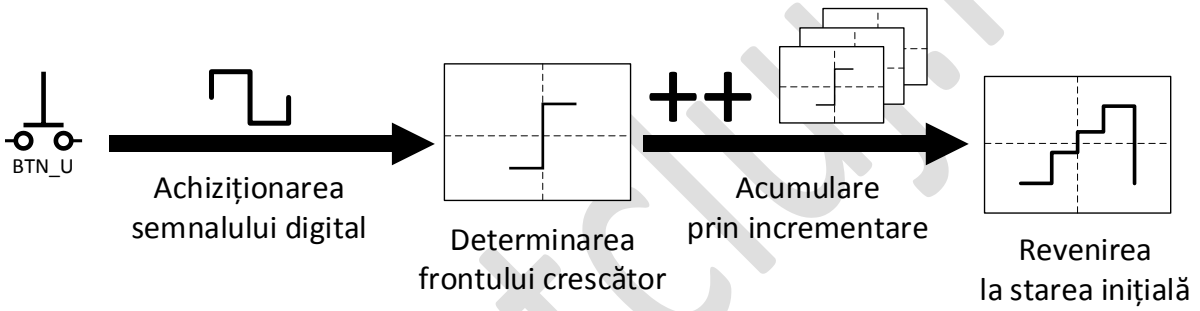


Fig. 5 – Principiul de funcționare al numărătorului

Astfel, prin apăsarea repetată a unui buton cu acțiune momentană, are loc incrementarea unei variabile, stocată într-o anumită zonă de memorie. Incrementarea variabile se va realiza până când condiția de re-inițializare nu este îndeplinită. Condiția de re-inițializare (eng. reset), poate fi indeplinită atât prin apăsarea unui buton fizic, cât și prin atingerea pragului maxim (ex. în cazul de față, valoarea 255).

Pentru a implementa funcția de numărător la nivelul platformei de dezvoltare ZedBoard se vor utiliza elementele de intrare și ieșire din componența plăcii. Simbolurile elementelor și denumirile terminalel sunt inscripționate pe mascajul fotolitografic.

Denumire element	Simbol element	Terminal atașat la nucleul FPGA
Buton cu apăsare și revenire	BTNU	T18
Buton cu apăsare și revenire	BTNC	P16
Indicator luminos LED	LD0	T22
Indicator luminos LED	LD1	T21
Indicator luminos LED	LD2	U22
Indicator luminos LED	LD3	U21
Indicator luminos LED	LD4	V22
Indicator luminos LED	LD5	W22
Indicator luminos LED	LD6	U19
Indicator luminos LED	LD7	U14

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

IMPORTANT: Tipul de date vehiculat în cadrul modelului, este Fixed Point, având reprezentare pe 8 biți (interval de reprezentare 0 la 255). Din acest motiv se vor utiliza blocuri de conversie al tipului de date vehiculat.

Pentru a permite interacțiunea dintre mediul de simulare Matlab – Simulink și mediul de programare Xilinx Vivado, este necesară stabilirea unui director principal de lucru, și introducerea comenzii următoare în consola de comandă Matlab:

```
hdlsetuptoolpath('ToolName','Xilinx
Vivado','ToolPath','C:\Xilinx\Vivado\2016.2\bin\vivado.bat');
```

Pentru a creea legătura dintre platforma de dezvoltare Zedboard Zynq 7000 și mediul Matlab – Simulink se va introduce următoarea comandă în consola Matlab:

```
z = zynq;
```

Pentru a reglementa parametrii de comunicare dintre platforma de dezvoltare și computerul gazdă se va introduce următoarea comandă în consola Matlab:

```
z = z.setupZynqHardware
```

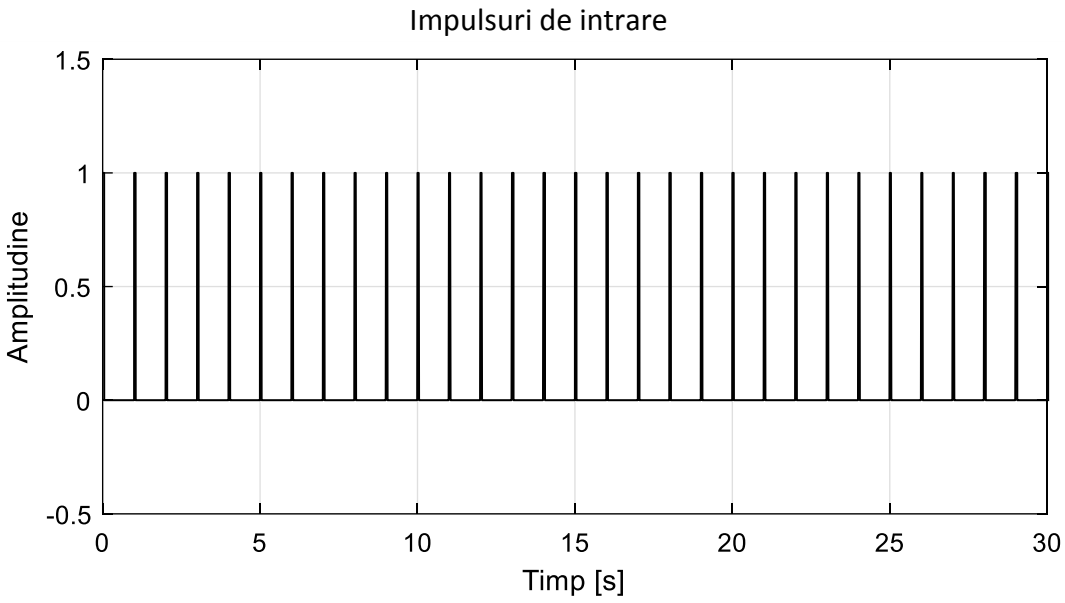
Ca și rezultat al introducerii comenzilor anterioare, mediul Matlab va returna în cadrul consolei de comandă toate datele de identificare și de conectare la platforma de dezvoltare:

