**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**SKAITMENINĖS LOGIKOS PRADMENYS(P175B100)**

***Laboratorinių darbų ataskaita***

Atliko:

IFF-8/11 gr. Studentas

Arnas Švenčionis

2019 m. balandžio 3d.

Priėmė:

dėst. Algirdas Dobrovolskis

**KAUNAS 2019**

[1. Įvadas 2](#_Toc8660425)

[1.1 Darbo tikslas 2](#_Toc8660426)

[1.2 Darbo užduotis 2](#_Toc8660427)

[2. Darbo atlikimas 3](#_Toc8660428)

[2.1 M1 skaitiklis; 3](#_Toc8660429)

[2.2 M1 ir M2 skaitiklių suma, skaičiuojanti iki JM1 modulio; 5](#_Toc8660430)

[2.3 M1, M2 ir M3 skaitiklių suma, skaičiuojanti iki JM2 modulio; 7](#_Toc8660431)

[3. Išvados 12](#_Toc8660432)

# 1. Įvadas

## 1.1 Darbo tikslas

Susipažinti su aparatūros aprašymo kalbomis ir jų galimybėmis. Aprašyti kombinacinę ir nuoseklios logikos schemą VHDL kalba, atlikti schemos sintezę ir išanalizuoti sintezės rezultatus.

## 1.2 Darbo užduotis

Laboratorinis darbas yra skirtas susipažinti su skaitikliklio realizacija naudojantis VHDL kodu.

1. VHDL kodu realizuoti M1 modulio skaitiklį. Sudaryti testinius rinkinius ir patikrinti schemos veikimą.
2. VHDL kodu realizuoti apjungtą skaitiklį susidedantį iš M1 ir M2 modulių skaitiklių, skaičiuojanti iki JM1 modulio. Sudaryti testinius rinkinius ir patikrinti sistemos veikimą.
3. VHDL kodu realizuoti apjungtą skaitiklį susidedantį iš M1, M2 ir M3 modulių skaitiklių, skaičiuojanti iki JM2 modulio. Sudaryti testinius rinkinius ir patikrinti sistemos veikimą.
4. Paruošti laboratorinio darbo ataskaitą. Ataskaitoje pateikti suprojektuotas VHDL kodą, Synplify-Pro sugeneruotą schemą, ir sistemų schemų funkcionavimo laiko diagramas.

146 variantas:

M1 = 12;

M2 = 31;

M3 = 50;

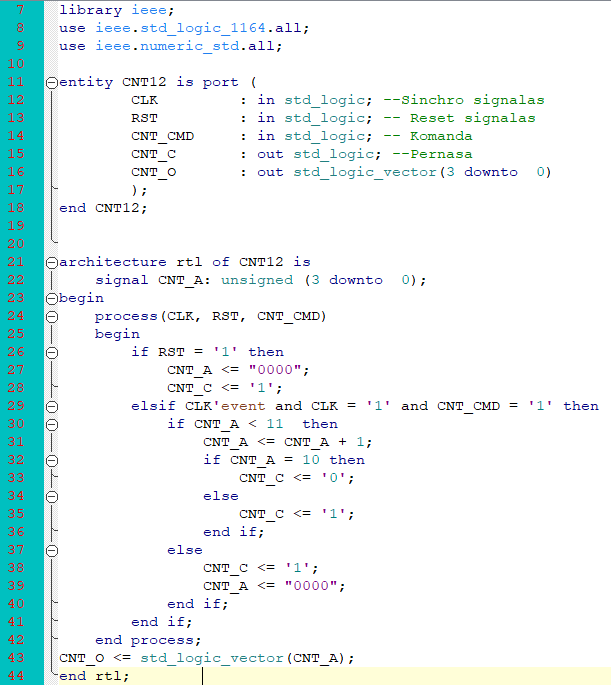
JM1 = 74;

JM2 = 3720;