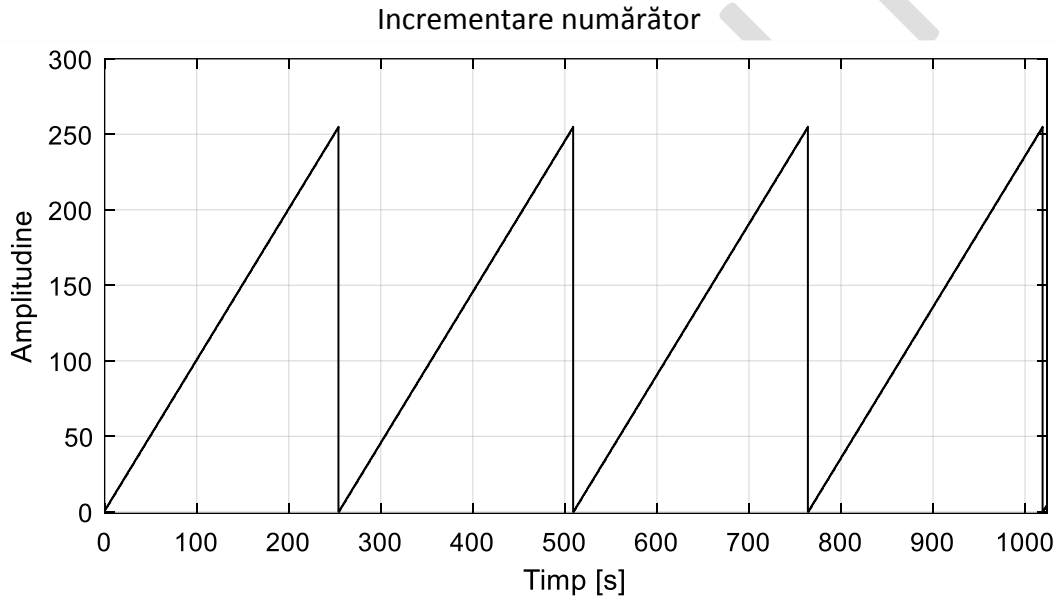
```
z =

  LinuxShell with properties:

    IPAddress: '76.45.136.111'
     Username: 'root'
         Port: 22
```

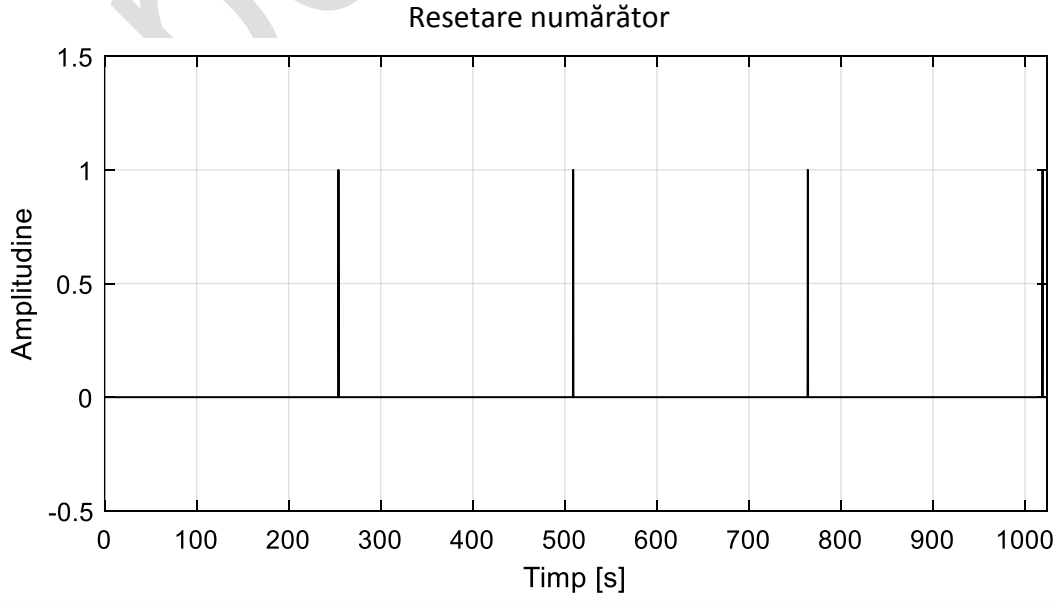
Înainte de a genera în mod automat atât fișierul BitStream specific procesului de re-configurare FPGA, cât și fișierul executabil necesar pentru partea de procesor ARM de aplicație, se va testa modelul implementat. Se vor stabili următoarele condiții de simulare:
- timp total de simulare: 1024 [s];
- metoda de rezolvare a ecuațiilor diferențiale (eng. Solver): discrete (no continupus states);
- tipul metodei de rezolvare a ecuațiilor diferențiale (eng. Solver Type): Fixed-step;
- timpul total de eșantionare (eng. Sample Time): 1e-4 [s];
- perioada de generare a impulsurilor de comandă (eng. Pulse Period): 1 [s];
- tipul de date vehiculat în model (eng. data type): Unsigned fixed point 8 bit: fixdt(0,8,0);
- condiția inițială de re-inițializare: zero;

Se vor obține următoarele forme de undă ca și rezultate de simulare:

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro







Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Rezultatele de simulare, reflectă faptul că, la fiecare impuls de intrare înregistrat, numărătorul incrementează variabila de ieșire. Incrementarea are loc până la valoarea 255, deoarece numărătorul are rezoluția maximă de 8 biți (adică (2^8)-1 = 255). Condiția de resetare este întrunită atunci când valoarea numerică rezultantă este egală cu 255.

Cu ajutorul unui registru de opt indicatori luminoși LED (8 biți), se vor putea descrie toate combinațiile binare de la 0 la 255. Cei opt indicatori luminoși LED se află în dotarea platformei ZedBoard, și vor fi utilizați în acest scop. La fel, se vor utiliza și două butoane cu apăsare și revenire. Un buton pentru incrementare, iar celălalt pentru re-inițializare.

Pentru a implementa modelul la nivel de platformă de dezvoltare, se va selecta blocul – sub-sistem „FPGA”, iar cu ajutorul operației „clic dreapta” se va alege (din meniul contextual), opțiunea „HDL Workflow Advisor”.

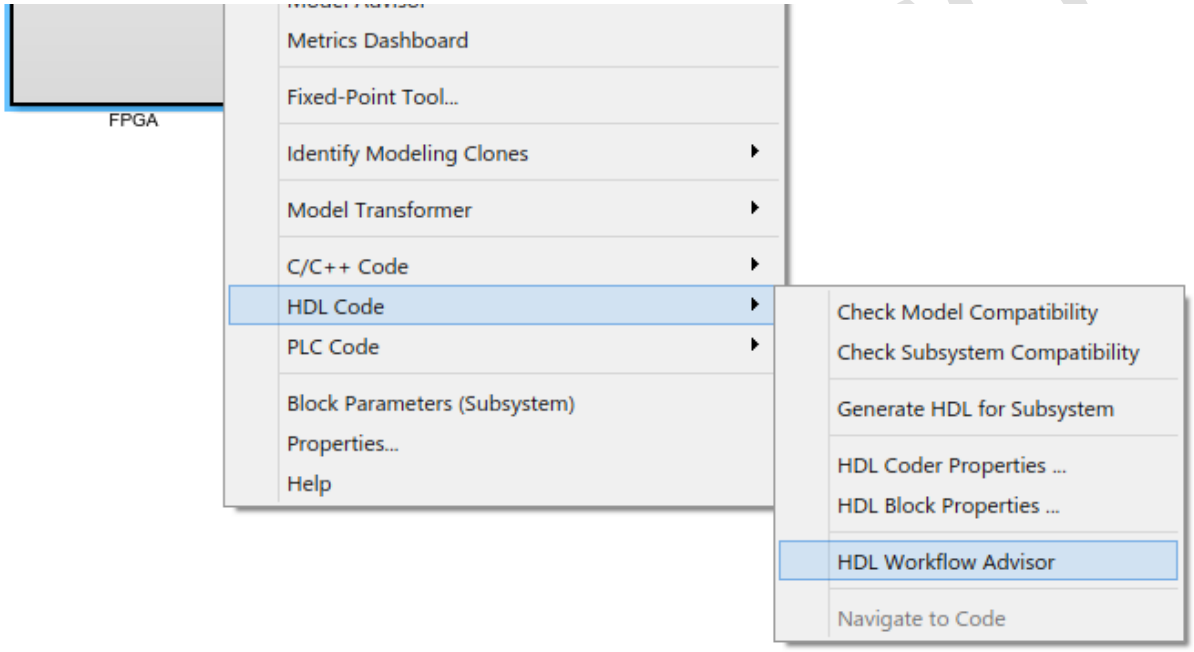


Fig. 6 – Apelarea îndrumătorului pentru generarea automată a codului VHDL [1]

Etapele de execuție pentru procesului de generare automată a codului sunt structurate pe capitole principale precum:
- „Set Target” (configurarea platformei de dezvoltare);
- „Prepare Model for HDL Code Generation” (etapa pregătitoare pentru generarea de cod);
- „HDL Code Generation” (generarea efectivă a codului HDL);
- „Embedded System Integration” (implementarea aplicației la nivelul platformei);

Prin urmare, se vor parcurge etapele respective pentru a genera aplicația executabilă. În cadrul capitolului „Set Target”, în etapa „Set Target Device and Synthesis Tool” se vor selecta următoarele opțiuni:
- Target workflow: IP Core Generation (generare nucleu cu proprietate intelectuală dedicată);
- Target platform: ZedBoard;
- Synthesis tool: Xilinx Vivado;