# 2. Darbo atlikimas

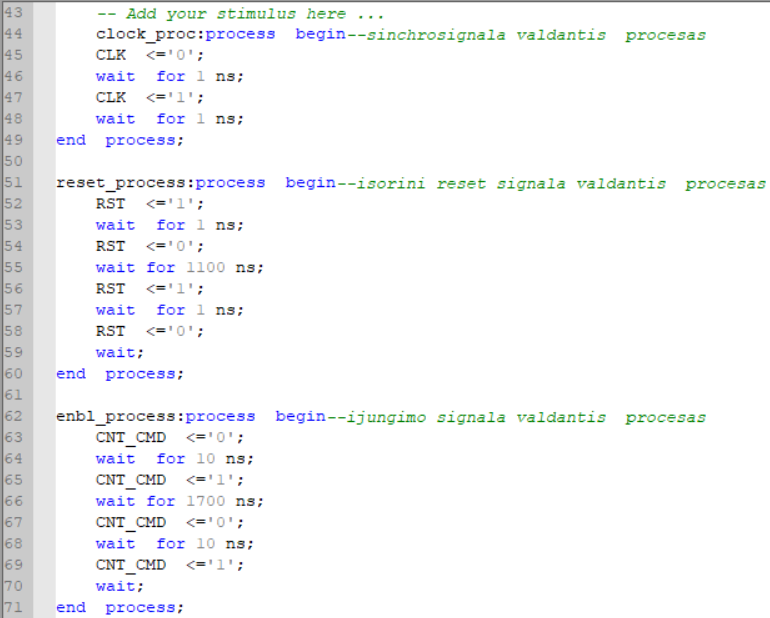
## 2.1 M1 skaitiklis;

Užduotyje M1 reikšmę gavau 12. Tai reiškia skaitiklis turi eiti nuo 0 iki 11 (12 vienetų). Pakoregavęs moodle pateiktą kodą, gavau jį tokį:



1pav. M1 kodas

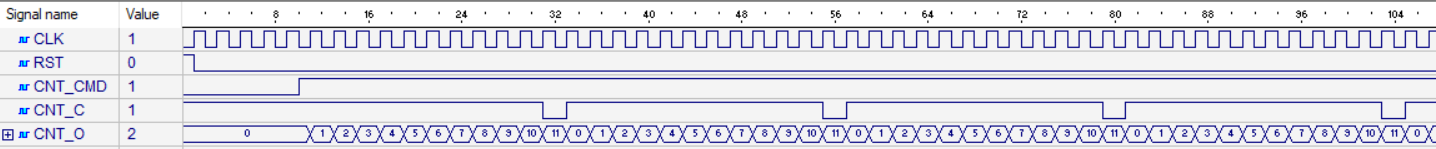
Kadangi maksimalus skaicius yra 12, o 12 dvejetainėje sistemoje yra 1100, tai bitų skaičius yra 4. Kol skaitliuko reikšmė yra mažesnė už 11, prie jo yra pridedamas vienetas. Kai jis pasiekia 10, CNT\_C (pernašos) reikšmė yra pakeičiama į nulį.



2pav. Kodo testavimui naudotas stimulas

Stimulas sutampa su

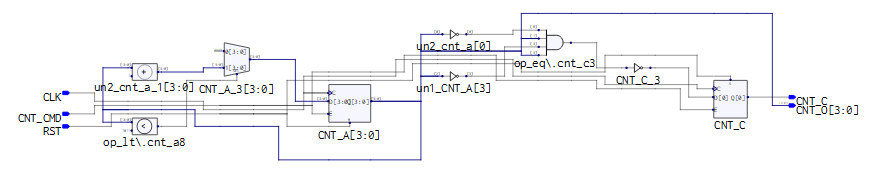
**M1 skaitiklio testavimas;**



3pav. M1 kodo testavimas

Kaip matome 3pav. laiko diagramoje, kai skaitiklis pasiekia 11 reiksšmę, pernašos signalas nusileidžia iš 1 į 0 ir yra pradedama skaičiuoti iš naujo.