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

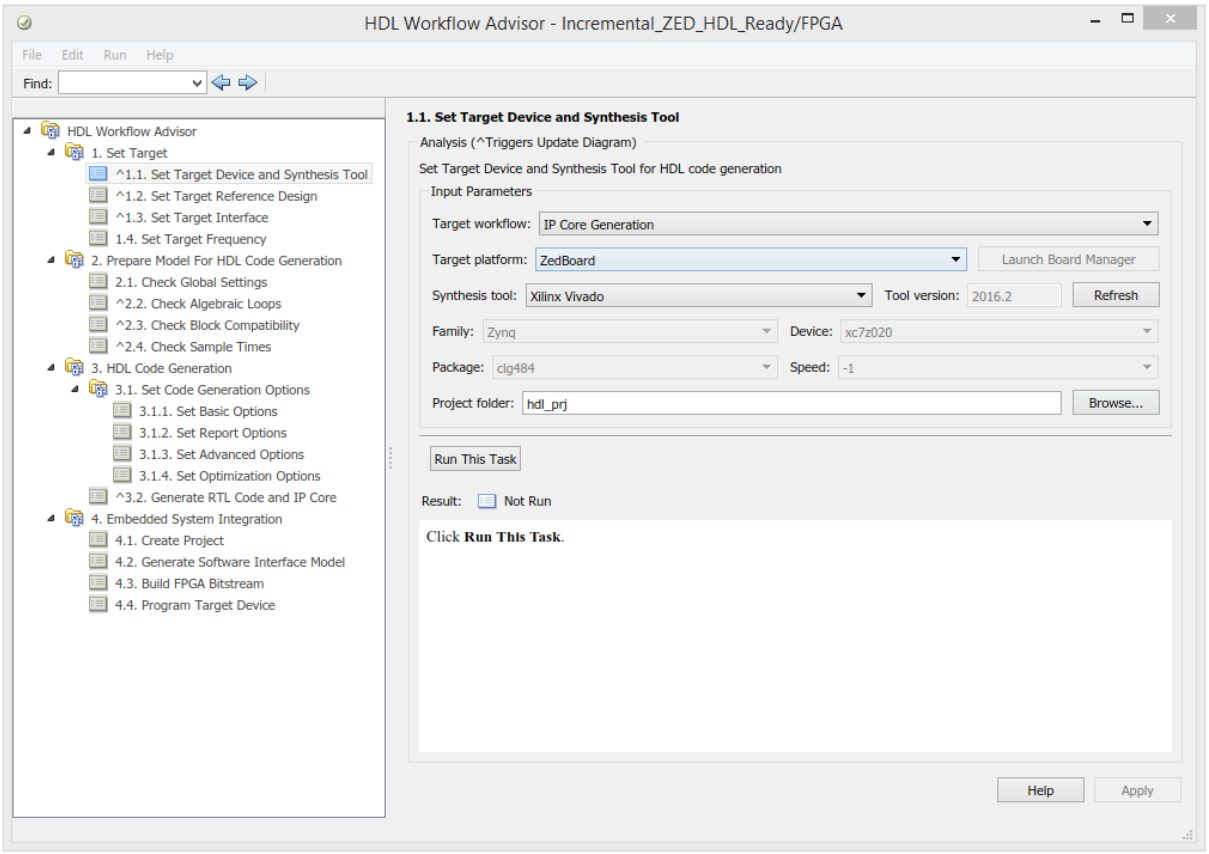


Fig. 7 – HDL Workflow Advisor - capitolul „Set Target”,
etapa „Set Target Device and Synthesis Tool” [1] [3]

În cadrul etapei „Set Target Reference Design”, se vor păstra setările implicite, anume, „Reference design: Default system” (păstrarea valorilor implicite pentru proiectarea arhitecturii interne a platformei cu FPGA).

În cadrul etapei „Set Target Interface” ca și metodă de sincronizare se va alege opțiunea „Processor / FPGA synchronization: Free running”. Tot în cadrul acestei etape, se vor configura elementele periferice fizice (eng. hardware) utilizate în acest proiect, prin intermediul opțiunilor „Target platform interface table”. Se vor utiliza următoarele valori:

- Pulse_in: Push Buttons L-R-U-D-S [0:4];
- Reset_in: Push Buttons L-R-U-D-S [0:4];
- Count_out: LEDs General Purpose [0:7];
- Count_monitor: AXI4-Lite;
- Pulse_monitor: AXI4-Lite;
- RESET_monitor: AXI4-Lite;

În categoria „Set Target Frequency” se vor păstra valorile implicite, anume „50 [MHz]. Pentru a aplica toate modificările din cadrul acestui capitol, se va efectua operația „click – dreapta” asupra titlului capitolului, anume „Set Target” iar din meniul contextual, se va alege opțiunea „Run All”.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

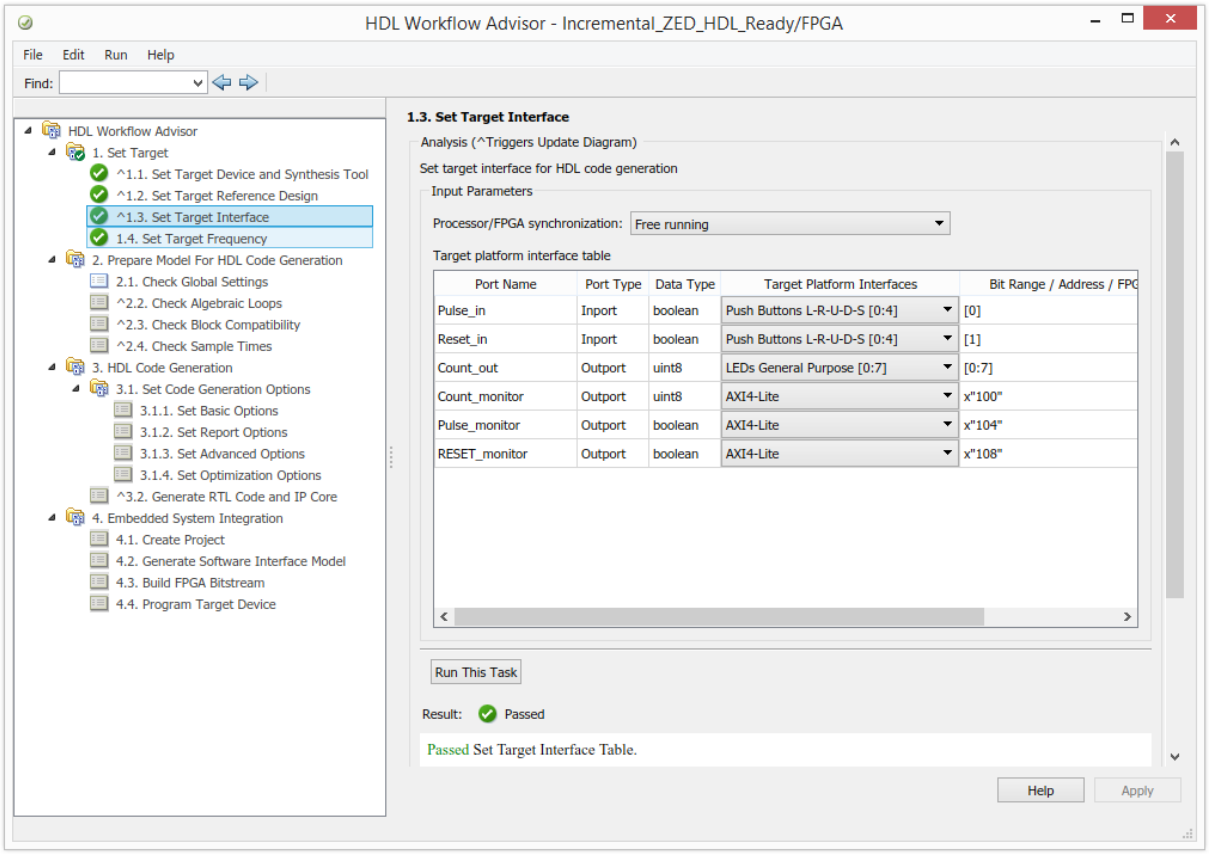


Fig. 8 – HDL Workflow Advisor - capitolul „Set Target”, etapa „Set Target Interfaces” [1] [3]

Pentru capitolul „Prepare Model for HDL Code Generation” se vor păstra valorile implicite recomandate, și se va parcurge etapa de verificare și executare prin selectarea opțiunii „Run All” din cadrul meniului contextual obținut prin operația „click – dreapta”.

Se va proceda în mod similar pentru etapa „HDL Code Generation”. La finalizarea execuției etapei respective, se va obține o fereastră de raport care va descrie tot procesul care a avut loc în cadrul etapei pentru generare a codului – program.

Ultimele etape, din cadrul ultimului capitol, anume „Embedded System Integration”, se vor parcurge fiecare în parte, pe rând, prin selectarea opțiunii „Run This Task” regăsită sub formă de buton în cadrul etapei respective. Se vor parcurge astfel etapele:

- „Create Project” (creeare proiect Vivado);
- „Generate Software Interface Model” (generare model Simulink pentru calculatorul gazdă);
- „Build FPGA Bitstream” (creearea fișierului binar pentru configurarea ariei de porți);
- „Program Target Device” (programarea sau implementarea fișierului binar – executabil);

În etapa de generare a fișierului binar - executabil pentru configurarea ariei de porți, va fi invocat mediul de programare HDL Vivado 2016.2 la nivel de consolă de comandă. Prin intermediul mediului Vivado 2016.2, se va parcurge procesul de sintetizare, generare și implementare a codului HDL generat.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

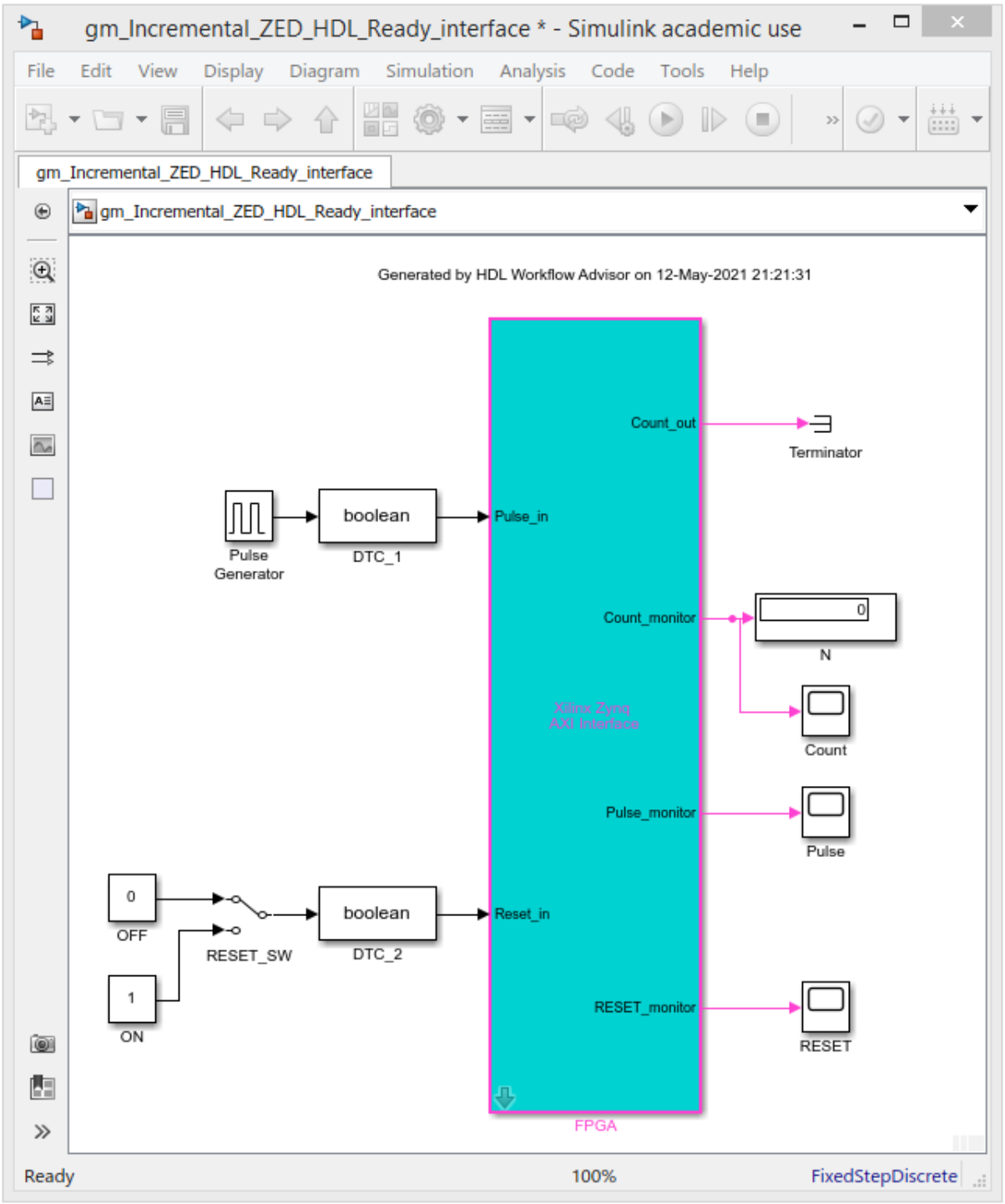


Fig. 9 - Modelul Simulink pentru calculatorul gazdă pentru controlul aplicației în timp real [1]