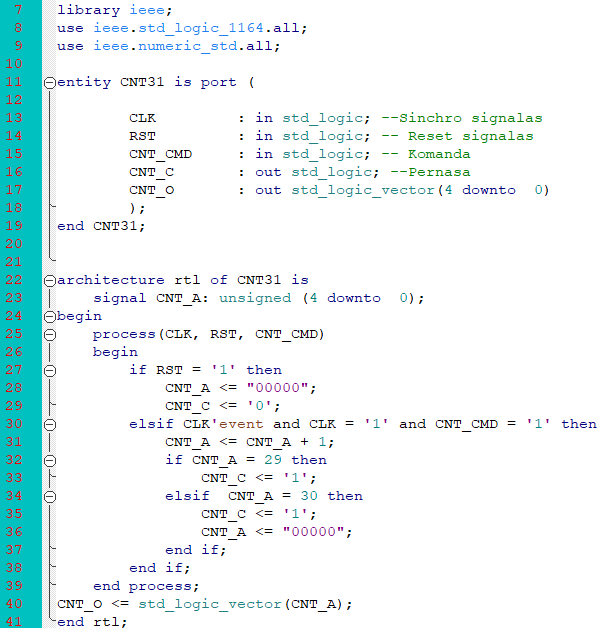
**Sudaryta schema;**



3pav. Synplify pro for Lattice sugeneruota M1 schema

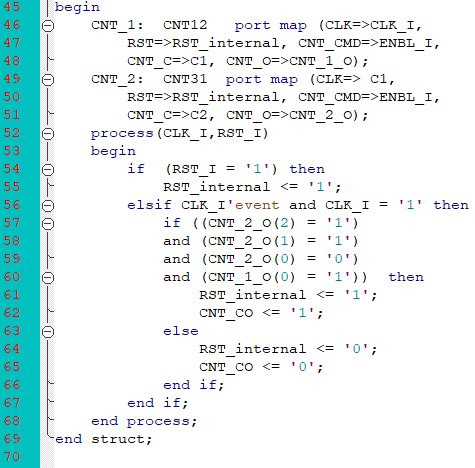
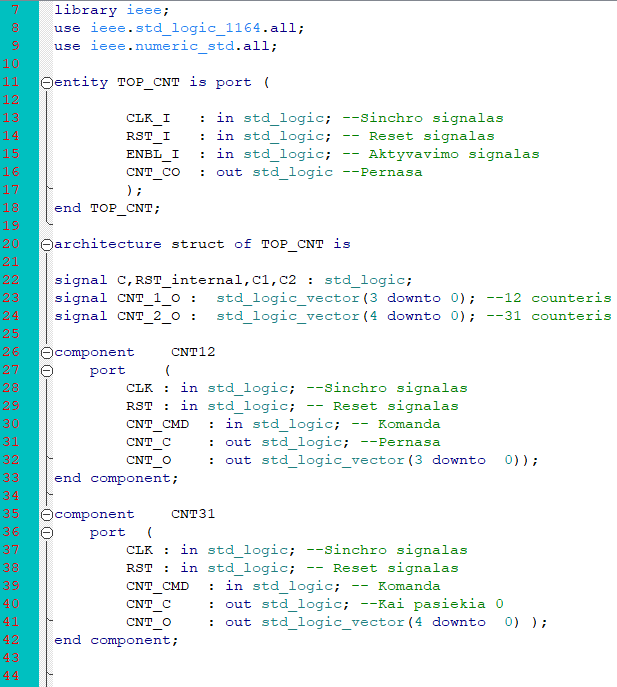
## 2.2 M1 ir M2 skaitiklių suma, skaičiuojanti iki JM1 modulio;

M1 = 12, M2 = 31. M1 kodas yra toks pats kaip ir 2.1 punkte 1 pav. Tačiau M2 kodas šiek tiek skiriasi. M2 skaitiklis skaičiuoja kiek kartų buvo paduota pernaša iš M1 skaitiklio. Tai reikškia M2 skaičiuoja kiek kartų M1 pasiekė savo maksimalią reikšmę.



4pav. M2 kodas

Panašiai kaip ir M1 kode, skaitikliui pasiekus 30 reikšmę, pernaša CNT\_C keičama iš 0 į vienetą.



5pav. JM1 kodas.