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

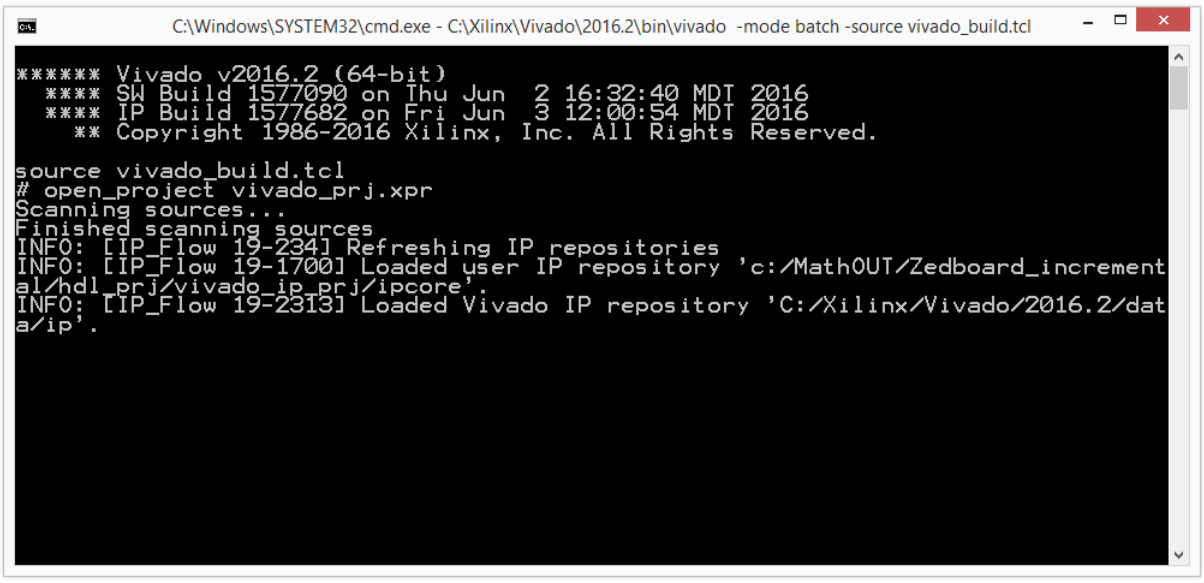


Fig. 10 – Invocarea mediului Vivado 2016.2 la nivel de consolă în vederea generării fișierului binar - executabile pentru configurarea ariei de porți [1] [3]

Operația post – sinteză de programare efectivă (de transferare în memoria platformei a fișierului binar - executabil), se va realiza prin intermediul ultimei etape „Program Target Device”. Se va selecta operația, după care, se va parcurge prin opțiunea „Run This Task”. În urma efectuării tuturor etapelor, îndrumătorul HDL Workflow Advisor va bifa toate etapele.

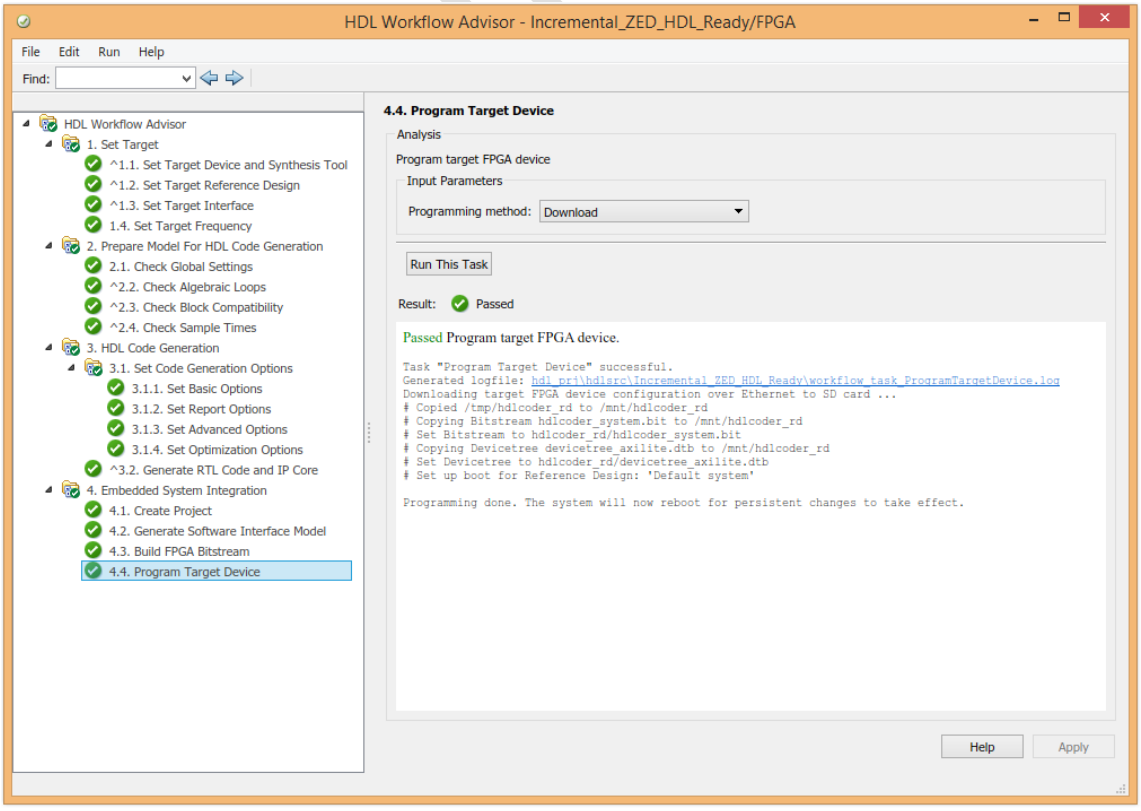


Fig. 11 - Parcurgerea completă a tuturor etapelor îndrumătorului HDL Workflow Advisor [1] [3]

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Pentru a iniția aplicația la nivel de platformă de dezvoltare, este necesară, programarea și a procesorului de aplicație ARM. Acest lucru, se poate realiza, prin intermediul modelului Simulink pentru calcualtorul gazdă, generat anterior. În cadrul modelului respectiv, se va alege timpul total de simulare „inf" (infinit), iar modul de execuție „External" (extern – executare la nivel de platformă de dezvoltare).

În urma efectuării parametrizărilor indicate asupra modelului de control, din cadrul paletei de instrumente se va alege opțiunea „Deploy to Hardware" (generarea și încărcarea codului program la nivelul procesorului de aplicație ARM al platformei ZedBoard).