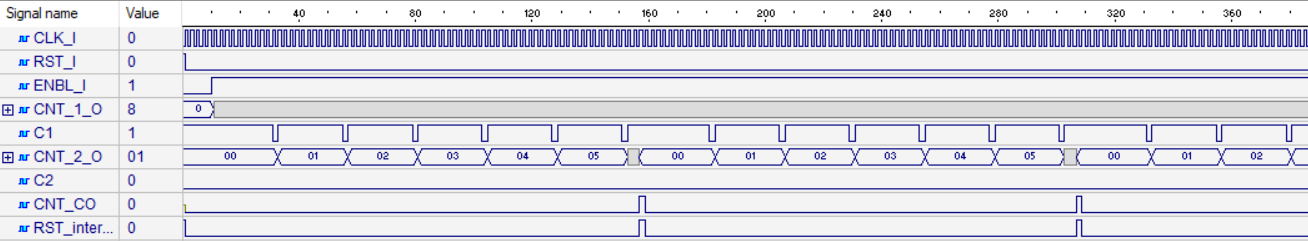
Sąlygoje JM1 reikšmė yra 74. Tai yra reikšmė, kurią pasiekus sistema turi būti restartuojama, pradedama skaičiuoti iš naujo. Po kiek pernašų kiekviename skaitiklyje tai atsitiks galime paskaičiuoti nesudėtingai.

Žinome, kad M1 sukasi kas 12 vienetų. Tai galime paskaičiuoti kiek sveikų ciklų:

74 = x \* 12; x = 74 / 12 = 6.17; Reiškia M2 reikšmė turės būti 6;

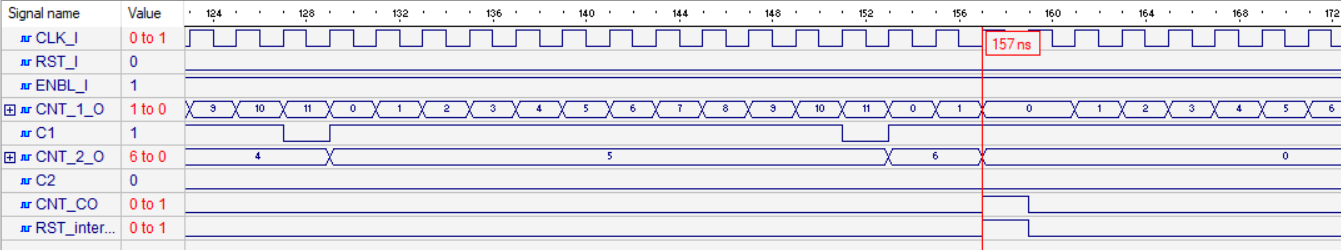
Dabar paskaičiuojame kiek kartų turės padidėti M1 reikšmė:

6 \* 12 = 72; 74 – 72 = 2; Tai reiškia CNT\_CO signalas turi pakilti kai M2 bus lygu šešiems, o M1 – vienam.



6pav. JM1 laiko schema

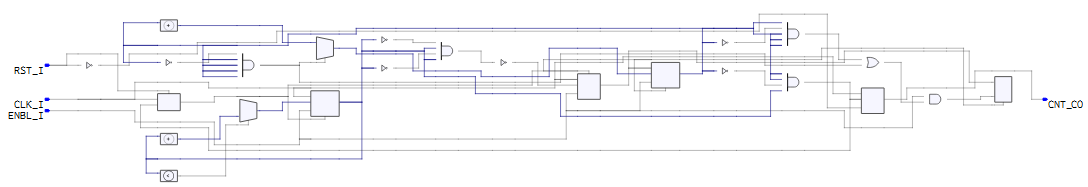
6pav. matome, kad CNT\_2\_O (M2 skaitiklis) pliusuojasi kas vieną M1 skaitiklio pernašą.



7pav. Priartinta JM1 laiko schema

Priartinus schemą (7pav) galime matyti, kad M1 skaitiklis dirba kaip ir turėtų (eina iki 12, pakeičia pernašos signalą ir resetinasi) Čia taip pat matome, kad M2 pasiekus 6 ir M1 pasiekus 1 CNT\_CO pakyla, resetina visa sistemą. Kodas veikia.

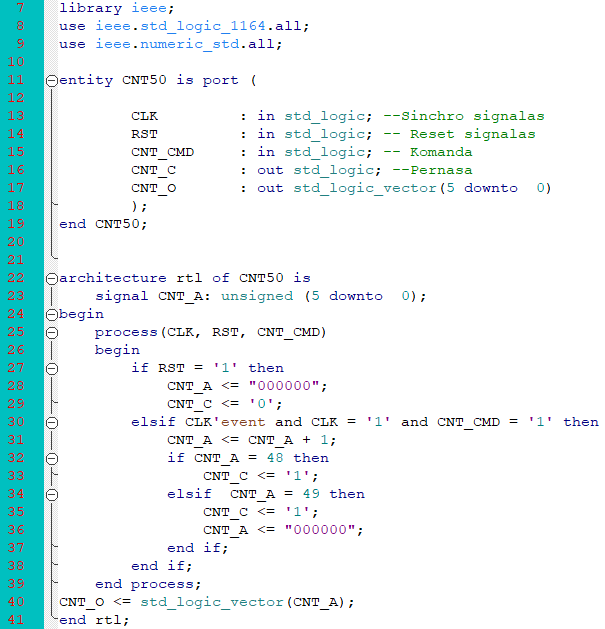
**Sudaryta schema;**



8pav. Synplify pro for Lattice sugeneruota M1 schema

## 2.3 M1, M2 ir M3 skaitiklių suma, skaičiuojanti iki JM2 modulio;

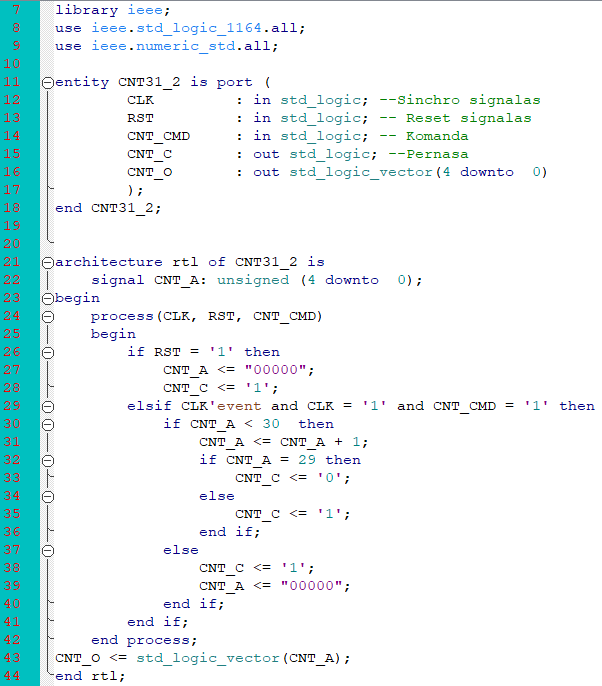
Trečia užduotis reikalauja trečio skaitiklio. Užduotyje jo maksimali reikšmė – 50. Kaip anksčiau M2 skaitiklis skaičiavo M1 skaitiklio pernašų skaičių, taip ir M3 skaitiklis skaičiuos M2 pernašų skaičių. Reikia pasiekti JM2 modulį, duotą sąlygoje (3720).Trečiam skaitikliui reikėjo parašyti atskirą kodą:



9pav. M3 skaitiklio kodas

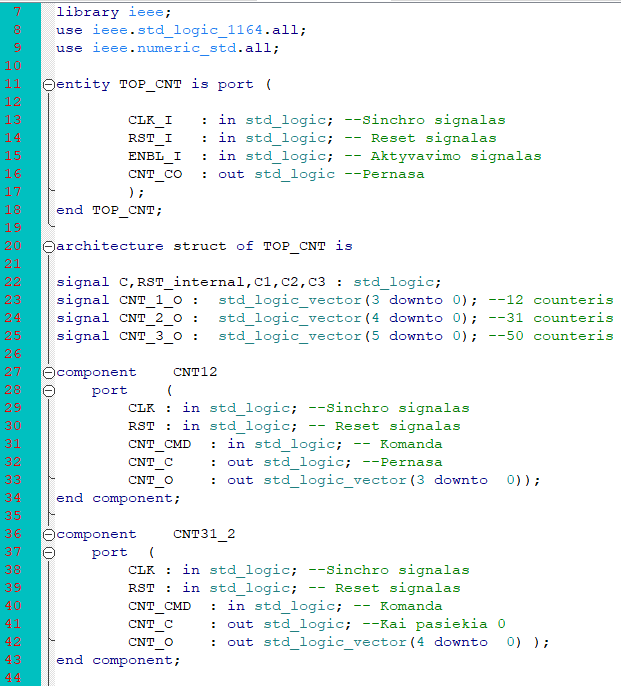
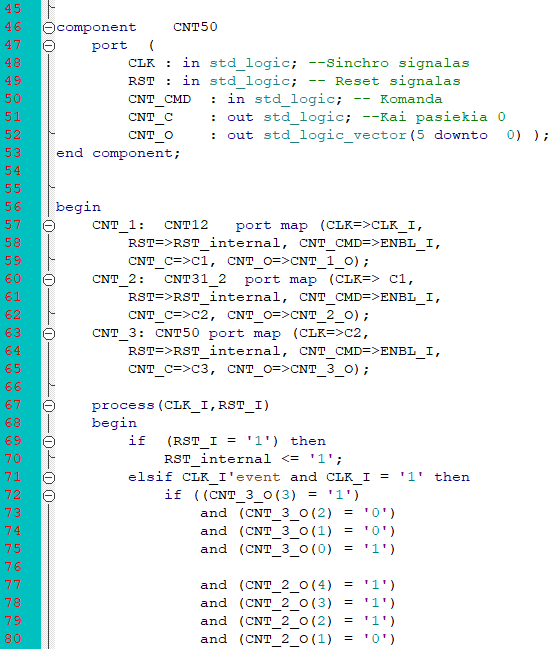
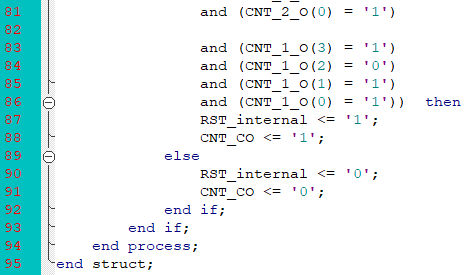
Jis yra panašus į anos schemos M2 kodui, tik pamodifikuotas pagal savo reikšmę ir bitų skaičių.

Tačiau, kad kodas veiktų pamodifikuoti reikia M2 skaitiklio kodą:



10pav. M2 modifikuotas kodas

Jis yra panašus į M1 kodą. Pakeistas bitų skaičius ir maksimali reikšmė.

11pav. JM2 kodas

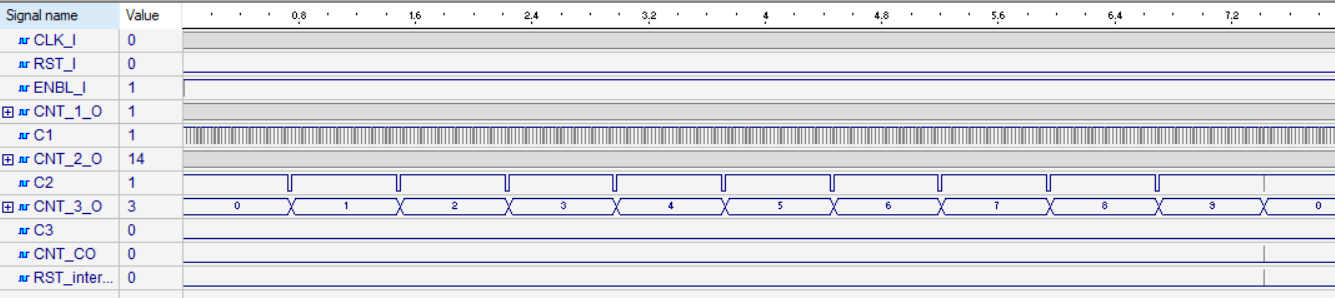
Kaip ir praeitame punkte, reikia paskičiuoti kiek M2 ir M1 pernašų turi skaičiuoti skaitiklis M3 prieš pasiekiant sąlygoje duotą JM2(3720) ir pakeliant CNT\_CO, Resetą. Paskaičiuojame pasinaudoję formulėmis.

3720 = 12 \* 31 \* M3 + M1kiek = 372 \* M3+ M1kiek;

Visų pirma paskaičiuojame kiek sveikų kartų M3 padidės:

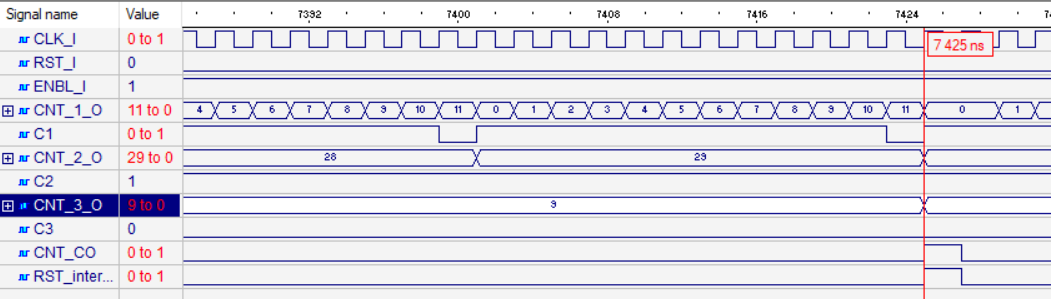
3720 = 372 \* M3; M3 = 3720 / 372 = 10; Gavome sveiką skaičių. Tai reiškia, kad M1kiek = 0;

CNT\_CO turės pakilti M3 pasiekus 10 reikšmę.



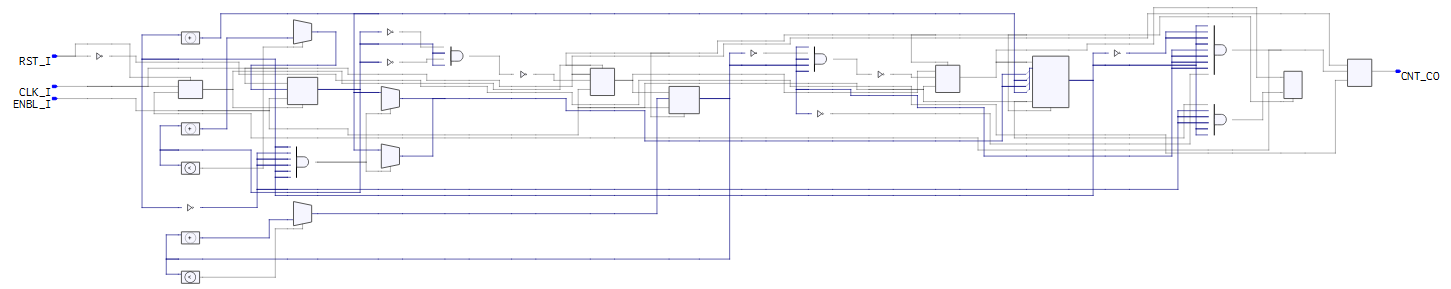
12pav JM2 laiko funkcija

Šioje schemoje matome, kad CNT\_CO pakyla CNT\_3\_O (M3 skaitikliui) pasiekus 0.



13pav. Priartinta JM2 laiko funkcija

Priartinus (13 pav) Aiškiau matome, kad veikia kaip turėtų M1 skaitiklis (eina nuo 0 iki 11, pasikeičia jo pernaša, jis grįžta į 0), M2 skaitiklis eina nuo 0 iki 30, pasikeičia jo pernaša, jis grįžta į 0) ir M3 skaitiklis. Jis kyla kai M2 skaitiklis paduoda pernašą.



14pav. Synplify pro for Lattice sugeneruota JM2 schema

# 3. Išvados

Sužinojau VHDL kodavimo pradmenis, kam gali būti panaudotas kodas, kaip jis gali palengvinti darbą. Sužinojau kam naudojami skaitikliai, kaip jie veikia. Susipažinau su Synplify pro galimybėmis, išmokau sugeneruoti schemą panaudojus VHDL kodą.