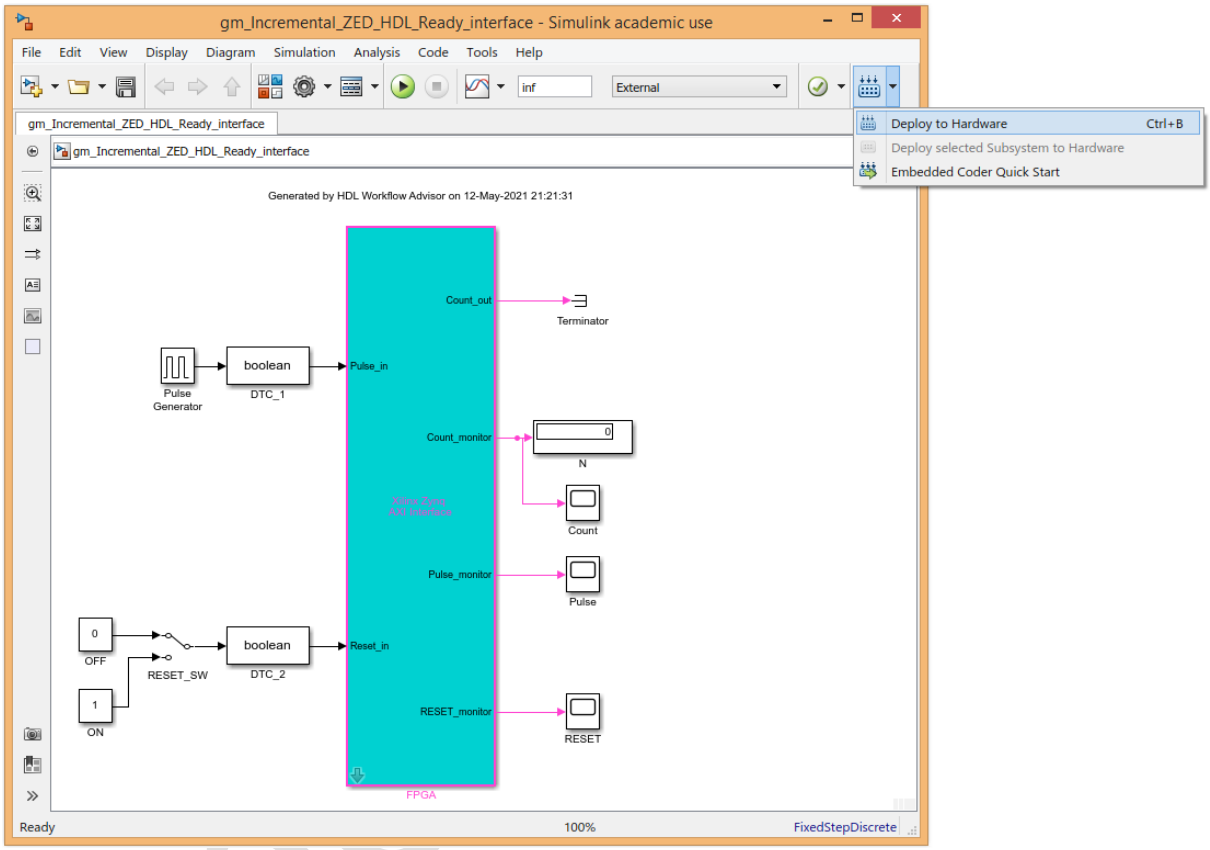


Fig. 12 – Generarea automată a fișierului executabil specific procesorului de aplicație ARM [1]

Pentru a interacționa în timp real cu parametrii programului executat la nivel de platformă de dezvltare se va alege opțiunea „Run" (butonul PLAY verde din barea de instrumente Simulink).

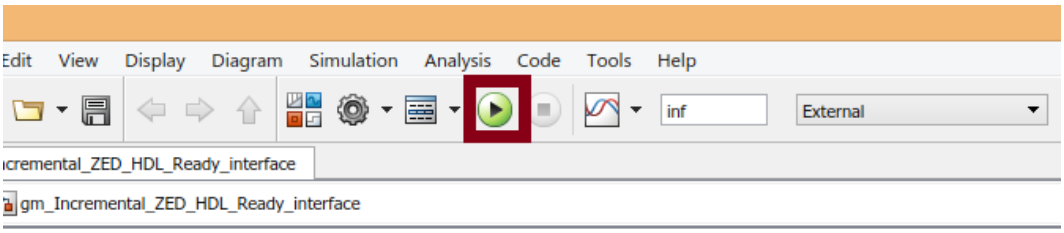


Fig. 13 – Executarea aplicației în timp real în modul Extern (pe procesorul platformei) [1]

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP"
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

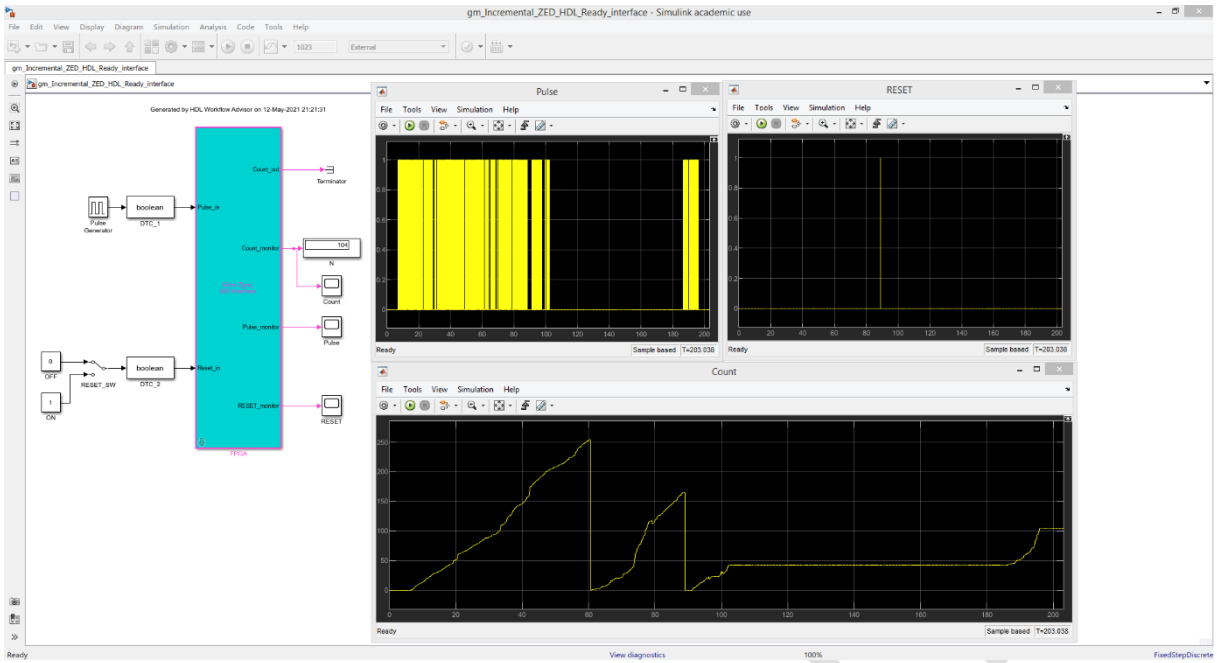


Fig. 14 – Executarea aplicației în timp real

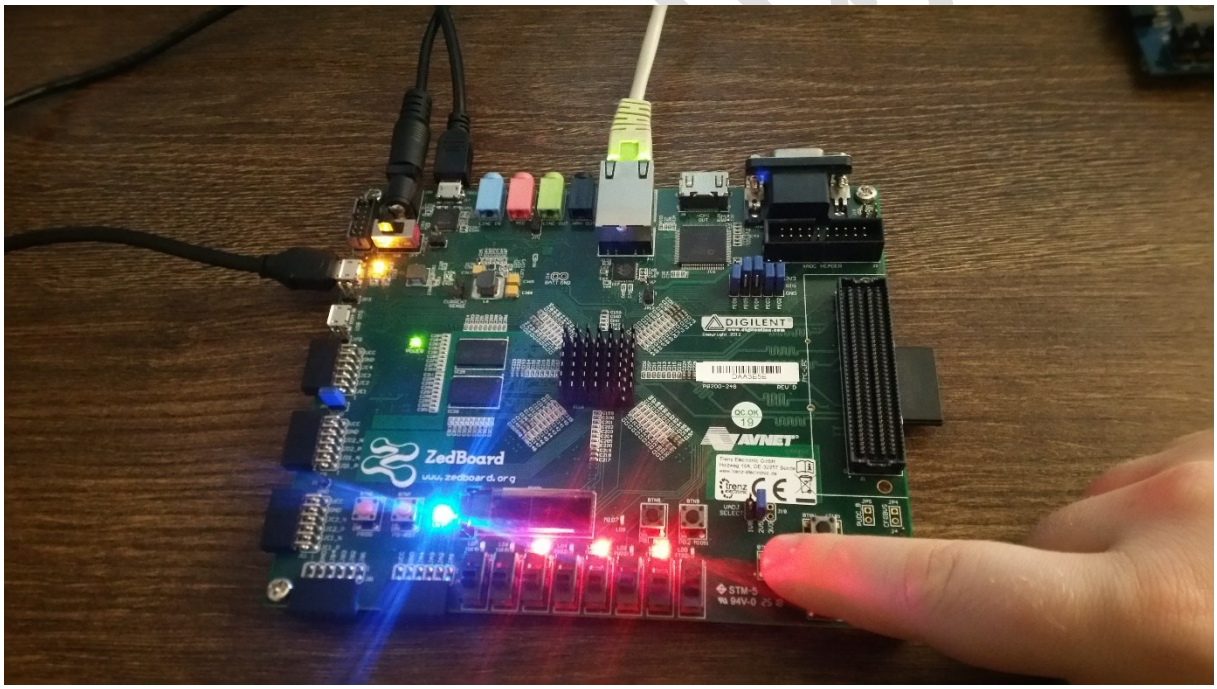


Fig. 15 – Testarea operației de incrementare prin apăsarea repetată a butonului

III. CONCLUZIE:

Generatorul automat de cod HDL Coder, reprezintă o soluție universală, care permite generarea fișierului executabil BitStream pentru configurare a ariei de porți, indiferent de producător. Majoritatea blocurilor din bibleoteca HDL Coder sunt similare cu cele uzuale din Simulink. Astfel, modelul creeat poate fi compatibil cu orice platformă de dezvoltare cu FPGA.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

IV. BIBLIOGRAFIE:

1. MathWorks.com – „Run a Simulink Model on Zynq – series” – Videos and Webinars: (https://www.mathworks.com/videos/run-a-simulink-model-on-zynq-introduction-and-requirements-1-of-4-89508.html);

2. Digilent Inc. – ZedBoard: (https://reference.digilentinc.com/reference/programmable-logic/zedboard/start);

3. MathWorks.com – „Xilinx Zynq Support from MATLAB and Simulink” – Hardware Support: (https://www.mathworks.com/hardware-support/zynq.html);

4. Teodor Crișan Pană – „Sisteme de calcul cu microprocesoare, FPGA și DSP” – Editura UTPRESS, Cluj – Napoca, 2016 – ISBN 978-606-737-206-9;

5. Ioana – Cornelia GROS, Lucian – Nicolae PINTILIE, Teodor Crișan PANĂ – „SISTEME EMBEDDED ÎN INGINERIE ELECTRICĂ - GHID DE APLICAȚII” – Editura UTPress Cluj – Napoca, 2020 ISBN 978-606-737-431-5: (https://biblioteca.utcluj.ro/files/carti-online-cu-coperta/431-5.pdf);

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro