БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ
ИНСТИТУТ ПО ИНФОРМАЦИОННИ И КОМУНИКАЦИОННИ ТЕХНОЛОГИИ

Владимир Николаев Иванов

РАЗРАБОТВАНЕ НА ПРОГРАМНИ СРЕДСТВА ЗА МОДЕЛИРАНЕ НА МНОГОФУНКЦИОНАЛНИ ЕЛЕКТРОННИ СХЕМИ

ДИСЕРТАЦИЯ

За получаване на образователна и научна степен

„ДОКТОР”

По направление:
5.3 Комуникационна и компютърна техника
Научна специалност : 02.07.20. Комуникационни мрежи и системи

Научен ръководител:

проф. Тодор Стоилов

София
2015

Съдържание

Списък на използвани съкращения и означения

AES - *Advanced Encryption Standard* – стандарт за криптиране на данни.
AHB - *Advanced high-performance bus*-високо скоростна микропроцесорна магистрала
ALU - Аритметично Логическо Устройство
AMBA - *Advanced Microprocessor Bus Architecture* – микропроцесорна магистрала с подобрена архитектура.
ARTEMIS - *Advanced Research and Technology for Embedded Intelligence and Systems* – название на европейска програма за вградени системи
ASMBL - *Advanced Silicon Modular Block* – модул с подобрена апаратна архитектура
AVB - *Audio Video Bridging* - аудио видео мост
АЦП - Аналогово-цифров преобразувател
barrel shifter - схема за едновременно преместване на съдържанието на регистър на определен брой позиции за един такт.
backhaul - *hierarchical telecommunications network* – йерархична телекомуникационна мрежа
BRAM - блокова памет в FPGA прибори.
ВС - Вградена система
CAN - *controller area network* – локална мрежа между управляващи устройства
COM - *Communication port* – изход за обмен на данни
CLB - конфигурируеми логически блокове
CLKDV - програмируем делител на честоти
CPLD - *complex programmable logic device* – съвременен препрограмируем прибор
DCM - *Digital Clock Manager* – блокове за обработка на тактови последователности
DCS - *Device Control String* –командна последователност от данни
DSP - *Digital signal processor* – процесор за цифрова обработка на сигнали
EDA - *Electronic Design Automation* – автоматизиране на процеса на проектиране на електронни устройства
EMI - *electromagnetic interference* – електромагнитни смущения
ESL -*Electronic System Level* – инициатива на Xilinx за автоматизация на инженерния труд
FPGA -*Field Programmable Gate Array* – съвременен препрограмируем прибор с голям логически капацитет
FIFO- *First In, First Out* – регистър за съхранение на данни
FIT -*Failures In Time* – брой грешки за определено време
FTDI- *Future Technology Devices International* – фирма за производсво на интегрални схеми
HDL – *High Description Language* – Език за описание от високо ниво
HW/SW co-design – *hard Ware/Soft Ware co design* – едновременно проектиране на апаратни и програмни средства
I/O - *input/output* - входно изходни портове
IP - *intellectual properties* – интелектуална собственост
LMB -*Local Memory Bus* – магистрала за обмен на данни при вградени процесори
LogiCore - система за генериране на логически елементи
LUT - таблица на истинност
LVDS, GTL, HSTL - стандарти за предаване на данни

MAC – *Multiply Accumulate Cell* – клетка изпълняваща операцията умножение с натрупване

MMU - *memory management unit* – устройство за управлението на памет.

MicroBlaze - 32 bit.-програмен процесор предлаган от Xilinx.

МПО - малки предметни области

ModelSim- програмна среда за симулация на хардуер

MILCOM - *Military Communications* – военни комуникации

MSPS – *Mega Sampels Per Second* - величина указваща бързодействие на аналого-цифров преобразувател

OPB - *On chip Peripheral Bus* – магистрала за обмен на данни между вградени периферни устройства

ОЗУ - Оперативно запомнящо устройство

OrCad - система за проектиране на електрони схеми и печатни платки

OTN - *Optical Transport Network* – оптична мрежа за предаване на данни

Overbuilding – несанкционирано копиране на проект

SoC - *System on Chip* – Система реализирана в рамките на един чип

pBlazIDE - програмна среда за проверка и настройка на асемблерската програма на микропроцесора picoBlaze

PCI Express- *Peripheral Component Interconnect Express* – стандарт за обмен на данни.

PicoBlaze.- 8 bit.-програмен процесор предлаган от Xilinx

PicoTerm - комуникационна програма

PLB -*Processor Local Bus* –локална за даден процесор магистрала

Port Map - оператор задаващ връзките в един проект

Reverse engineering – е процес на извличане на данни и информация от даден проект с цел не позволено копиране.

RISC- *Reduced instruction set computing* – процесор с ограничен набор от команди

Verilog - език от високо ниво за описание на електрони схеми.

Scratchpad RAM - малка оперативна памет при вградени процесори

slices – съставна част на конфигурируемия логически блок

SLISEM - част от конфигуриуем логически блок *(CLB)*

Spartan , **Virtex** - семейства FPGA прибори предлагани от Xilinx

SPI- *Serial Peripheral Interface* – стандарт за обмен на данни

UART - *universal asynchronous receiver/transmitter* – асинхронно устройство за обмен на данни по сериен канал

UCF - *user constraint file* – файл задаващ физически ограничения към проект

USB- *Universal Serial Bus* – стандарт за обмен на данни

VHDL- език от високо ниво за описание на електрони схеми

Увод

Цел на дисертационния труд е разработката на програмна среда за автоматизиране на инженерния труд при проектиране на вградени системи базирани на процесорно ядро PicoBlaze.

Актуалност на темата.

По своята същност понятието многофункционалност указва за възможността един обект да изпълнява няколко различни функции [1]. Многофункционалността позволява да се подобрят потребителските свойства, да се намали масата, заемания обем, бройя съставящи елементи, връзки, да се повиши надеждността и безопасността.

Доминираща роля по отношение на многофункционалността при цифровите електронни схеми имат съвременните препрограмируеми схеми от типа FPGA. Те позволяват практическо реализиране на паралелни изчислителни алгоритми. Успешно се използват като среда за изграждане на различни по сложност многофункционални устройства и системи на базата на „вградени" микропроцесорни ядра. Вградените микропроцесорни ядра имат възможност за ре-конфигурация и развой, което ги определя като основни градивни блокове на всички настоящи и бъдещи високо технологични, информационно - управляващи системи и устройства.

Процесът на проектиране на вградени системи, базирани на програмни микропроцесорни ядра, изисква детайлно познаване на структурата и особеностите на използваните FPGA прибори и средите за работа с тях, като отчита различните скорости на тяхното развитие. На практика, средите за работа с FPGA прибори не могат да следват достатъчно бързо темповете на тяхното развитие. Това силно затруднява практическата реализация на вградените системи и налага

разработването на методи, използващи абстракции с по-висока степен, които подобряват показателите на средите за работа с FPGA прибори и ускоряват инженерния труд при проектиране на многофункционални електронни схеми с използването на вградени микропроцесорни ядра. Тяхното въвеждане минимизира влиянието на индивидуалните дадености на проектанта върху процеса на проектиране и открива допълнителни възможности за неговото автоматизиране.

Обзор на основните резултати в областта

Обща особеност на съвременната методология за проектиране и синтез на електронни схеми с повишена многофункционалност е използването на HDL-базирани средства. По мнението на експерти, в рамките на близките 5-10 години тази методология за проектиране ще се превърне в задължителна за всички проектанти на електронни схеми. Тя подобрява качеството и намалява продължителността на процеса на проектиране. Открива възможност при създаването на ново електронно устройство да се използват предварително разработени компоненти (IP), в това число и такива, съдържащи в себе си апаратни или програмно изграждани процесори.

Основните тенденции при средствата за разработка на програмно осигуряване се свързват с активното внедряване на езици за програмиране от високо ниво (C, C++) и метаезици, използвани съответно за създаване на компилатори и описание на поставената задача [142].

Проблем при проектирането на вградени системи е тяхната верификация [119], заемаща до 80% от времето за изготвяне на проекта. Минимизирането на тази стойност изисква средства, които използват абстракции от високо ниво и унифицирани HW и SW

компоненти и осигуряват интеграция с програмното осигуряване, работещо на ниските нива от цикълът на проектиране на ВС.

Тези изисквания имат място в системите за развой на фирмите предлагащи FPGA прибори (Mico System Builder на Lattice [33], Quartus II [32], на Altera и ISE, WebPack и Embedded Development Kit на Xilinx). Това са среди, позволяващи интеграция с инструментариуми на водещи EDA производители (Mentor Graphics, Synplicity MathWorks, System Generator, Accel и др).

Те позволяват бързо придвижване от идея до реализация. Предназначени са за работа на екип и са много скъпи.

За Xilinx решението на проблемите в проектирането и разработката на нови средства за проектиране се свежда до установяване на партньорство със заинтересовани от нововъведения производители EDA в рамките на специална програма. Тази програма се нарича "ESL инициатива"(Electronic System Level, ESL). Нейната основна цел се свежда до разработка на нови програмни средства от системно ниво, които да направят методологията за проектиране максимално "близка" за проектанта. Слаба страна на инициативата ESL представлява нейната ориентация към проекти, изискващи значителни апаратни ресурси. Това поставя под въпрос нейната ефективност при реализиране на проекти с малки вградени процесори [62,63]. На тази основа реализирането на програмна среда за автоматизирано проектиране на вградени, PicoBlaze базирани системи, посредством представянето им във форма с повишена степен на абстрактност, има своето място. Тя позволява да се постигне автоматизация на процеса на инженерния труд и ускорява реализирането на разработки, използващи това микропроцесорно ядро.

Задачи на дисертацията

За постигането на целта на дисертациония труд са формулирани следните задачи :

1. Да се анализира и обоснове необходимоста от използване на вградените процесори като алтернатива на конвенционалните и специализирани микропроцесори и прилагането им като основни градивни блокове на високо технологични, информационни управляващи системи и устройства.
2. Да се разработи програмна среда за автоматизиране на инженерния труд при проектиране на системи с вградени процесори.
3. Да се проектират технически средства с използване на разработената среда.
4. Да се разработи алгоритъм за оценка на ефективността на програмната среда за автоматизиране на инженерния труд при проектиране на системи с вградени процесори.

Методология на изследването

По своята същност, процесът на проектирането на завършени, апаратно-програмни вградени системи, използващи като среда за реализиране FPGA чип представлява многопланова задача. Тя изисква използването на комплексен подход, който да обедини традиционите апаратни и програмни съставящи на системата в набор от архитектурни абстракции, описващи инфраструктурата на разработваната система. Комплексният подход позволява използването на готови апаратно програмни платформи, унифицира създаването на HW и SW компоненти и канализира проектирането на вградени системи в следните направления :

- развитие на технологиите за HW/SW co-design;
- повторно използване на компоненти;
- проверка за достоверност и верификация;
- създаване на средства за моделиране на нефункционални свойства (надеждност, консумирана мощност, габарити и др.).

Тези направления показват необходимоста от проектиране с използване на независими спрямо елементната база описания от високо ниво, които адекватно отразяват функционалността на системата.

В качеството на основа за изграждане на средата за автоматизирано проектиране на PicoBlaze базирани системи е използвана създадената в ИППИ-РАН теория за малки предметни области (МПО) [67,68]. Съгласно тази теория, решението на произволна задача се представя в абстрактно пространство, формирано от МПО [67].

Това представяне позволява използването на технологии за HW/SW co-design, допуска извършване, позволява проверки за достоверност и верификация, както и повторно използване на предварително създаден и проверен код. Така съществуващи проекти могат да се разширяват или модифицират без повторно моделиране. Елиминират се всички, незабележими в началните етапи на проектирането неопределености, чиито последствия се откриват едва в последните стадии на процеса. По този начин се постига минимизиране на броя на циклите възникващи в процеса на проектиране, които напрактика определят неговата продължителност.

Структура на дисертационния труд

Дисертационният труд се състои от увод и четири глави. Дисертацията съдържа 146 страници, 64 фигури, 17 таблици, 136 цитирани източника и приложения.

В първа глава е извършен анализ на процеса на проектиране на многофункционални електронни схеми. Представена е същността на методите за проектиране. Представена е необходимостта от разглеждането на многофункционалността при анализа и проектирането на електронни схеми с използването на елементи с повишена многофункционалност от типа ASIC, CPLD и FPGA. Сравнени са основните системни показатели на FPGA приборите на фирмите Xilinx и Altera. Показано е, че FPGA приборите на фирмата Xilinx се очертават като основа за най-новото поколение многофункционални електронни схеми позволяващи реализирането на вградени процесори и системи. Разгледан е високо производителният 8 битов RISC процесор PicoBlaze, който по най-ефективен начин използва вътрешната структура на FPGA прибори. Също така са разгледани развойните средства за съставяне, моделиране и верификация на електронни схеми с повишена многофункционалност. Показани са основни етапи на съвременната методология за проектиране и синтез на електронни схеми с повишена многофункционалност използващи HDL-базирани средства. Разгледани са основните тенденции при средствата за разработка на програмно осигуряване за автоматизираното проектиране на електрони устройства. Обърнато е внимание на използването на езици за програмиране от високо ниво (C, C++) и създаването на метаезици,

позволяващи описване на поставената задача и генериране на необходимото програмно осигуряване.

В Глава 2 са представени теоретичната основа и практическите аспекти на разработването на програмна среда за синтез на PicoBlaze базирани устройства. Представен е класическият алгоритъм за разработка на вградени процесори. Изяснено е значението и ролята на предварителната обработка на изискванията и особеностите на реализирания проект. Показана е необходимоста от проектиране с използване на независими спрямо елементната база абстракции от високо ниво.

Глава 3 представя реализирането на инженерни приложения с разработената среда. Показани са процесите на разработка и генериране на МПО модули и управляващата програма на процесора. Дадено е описание на разработената специализирана апаратна среда за проверка и настройка на апаратното и програмното осигуряване на вградени PicoBlase базирани системи. Представени са практически разработки на вградени системи за светлинни ефекти в домове на бъдещето и генератор на случайни числа GENAP, реализирани на базата на процесора PicoBlaze. Направено е сравнение и оценка на разработените МПО модули. Представени са структурата на разработената програмна среда, резултатите от направения тест за бързодействие и ръководство за работа със средата.

Глава 1 Анализ на процеса на проектиране на многофункционални електронни схеми

Процесът на съставяне на описание, необходимо за създаване на още несъществуващ обект по неговото първично описание и алгоритъм на функциониране, чрез тяхното преобразуване, оптимизация, отстраняване на некоректности и последователно представяне в рамките на един език се нарича проектиране. Продуктът на процеса проектиране е резултат от изпълнението на комплекс от изследователски, разчетни, конструкторски и описателни работи, по които този обект може да се изработи.

Съществуват различни методи за проектиране. Традиционният метод за проектиране се базира на чертежи. Той възниква на стадия на машинното производство. При него проектирането се свежда до създаване на чертежи на обекта в определен мащаб. Характерното за метода е, че във всеки един момент той разглежда само една концепция на обекта. Този подход дава добри резултати на ниво изделия и техните части.

Съвременните методи за проектиране позволяват да се разглеждат множество концепции на обекта. Това се постига чрез разширяване на пространството на решения, в което се провежда търсенето на нови структури. Те представляват формални схеми, които позволяват разделянето на задачата за проектиране на части и указват взаимните връзки между тях. Така се получават технически задания на проектираните части, които отчитат тяхното съединение. Обемът от информация необходим за вземане на решение на всяко ниво от процеса на проектиране се осигурява на базата на съвременните

информационни технологии, което води до автоматизация на процеса на проектиране.

Процесът на проектиране се нарича автоматизиран, когато той се осъществява от човек, във взаимодействие с компютър.

Степента на автоматизация може да бъде различна. Тя се оценява на базата на частта от проектните работи δ, изпълнявани от компютъра без участието на човека. Когато $\delta = 0$ проектирането се нарича неавтоматизирано. В другия граничен случай, когато $\delta = 1$ проектирането се нарича автоматично.

Основната цел на автоматизираното проектиране на сложни многофункционални електрони системи е преминаване от автоматизация на отделните информационни процеси, протичащи в обекта към тяхната цялостна автоматизация.

Тази концепция извежда същността на процеса на проектиране до ниво на универсален конструктор на електронни модели с единна форма за представяне на данни. Така се разширяват вариантите на проектните решения, намалява се броят на грешките при проектиране и се съкращават сроковете за внедряване.

1.1 Многофункционалността като категория

Понятието многофункционалност е атрибут, който показва възможността на един обект да изпълнява няколко различни функции [1]. Многофункционалността подобрява потребителските свойства на обекта и разширява кръга на решаваните с негова помощ задачи.

Това обуславя необходимостта от нейното разглеждане при анализа и проектирането на електронни схеми.

Способите за достигане на многофункционалност са много. Сред тях, за анализа и проектирането на електронни схеми най-често се използва подходът “от взаимовръзки, взаимодействия, условия, свойства и

ресурси - към функции” [1]. Този подход позволява търсенето на нови, оптимални за потребителя функции. Допринася за повишаване на нивото на интеграция на системите и техните елементи. Така многофункционалността се реализира като необходим за всяка система атрибут.

1.2. Елементна база за електронни схеми с повишена многофункционалност

Основен фактор за създаване на многофункционални електронни схеми е структурният излишък на използваните градивни елементи.

По закона за преминаване на количествените натрупвания в качество, този структурен излишък позволява на градивните елементи да получат нови показатели и свойства. Тази тенденция лесно може да се проследи в различните епохи от развитието на градивните елементи в електрониката.

С добавянето на решетки в конструкцията на радиолампите се появяват тетроди и пентоди, добавянето на преходи при транзисторите води до получаването на тиристори и триаци, а структурният излишък от транзистори генерира интегрални схеми.

Благодарение на бързо развиващите се технологии за тяхното производство многообразието от видове интегрални схеми се разширява значително. Броят на различните TTL и CMOS интегрални схеми надхвърля 6000. Усъвършенстваната технология за производство на различни видове памет ускорява появата на конвенционалните и специализираните микропроцесори, а в последствие и на прибори от типа ASIC, CPLD и FPGA.

От тях, днес доминираща роля в областта на цифровите електронни схеми имат схемите от типа FPGA.

Съвременните FPGA прибори съдържат милиони логически клетки и хиляди битове памет [9]. Тези количествени натрупвания материализират представата за многофункционалност на FPGA приборите. Сега те притежават вградени специализирани апаратни възли, голямо бързодействие и нищожна консумация на енергия.

FPGA приборите позволяват реализиране на отказоустойчиви системи и сложни проекти в рамките на един кристал. Допускат разпаралелване на процесите, както и осъществяване на реконфигурация на вътрешната архитектура в процеса на функциониране на системата.

FPGA приборите –отговарят на изискванията на стандартите MIL-Std 883 клас B и MIL-PRF-38535 клас Q и N, задаващи съответно методите и процедурите за контрол на микроелектронни устройства, използвани във военни и аерокосмически системи и нормите за тяхната надеждност. Тези стандарти, гарантират на FPGA приборите надежност, допускаща само 10 отказа за 10^9 часа работа, която е с няколко порядъка по-добра спрямо обикновените полупроводникови елементи. Тези показатели определят FPGA приборите като оптимална елементна база за разработката на отказоустойчиви системи и такива с критично приложение [94]. FPGA прибори могат да се открият в системите за управление на вертолети APACHE и самолетите B-52, F-14 и F-16, системите за радиоелектронна борба и управлението на радари, управлението и насочването на ракети PATRIOT, TOMAHAWK, STINGER, в системите за управление на космическата совалка Space-Shuttle и др. [95].

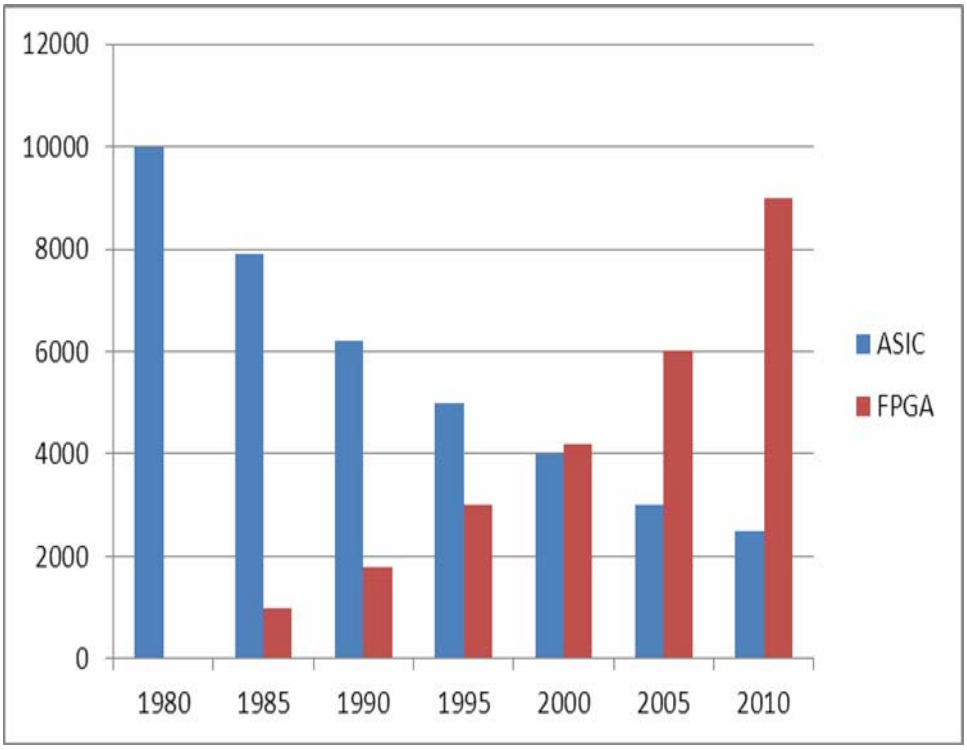
Сравнителният анализ относно използването на FPGA и микропроцесори при разработка на системи, осигуряващи безопасността на АЕЦ [94], (даден в таблица 1.1) показва, че използването на FPGA като алтернатива на микропроцесорите,

позволява да се минимизират 10 от 16 най-възможни рискове, свързани с внедряването им в такива системи.

FPGA позволяват реализирането на структури с отказоустойчивост и частично изключване на частта, явяваща се причина за появата на грешки.

Това открива възможност за реализиране на "динамическо реконфигуриране", което позволява автоматично преразпределение на функциите на дефектиралия модул сред изрядните [95].

Израз на тези свойства е сравнението на тенденциите за развитие на проектите, реализирани като ASIC и на базата на FPGA прибори. Нейните стойности за периода до 2010г. са показани на фигура 1.1 [97].



Фигура 1.1. Тенденция за броя на инициираните ASIC и FPGA базирани проекти.

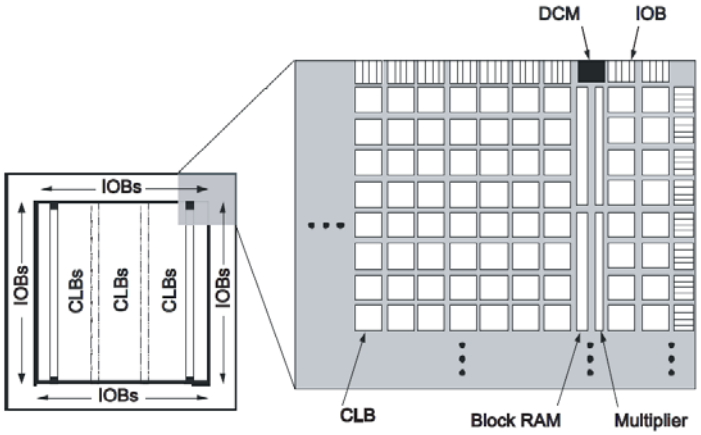
Таблица 1.1. Резултати от сравнението на рисковете, свързани с използването на FPGA и микропроцесори

Видове риск	Резултати от сравнителния анализ
1. Рискове, свързани със свойствата на обектите (FPGAs и микропроцесори(МП))	За тази група използването на FPGA позволява понижение на четири от деветте видовете рискове. за останалите стойностите на рисковете са идентични за FPGAs и микропроцесорите
рискове от нарушаване на изискванията за появата на откази по обща причина	Прилагането на FPGA позволява да се намалят рисковете от този вид в сравнение с рисковете за микропроцесорно базирани ИУС
рискове за нарушаване на изискванията от времеви характеристики	Прилагането на FPGA позволява да се намалят рисковете от този вид в сравнение с рисковете за микропроцесорно базирани ИУС
рискове от нарушаване на изискванията за надеждност	Прилагането на FPGA позволява да се намалят рисковете от този вид в сравнение с рисковете за микропроцесорно базирани ИУС
рискове за нарушаване на изискванията за защита от изкривяване на входната информация	Стойностите на тези рискове са еднакви за FPGA и МП
рискове за нарушаване на изискванията за защита на неоторизиран достъп	Стойностите на тези рискове са еднакви за FPGA и МП
рискове от нарушения на изискванията за устойчивост на външни влияния	Стойностите на тези рискове са еднакви за FPGA и МП
рискове от нарушения на изискванията за устойчивост при промяна на параметрите на захранването	Стойностите на тези рискове са еднакви за FPGA и МП
рискове от нарушаване на изискванията за електромагнитни влияния	Стойностите на тези рискове са еднакви за FPGA и МП
рискове от нарушения на изискванията за техническа диагностика	Прилагане на FPGA позвалява да се намалят рисковете от този вид от в сравнение с риска за ИУС базирани на МП
2. Рискове, свързани с изпълнението на процеси на жизнения цикъл (FPGA микропроцесори)	За тази група използването на FPGA позволява понижение на 6 от 7 видове рискове. за останалите стойностите на рисковете са идентични за FPGAs и микропроцесорите
рискове от нарушаване на изискванията на процеса на разработка	Прилагане на FPGA позволява да се намалят рисковете от този вид в сравнение с риска за ИУС базирани на микропроцесори.
рискове от нарушаване на изискванията на процеса на проверка	Прилагане на FPGA позволява да се намалят рисковете от този вид в сравнение с риска за ИУС базирани на микропроцесори.
рискове от нарушаване на изискванията на процеса на експлоатационен	Стойностите на тези рискове са еднакви за FPGA и микропроцесори.
рискове, свързани с използването на предходни проекти	Прилагане на FPGA позволява да се намалят рисковете от този вид в сравнение с риска за ИУС базирани на микропроцесори.
рискове, свързани с използването на програмното обезпечение	Прилагане на FPGA позволява да се намалят рисковете от този вид от в сравнение с риска за ИУС базирани на микропроцесори.
рискове, свързани с използването на прекъсвния	Прилагане на FPGA позволява да се намалят рисковете от този вид в сравнение с риска за ИУС базирани на микропроцесори.
рискове, свързани с използването на средства за развой и проверки	Прилагане на FPGA позволява да се намалят рисковете от този вид в сравнение с риска за ИУС базирани на микропроцесори.
3. Специфичните рискове, свързани с реализацията на схемни решения базирани на FPGA	Няма специфични рискове, свързани с употребата FPGA които не могат да бъдат намалени до приемливи нива при използване на стандартни или специални решения.

1.2.1. FPGA прибори

Приборите от класа FPGA са програмируеми интегрални схеми, притежаващи големи логически ресурси, възможност за неограничен брой програмирания, висока надеждност, малка консумация на енергия и ниска цена.

Типичната структура на един FPGA прибор [4,5,6] е показана на фигура 1.2. Основен елемент в структурата на съвременните FPGA прибор е двумерен масив от еднакви конфигурируеми логически блокове (CLB). Броят на тези блокове в чипа варира в зависимост от неговата големина от няколко десетки до няколко хиляди. В рамките на този масив се вграждат още голямо количество DSP блокове, статична памет и програмируеми входно изходни блокове, които осъществяват връзката между външния свят и изчислителните ресурси на FPGA прибора.



Фигура 1.2. Типична структура на FPGA прибор

Наличието на значително количество еднотипни структурни блокове в архитектурата на съвременните FPGA прибори, е важна предпоставка, която определя тяхната многофункционалност.Тази многофункционалност позволява намиране на решение на голямо множество от задачи с практическа насоченост. Цифровата обработка на сигнали добива реални измерения. Телекомуникационните и информационните технологии са едни от най-динамичните и бързо

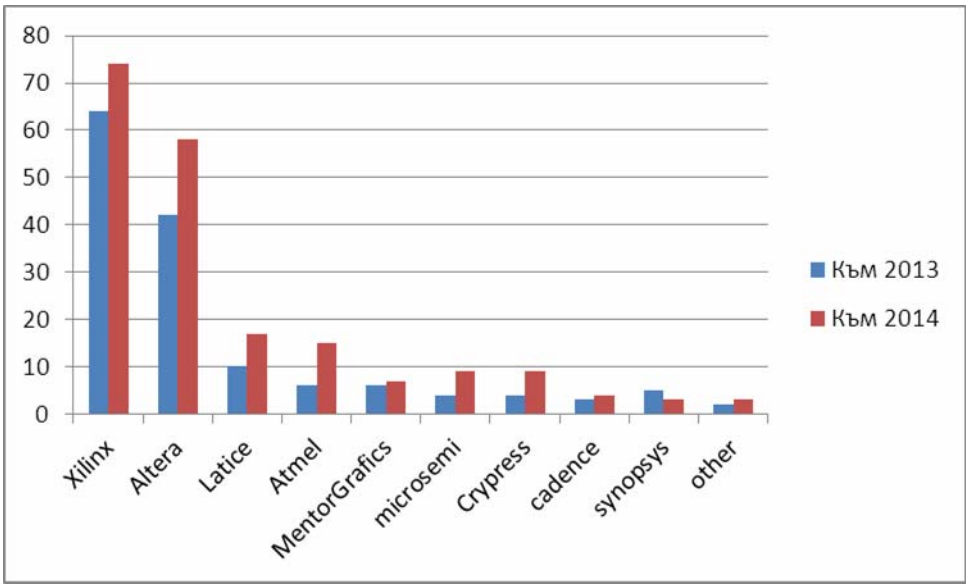
развиващи се технологии в световен мащаб. Всички задачи в системите за управление, акустиката, сеизмологията и др., изискващи обработка на големи потоци и обеми от информация в режим на реално време намират своето практическо решение.

По данни за 2009г., делът на FPGA прибори в средната цена на един автомобил е около 36%, 22% от цената на промишлени системи за автоматизация, 41% от цената на потребителската електроника, 33% от разходите за медицинско оборудване. При прогнози за среден годишен растеж от 10% до 2020 година броят на използваните FPGA прибори във всички области на земята ще бъде над 40 милиарда.

FPGA прибори се предлагат от няколко фирми. Класирането на тези фирми по използваемост на техните FPGA прибори според проучване на UBM за 2013г. [3] е показано на фигура 1.3. Тази диаграма позволява да се направят два извода:

1. Водещи по използваемост са фирмите Xilinx и Altera.

2. Използваемостта на приборите на Xilinx превишава тази на Altera средно 1.34 пъти.



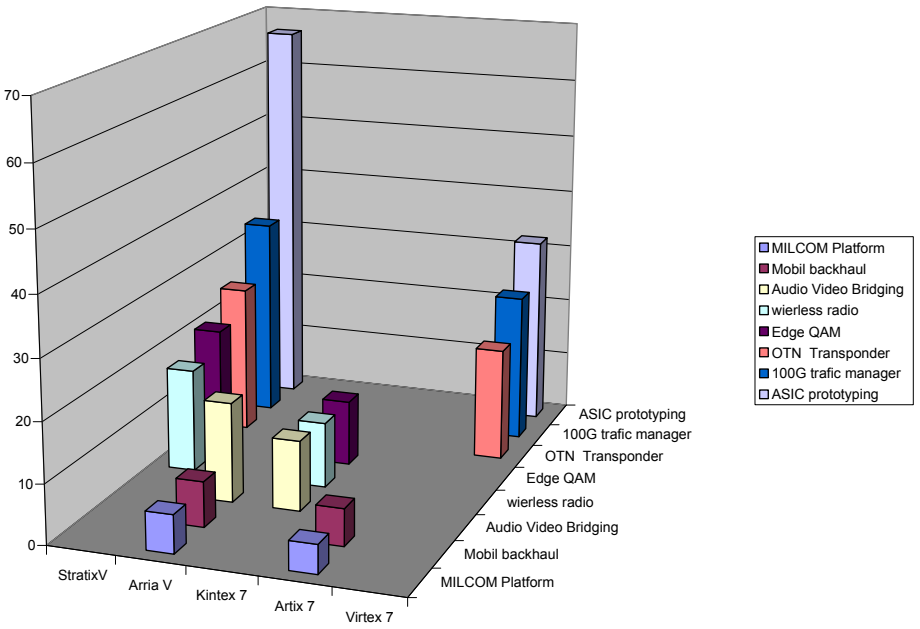
фигура 1.3.използваемост на FPGA прибори

Ключов фактор за всички FPGA прибори е консумираната енергия. Тя е основно предизвикателство, пред което са изправени както производители, така и проектанти. По тази причина, на консумираната енергия се отрежда едно от първите места в списъка на показатели за сравнение на продуктите, реализирани по суб микронна технология. Оценката на този показател за FPGA приборите на Altera и Xilinx, използвани в реални устройства от различни сфери на приложение са приведени в [110]. Тези резултати са показани в таблица 1.2, а като графика на фигура 1.4. Те позволяват да се направи извода, че приборите от серия 7 на Xilinx консумират от 1.32 до 2.08 пъти по малко енергия. На фигура 1.5. е показано сравнение на основните системни показатели на най-новите FPGA приборите на Xilinx и Altera [107].

Таблица 1.2 Консумирана мощност от FPGA приборите на Altera и Xilinx[W]

	StratixV	Arria V	Kintex 7	Artix 7	Virtex 7
OTN Transponder	25,08				19
Audio Video Bridging (AVB)		16,756	11,8		
MILCOM Platform		6,384		4,8	
Mobil backhaul		7,564		6,2	
wierless radio	17,38		11		
Edge QAM	20,9		11		
100G trafic manager	34				25
ASIC prototyping	66,56				32

Според това сравнение, системните показатели на приборите на Xilinx превъзхождат тези на Altera в границите от 1.25 до 2.5 пъти.



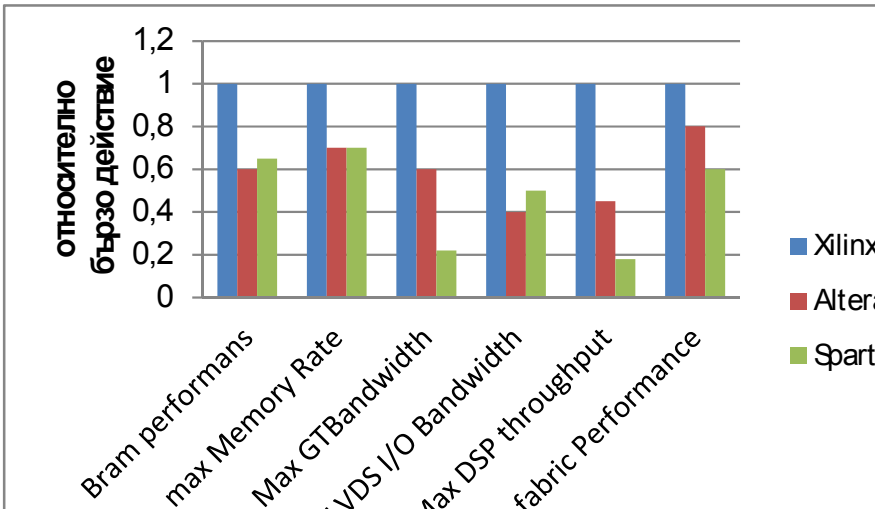
Фигура 1.4. Консумирана мощност от FPGA приборите на Altera и Xilinx[W]

Друг показател за сравняване на FPGA прибори е количеството на притежаваните от тях таблици на истинност. По този показател, архитектурата на Xilinx се оказва приблизително 1.2 пъти по-добра спрямо тази на Altera [104].

С оглед придобиване на по-пълна представа на съотношението между FPGA прибори на фирмите Xilinx и Altera, в таблица 1.3 са представени техните базови структурни показатели.

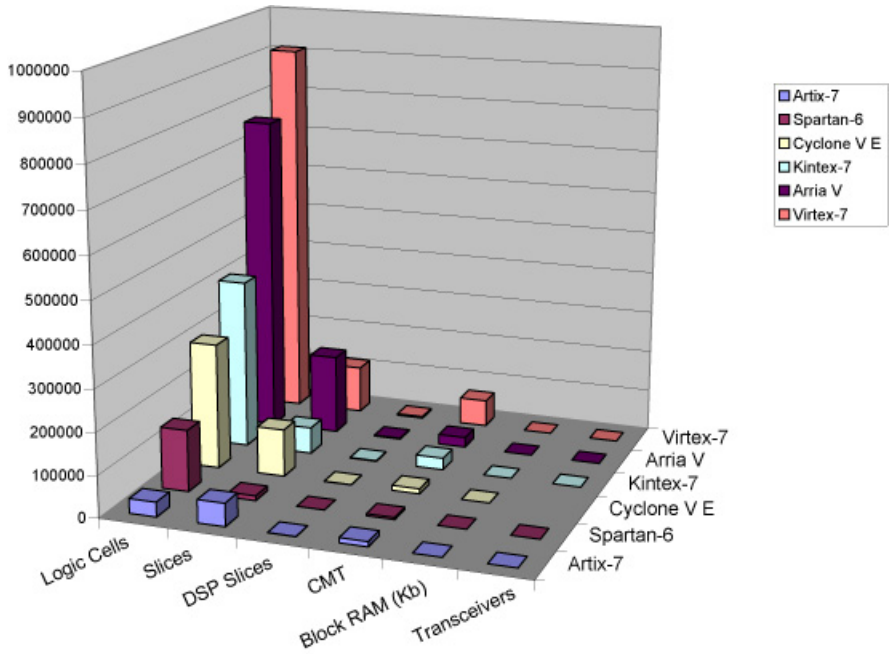
Таблица 1.3. Базови показатели на FPGA приборите на Xilinx и Altera.

Устроиства	**Параметри**					
	Logic Cells	Slices	DSP Slices	Block RAM (Kb)	CMT	Transceivers
Spartan-6	147443	11519	180	4824	6	8
Artix-7	35232	55050	700	12060	10	4
Kintex-7	406720	63550	1540	28620	1	16
Virtex-7	910080	113760	3960	64800	18	72
Arria V	760960	190249	2312	24140	16	36
Cyclone V E	301000	113560	684	12200	8	



Фигура 1.5. Сравнение на основни системни показатели.

В графична форма същите показатели са показани на фигура 1.6.



Фигура 1.6. Базови показатели на FPGA приборите на Xilinx и Altera.

Обобщението на направените съпоставки потвърждава водещите позиции на фирмата Xilinx в разработката на нови структурни и схемни

решения, увеличаващи многофункционалността на нейните FPGA прибори.

Най-масовата серия FPGA прибори на Xilinx е Spartan. Те позволяват обработката на високоскоростни еднополярни и диференциални сигнали, вграждане на 8 и 32 битови ядра на софтуерни процесори и защита на вложения в чипа проект срещу reverse-engineering, клониране и overbuilding [4,6]. Архитектурата на семейството Spartan е изградена от пет основни функционални елемента: Входно изходни блокове (IOB), Конфигурируеми логически блокове (CLB), вградена блокова RAM памет, DSP и DCM блокове.

Входно изходните блокове са предназначени да управляват потока от сигнали и данни между изводите и вътрешната структура на чипа. Те имат възможност за цифрово съгласуване на импеданса по програмен път. Поддържат еднополярни и диференциални стандарти за предаване на сигнали.

Конфигурируемите логически блокове съставляват главният ресурс на приборите. Използват се за реализиране на логически функции, паралелни суматори, броячи, АЛУ и разпределена RAM памет.

Вградената в FPGA блокова RAM памет е двупортова, с обем 18-Kbit. Тя е типичен пример за вграден апаратен ресурс, който повишава ефективността на FPGA прибори при решаване на типови задачи на схемотехниката. Броят на блоковете памет в чипа зависи от неговата големина.

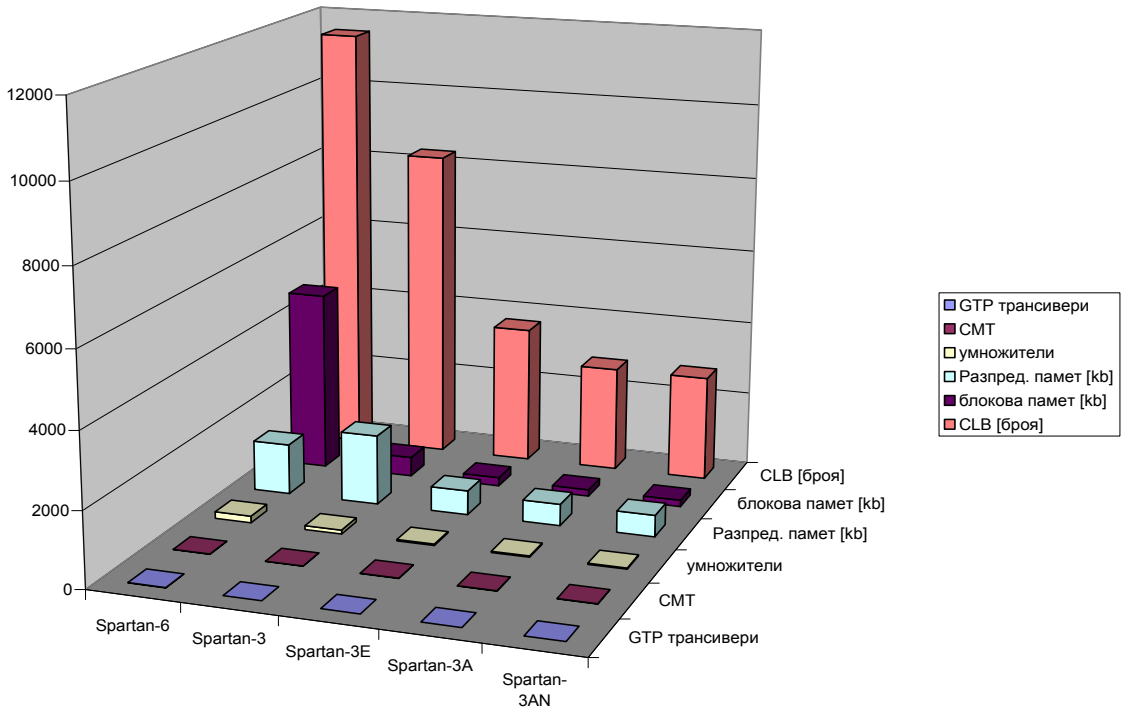
Блоковете DCM извършват закъснение, умножение и деление на тактовите честоти.

DSP блоковете съдържат 18 битов апаратен умножител и 48 битов акумулатор, които предоставят широки възможности за подобряване на неговата ефективност.

Връзките между петте функционални елементи се извършват посредством богата мрежа от превключващи елементи. Броят на потребителските изводи за приборите от това семейство е увеличен до 499.

Намират приложение в устройства с широколентов достъп с откриване и отстраняване на грешки, домашни системи за кино, цифрово телевизионно оборудване и др. [4].

В графична форма количественото представяне на показателите на приборите от серия Spartan е показано на фигура 1.7.

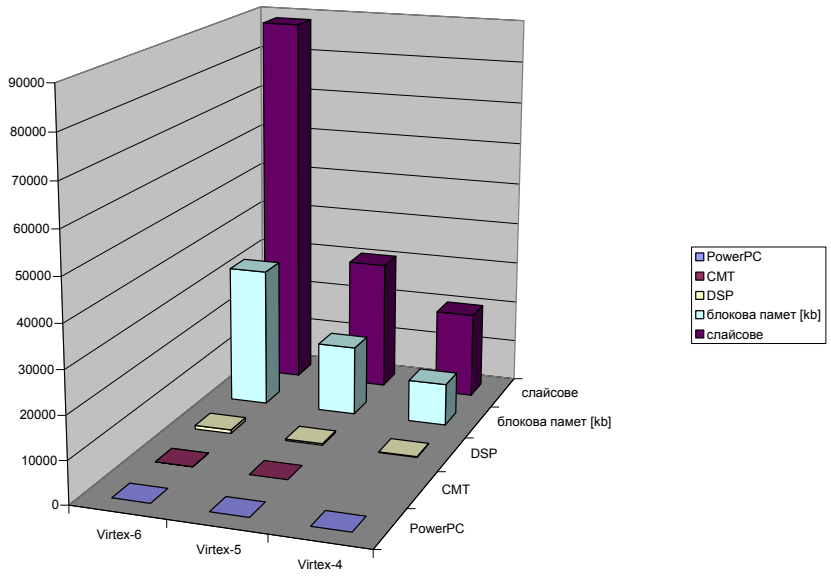


Фигура 1.7. Основни характеристики на семейството Spartan.

За професионални цели Xilinx предлага семейството Virtex. То е предназначено за реализация на висококоростни системи за цифрова обработка на сигнали [7,8]. Приборите от това семейство са реализирани на базата на ASMBL (Advanced Silicon Modular Block) архитектура. Те използват CLB блокове с 6-входови LUT и двупортова

RAM блок памет с обема 36 Кбит. Притежават DSP и DCM модули, висикоскоростен трансивер със скорост на предаване до 11.1 Gb/s и три режимен апаратен Ethernet MAC модул, съвместим със стандарта IEEE 802.3. В тях се вграждат процесорните ядра PowerPC 405 и PowerPC 440 на IBM.

В графична форма количественото представяне на показателите на приборите от серия Virtex е показано на фигура 1.8.

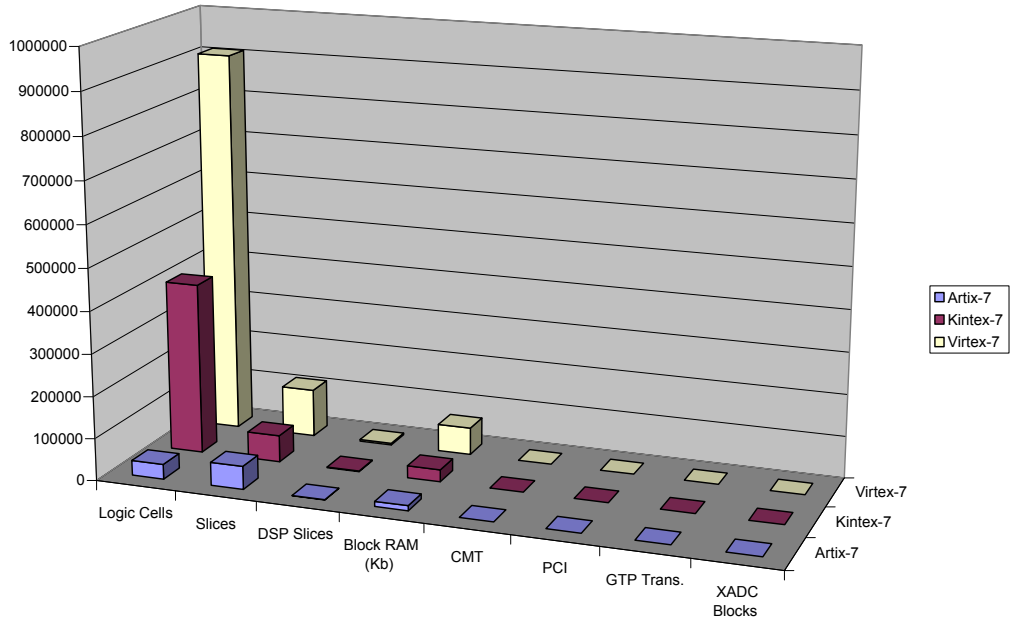


Фигура 1.8. Основни характеристики на семейството Virtex.

Най-новото семейство FPGA прибори на Xilinx носи името „серия 7”. Тя е изградена по технология 28 Nm, която позволява достигане на висока плътност и обем на логическите ресурси и осигурява пропускателна способност от 2.4 Tb/s при 50% по-малко консумирана мощност, спрямо тази на предишното поколение прибори. В приборите от това семейство се вграждат 36 Kb двупортова блок RAM памет с FIFO логика, два 12-битови, 1 MSPS АЦП с общо предназначение, вградени датчици за температура и контрол на захранващите

напрежения и вграден PCI Express блок. Те поддържат 256-битово AES криптиране с автоидентификация.

В графична форма количественото представяне на показателите на приборите от „серия 7” е показано на фигура 1.9. Данните показват, че FPGA приборите на фирмата Xilinx се очертават като един от ключовите фактори за формиране развитието на електроните системи и тяхната многофункционалност.



Фигура 1.9. Основни характеристики на семейството „серия 7”

Това гарантира техните позиции на основна среда за изграждане на различни по сложност вградени микропроцесорни ядра, системи и SoC [4,6].

1.3. Многофункционалност на вградените процесори и системи

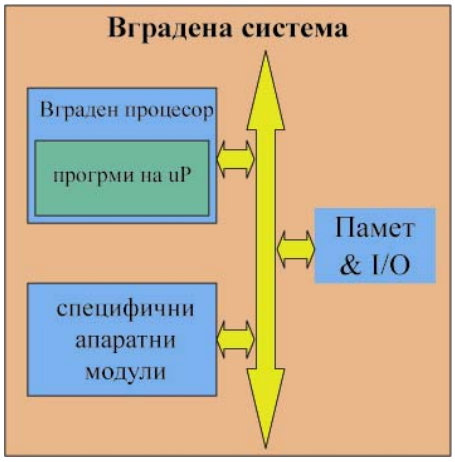
Изразите „вграден процесор” и „вградена система” са понятия, добили голяма популярност. Изделията от този клас често се анонсират и възприемат като ново направление в развитието на електронната техника, призвано да измести „класическите” големи и свръхголеми интегрални схеми [98]. За тези понятия съществуват много дефиниции. Според [99], под „вграден процесор” и „вградена система” следва да се разбират съответно процесорен чип, респективно електронна система, които са използвани в прибор, който не е работна станция, настолен или преносим компютър. В [24], понятията „вграден процесор” и „вградена система” се отнасят до изградени в рамките на един единствен FPGA чип процесорно ядро, набор периферии, памет, интерфейси към външни памети или устройства.

По аналогичен начин в [111], терминът „вградена система” се определя като свръхголяма интегрална схема интегрираща в рамките на кристала на чипа различни функционални блокове, които образуват завършено изделие за автономно използване в електрона апаратура. Обобщението на тези описателни определения се свежда до разбирането, че термините „вградени процесори и системи” отразяват характера на решаваните задачи, които в по-голямата си част са свързани с изпълнение на специализирани фонови функции, незабележими за човека при условие на коректно изпълнение. Типовата структура на една вградена система е показана на фигура 1.10.

Следва да се отбележи и фактът че за много проектанти, терминът вграден процесор, се свежда до организирани по особен начин цифрови устройства, реализирани на базата на стандартни за FPGA логически клетки, който притежава всички атрибути, присъщи на един класически

процесор (регистри, система от команди, възможност за изпълнение на програми записани в памет, и др.).

Важна страна на подобно представяне е, че подобни проекти се реализират с помощта на HDL езици за описание на апаратура. Това им придава определена имагинерност, която гарантира отсъствието на привързаност към FPGA на конкретен производител. Позволява тяхната реализация и функциониране в качеството на структури, устройства за управление или други цифрови възли.



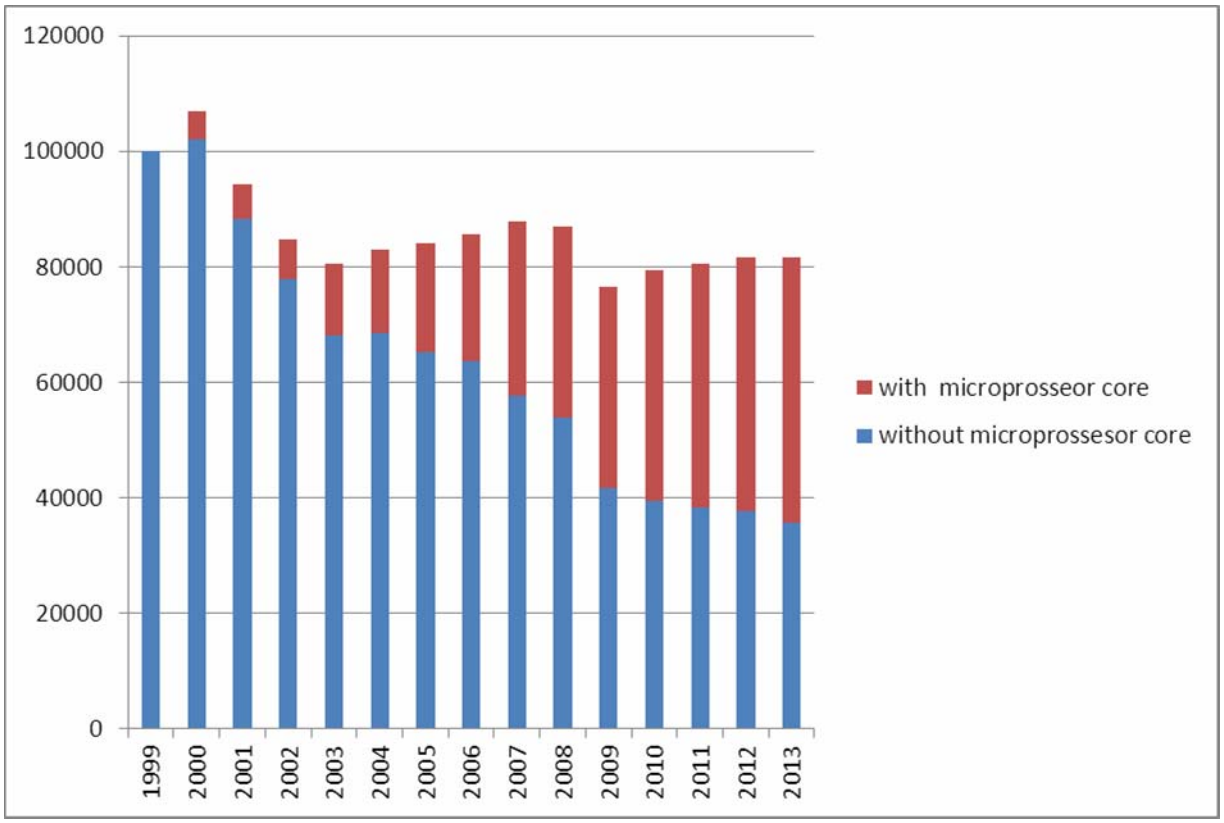
Фигура 1.10. Типова структура на вградена система

Разработката на вградени процесори не е процес, достъпен само за висококвалифицирани инженери с богат опит. Това обяснява и наблюдаваната тенденция към универсализация на вградени процесори и тяхното използване за решаване на различни типове задачи. Вградените процесори са средство за разширяване и постигане на реални измерения за многофункционалност на електроните системи и устройства.

По данни на Gartner [10] (вж. фигура 1.11) броят на новите проекти включващи в себе си вградени процесори нараства по закон близък до

експоненциалния. Вградените микропроцесорни ядра не се ограничават от производствени лимити и приложения.

В съвременния автомобил има над 20 вградени процесора, заети с управлението на спирачната система, двигателя и др.

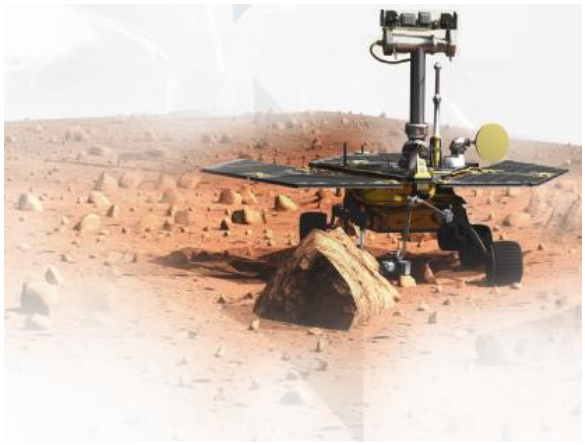


Фигура 1.11. Тенденции на използването на вградени процесори.

Многофункционалността на вградените процесори намира изява и в системи с критични приложения, работещи в екстремални условия. Управляващият блок на телескопа Хабъл Spacecube ползва 4 вградени процесора (вж. фигура1.12) [10].

Фигура 1.12. Управляващ блок Spacecube

Системите за кацане и управление на марсохода Curiosity (вж. фигура 1.13) са реализирани на базата на 3 FPGA прибора от серията Virtex [10].



Фигура 1.13. Марсохода Curiosity

Над 1000 вградени процесори [10] са използвани за реализация на светлинното табло на площад Тайм скуеър в Ню Йорк (вж. фигура1.14). По данни на Кен Чапман, [123] за постигане на висока многофункционалност в рамките на един FPGA чип са вградени 3600 микропроцесора.

Приведените примери показват стремеж за постигане на оптималност и функционална пълнота на потребителските функции на системите, изграждани на базата на вградени процесори. Това показва, че е на лице

тенденция на плътно сближаване на интересите на потребители и производители, която на практика превръща програмно реализируемите, вградени микропроцесорни ядра в ефективно доминираща алтернатива, както над конвенционалните, така и над специализираните микропроцесори. В тази насока, вградените процесорни ядра могат да се считат за основни градивни блокове на всички настоящи и бъдещи високо технологични, информационно-управляващи системи и фактор за тяхното непрекъснато усъвършенстване.

Фигура 1.14. Светлинно табло на площад Тайм скуеър в Ню Йорк

Значението на развитието на вградени системи за растежа на производителността на икономиката е оценено отдавна.

През 2001г. Националната академия на науките на САЩ публикува програма за научни изследвания, наречена „Embedded Everywhere” [12], която подчертава важната роля на научните изследвания в областта на вградените процесори. В тази насока през 2004г Европейският съюз стартира инициативата, наречена ARTEMIS (Advanced Research and Technology for Embedded Intelligence and Systems). Тя фокусира

различните аспекти от създаването и прилагането на координирана европейска стратегия в тази област, обхваща изследователските приоритети и създаване на необходимата инфраструктура за стандартизация, политика и образователни програми [13].

Подобни програми с държавна подкрепа на тази тенденция има в Япония, Корея, Китай и Русия. Всяка една от тях има свои собствени характеристики и акценти, насочени към масово въвеждане на вградените системи в широк спектър от задачи, свързани с промишлеността, здравеопазването, развлекателната индустрия, фиксирани и мобилни мрежи и др.

1.3.1. Вградени процесори

Важна роля за разширяване на многофункционалността на съвременните FPGA прибори играе възможността за използването им като среда за изграждане на различни по сложност и бързодействие системи и „вградени" микропроцесорни ядра. Те са основната причина за широкото навлизане на FPGA приборите в потребителската електроника, телекомуникациите, и различните интернет устройства.

По своята същност, изразът „вграден процесор" обединява в себе си понятие, обхващащо изградените в рамките на един единствен FPGA чип процесорно ядро, набор периферии, памет, интерфейси към външни памети или устройства. Съществуват два вида процесори, които могат да се вграждат в FPGA приборите – апаратни и програмни.

Независимо от начина си на генериране, тези микропроцесорни ядра позволяват да се ускори цикълът на адаптацията им към изискванията

на конкретния случай и гарантират устойчивост на процесорното ядро в процеса на настройка.

1.3.2. Апаратни вградени процесори

За апаратни се считат онези процесори, чието ядро е вградено физически в структурата на FPGA прибора още в процеса на неговото производство [24,25]. Този вид процесори се предлагат от фирмите Xilinx и Altera.

В своите FPGA прибори Altera вгражда апаратния процесор ARM922T [26]. Това е бързодействащ 32 битов RISC процесор, проектиран за изграждане на системи от типа System-on-Chip (SoC). Процесорът поддържа AMBA и AHB магистрали и 32-bit инструкции. Работи с тактови честоти до 800 MHz. Притежава кеш-памет с две нива, блок за работа с плаваща запетайка с двойна точност и спомагателен процесор NEON за обработка на мултимедийни данни. Може да взаимодейства с други апаратни модули, реализирани в структурата на FPGA. Вгражда се в семействата CycloneV, StratixV и ArriaV.

В качеството на апаратен процесор, Xilinx вгражда в своите FPGA прибори от семейството Virtex процесорните ядра на фирмата IBM PowerPC405 и PowerPC440 [27,28,29] и ARM Cortex-A9.

PowerPC 405 е 32-разрядна архитектура, която е предназначената за вграждане в FPGA Virtex-4.

Power PC 440 е 32-битов процесор от серията PowerPC 440 на IBM, който се вгражда в семейството FPGA прибори Virtex-5 FXT.

И двата процесора съдържат конвейерно процесорно устройство, което заедно с елементите за управление на паметта и кеша, таймерите и средствата за настройка и контрол позволява осъществяване на вградени системи от типа SoC. PowerPC 440 предлага три независими 128-bit PLB интерфейса, интерфейс за специализирани ко-процесори,

функции за плаваща запетая и два независими 32KB кеша за инструкции и данни. Намира приложение в системи, изискващи висока степен на сигурност.

Едно обобщение на основните показателите на процесорите PowerPC 405 и PowerPC 440 е показано в таблица 1.4 [30].

В детайли свойствата на вградения процесор PowerPC 405 и PowerPC 440 могат да се открият в [27,28,29].

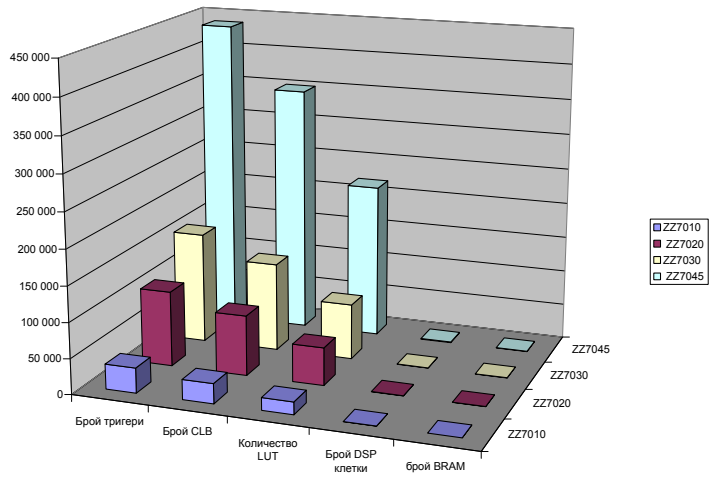
Таблица 1.4. Основни показатели на процесорите PowerPC 405 и PowerPC 440

	Architecture	Pipline	Caches – I/D	MMU	DMPS Estimate
PPC405 (Virtex-4 FX)	32-bit instruction, 32-bit address, 64-bit data	Single instruction /cycle, five-stage pipeline, in-order	16K/16K, two way set associative , no locking	Page size: 1 KB to 16 MB	700+ DMIPS
PPC440 (Virtex-5 FXT)	32-bit instruction, 36-bit address, 128-bit data, Book E compliant	Two instructions /cycle, seven-stage pipeline, out-of-order	32K/32K, 64-way set associative, locking	Page size: 1 KB to 256 MB	1000+ DMIPS

Zynq-7000 е най-новото семейство FPGA прибори предлагани от Xilinx. То е базирано на прибори от серия 7 и притежава вграден двуядрен процесор ARM® Cortex-A9 [103]. Ресурсите на семейството са обобщени в таблица 1.5 и на фигура 1.15.

Таблица 1.5. Основни характеристики на семейството Zynq-7000

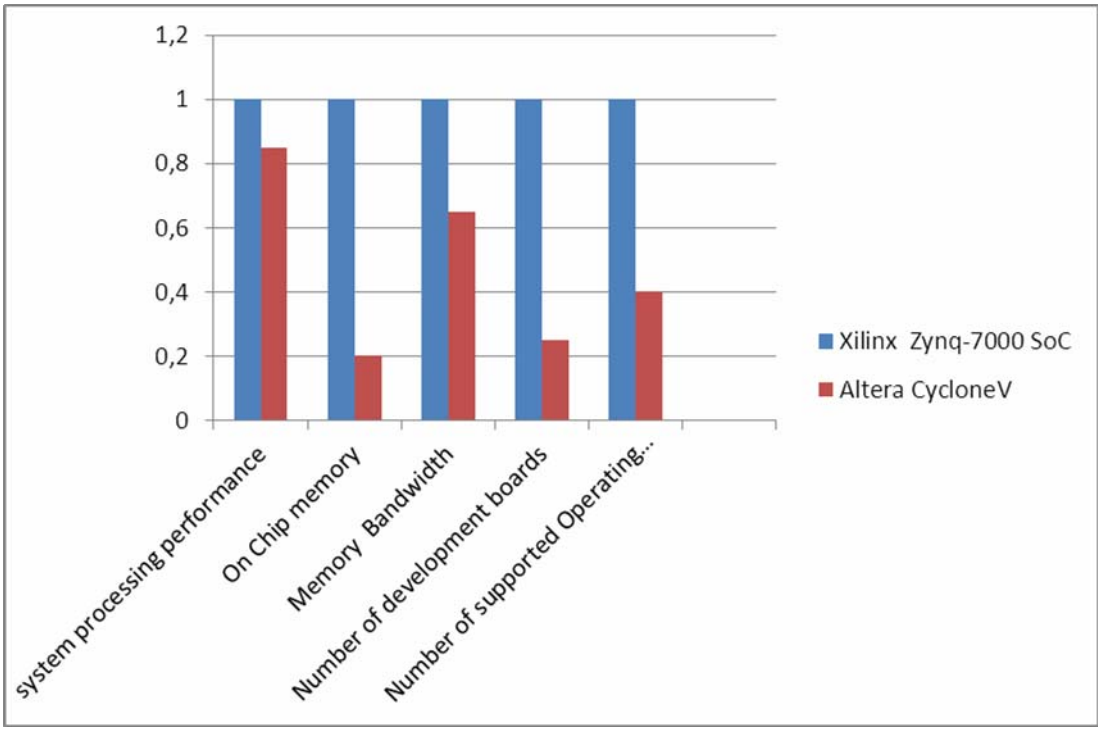
	ZZ7010	***ZZ7020***	***ZZ7030***	***ZZ7045***
Брой CLB	28 K	85K	125 K	350 K
Количество LUT	17 600	53 200	78 600	218 600
Брой тригери	35 200	106 400	157 200	437 200
брой BRAM (36кb)	60	140	265	545
Брой DSP клетки	80	220	400	900



Фигура 1.15. Основни характеристики на семейството Zynq-7000

Обобщеното сравнение на основните системни показатели на SoC реализирани с FPGA приборите от серия Zynq-7000 на Xilinx и с CycloneV на Altera [110] е показано на фигура 1.16.
Според него, системните показатели на SoC, реализирани на база Zynq-7000, превъзхождат показателите на SoC реализирани с CycloneV в границите от 1.25 до 5 пъти

Сферата на приложение на членовете на семейството Zynq7000 е широка. Тя обхваща транспортни системи, промишлена автоматика, обработка на изображения, сигнали, комуникационни системи, битова електроника и др.



Фигура 1.16. Сравнение на системни показатели на SoC реализирани с Zynq-7000 и с CycloneV

1.3.3. Програмно изграждани процесори

За разлика от апаратно вградените процесори, ядрото на всеки един от програмно изгражданите процесори предварително се моделира, синтезира и тества извън ресурсите, предлагани от структурата на избрания FPGA прибор. За да отговорят на съвременните тенденции за използване на вградени процесори, фирмите, предлагащи FPGA прибори разработват свои програмни процесорни ядра.

Altera предлага своя програмен процесор Nios® II. Той е предназначен за вграждане в семействата Stratix-II и Cyclone [32]. Nios II е 32 битов RISC процесор с общо предназначение. Притежава 32 регистъра с общо предназначение и позволява непосредствен достъп до периферииите, разположени в чипа [32].

Друг програмен 32 битов процесор е LEON3. Той притежава апаратен умножител, MAC клетки, конфигурируеми кеш и scratch pad памети за

инструкции и данни. Предлага се за вграждане в FPGA приборите на фирмата Actel. Намира приложение в автоматизацията, безжични телекомуникации и за изграждане на SOC.
LatticeMico32 е 32 битов процесор с общо предназначение. Той притежава Harvard архитектура, 32 регистри и може да обработва до 32 външни прекъсвания. Вгражда се в FPGA прибори на фирмата Lattice.

LatticeMico8 е 8-битов микропроцесор, оптимизиран за вграждане в FPGA приборите на Lattice [33]. Притежава 32 регистъра с общо предназначение и 32 байта вътрешна Scratch Pad памет. Поддържа до 256 входно изходни порта. За реализирането му са необходими около 200 LUT. Предлага се като лицензиран Verilog файл.

Xilinx предлага две ядра MicroBlaze и PicoBlaze. Всяко едно от тях притежава мощна, компактна архитектура и заема значително по-малко ресурси в FPGA, отколкото кое да е сходно нему по показатели микропроцесорно ядро. Вграждат се във всички FPGA семейства на Xilinx [34,35].

MicroBlaze е софтуерно ядро на 32 битов RISC процесор с Харвард архитектура. Притежава отделни 32 битови магистрали за адреси и данни, съответстващи на изискванията за OPB (On-chip Peripheral Bus) на IBM. Той предоставя възможности за пряк достъп до 4GB външна памет, а посредством магистралата LMB (Local Memory Bus), си осигурява и пряк достъп до ресурсите на блоковата памет на FPGA.
MicroBlaze е едно от най-бързите технически решения способно да работи в промишлени условия [35]. Използва се за изграждане на комплексни системи от мрежи за пренасяне на данни, телекомуникации, и др. Съпоставката на неговите качества с

процесорите NiosII, LatticeMico32 и LEON3 представена в таблица 1.6 показва че той изисква най-малко ресурси за реализация.

Таблица 1.6. Основни характеристики на 32 битови S/W вградени процесори

Ядро [бита]	Работна честота, [MHz]	производителност, [MIPS]	Брой slices за реализация
LatticeMico32[32]	100	-	2230
LEON 3 [32]	150	150	3500
MicroBlaze [32]	100 - 200	166	1250
Nios II F [32]	185	218	1800

PicoBlaze е високо производителен 8 битов RISC процесор, който по най-ефективен начин използва вътрешната структура на FPGA чипа. Неговото ядро се счита за един от най-успешните отговори на съвременните тенденции за изграждане на програмно реализирани вградени процесори. Той изисква значително по-малко ресурси спрямо кой да е друг сходен нему традиционен микропроцесор. Вътрешната архитектура на PicoBlaze позволява пълен достъп до всички апаратни особености на чипа. Допуска вграждане на няколко негови копия в структурата на един и същ FPGA прибор. PicoBlaze се оказва единственият процесор, чието използване доминира при осъществяване на паралелни алгоритми в пределите на един FPGA прибор и изграждането на SoC. Позволява използването му като базова градивна единица при изграждане на многопроцесорни системи. Това разкрива многофункционалността на процесора и обяснява причината за неговото широко приложение [36] и превръщане то му в основно средство за обучение по компютърна техника [140].

1.3.4. Микропроцесор PicoBlaze

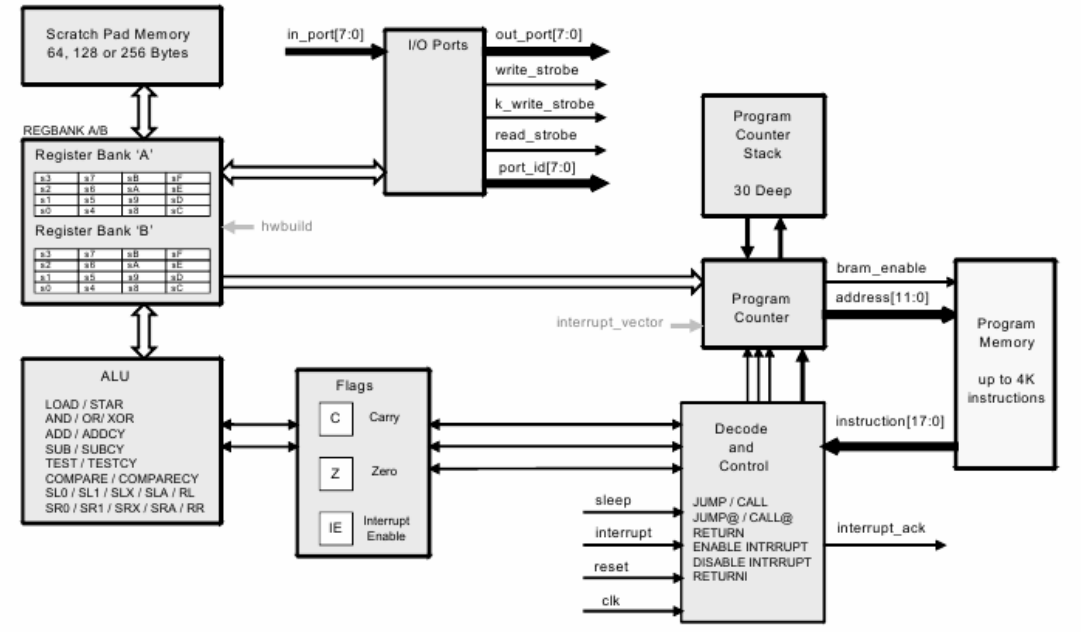
Микропроцесорното ядро PicoBlaze е разработено от Кен Чапман [36]. Структурната схема на PicoBlaze е изключително проста. Показана е на фигура 1.17 [45]. Тя съдържа две банки 8 битови регистри с общо предназначение. Броят на регистрите във всяка банка е 16 и са напълно

равностойни. Никой от тях не е отделен за специални цели и не притежава приоритет спрямо останалите. Няма специализиран акумулатор и всеки резултат се записва директно в указания в инструкцията регистър.

С помощта на асемблерска директива регистрите могат да бъдат преименувани, което допринася за постигане на по-добра яснота и читаемост на програмата.

По подразбиране, размера на програмната памет на микропроцесора е 1к 18 битови думи. Аритметично-логическото устройство на микропроцесора е 8-битово и изпълнява всички основни аритметични и логически действия.

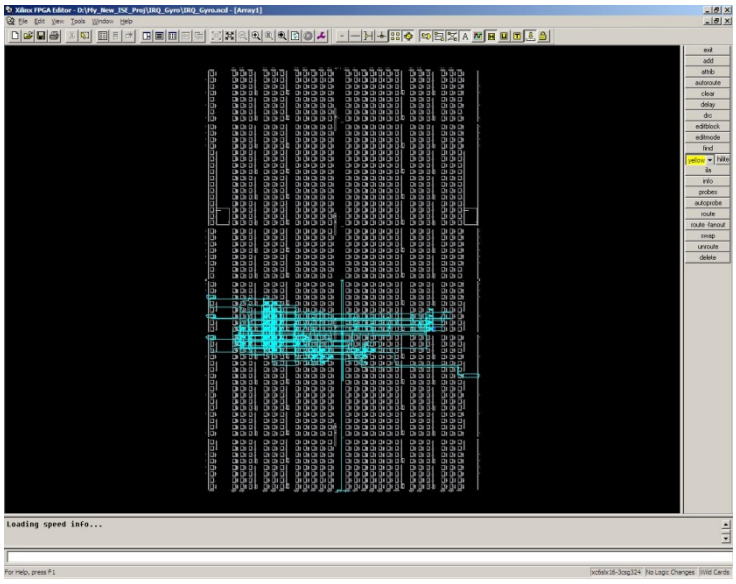
PicoBlaze притежава ОЗУ (Scratchpad RAM), чиито размер може да приема стойности до 256 байта. Достъпът до нея се осъществява чрез косвена адресация. PicoBlaze поддържа до 256 входни и 256 изходни порта. Входно изходните портове на микропроцесора разширяват неговите възможности. Чрез тях той може да се свързва със специализирана периферия или с друга логика вътре или извън FPGA.



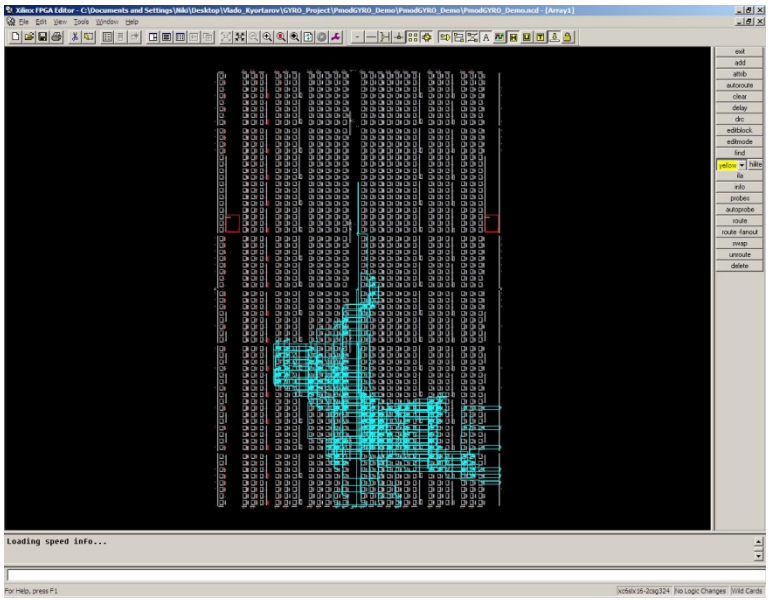
Фигура 1.17. Структурна схема на микропроцесорното ядро PicoBlaze

PicoBlaze се разпространява свободно под формата на VHDL файл. Не изисква лицензионни права. Неговото ядро изисква най-малко ресурси за реализация (96 slices при вграждане в Spartan-3 [39] и 26 slices при вграждане в Spartan -6, Virtex, и Серия 7). При вграждане в Spartan-3 бързодействието на PicoBlaze е в границите на 44 MIPS и над 200MIPS при вграждане в Virtex и Seria 7. PicoBlaze притежава висока надеждност, която се определя по надеждността на използваният FPGA прибор [42].

PicoBlaze е най-добрата илюстрация за необходимостта от вграждане на малки процесори в FPGA базирани проекти. Тя почива на факта, че използването на процесор за изграждане на модули, ориентирани към реализиране на последователни функции или на апаратни ресурси, изпълняващи няколко задачи в режим на времеделение, се оказват много по-ефективни, отколкото тяхното апаратно реализиране с ресурсите на FPGA чипа. В качеството на пример, на фигури 1.18 и 1.19 са показани ресурсите заети от реализацията на един и същ проект, изпълнен съответно в апаратна и програмна форма. Разликата в ресурсите заемани от проектите е двукратна.

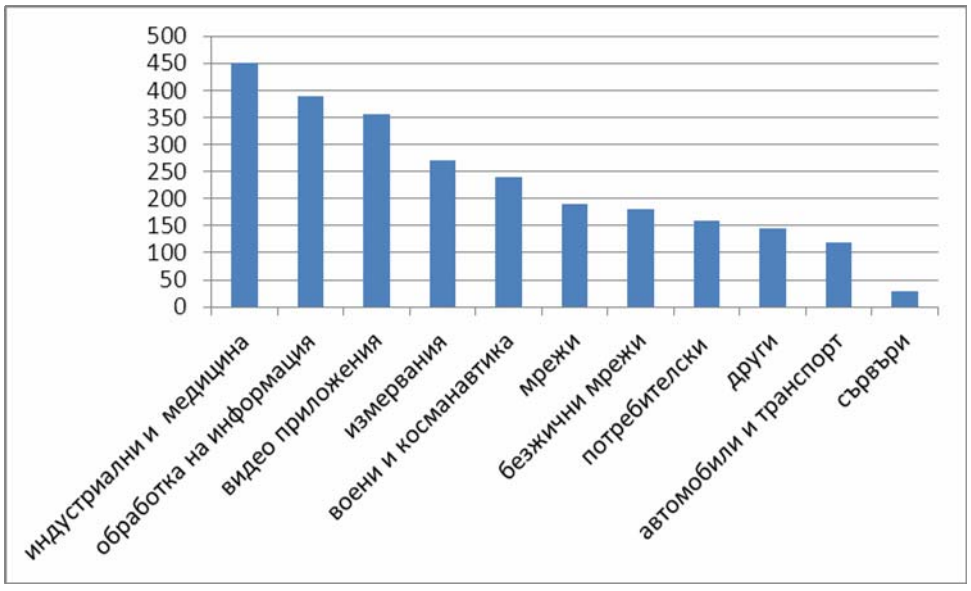


Фигура 1.18. Реализиране на SPI контролер посредством микропроцесора PicoBlaze



Фигура 1.19. Реализиране на SPI контролер с ресурсите на FPGA чип.

Един опит за илюстриране на широкия спектър от различни практически приложения на PicoBlaze е показан на фигура 1.20 [41].



Фигура 1.20. Практическите приложения на PicoBlaze

PicoBlaze позволява постигането на драстично съкращаване на продължителността на цикъла за проектиране. По тази причина, много

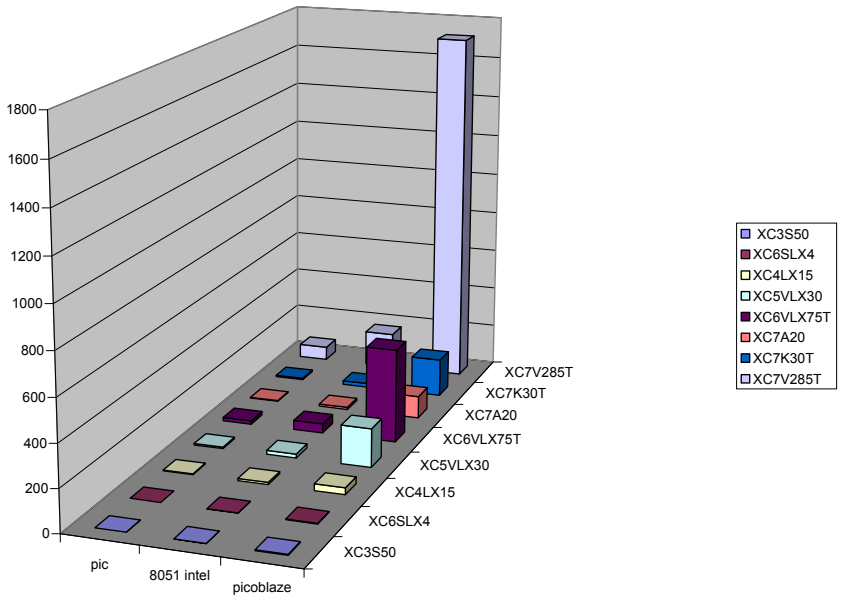
фирми и отделни производители, в чиито производствени програми класическите микропроцесори от типа PIC, 8051 и др., имат дългогодишни традиции, намират за рационално да реализират техните ядра в рамките на един FPGA чип [14,15].

На тази основа логично възниква въпросът за необходимостта от разработване и използване на нови микропроцесорни ядра.

Отговорът на този въпрос следва да се търси в стратегията за реализиране на класически микропроцесорни ядра в рамките на един FPGA чип. Същността на тази стратегия съвпада с изработването на копия и репродукции на картини от известни автори. При тази стратегия новото се свежда до неколкократно повишение на бързодействието, но всички характерни белези и недостатъци на породилата го архитектура остават. В таблици 1.7 и 1.8 е представено сравнение на броя на процесорите тип PIC, 8051 и PicoBlaze в FPGA прибори от основните семейства на Xilinx, притежаващи съответно минимални и максимални ресурси. В графична форма, съдържанието на тези таблици е представено съответно на фигури 1.21 и 1.22.

таблица 1.7. Брой процесори за вграждане

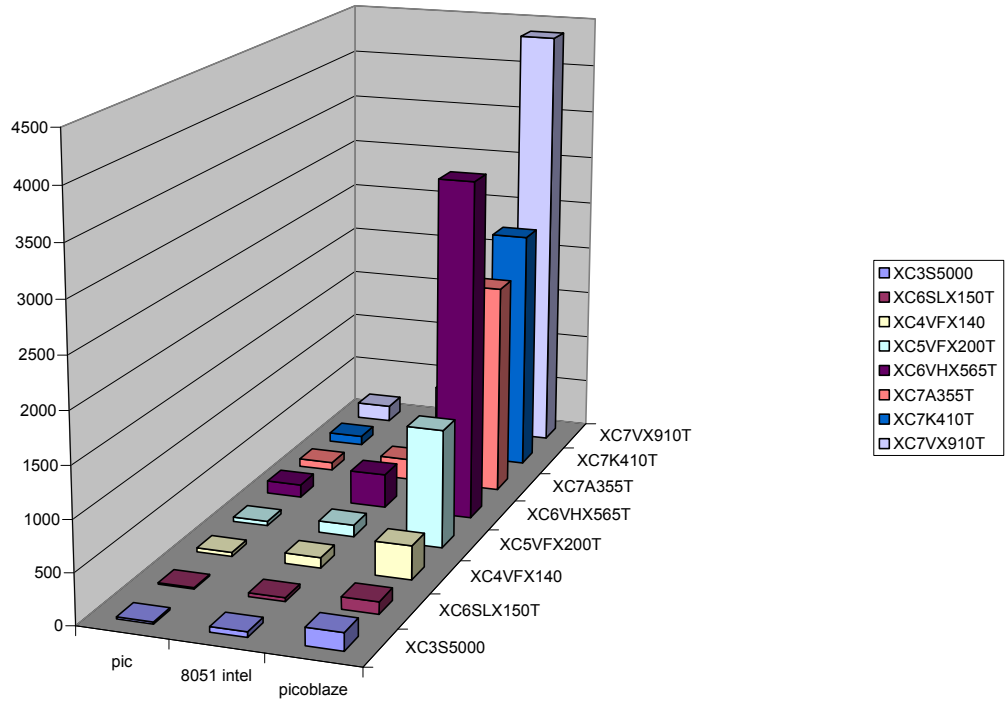
Семейства FPGA	Брой процесори за вграждане		
	pic	8051 intel	picoblaze
XC3S50	0	1	4
XC6SLX4	0	1	3
XC4LX15	4	10	32
XC5VLX30	7	19	185
XC6VLX75T	17	45	448
XC7A20	4	11	108
XC7K30T	7	18	183
XC7V285T	65	172	1719



Фигура 1.21. Брой процесори за вграждане в FPGA прибори от основните семейства на Xilinx с минимални ресурси.

таблица 1.8. Брой процесори за вграждане

Семейства FPGA	Брой процесори за вграждане		
	pic	8051 intel	picoblaze
XC3S5000	20	54	173
XC6SLX150T	14	37	120
XC4VFX140	39	104	336
XC5VFX200T	45	119	1182
XC6VHX565T	130	342	3406
XC7A355T	81	212	2117
XC7K410T	93	245	2444
XC7VX910T	166	439	4375



Фигура 1.22. Брой процесори за вграждане в FPGA прибори от основните семейства на Xilinx с максимални ресурси.

На базата на данните от [15], в таблица 1.9 е направена съпоставката на показателите на микропроцесора 8051 при вграждането му в FPGA приборите Xilinx. За сравнение в същата таблица са показани показателите на двете последни версии на PicoBlaze.

Таблица 1.9. Показатели на 8051 при вграждане в FPGA

	Работна честота, [MHz]	производителност, [MIPS]	Брой slices за реализация
8051 Intel	12	1,3	1600
8051 FPGA emulation	40	5	311
PicoBlaze v7	250	200	26
PicoBlaze v3	250	125	96

Очевидно е, че използването на вградения процесор PicoBlaze много по- добре отговаря на изискванията по отношение на ресурси, бързодействие и бъдещи промени. Тези предпоставки определят неговото използване от хиляди инженери по света в различни устройства и системи, а с приложението му в рамките на различни

учебни заведения, потенциалният брой на реализираните проекти става огромен.

Интересно е да се отбележи, че присъствието на PicoBlaze в един проект като канал за обмен на информация с PC значително ускорява итерационните процедури без ущърб на тяхното качество и процеса на неговото реализиране като цяло. Това открива възможност за автоматизиране на проектираната система чрез създаване на затворен кръг за нейната верификация и настройка.

1.4. Развойни средства за електронни схеми с повишена многофункционалност

Съвременната методология за проектиране и синтез на електронни схеми с повишена многофункционалност включва в себе си два основни етапа.

1. въвеждане, описание и синтез на проекта.
2. настройка и верификация на проекта.

Обща особеност за двата етапа е използването на HDL-базирани средства.

По мнението на експерти, в рамките на близките 5-10 години тази методология за проектиране ще се превърне задължителна и ще се използва от всички проектанти.

Многофункционалността на съвременните градивни елементи, извежда на преден план проблемите за качеството и продължителността на процеса на проектиране.

Първия опит за решаване на тези проблеми е въвеждането на стандартизация. Тя позволява при създаването на ново електронно устройство да се използват готови, предварително разработени компоненти, в това число и такива, съдържащи в себе си апаратни или програмно изграждани процесори.

Следващ етап в развитието на средствата за проектиране на хардуер е ориентирането към използване на платформи.

При този подход обемът на готовото проектирано и верифицирано апаратното осигуряване на разработваната система може да достигне до 80%.

При средствата за разработка на програмно осигуряване се наблюдават две тенденции:

Първата се свързва с активното внедряване на езици за програмиране от високо ниво (C, C++). При нея за всяка архитектура се развиват средства от типа на компилатори и симулатори.

Втората тенденция се свързва със създаването на метаезици (например UML - Unified Modeling Language), позволяващи описване на поставената задача, а в последствие и генериране на необходимото програмно осигуряване [98].

Тези тенденции ускоряват технологиите за автоматизираното проектиране на електрони устройства. Те позволяват решаването на задачи с повишена сложност. Ползват абстракции и автоматизирани процеси за реализиране на дребните детайли на проекта. Позволяват извършването на функционални проверки в началните етапи на проектиране, когато внасянето на изменения не води до усложнения. Силните им страни проличават когато количеството на логически клетки в FPGA надхвърли границата от 100000.

Тази концепция извежда същността на процеса на проектиране до ниво на универсален конструктор на електрони модели. Универсалният конструктор ~~Той~~ притежава единна форма за описание на взаимодействие и представяне на данни на създаваните електрони модели. При него електроните модели представляват компактни многофункционални програмни модули, способни да представят, създават и трансформират произволни данни. Това разширява

методологията и функционалните възможности на процесите за проектиране и моделиране. Намалява възможността за поява на грешки. Съкращава сроковете за внедряване.Осигурява средствата за проектиране с авангардни методи за управление и оптимизация на проекта.

1.4.1. Развойни средства за проектиране

Увеличената многофункционалност на вградени системи (ВС), превърна необходимостта от тяхното проектиране, верификация и настройка в приемливи от гледна точка на пазара срокове в сериозен проблем. Оценките на този проблем [119] показват, че към момента функционалната верификация и настройката заемат до 80% от времето за изготвяне на проекта. Това води до възникване на изоставане на възможностите на средите за изграждане на ВС, спрямо тези на средствата за проектиране (EDA - Electronic Design Automation). При средна скорост на нарастване на възможностите на средите за изграждане на ВС от 58 % /година и 21% / година на EDA [113], величината на това изоставане добива стойности, които заслужават внимание. За неговото елиминиране е необходимо създаването на нови средства за автоматизация, които осигуряват изцяло интегрирано проектиране. От средствата за автоматизация се изисква осигуряване на възможност за едновременно провеждане на разработка, проектиране, тестване, верификация и настройка на програмната и апаратната част на проекта в рамките на целия цикъл на проектиране на ВС, от идея до реализация.

По този начин същността на проектирането на ВС се измества от областта на RTL в тази на програмните дейности. Ето защо към

методите и средствата за проектиране се предявяват изисквания за подобряване, усъвършенстване и създаване на нови методики и програми. При тяхното създаване се използват теории, които позволяват изпълнението на формален анализ и адекватно отразяват функционалността на системата.

Комплексният подход за проектиране е практическата реализация на тези изисквания. Неговото прилагане намалява риска от грешки, възникващи при едновременното създаване на апаратна и програмна част на проектираните ВС. Позволява използването на абстракции от високо ниво и унификации на HW и SW компоненти на системата. Осигурява интеграция с програмното осигуряване работещо на по ниските нива. Това означава необходимост от създаване на стандартизация, която позволява използването на готови, предварително разработени IP компоненти.

Част от тези изисквания имат място в системите за развой на фирмите предлагащи FPGA прибори. За изграждане на ВС фирмата Lattice [33], предлага интегрирана среда, обхващаща три инструмента: Mico System Builder (MSB), C / C++ Software Project Environment и дебъгер. Тя позволява изграждането на вградени FPGA базирани системи, създаване и отстраняване на грешки в управляващия софтуер.

Средата за проектиране на Altera Quartus II [32] също позволява бързо придвижване от идея до реализация. Тя предлага цялостно автоматизиране на процеса на създаване и имплементация на ВС, без необходимост от използване на описания на ниско или схемно ниво. Позволява също така интеграция с инструментариуми на водещи EDA производители (Mentor Graphics и Synplicity).

Xilinx предлага различни средства и платформи за разработка. Платформата XtremeDSP Development Kit служи за професионална разработка на проекти на базата на FPGA прибори от серията Virtex-II

[7]. Тя включва инструментални средства от типа MATLAB/SIMULINK на фирмата MathWorks, System Generator и др. Пакетите за програмно осигуряване на Xilinx ISE и WebPack допускат интеграция със средата за развой на други EDA производители Synplify Pro на фирмата Synplicity и FPGA Advantage на фирмата Mentor Graphics [65]. Xilinx отделя внимание и на разработката на взаимно допълващи се технологии. Пример за това е пакетът AccelDSP Synthesis, който предоставя апаратна реализация на алгоритми с плаваща запетая, създадени в пакета MATLAB.

За проектиране на вградени микропроцесорни системи, Xilinx предлага средата Embedded Development Kit (EDK). Тя подпомага разработката и настройката на апаратната и програмната части на микропроцесорни системи, ползващи ядрата MicroBlaze или PowerPC при вграждане в семействата Spartan и Virtex. Характерните особености на тази среда са :

● възможност за съвместна разработка и настройка на програмната и апаратната части :

● поддържка на различни способи за описание на апаратната част на разработваната система и тясна интеграция със стандартните средства за разработка ISE WebPack и ISE Foundation;

● наличие на библиотеки съдържащи IP ядра на различни компоненти.

● възможност за верификация на апаратната част в средата ModelSim;

В състава на средата Xilinx Embedded Development Kit влизат:

● средства за разработка на вградени системи (ВС) Embedded System Tools (EST), библиотека с IP компоненти за вградени микропроцесорни системи, както и комплект от драйвери и библиотеки към средствата за разработка на ПО проектируемите системи, С компилатор и средства за настройка на програмите, примерни проекти и документация.

Средствата за разработка EST на Xilinx са комплекс от инструменти, реализирани като програмни модули. Те са предназначени за определени етапи от проектирането на вградени микропроцесорни системи. Разпределят се в две групи. Първата група обхваща програми, необходими за проектиране на апаратната част на разработваната система. Те позволяват създаване, редактиране и формиране на спецификациите, HDL описанията и моделите на апаратната част. Втората група обхваща средства, необходими за проектиране на програмната част на разработваната система. Те позволяват създаване, редактиране и формиране на спецификации, библиотеки и драйвери за програмната платформа на проектираната система.

Средата EDK поддържа инструментариуми и развойни платки на фирмите Avnet, Digilent, Memec и Xilinx.

Xilinx не предлага специални среди за разработка на 8 разрядните вградени системи, базирани на ядрото PicoBlaze. За разработката на апаратната част се използват инструментариума на пакета Webpack.

Като средства за развой на ПО (програмно осигуряване) се предлага само асемблер. За проверка и настройка се ползват интегрираните среди pBlazeIDE или openPICIDE. Те се предлагат от трети фирми. Асемблерът за този процесор предлага възможност за откриване и отстраняване на синтактически грешки, но не може да оцени съответствието на нейното функциониране съобразно желания алгоритъм. Недостатъчно ефективно за настройка на програмите е използването на инструментариумите Xilinx или ModelSim. Те не предлагат нагледна форма за представяне на резултатите от моделирането на програмата. Изискват тя да бъде готова преди процеса на проектиране на апаратната платформа.

По тази причина разработката на програмно осигуряване на вградени PicoBlaze базирани системи не може да мине без използването на

средите pBlazeIDE или openPICIDE. Те поддържат командите и директивите на асемблера за ядрото PicoBlaze и позволяват интуитивен режим на работа, което намалява общото време за разработка.

За Xilinx решението на проблемите в проектирането и разработката на нови средства за проектиране се свежда до установяване на стабилно партньорство със заинтересовани от нововъведения EDA производители в рамките на специална програма. Тази програма се нарича "ESL инициатива"(Electronic System Level, ESL). Нейната основна цел се свежда до разработка на нови програмни средства от системно ниво, които да направят методологията за проектиране максимално "близка" за проектанта, сходна с тази, използвана при програмиране на езика С.

В рамките на програмата ESL важно място заема обучението и получаването навици за работа със средствата за проектиране от високо ниво, които позволяват на проектантите леко да реализират своите FPGA базирани проекти. За тази цел фирмите, доставящи ESL средствата за проектиране предлагат широк избор от взаимно допълващи се решения, оптимизирани спрямо различни приложения, платформи и крайни потребители. В основата си това са средства, ориентирани към преобразуване на C/C++ разработки във вградени, апаратно реализирани FPGA процесори и системи. В този смисъл техните усилия се оказват съсредоточени към създаването на платформа, ориентирана към проектантите на програмно осигуряване.

Неприятна страна в инициативата ESL е нейната ориентация към проекти, изискващи значителни апаратни ресурси, което поставя под въпрос нейната ефективност при реализиране на проекти с малки вградени процесори [61,62].

Независимо от това, етапите на проектиране, формализация и автоматизация на ВС на високо ниво придобиват все по-голямо

разпространение. Създават се архитектурни абстракции, в рамките на които се постига обединение на традиционните апаратни и програмни съставящи на ВС. Действието на архитектурните абстракции обхваща всички обекти от ниско системотехническо ниво до тези, отговарящи за общуването с потребителя. Те позволяват създаването на ефективни езици за общуване в областта, занимаваща се с проектиране на ВС, подобряват технологиите и качеството на техническата документация.

В тази насока, реализирането на програмна среда за автоматизирано проектиране на вградени, PicoBlaze базирани системи, посредством представянето им във форма с повишена степен на абстрактност, има своето място и значение в процеса на автоматизация на инженерния труд. Тя позволява да се постигне ускоряване на процеса на реализиране на разработки, използващи това микропроцесорно ядро.

1.4.2. Средства за верификация

Проблемът с времето за функционална верификация и настройка добива изключителна актуалност в случаите, когато за обследване на разработка, включваща едно или няколко микропроцесорни ядра, разположени на един кристал се използват отделни средства за апаратната и програмните й модули [119].

Традиционният подход в това направление е универсализация на средствата за настройка на апаратната и програмната части на проекта. Универсализацията намалява стойността на средствата за настройка и значително съкращава времето за обучение при преминаване от една архитектура в друга.

Съвременното разбиране за минимизиране на времето за верификация и настройка се базира на методология, която позволява едновременно

провеждане на разработка, верификация и настройка на програмното и апаратното осигуряване на всички етапи на разработката, като се започне от идеята и ескизното проектиране до самата физическа реализация.

Използва езици и инструменти за описание и генериране на модели на процесорни ядра и периферийни устройства, изграждани в рамките на кристала. Поддържа интеграция на външни компилатори, симулатори, синтезатори, средства за редактиране и HDL езици. Допуска използването на модели от високо ниво. Позволява разпаралелване на работата и използване на външни устройства за симулация и за визуализация. Осигурява безшевна интеграция с програмното осигуряване, работещо на най-ниски нива. Взаимодейства с външни апаратни системи.

Тази методология гарантира на проектантите на вградени FPGA базирани системи значително ускорение на процеса на верификация и настройка на програмното и апаратното осигуряване.

1.5. Обобщения и изводи

Възможността на една електронна схема да изпълнява няколко различни функции представлява основният атрибут, който определя нейната многофункционалност. Тя позволява да се разшири кръгът на решаваните от нея задачи, което наложи разглеждането й при проектирането на електронни схеми в рамките на тази глава. Показано е, че основните фактори, които определят многофункционалността на електроните схеми са подходите за проектиране, елементната база и развойните средства за тяхното реализиране. На базата на примери от развитието на елементната база е показано, че доминираща методика за увеличаване на многофункционалността се свежда до въвеждане на

структурен излишък и увеличение на броя на интегрираните еднотипни елементи.

В главата е разгледана ролята на препрограмируемите схеми от типа FPGA за повишаване на многофункционалността на електронните схеми. Показано е, че използването на FPGA като алтернатива на микропроцесорите, позволява да се минимизират 10 от 16 най-възможни рискове, свързани с внедряването им в системи с критични приложения, работещи в екстремални условия, атомна енергетика, космически апарати, жп транспорт и други.

Показана е възможността за използване на FPGA приборите като среда за изграждане на различни по сложност и бързодействие процесорни ядра и SoC.

Посочени са дефиниции на понятията „вграден процесор" и „вградена система". Обяснено е, че те се отнасят до изградени в рамките на един единствен FPGA чип процесорно ядро, набор периферии, памет, интерфейси към външни памети или устройства и отразяват характера на задачи, които в по-голямата си част са свързани с изпълнение на специализирани фонови функции, незабележими за човека при условие на коректно изпълнение.

Показана е необходимостта от използване на съвременната методология за проектиране и синтез на електронни схеми с повишена многофункционалност от високо ниво на базата на езици VHDL и Verilog.

Анализирани са съвременните методологии за проектиране и синтез на електронни схеми с повишена многофункционалност. Показано е тяхното влияние върху качеството и продължителността на разработка на апаратно и програмно осигуряване и необходимостта от използването на HDL-базирани средства за целите на моделиране и верификация с цел намаляване на грешки.

Направен е преглед на развойни средства за проектиране на вградени системи на фирмите Lattice, Altera и Xilinx предлагащи FPGA прибори. Отбелязано е наличието на изоставане на възможностите на средите за изграждане на ВС спрямо тези на средствата за проектиране (EDA - Electronic Design Automation).

Описани са показатели на средствата за проектиране на вградени микропроцесорни системи с ядрата MicroBlaze или PowerPC, предлагани от Xilinx, при вграждане в семействата Spartan и Virtex.

Посочено е отсъствието на среда за разработка на 8 разрядните вградени системи, базирани на ядрото PicoBlaze в средствата за развой на Xilinx. Обоснована е необходимостта от създаване на специализирана апаратно-програмна среда за автоматизирано проектиране на вградени, PicoBlaze базирани системи, която съдържа архитектурни абстракции, обхващащи всички обекти от ниско системотехническо ниво до тези отговарящи за общуването с потребителя.

Изследванията направени в тази глава са представени в публикации [2,4] от списъка с публикациите свързани с дисертацията

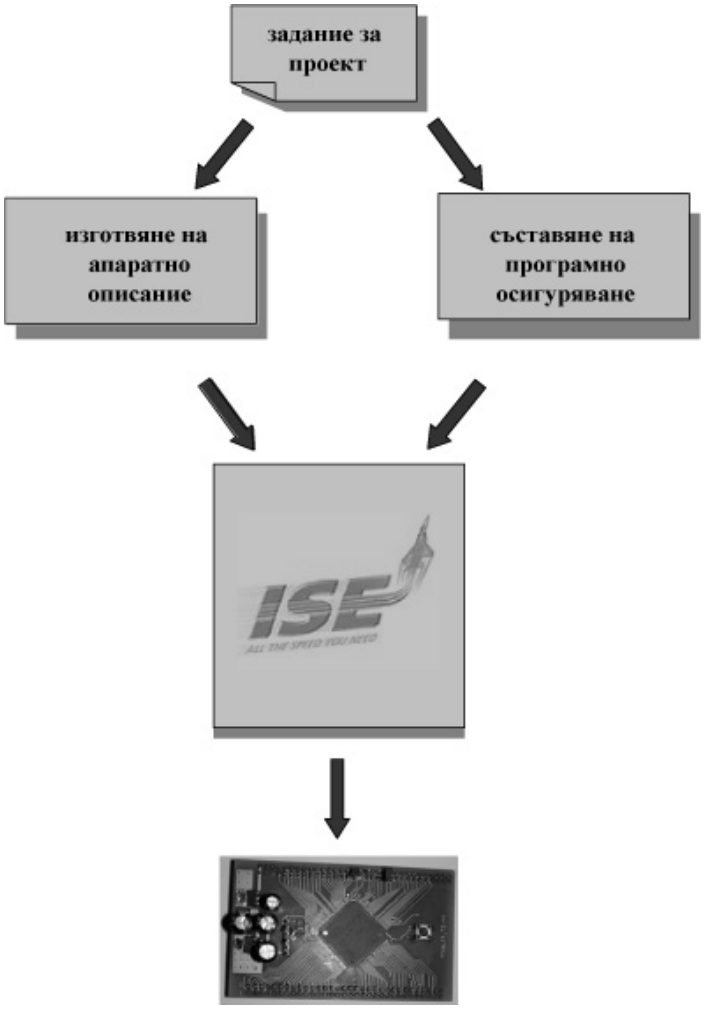
Глава 2 Разработване на програмна среда за синтез на PicoBlaze базирани устройства.

Процесът на проектиране на завършени, апаратно-програмни вградени системи, използващи като среда за реализиране FPGA прибори, представлява много планова задача.

Класическият алгоритъм за тяхната разработка е показан на фигура 2.1. Този алгоритъм се ползва от много проектанти. Той не отразява етапа на предварителната обработка на изискванията и особеностите на реализирания проект [66]. Процесът на предварителната обработка е задача, по чието решение системният инженер на проекта изготвя изискванията към апаратното и програмното осигуряване на проектираната система.

Решението на тази задача изисква от системния инженер на проекта да постигне пълно обхващане на всички дейности, свързани със систематизирането и представянето на проекта в рамките на отделни задачи.

Броят на екипите, които системният инженер трябва да формира за реализирането на проекта е в пряка зависимост от неговата сложност. При проекти с малка сложност проектирането се поема от един изпълнител. В останалите случаи дейностите се поемат от три екипа. Единият отговаря за вградения процесор и необходимата периферия, вторият за програмното осигуряване, а третият за изграждането на неговата логистична среда.



Фигура 2.1. Опростен алгоритъм за разработка на практически приложения базирани на вградени процесори

Работата на екипа, отговарящ за апаратната част на проекта започва с въвеждане и моделиране на структурата на разработваната система и изграждащите я модули.

За въвеждане на проекта обикновено се използват езици от високо ниво (VHDL, Verilog, SystemVerilog или System C) или принципни схеми.

Моделирането се извършва в рамките на средствата предлагани от използваната развойна система (XST на Xilinx) или такива на трети фирми, например ModelSim [65].

Работата на втория екип, отговарящ за програмното осигуряване, е тясно свързана с поведението на функционално завършените модули, формиращи структурата на проекта. На тази база екипът подбира най-подходящия начин за програмиране на дейностите и поведението на отделните модули на проекта, който гарантира поместването на кода в обема на предлаганата от конкретния FPGA чип памет.

Работата на екипа, отговарящ за логистичната среда на проекта, обхваща всички дейности по избора на входно-изходните сигнали, извършването на оценки за тяхното качество, разпределението им по изводите на чипа, физическото му позициониране върху печатната платка, нейната реална изработка и др.

Тези дейности имат итеративен характер и се изпълняват до окончателното завършване на топологията на печатната платка при съблюдаване на поставените изисквания от гледна точка на импедансно съгласуване, появата на паразитни EMI излъчвания и прослушвания между сигналите. Едва след това може да се пристъпи към физическото проектиране и реализиране на печатната платка на проекта.

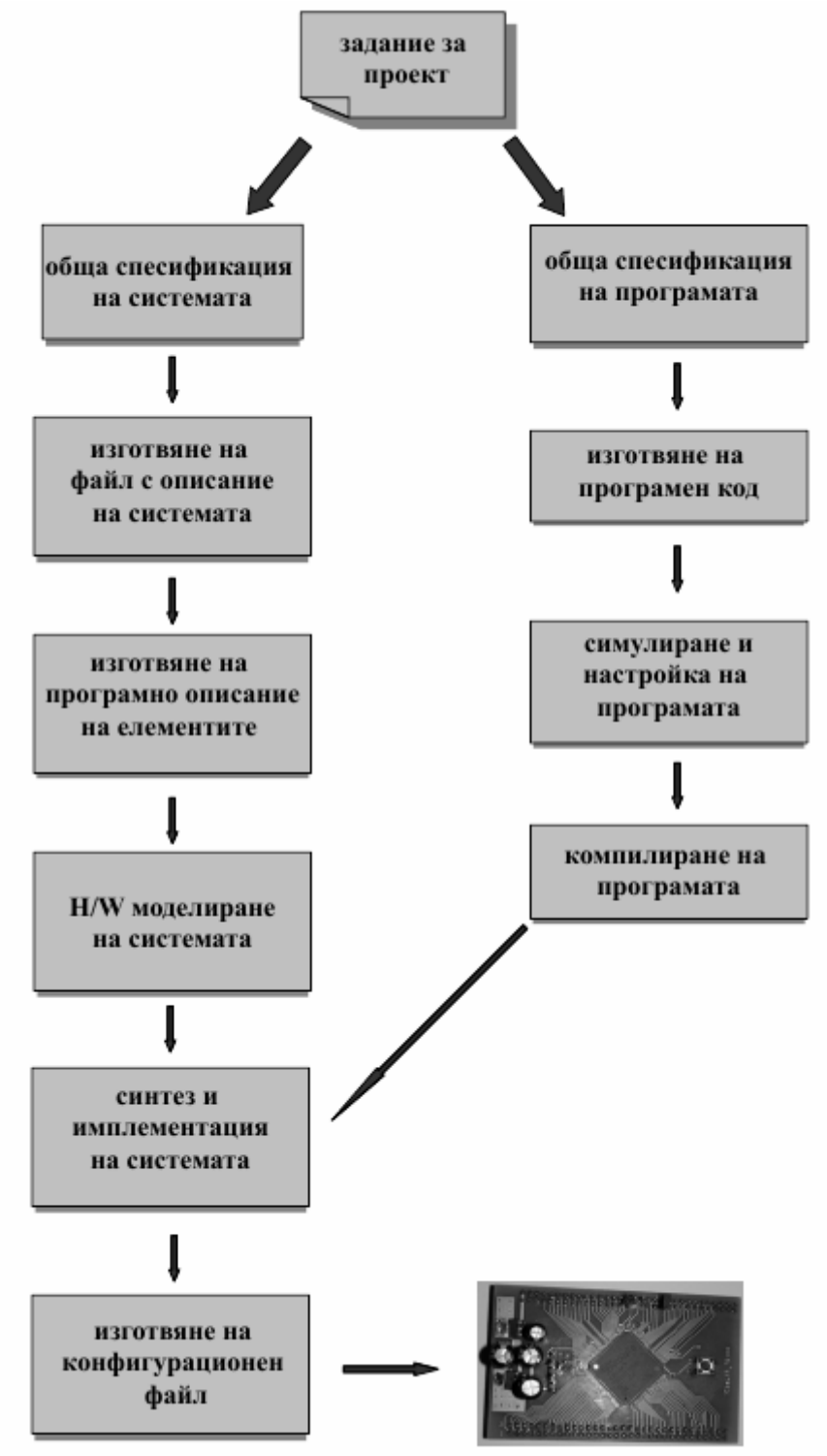
Последният етап от процеса на реализиране на един проект на базата на вградени процесори, представлява неговата верификация. Този етап от реализирането на проекта е от особена важност тъй като изисква време, съизмеримо с това на самото проектиране [66]. На тази основа реалната последователност за проектиране на вградена система добива вида, показан на фигура 2.2. Обикновено тя се изпълнява ръчно. Този начин на работа не изключва появата на грешки, забавящи цялостния процес на проектиране. По тази причина, всички опити за създаване на ефективни средства, допринасящи за ускоряване на процеса на проектиране, срещат разбиране и практическа поддръжка.

За подобряване на процеса на проектиране на вградени системи е необходимо използването на комплексен подход. Той трябва да

обедини традиционите апаратни и програмни съставящи на системата в набор от архитектурни абстракции, които да обхванат цялото практическо разнообразие от потенциално възможните варианти на архитектурата и инфраструктурата на разработваната система. Прилагането на комплексен подход позволява използването на готови апаратно програмни платформи, унифицира създаването на HW и SW компоненти и канализира проектирането на вградени системи в следните направления :

- развитие на технологиите за HW/SW co-design;
- повторно използване на компоненти;
- проверка за достоверност и верификация;
- създаване на средства за моделиране на нефункционални свойства (надеждност, консумирана мощност, габарити и др.);

Тези направления формират функционалността на системата и показват необходимостта от проектиране с използване на независими спрямо елементната база абстракции от високо ниво.



Фигура 2.2. Детайлен алгоритъм за разработка на вградена система

2.1. Програма за проектиране на PicoBlaze базирани системи

В качеството на основа за изграждане на разработената в дисертациония труд програма за автоматизирано проектиране на PicoBlaze базирани системи се използва създадената в ИППИ-РАН теория за малки предметни области (МПО) [67,68]. Тя е разработена от В.С. Файн за целите и задачите на изкуствения интелект.

Съгласно теорията за МПО, решението на една задача може да се получи чрез представяне и моделиране в абстрактно пространство, формирано от МПО. Тази теория гарантира трансформирането на безкрайното множество от задачи в крайно множество от взаимно независими МПО [67].

Използването на тази теория при разработването на PicoBlaze базирани системи изисква наличие на независими МПО модули. За доказване на независимостта на отделните МПО модули, използвани при изграждането на PicoBlaze базирани системи е използвана математическата концепция "алгебрична система" [50,51].

Това е абстрактна конструкция, която позволява обхващане на цялото многообразие на отличителните белези на всеки един член на множеството от МПО. Тя позволява да се развият и навлязат в различни области на познанието абстрактните клонове на съвременната математика. По определение [48], представата за алгебрична система се дава като подредена тройка обекти :

$$U \Leftrightarrow \langle A, O_f, O_p \rangle$$

Където с U е означението за алгебрична система, A е множеството, върху което е определена алгебричната система, O_f е множество от операции, дефинирани в A, O_p е множество от ограничения дефинирани в A. По определение[48] множество се нарича обединение на предмети, данни от опити или други обекти на мисленето, еднозначно определени чрез дефинирани условия, признаци и свойства.

Основен елемент, еднозначно определящ съществуването на една алгебрична система е множество (наричано още като носител), в рамките на което са дефинирани всички апаратни особености, входни въздействия и изходни реакции на моделираната система. В зависимост от техният тип, ролята на носител за една реална алгебрична система може да се изпълнява от множеството на реалните числа{ R }, множеството на целите числа { Z }, множеството на комплексните числа { C } или някои от техните подмножества. Така, съобразно условията за съществуване на алгебрична система, изборът на носител еднозначно определя типа на математическите операции и логическите отношения.

В конкретния случай, предвид типа и характера на градивните блокове на разработваната система, типът на носителя трябва да бъде множеството на целите числа { Z }. Следователно, всеки един от МПО модулите ще представлява алгебричната система от целочислен вид и ще се описва с израза :

$$U \Leftrightarrow \langle Z, Z_f, Z_p \rangle$$

Във всички останали случаи представянето се явява некоректно, защото изискването за затвореност на математическите операции няма да се съблюдава и ще се получи противоречие с постулатите за съществуване на алгебра.

Този избор автоматично определя съдържанието на множеството Z_f, включващо правилата за действие на конкретния модул, с оглед реализирането му като реално функциониращо устройство.

Що се отнася до множеството от ограничения Z_p, то се генерира от спецификата на конкретния модул и включва в себе си всички онези

особености, които могат да възникнат при практическото реализиране на една система.

Така се стига до генерирането на две нови множества: $Z_f(i)$ и $Z_p(i)$ отразяващи правилата за действие и ограниченията на множеството от МПО модули. Вследствие специфичните особености на правилата за действие и ограниченията на отделните МПО модули, за веки два произволно избрани члена на тези множества може да се запише:

$$Z_f(i) * Z_f(j) = 0$$
$$Z_p(i) * Z_p(j) = 0,$$

където * е знак за скаларно произведение.

Тези изрази показват взаимната независимост на елементите от множествата $Z_f(i)$ и $Z_p(i)$, следствие на което всеки един МПО модул се оказва капсулиран. В случаите, когато тези изрази не се изпълняват, съответните МПО се оказват съвпадащи, което противоречи на първоначалното предположение че МПО са различни.

Това доказва приложимостта на теорията на МПО за генериране на PicoBlaze базирани системи.

Прилагането на теорията за МПО позволява разработваната система да използва математически модели в технологиите за HW/SW co-design на системните компоненти. Позволява проверки за достоверност и верификация и повторно използване на предварително създаден и проверен код. Така съществуващи проекти могат да се разширяват или модифицират без повторно моделиране. Елиминират се всички, незабележими в началните етапи на проектирането неопределености, чиито последствия се откриват едва в последните стадии на процеса на проектиране. Така се постига минимизиране на броя на итерациите,

възникващи в процеса на проектиране, които определят неговата продължителност.

В теоретичен аспект, принципите за решаване на задачата за автоматизирано проектиране на вградени PicoBlaze базирани системи са два: евристичен и имитационен [67].

Евристичният принцип намира приложение при решаването на произволни, некоректно зададени задачи. При него, чрез внасяне на допълнителна информация и данни, несъществуващи в изходните условия, се извършва до определяне на поставената задача. Ето защо, творческата функция за тяхното откриване, предварителна оценка и въвеждане все още не може да се автоматизира и се извършва от човек. Приложението на евристичния принцип има място при въвеждане в употреба на нови вградени процесори или, когато с тях се усвояват нови области на приложение. В останалите случаи, доминираща роля има имитационният принцип.

При имитационния принцип, подходът за решение на всяка възникнала задача изисква да се изясни решавана ли е вече подобна задача. Когато резултатът от извършената проверка е положителен, това означава че може да се използва вече известен алгоритъм като му се подадат аргументите от новата задача. В този случай, имитационният принцип позволява задачата за автоматизирано проектиране на вградени PicoBlaze базирани системи да се сведе до направата на подбор на нови МПО, съобразени с началните условия на конкретната задача.

2.2. Изграждане на среда за автоматизирано проектиране на PicoBlaze базирани системи

На практика изборът на имитационния или на евристичния принцип за решаването на конкретната задача зависи от заложения в нейното формулиране смисъл.

По своята същност понятието смисъл е абстрактната представа, отъждествяваща се със сложна структура, която решаващият задачата човек трябва да изгради в своя мозък, с оглед реализиране на една сложна, разностранна реакция с всичките нейни делови, емоционални, логически, асоциативни и много други компоненти [67].

Според тази концепция, независимо от начина за нейното представяне физическата формулировка на задачата, не съдържа в себе си това, което може да се нарече СМИСЪЛ. По тази причина, физическата формулировка на задачата обикновено представлява израз на определена цел, преследвана от нейния автор, а формирането в мозъка на получателя на формулировката "смисъл", винаги се явява само средство за изработване на последваща адекватна реакция.

Казано с други думи, формулировка на задачата не съдържа и не предава смисъл, а се явява само инструмент за съобщаване на получателя каква реакция иска да получи от него авторът на представената формулировка [67].

Това е причината, поради която всички начални етапи от проектирането на вградени системи все още не могат да се извършват автоматично, без човешка намеса.

В този смисъл творческата функция по избор на необходимите за изграждане на една вградена PicoBlaze базирана система МПО, които гарантират точно и еднозначно решение на поставената задача, все още не може да се формализира и се извършва от системния инженер. По

тази причина, въпросът за пълно автоматизиране на началния етап от процедурата за проектиране на вградени системи не може да се осъществи цялостно. Определено приближение до автоматизиране на процедурата за проектиране може да се постигне в случаите, когато броят на МПО е много малък и позволява използването на таблици. За останалите аспекти от процеса на проектиране на вградени системи, отнасящи се до изграждането на тяхната логистична среда, програмното осигуряване и верификация, предвид итеративният им характер и различните целеви функции, преследващи специфични за всяка една от тях цели, може да се реализира частична автоматизация.

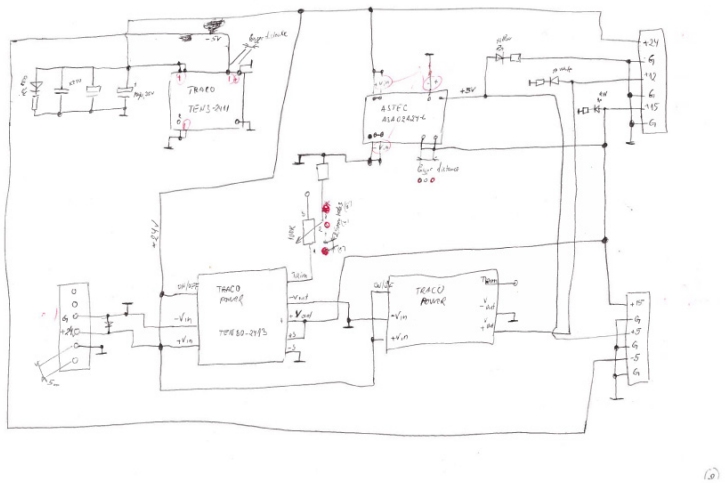
Следващата трудност, възникваща при изграждане на една система, базирана на вградени процесори, след приключване на избора на необходимите МПО, е свързана със съставянето на конфигурационен файл, които отразява описанието на връзките между отделните компоненти и модули на системата. Съставянето на този файл представлява първата стъпка от общата последователност при проектирането на една вградена система.

Правилно съставяне на конфигурациония файл е от особена важност за началните етапи от изграждането на всеки един проект. Той налага определена подредба в процеса на конструиране и формира структурата на проекта, като съвкупност от МПО обекти, способни да обменят информация съобразно правила и ограничения.

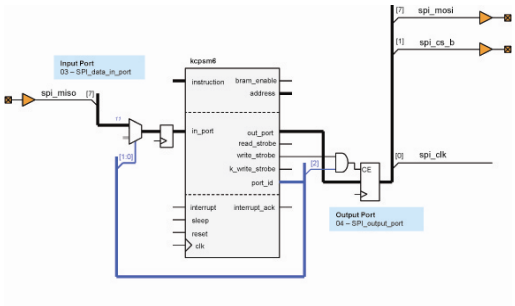
При съставянето на конфигурационния файл проектантът не се интересува какво точно има в модулите. Неговата задача е точно физическо описание на връзките между отделните модули на системата. Независимо от това, при разработката на практически вградени системи, изграждането на файла, отразяващ описанието на връзките между отделните компоненти на системата е трудна задача,

както за начинаещи, така и за онези проектанти които имат определен опит в тази насока.

Крайната форма на този файл е текст, отразяващ конкретното описание на системата в термините на език от високо ниво. Тази дейност се извършва ръчно, с помощта на текстов редактор. За съставянето на този файл, се използват два начина. При първия като отправна точка се избира файла на друг проект, който се използва в качеството на шаблон. Този файл се записва под друго име и върху него се нанасят съответните промени и корекции. След няколко итерации се стига до желания резултат. При втория начин, описанието на връзките между компонентите на системата се изготвя директно, без използването на шаблон. Той е по-труден и се ползва от проектанти с по-голям опит. И при двата подхода съставянето на файла не е гарантирано срещу възникване на грешки. Обикновено, тяхното откриване става едва в процеса на синтезиране на проекта. Това усложнява процеса на проектиране, и води до по-голям брой итерации за тяхното отстраняване. За минимизиране на броят на грешките при въвеждане е желателно съставянето на скица или пълна схема на разработваните връзки (вж. фигури 2.4, 2.3), които да подпомагат създаването на файла.



Фигура 2.3. Скица на разработвана система.



Фигура 2.4. Подробна схема на разработваната система.

Тяхното наличие позволява на проектанта да обхване и проследи цялата същност на системата само с един поглед. По тази причина изготвянето на скица или схема се явява необходим, задължителен етап за изграждането на файла с описанието на системата. Формулировката на концепцията, залегнала в основата на предлагания в дисертациония труд метод за автоматично генериране на файл, съдържащ описанието на вградена, PicoBlaze базирана системата се свежда до:

„Възможно ли е генерирането на файла с описанието на системата да се извърши непосредствено от нейната блокова схема, която се съставя от проектанта ?”

Положителният отговор на този въпрос изисква наличието на предварително формирано множество, съдържащо отделните МПО. За формирането на това множество бе направен анализ на свободно предлаганите от Xilinx или описани в литературата проекти [70-74]. Списъкът на проектите, използвани за формиране на базовото множество с МПО модули е показан в таблица 2.1

Анализът на разгледаните проекти показва, че за изграждането на една вградена, PicoBlaze базирана система се използва ограничено, постоянно от гледна точка на тип и брой множество от модули. Тези модули формират базовото множество от МПО. В общия случай, представители на това множество са процесор, памет, един или няколко

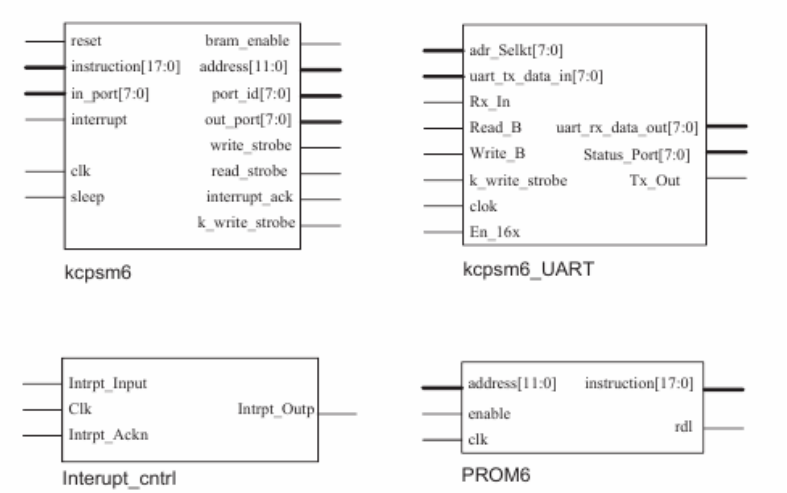
различни модула за серийна комуникация, часовник за реално време, специализирана периферия и др.

Таблица 2.1. Списък на проектите използвани за формиране на множество с МПО модули

Проект	PicoBlaze Процесор	LCD	LED	UART	SPI	I2C	Rotary Encoder
Initial Design for the Spartan-3E FPGA Starter Kit Board	да	да	да	да	да	да	да
Authentication for Spartan-3E FPGAs	да			да		да	да
Rotary Encoder Interface	да	да	да				да
PicoBlaze Процесор A/D Converter	да			да	да	да	
PicoBlaze Процесор D/A Converter	да		да	да		да	да
PicoBlaze RS-232 StrataFlash™ Programmer	да	да		да	да		
PicoBlaze Процесор SPI Flash Programmer	да	да		да	да		
PicoBlaze Процесор Frequency Generator	да	да	да	да			да
PicoBlaze Процесор Frequency Counter	да	да	да	да			да
PicoBlaze Процесор DS2432 Communicator	да	да	да	да			
PicoBlaze Процесор DS2432 SHA-1 Algorithm	да			да			
Pulse Width Modulation with PicoBlaze Процесор	да	да	да	да			да
PicoBlaze Процесор Real Time Clock	да		да	да			

За всеки един от тези модули за целите на дисертационя труд бе разработен съответен графичен примитив, който може да се използва в средите за въвеждане на електронни проекти. Тези модули са обединени в библиотека, оформена съобразно изискванията на средата OrCad. Библиотеката не е затворена и при спазване на определени правила всеки проектант може да добавя към нея свои собствени модули.

Графичните примитиви на основните модули използвани в процеса на създаване на блоковата схема на разработваните системи са показани на фигура 2.5.



Фигура 2.5. Графични примитиви на основни модули.

На базата на това множество и смисъла, заложен в заданието за разработка на вградена PicoBlaze базирана система, системният инженер на проекта, извършва формализация на задачата и изготвя описание на връзките между отделните МПО. Това описание може да се реализира като текст или графика.

С оглед минимизиране на броя на грешките, допускани в процеса на въвеждане, тази процедура в разработената в дисертациония труд програмна среда се извършва по графичен път в рамките на средата OrCad.

Тя позволява въвеждане на различни атрибути към връзките между отделните МПО модули, които в последствие се използват от разработеното програмно осигуряване.

В качеството на илюстрация, на фигура 2.6. е показано графичното представяне на структурната схема на вградена, PicoBlaze базирана системата, реализирана в средата OrCad.

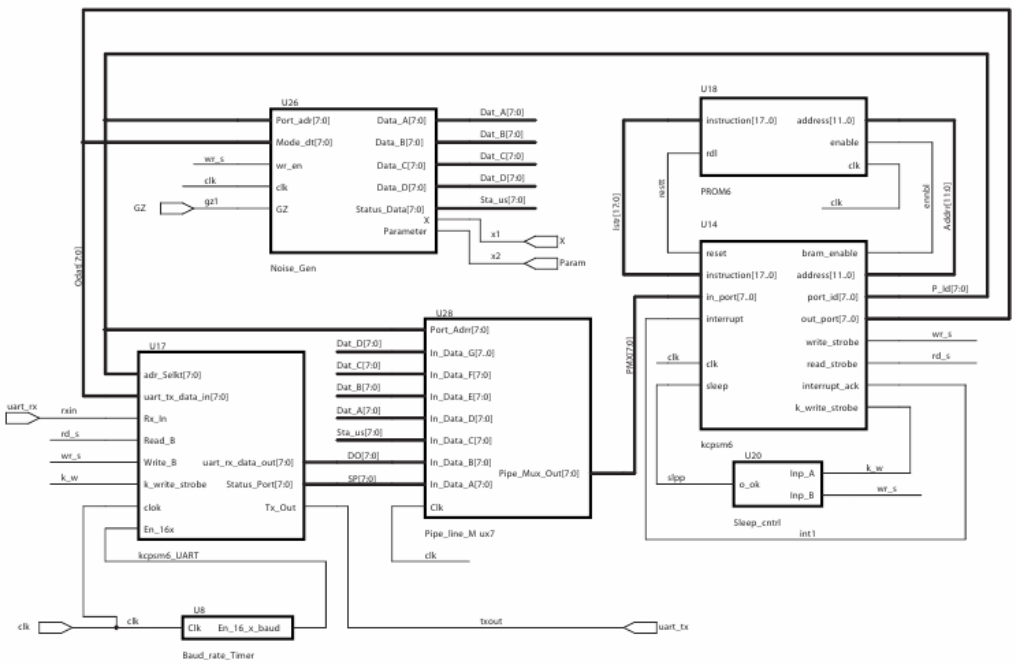
Тази схема се запазва под формата на файл, който може да се използва и за оформяне на конструктивна и потребителска документация. Разработената програмна среда обработва този файл и генерира конфигурационния файл на системата.

Така генерираният файл може да се подаде непосредствено към проект мениджъра на системата за работа с FPGA прибори. Съдържанието на файла, генериран от блоковата схема, представена на фигура 2.6. е приведен в листинг 1. Недостатък на така генерирания файл е подробното описание на връзките в магистралите присъстващи в изразите Port Map.

Наличието на подобен излишък не е проблем за работата на компилатора, но силно затруднява неговото ръчно изследване.

Разработената в дисертацията труд програмна среда елиминира това неудобство. Тя редуцира неговия обем и го подава към средата за работа с FPGA прибори (WebPack). Редуцираният вариант на файла е показан в листинг 2.

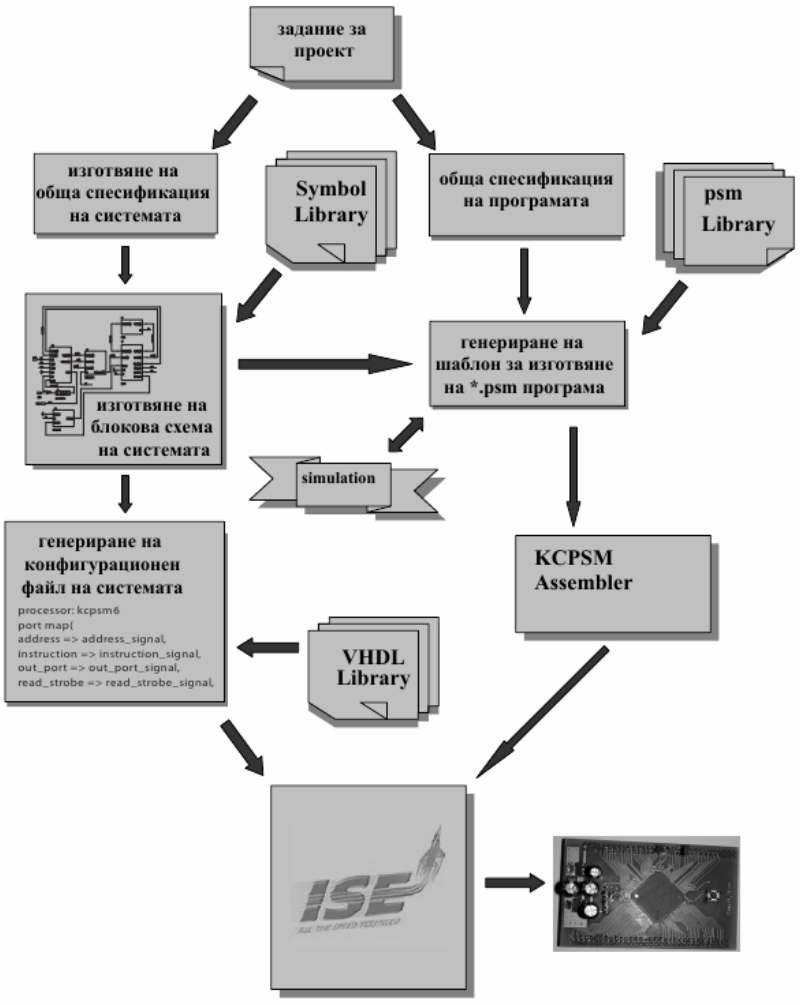
За правилната работа на средата WebPack, освен файла с описанието на проектираната система, е необходимо да се добавят апаратното VHDL описание на отделните МПО модули, и модул съдържащ програмата за работа на вградения микропроцесор.



Фигура 2.6. Графично представяне на вградена, PicoBlaze-6 базирана система.

VHDL описанието на МПО модулите се изготвят предварително при създаването на всеки един от тях. Така те формират втора системна библиотека, необходима за правилната работа на програмната среда за автоматизирано проектиране на вградени PicoBlaze базирани системи.

Характерна особеност за голяма част от МПО, използвани за формиране на вградени PicoBlaze базирани системи, представлява необходимостта от малки програми (драйвери), управляващи тяхната работа в обкръжението на използвания процесор. Тези програми се пишат на асемблер за процесора PicoBlaze и на практика формират трета библиотека, необходима за правилната работа на системата. От тази библиотека драйверите, се вмъкват в тялото на приложния асемблерски код, посредством директивата include. По този начин, концепцията за използването на МПО за изграждане на вградени, PicoBlaze базирани системи, се свежда до алгоритъма представен на фигура 2.7. Разработената програмна среда включва в генерирания конфигурационен файл и UART модул който осигурява възможност за комуникация с външния и редуцира времето, необходимо за верификацията на проекта.



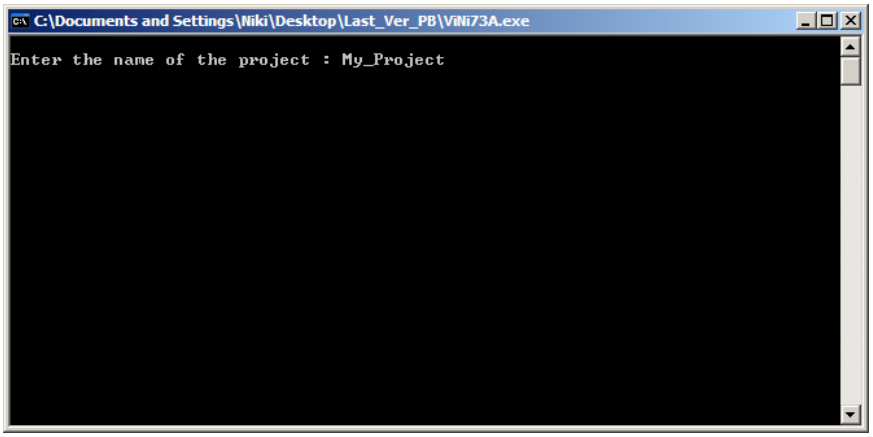
Фигура 2.7. Блоковата схема на разработената автоматизирана среда за генериране на вградени PicoBlaze базирани системи

2.3. Проектиране на вградени системи с средата ViNi73A

Работата с разработената в дисертаациония труд програмната среда за автоматично генериране на вградени PicoBlaze базирани системи започва с разархивирането в определена директория на нейния дистрибутивен файл. След тази операция в директорията се формират изпълнимият файл на програмата ViNi73A.exe и помощният файл PicoBlaze_EDK.rar.

Следващата стъпка от процеса на работа е стартирането на нейния изпълним файл (ViNi73A.exe). Това предизвиква отварянето на

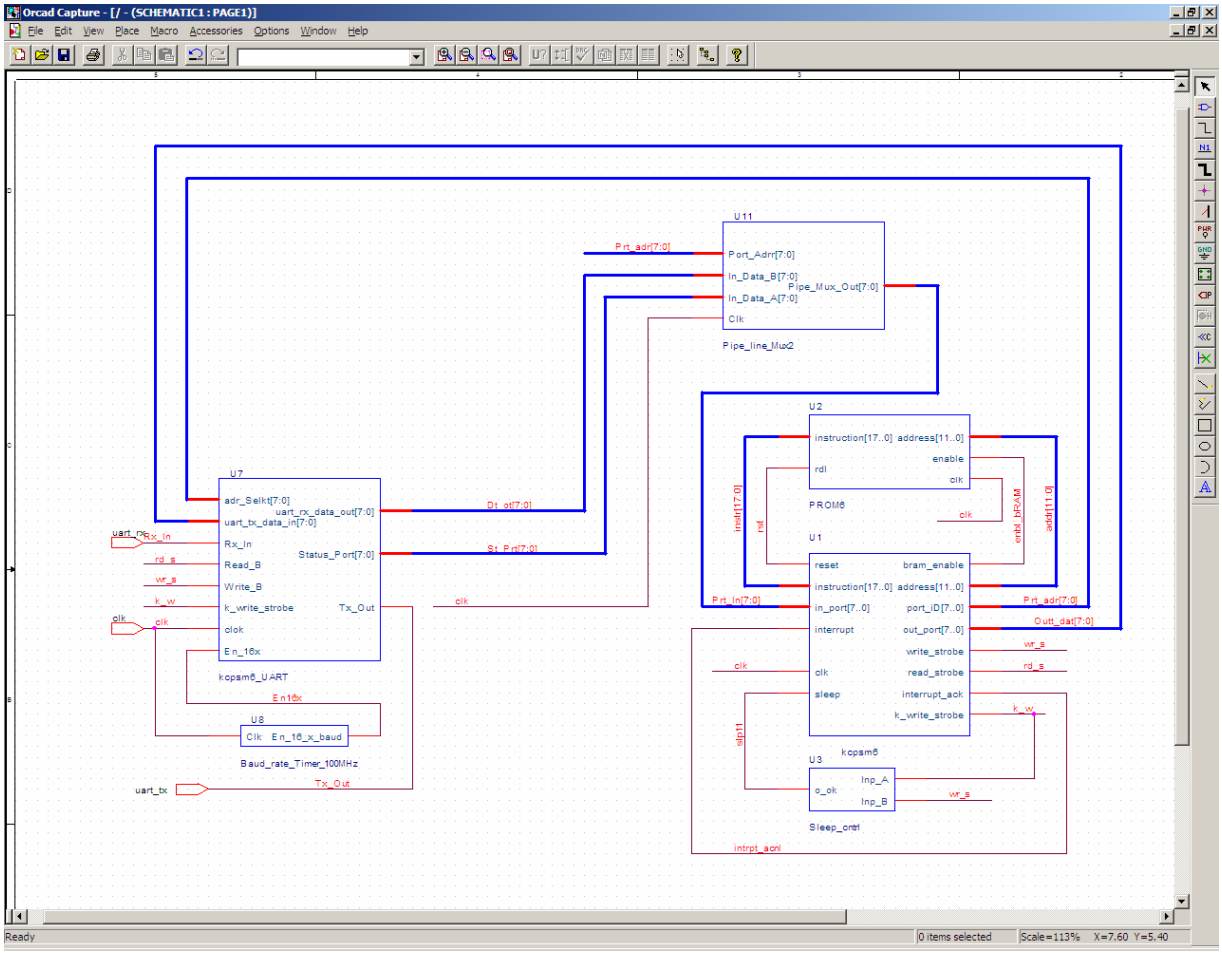
прозорец, в който трябва да се въведе името на разработвания проект (например My_Project) фигура 2.8.



Фигура 2.8. Въвеждане на името на проекта

Под това име програмата създава едноименна работна поддиректория на проекта и стартира средата OrCad, в която тя е генерирала блоковата схема на минимална вградена PicoBlaze базирана система фигура 2.9.

За потребителя тази схема е своеобразен шаблон, който може да бъде променен или допълнен с други модули. Библиотеката, съдържаща графични примитиви се намира в папка на средата. Нанасят се желаните корекции и схемата се съхранява. Натискат се последователно бутоните  и  , разположени в лентата с инструменти. Отваря се прозореца create netlist, от който се избира полето VHDL и се натиска OK. Затваря се средата OrCad. След това в работната директория се генерира под директория с името на проекта (например My_Project) и програмата прекратява своята работа. С това процесът на проектиране на вградена PicoBlaze базирана система е приключил и може да се пристъпи към нейното физическо реализиране.



Фигура 2.9. Блоковата схема на минимална вградена PicoBlaze базирана система

За тази цел е необходимо да се отвори и разгледа съдържанието на създадената под-директория с името на проекта. Тя съдържа изходни и изпълними файлове както и няколко поддиректории, които не трябва да се модифицират.

За физическото реализиране на разработваната вградена PicoBlaze базирана система важно значение имат изходните файлове ***My_file.vhd, My_file_PROM6.vhd, My_file_PROM6.psm, UCF_demo_file.ucf*** и под директориите ***Vova_VHDL*** и ***Vova_PSM***.

Файлът ***My_file.vhd*** е описание на най-горното ниво в йерархията на разработвания проект. Той съдържа цялостното структурно описание на разработвания проект, включващо набора от МПО и връзките между

тях. Файлът ***My_file_PROM6.vhd*** е ***VHDL*** описание на асемблерската програма на процесора. Той се генерира от изпълнимия файл на асемблера kcpsm6.exe. Файлът ***My_file_PROM6. psm*** е шаблон съдържащ описание на асемблерската програма на процесора. Генерира се автоматично от програмата ViNi73A.exe. Файлът UCF_demo_file.ucf съдържа описание на разположението на различните управляващи сигнали по изводите на чипа.

Под-директориите ***Vova_VHDL*** и ***Vova_PSM*** са библиотеки, в които се съдържат файлове с ***VHDL*** описание на предлаганите МПО модули и техните асемблерски драйвери.

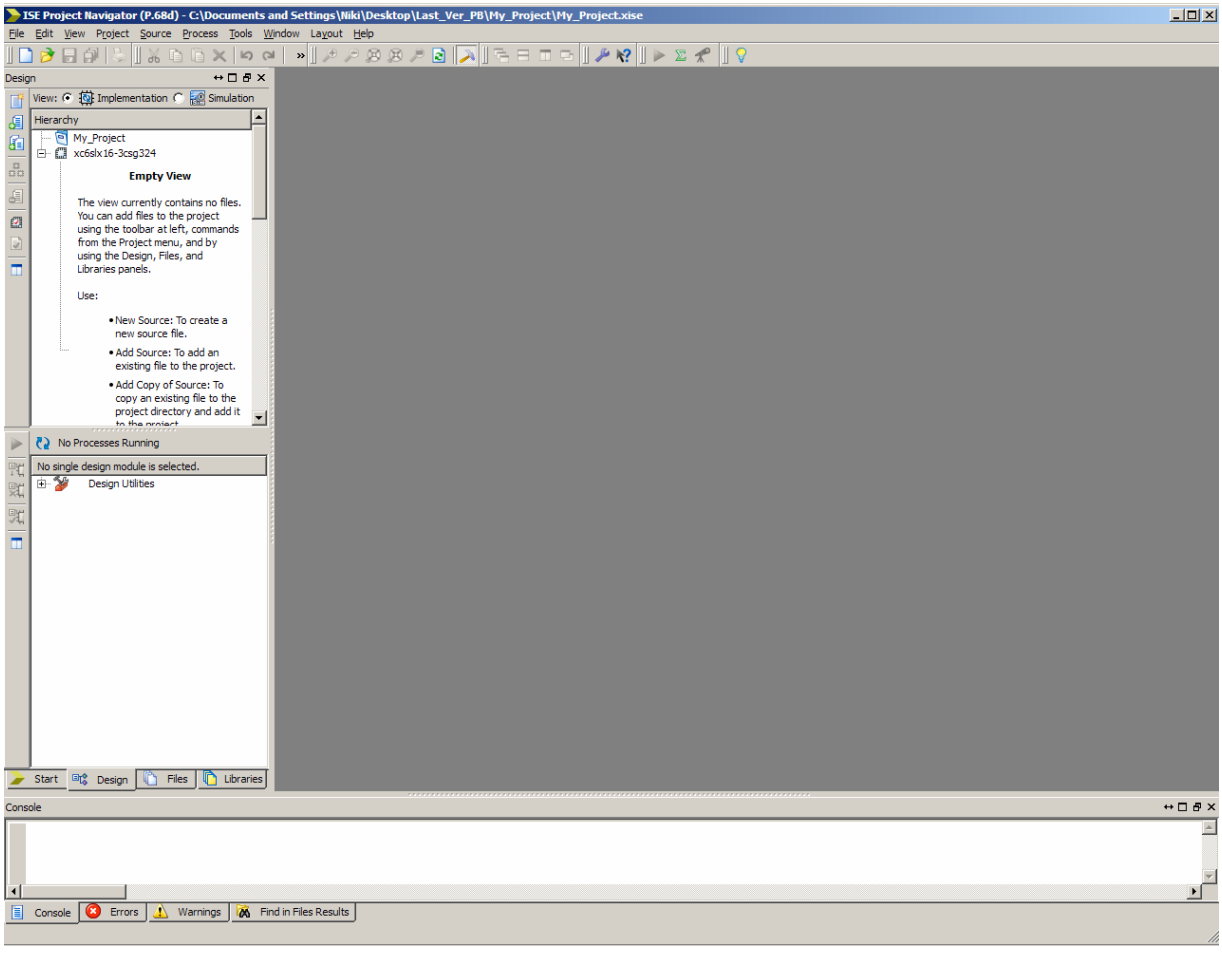
2.3.1. Формиране на йерархията на проекта

Следващият етап от разработката на вградената система е нейното имплементиране. То се извършва в рамките на специализираната среда за работа с FPGA прибори WebPack. За целта от поддиректорията с името на проекта се стартира файлът ***My_Project.xise***. Това е файл, който съдържа описание на разработвания проект съобразно изискванията на средата WebPack. Той се генерира автоматично от програмата ViNi73A.exe.

С неговото стартиране разработваният проект се отваря в работния прозорец на проект-навигатора на средата WebPack фигура 2.10.

Структурата на проекта се формира в прозореца hierarchy на проект-навигатора. На този етап този прозорец съдържа само обща информация за използвания FPGA чип, тъй като структурата на проекта не е отразена в него. Въвеждането на структурата на проекта в Навигатора на проекта (файла ***My_file.vhd***) се извършва със стандартните средства на средата WebPack. След неговото въвеждане структурата на проекта добива вида показан на фигура 2.11.

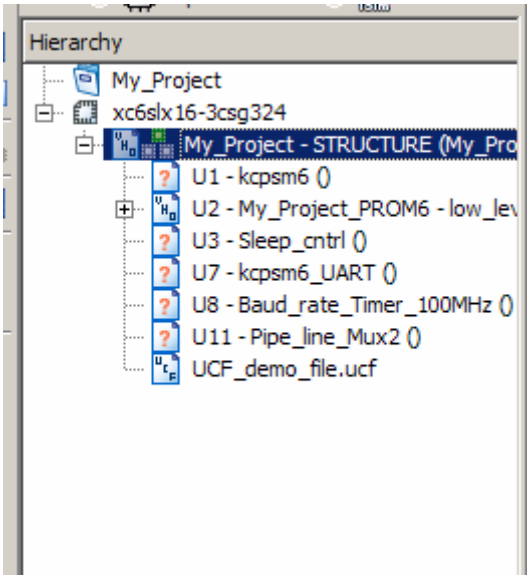
Всички файлове от тази структура, с изключение на главния (My_file.vhd) са неопределени за средата WebPack. Те са маркирани със знака “ ?” и следва да се добавят в системата. Тези файлове се намират в поддиректорията Vova_VHDL, от където техните изходни кодове следва да се добавят към разработвания проект. С оглед да се постигне пълно задаване на параметрите, необходими за правилната работа на средата WebPack към йерархията на проекта следва да се добавят автоматично генерираните от програмата файлове My_file_PROM6.vhd и UCF_demo_file.ucf, намиращи се в работната директория на проекта.



Фигура 2.10. Разработваният проект в прозореца на средата WebPack

Първият съдържа кода на процесорната програма, представена като VHDL модул, а вторият-разпределението на входно изходните сигнали

на проекта по изводите на FPGA чипа. Видът на файла UCF_demo_file.ucf е показан в раздел приложения.



Фигура 2.11. Структурата на проекта в прозореца hierarchy

2.3.2. Синтез на проекта

В процеса на синтез от изходните модули на проекта се формира файл, който съдържа цялата изходна информация, необходима на програмите за моделиране и опроводяване.

Активирането на процеса на синтез се стартира от полето на процесите с избора на команда Process/Run от основното меню или с двукратно натискане на левия бутон на мишката върху наименованието на процедурата Implement Design в прозореца на процесите (фигура 2.12).

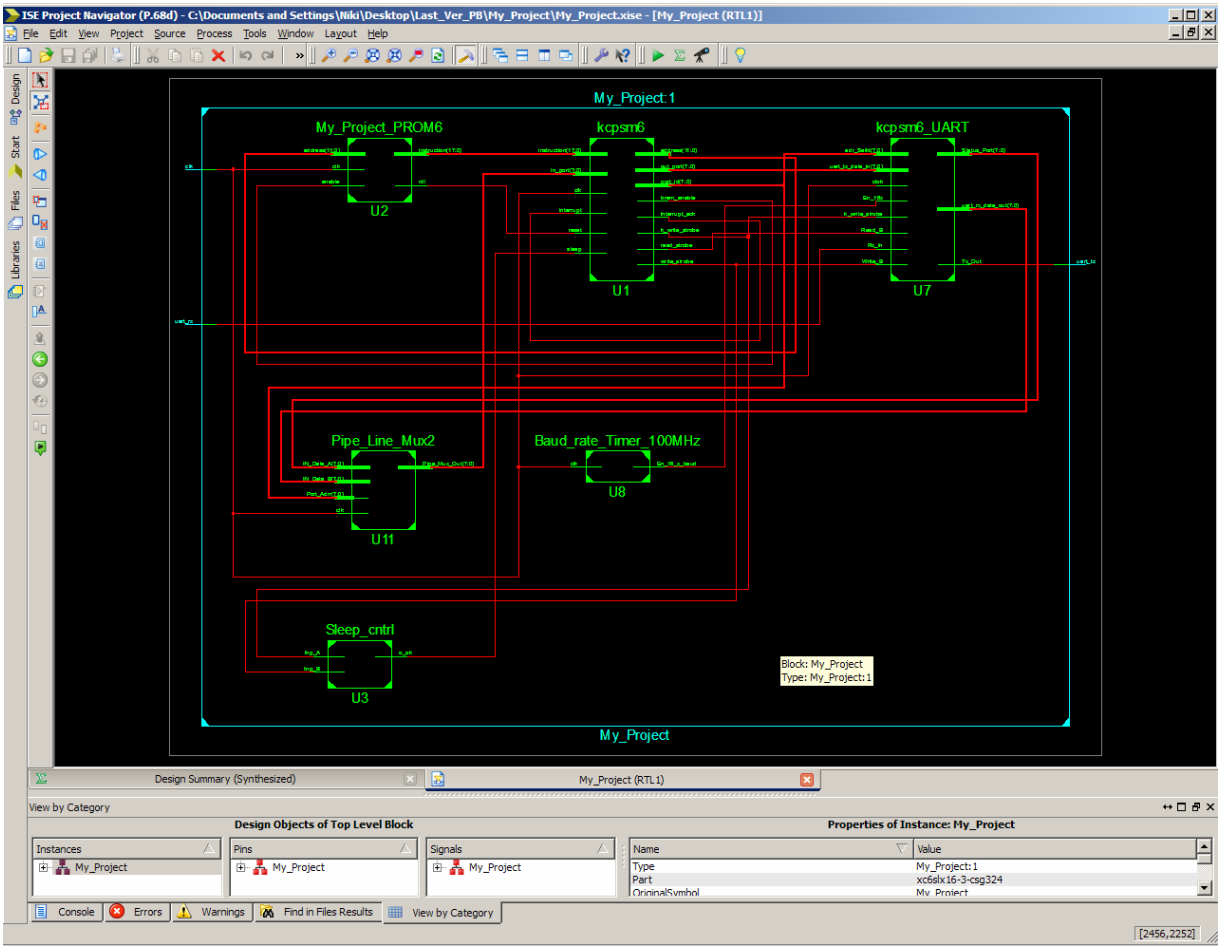
Всички грешки, възникнали в рамките на процеса се показват в прозореца за конзолни съобщения с указание за кода и мястото, където те са възникнали.

Фигура. 2.12. Активиране на процедурата Implement Design

По време на етапа на синтез се извършва опроводяване на проекта в чипа, съобразно изискванията заложени във файла UCF_demo_file.ucf и даденостите на използвания чип. В рамките на този етап се генерират и оценки за реалните стойности на закъсненията на разпространенията на сигналите, необходими за моделиране на устройството.

Основният резултат от етапа на синтез е формирането на *. bit файл, в който се съдържа информация за физическата конфигурация, на проектираното устройство в чипа.

Желателно е след успешното завършване на етапа на синтезиране да се извърши проверка на синтезираната структура на проекта. За целта в прозореца на процесите се отваря реда syntesize и се активира функцията View RTL Schematics. В появилия се прозорец се избира опцията по подразбиране start with a schematic of the top-level block. Маркира се появилият се блок с условно изображение на проекта като цяло и с десен бутон на мишката се отваря нов прозорец, от който се избира опцията show block contents. Появява се структурата на разработвания проект, генерирана от средата WebPack (фигура 2.13). Сравняването на тази структура с изходната, (фигура 2.9.) позволява да се получи ясна представа за съответствието между тях.

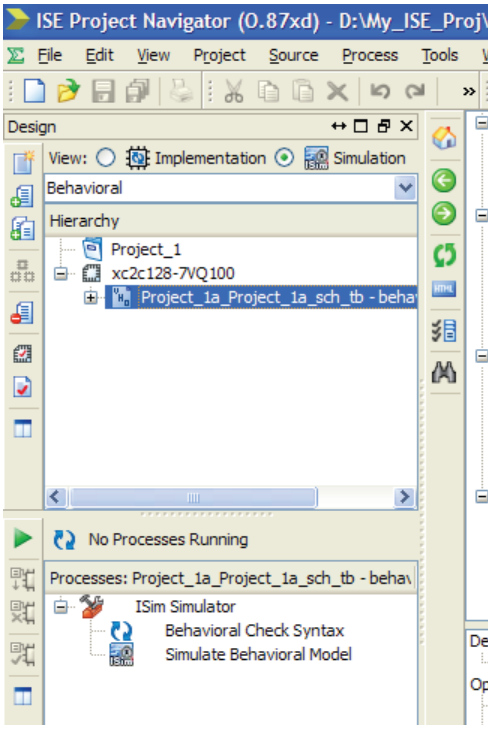


Фигура 2.13. Структура на проекта генерирана от средата WebPack.

2.3.3. Функционална проверка на проекта

Функционалната проверка на проекта се осъществява с помощта на предварително генерирани тестове. Генерирането на тест започва със създаването на файл, в който се записват желаните входни въздействия. За целта е необходимо да се стартира процедурата Project-> New Source. В появилия се прозорец трябва да се укажат името на файла и неговия тип (в случая VHDL Test Bench). След въвеждане на необходимата информация системата генерира шаблон, в който се указват времевите съотношения на входните въздействия. Желаните въздействия се записват на диска.

Активирането на процеса на симулация започва с маркирането на радио бутона simulation, разположен в основния прозорец на проект навигатора (фигура 2.14).

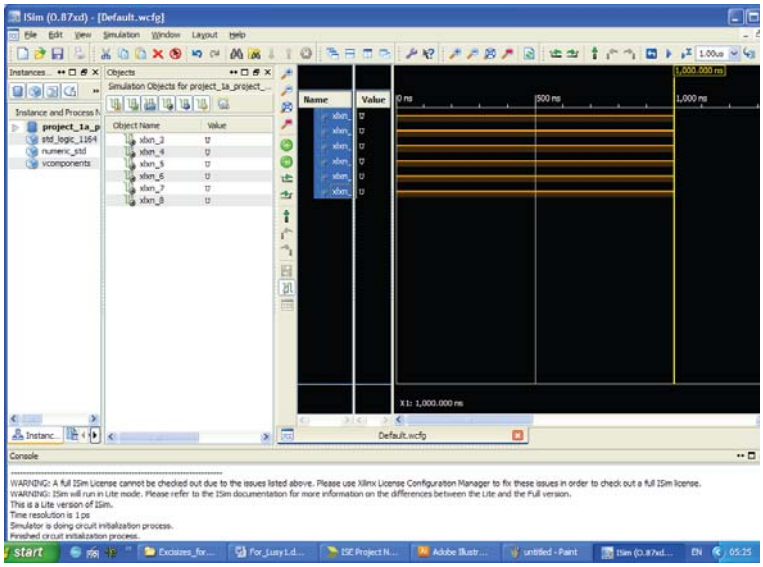


Фигура. 2.14. Активиране на процеса на симулация

В прозореца Hierarchy се маркира реда съдържащ файла с изходните въздействия. Веднага след неговото маркиране в прозореца Processes се появява иконката на симулатора ISim Simulator. Симулаторът предлага възможност за проверка за грешки във файла с входните въздействия (чрез стартиране на процедурата Behavioral Check Syntax). Самата симулация се стартира с активирането на процедурата Simulate Behavioral Model. Непосредствено след нейното активиране се отваря прозорец, в който са визуализирани входните и изходните сигнали, участващи в проекта (фигура 2.15).

За улеснение на анализа на симулацията е желателно да се натисне бутона  , разположен в лентата на инструментите. След успешното

завършване на етапа на моделиране може да се пристъпи към синтезиране на проекта.



Фигура. 2.15. Визуализиране на входните и изходните сигнали на проекта

Завършващ етап от процеса на разработка е зареждането на конфигурационните данни в чипа.

2.3.4. Програмиране на проекта в FPGA чипа.

Завършващ етап от процеса на проектиране на една вградена picoblaze базирана система представлява нейното зареждане в структурата на FPGA чипа. За целта програматорът се включва към съответния порт на компютъра (паралелен или USB) и към специализирания вход на програмируемата платка.

В прозореца на процеси се избира процеса Implement Design. Отваря се структурата на този процес и в появилата се дървовидна структура се стартира пиктограмата, означена като Configure Target Device. Появява се прозорецът на средата iMPACT. От менюто File се избира New. Следват се инструкциите на програмата. След успешното програмиране на чипа, зареждащият кабел се отстранява от изводите за

програмиране и се пристъпва към практическа функционална проверка на проекта с помощта на ресурсите на системата.

2.4. Обобщения и изводи

Използването на средства, притежаващи повишена степен на абстрактност в за проектиране на вградени системи е основен фактор за развитието, усъвършенстването и автоматизирането на този процес. Те осигуряват по-висока степен на адекватност на моделите на елементите изграждащи разработваната система и нейната физическа реализация. Предоставят възможност за автоматизация на процеса на проектиране, като се започне от формализацията и типизация на началните проектни процедури, разработката и изследването на модели, алгоритми, методи и средствата за реализация на отделни проектни решения.

В тази глава е описана, разработената в дисертациония труд автоматизирана програмна среда, предназначена за проектиране на вградени PicoBlase базирани системи.

Решението на тази задача се базира на теорията на МПО и концепцията алгебрична система.

Показано е, че вградените FPGA базирани системи представляват алгебрични системи от целочислен вид.

Показана е много плановата същност и практическите аспекти на процеса на автоматизирано проектиране на завършени, вградени FPGA базирани системи.

Представени са особеностите и практическите аспекти на процеса на проектиране на вградени системи.

Разработен е обобщен алгоритъм за автоматизирано проектиране на вградени PicoBlaze базирани системи.

Показана е причината, поради която началните етапи от проектирането на вградени FPGA базирани системи не могат да се извършват автоматично, без човешка намеса.

Разгледани са начините и необходимостта от предварително съставяне на графични скици при изграждане на конфигурационния файл на вградени системи.

Разработена е програмна среда за автоматично генериране на конфигурационния файл и шаблон за асемблерската програма на процесора от блоковата схема на разработваната система. Изследванията направени в тази глава са представени в публикации [1,5,6,7] от списъка на публикациите свързани с дисертацията

Глава 3. Реализиране на инженерни приложения с разработената среда.

3.1. Разработка на МПО модули

При генериране на приложения, реализирани с помощта на PicoBlaze базирани системи, е възможно възникването на необходимост от доработка на управляващата програма на процесора или създаване на нов МПО модул.

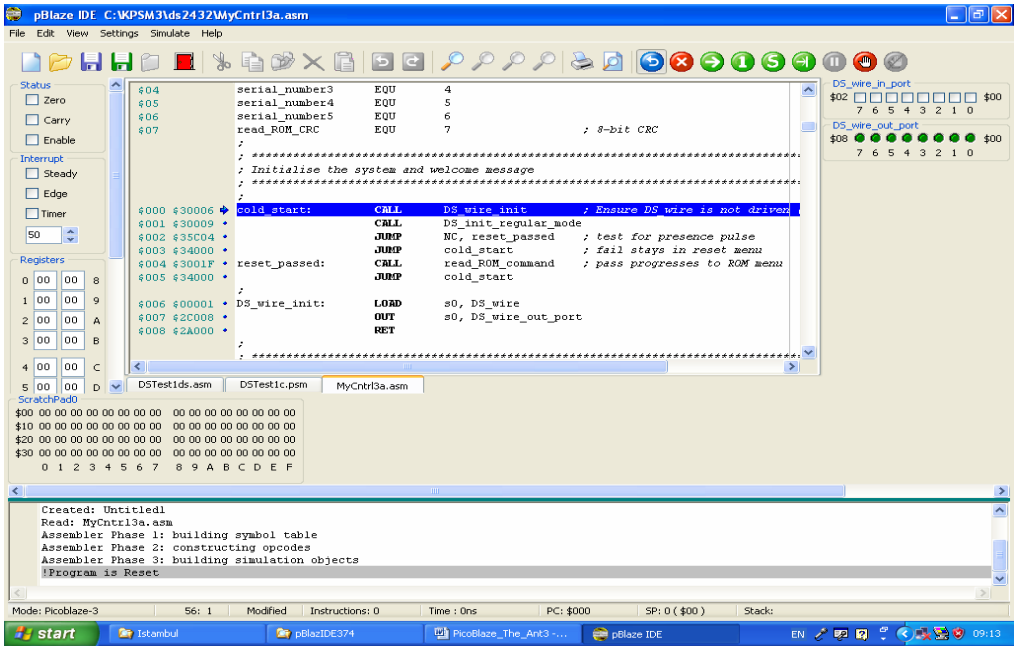
3.1.1. Разработка на управляваща програма на процесора

МПО модулът, съдържащ управляващата програма на процесора, се изготвя на асемблер. За основа се използва автоматично генерираният от програмата ViNi73A файл *.psm. Тои се надгражда с помоща на текстов редактор. Програмата се съхранява като файл, името на който не трябва да надхвърля 8 символа и има разширение *psm*.

За превръщане на асемблерския файл в VHDL МПО модул се използва асемблиращата програма KCPSM6.exe. Програмата работи като Windows приложение, в чийто прозорец се извеждат съобщенията за възникнали грешки, открити в процеса на транслация на програмата. Този транслатор не позволява провеждането на симулация на разработваната програма. За симулиране на разработваната програма се препоръчва използването на средата pBlazIDE (фигура 3.1), разработена от Mediatronix [82].

Това е безплатна среда работеща под Windows. Тя предлага непосредствена информация за състоянието на регистри, флагове, стойности на паметта, и прекъсвания. Всеки един от тези ресурси може

да се променя на място, което като цяло много улеснява и ускорява процеса на развой. По тази причина тя се счита за най-добро средство за моделиране на поведението на управляващата програма на процесора.



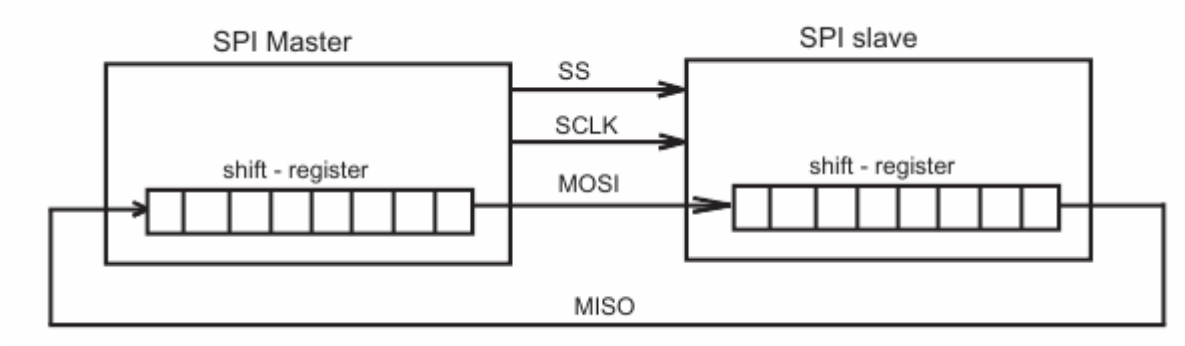
Фигура 3.1. Симулация на програма със средата pBlazIDE

3.1.2. Разработка на МПО модул SPI

Един от интерфейсите, придобил най- широка популярност при изграждане на серийна комуникация между чипове, е SPI. Той е въведен в употреба от Motorola и в момента се използва в продуктите на много производители. Неговото име е акроним за " Serial to Peripheral Interface ", което отразява неговата мисия – магистрала за свързване на външни устройства. SPI е организиран на принципа " водещ - подчинен " . В качеството на водещ може да се използва Микро контролер, DSP контролер или ASIC, а в ролята на подчинени устройства могат да служат различни видове устройства (EEPROM,

Flash - памет, SRAM), часовник за реално време (RTC), ADC / DAC, цифрови потенциометри, специализирани контролери и т.н.

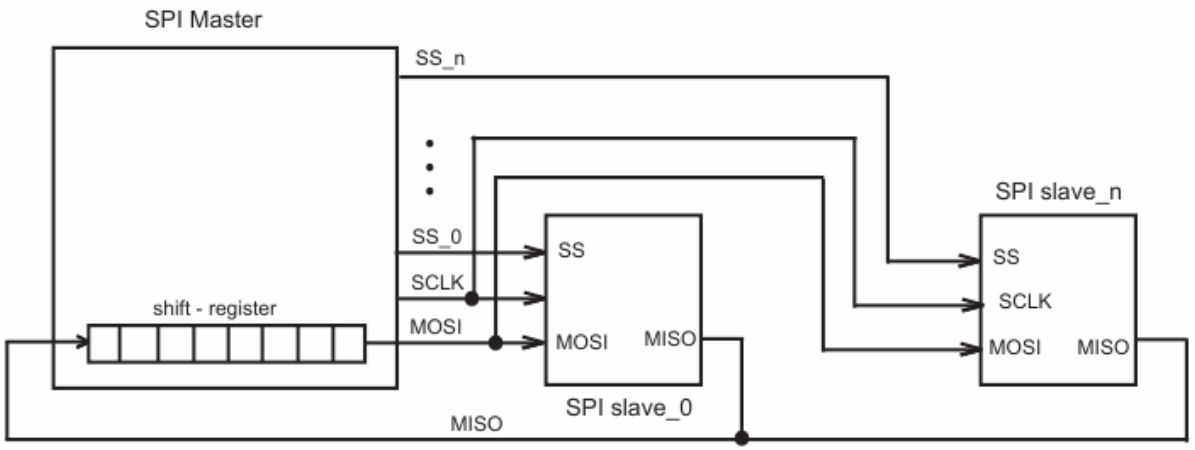
Основната съставна единица на този интерфейс е традиционен преместващ регистър и набор от четири сигнала, чието свързване е показано на фигура 3.2



Фигура 3.2. Класическа схема за обмен на информация по SPI.

По своята същност това е най-простото включване, в което участват само два чипа. При него водещият чип предава данните по линията MOSI синхронно с генерирания от него сигнал SCLK, а подчиненият приема предаваните битове от данни по определен фронт на сигнала за синхронизация и едновременно с това, по другия фронт изпраща своите данни по линията MISO.

Както се вижда от фигурата протоколът за предаване на данни по интерфейса SPI е пределно прост и по същество е идентичен с логиката на работа на преместващ регистър. Характерна особеност на този интерфейс е, че приемането и предаването на данни винаги се изпълнява по противоположните фронтове на синхронизиращия сигнал, с оглед да се гарантира надеждното им установяване. С въвеждането на незначителни усложнения този интерфейс позволява паралелна работа на няколко устройства. Този режим на работа е показан на фигура 3.3.

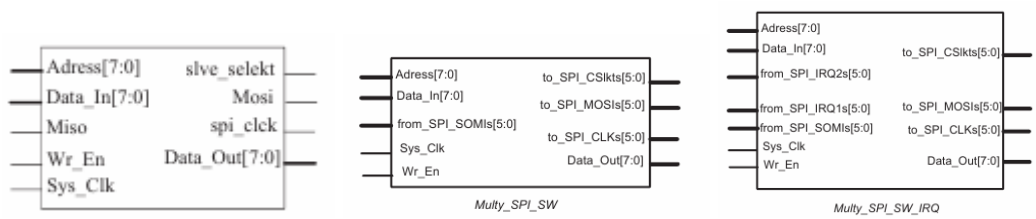


Фигура 3.3. Паралелна работа на няколко SPI устройства.

Основните преимущества на SPI интерфейса са:

1. комуникация в режим Full duplex
2. високо бързодействие
3. лесна галваническа развръзка
4. проста програмна реализация
5. по висока пропускателна способност спрямо протокола I²C
6. може да предава произволен брой битове
7. прост апаратен интерфейс
8. по ниска консумация спрямо протокола I²C
9. не изисква арбитраж
10. не изисква прецизен тактов генератор, тъй като подчиненият прибор използва такта на водещия.
11. не са необходими трансивери
12. за всеки прибор е необходим само един уникален сигнал (CS).

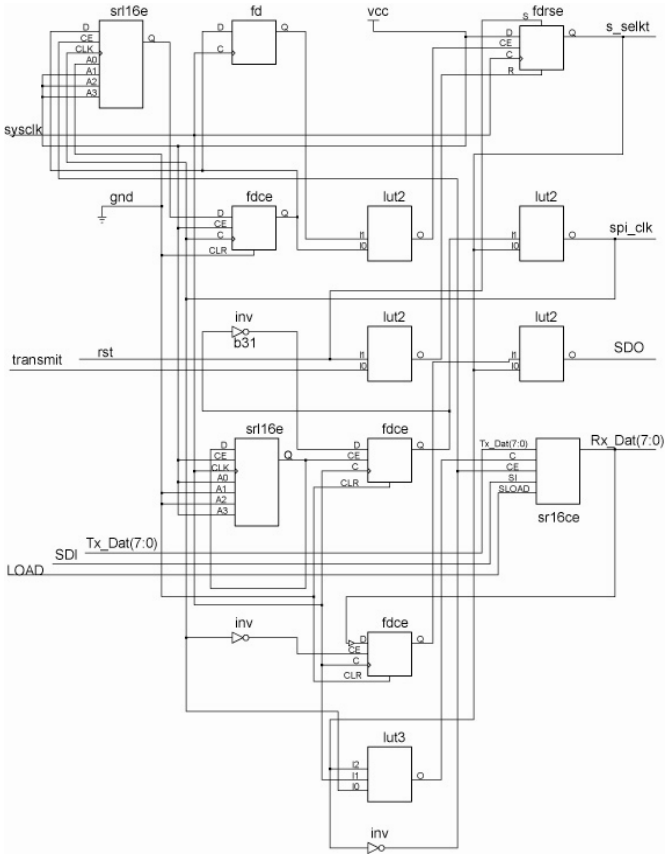
Основенен мотив за неговото присъствие в библиотеката с градивни елементи на разработения в дисертациония труд автоматизирана среда за генериране на вградени PicoBlaze базирани системи е голямото многообразие от периферийни устройства използващи този интерфейс. За библиотеката с градивни елементи в дисертациония труд са разработени един апаратен и два програмни SPI модула. Техните условни изображения са показани на фигура 3.4.



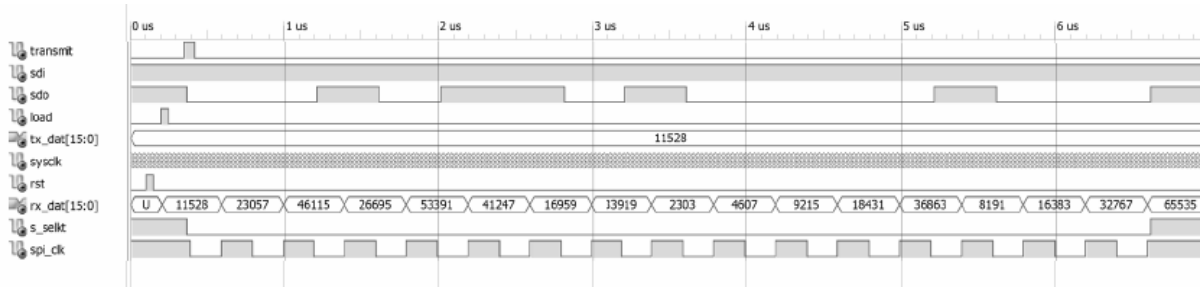
Фигура 3.4. Условни изображения на SPI модули

Апаратен SPI модул

Апаратният SPI модул е реализиран под формата на VHDL описание на ниско ниво и може да се вмъква непосредствено във всеки един проект. Структурната схема на модула е показана на фигура 3.5, а неговата симулация на фигура 3.6.



Фигура 3.5. Структурна схема на SPI модула



Фигура 3.6. симулация на работата на SPI модула

Решение за изграждане на програмен SPI модул е предложено от Кен Чапман [45]. За управлението на протокола то използва два 8 битови порта. Разпределението на битовете в тези портове са дадени в таблица 3.1.

Таблица 3.1. Разпределение на сигнали във входно - изходните портове

bit No	signal name	
	изходен порт	входен порт
7	SPI_MOSI	SPI_MISO
6	nc	Nc
5	nc	Nc
4	nc	Nc
3	nc	Nc
2	nc	Nc
1	SPI_CS_b	Nc
0	SCLK	Nc

На базата на тази програма в рамките на дисертационния труд са разработени два програмно реализуеми SPI модула с разширени възможности.

Същността на първия програмен модул е показана в таблица 3.2. От нея се вижда, че за управление на подчиненото SPI устройство от изходния порт се използват само битове 0, 1 и 7. Останалите битове 2,3,4,5 и 6 са свободни. В разработената програма тези свободни битове се асоциират със сигнали за разрешение на подчинени устройства. Това позволява да се постигне управление на 6 независими SPI прибора, включени

паралелно (вж. фигура 3.3). За целта в кода на програмата трябва да се добави само един оператор и шест константи.

Таблица 3.2. Разпределение на сигнали във входно - изходните портове

Kit	signal name	
	изходен порт	входен порт
7	SPI_MOSI	SPI_MISO
6	SPI_CS_5	nc
5	SPI_CS_4	nc
4	SPI_CS_3	nc
3	SPI_CS_2	nc
2	SPI_CS_1	nc
1	SPI_CS_0	nc
0	SCLK	nc

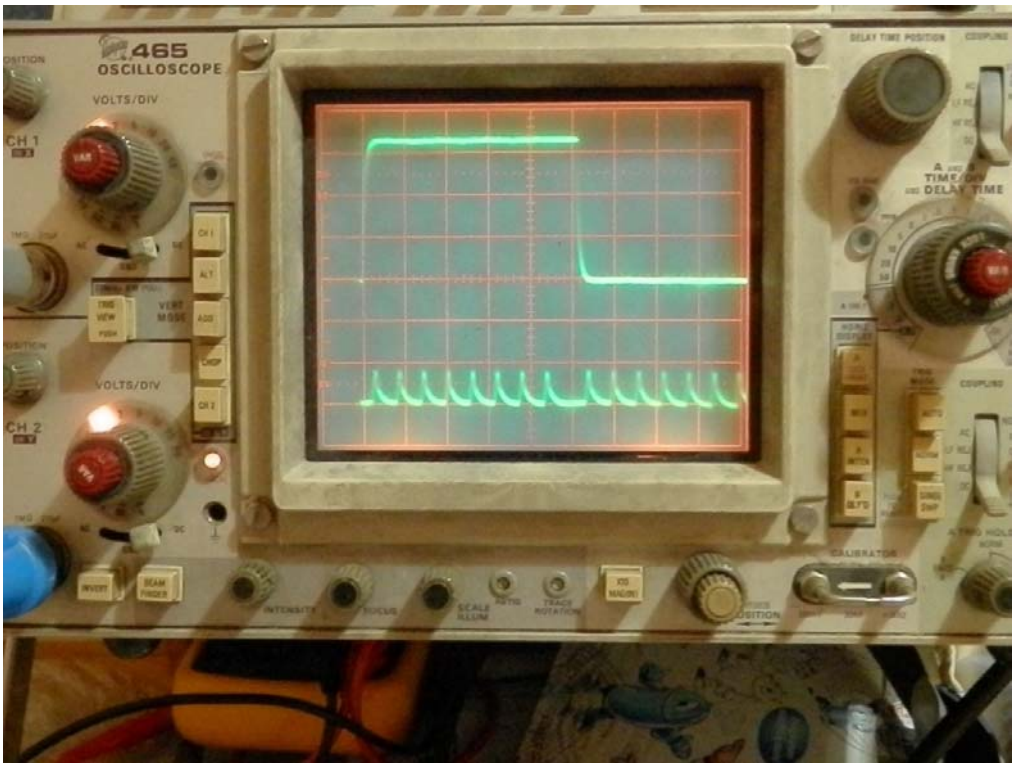
```
;
CONSTANT spi_clk,    00000001'b  ;   spi_clk - bit0 (SPI_output_port)
CONSTANT spi_cs_b, 00000010'b ;  spi_cs_b - bit1 (SPI_output_port)
CONSTANT spi_mosi, 10000000'b ;  spi_mosi - bit7 (SPI_output_port)
CONSTANT spi_miso, 10000000'b ;  spi_miso - bit7 (SPI_data_in_port)
;.
;-------------------------------------------------------------------------------
CONSTANT CS_SPI_Dev_0, 11111100'b  ;
CONSTANT CS_SPI_Dev_1, 11111010'b  ;
CONSTANT CS_SPI_Dev_2, 11110110'b  ;
CONSTANT CS_SPI_Dev_3, 11101110'b  ;
CONSTANT CS_SPI_Dev_4, 11011110'b  ;
CONSTANT CS_SPI_Dev_5, 10111110'b  ;
;-------------------------------------------------------------------------------
                ;       Register used s0,s1,s2,s3,s4
                                                        ;               ....
SPI_tx_rx:      LOAD s1, 08                     ;8-bits to transmit and receive
next_SPI_bit:   LOAD s0, s2                     ;prepare next bit to transmit
                AND s0, spi_mosi                ;isolates data bit and spi_cs_b = 0
                OR  s0, s4                      ;select desire SPI chanel
                OUTPUT s0, SPI_output_port      ;output data bit ready to be used on rising clock edge
                INPUT s3, SPI_data_in_port      ;read input bit
                TEST s3, spi_miso               ;carry flag becomes value of received bit
                SLA s2                          ;shift new data into result and move to next transmit bit
                CALL SPI_clock_pulse            ;pulse spi_clk High
                SUB s1, 01                      ;count bits
                 JUMP NZ, next_SPI_bit          ;repeat until last bit
                RETURN
```

Листинг 4.1 програма за управление на SPI модул.

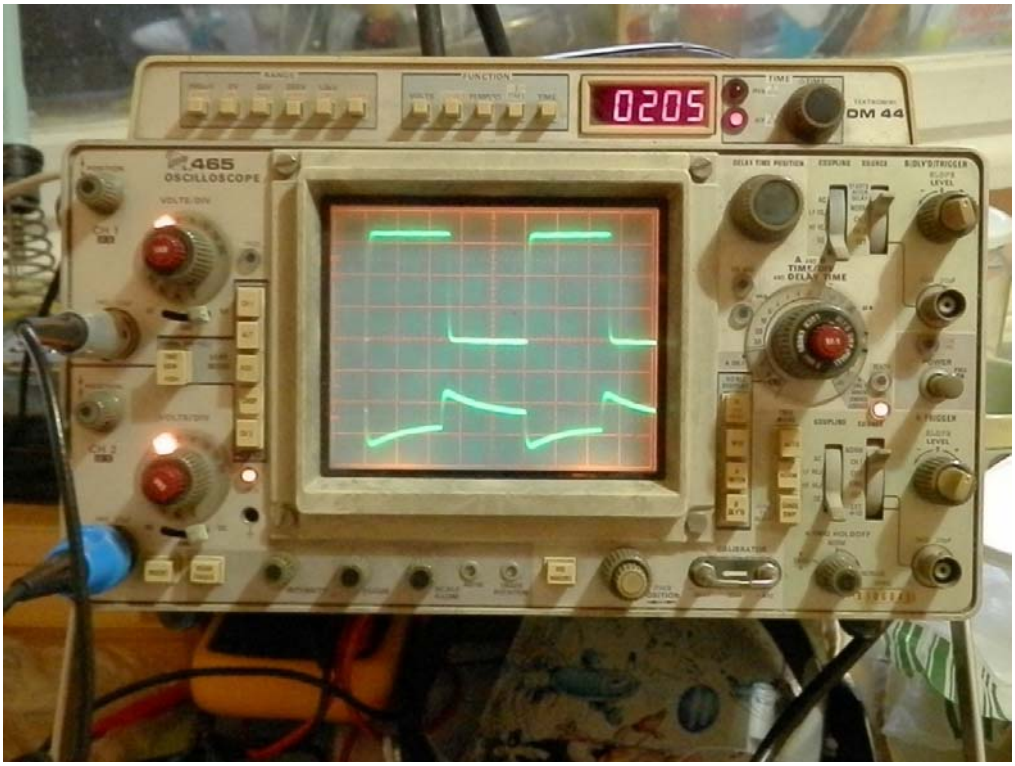
В този случай обръщението към програмата би изглеждало така:

```
;exampl for use
;               ...
;               load s4,CS_SPI_Dev_n
;               call SPI_tx_rx
;               ...
```

Осцилограмите, илюстриращи работата на тази програма за два SPI прибори са показани на фигури 3.7 и 3.8.

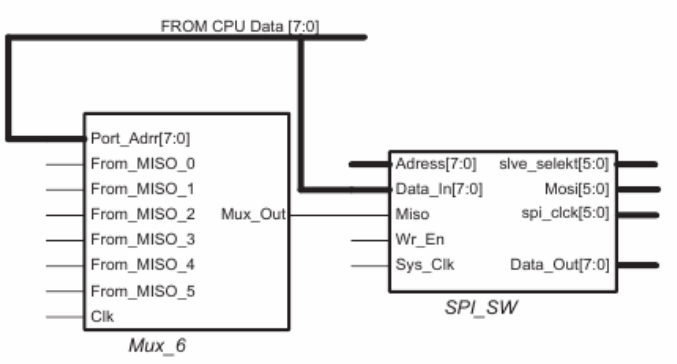


Фигура 3.7. Развивка 0,5 us/cm.



Фигура 3.8. Развивка 1 us/cm.

Еквивалентната структурна схема на този модул е показана на фигура 3.9.



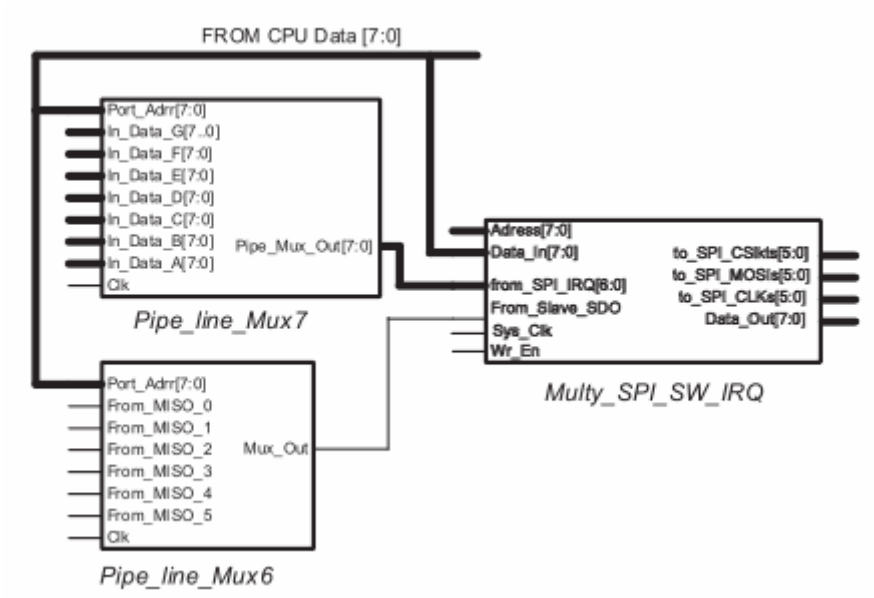
Фигура 3.9. Схема на многоканален SPI модул

Същността на втория програмен модул е показан в таблица 3.3. Той е ориентиран към използване за работа в режим на прекъсване. Този модул позволява работа и поддържа до 7 различни прекъсвания.

Таблица 3.3.Рразпределение на сигнали във входно - изходните портове

bit No	signal name	
	изходен порт	входен порт
7	SPI_MOSI	SPI_MISO
6	SPI_CS_5	IRQ6
5	SPI_CS_4	IRQ5
4	SPI_CS_3	IRQ4
3	SPI_CS_2	IRQ3
2	SPI_CS_1	IRQ2
1	SPI_CS_0	IRQ1
0	SCLK	IRQ0

Еквивалентната структурна схема на този модул е показана на фигура 3.10



Фигура 3.10. Схема на многоканален SPI модул с прекъсване.

Обръщението към този програмен модул в рамките на асемблерската програма изглежда така:

```
;exampl for use
            ...
                    load s4,CS_SPI_Dev_n
wait_for_IRQ:       Input s3, SPI_data_in_port   ;    ;read input IRQ data
                    compare s3, IRQ_data         ;    ;compare for expected IRQ
                    jump nz, wait_for_IRQ        ;    ;repeat until last bit
                    call IRQ_develop             ;   develop IRQ
            ...
```

Проведените експерименти показват, че и при двата програмно реализируеми SPI модула скоростта остава близка до 3,3 Mb/sec

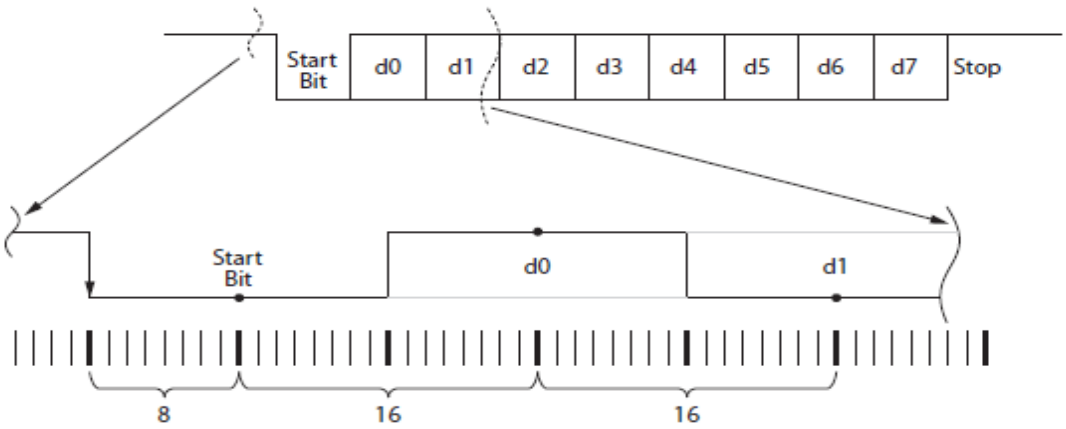
3.1.2. Разработка на МПО модул Baud rate

Основна роля за правилното функциониране на всеки UART модул представлява честотата на стробиране на информацията, постъпваща от и към него.

Основната идея за използване на стробиращата честота при изграждане на UART комуникации е показана на фигура 3.11.

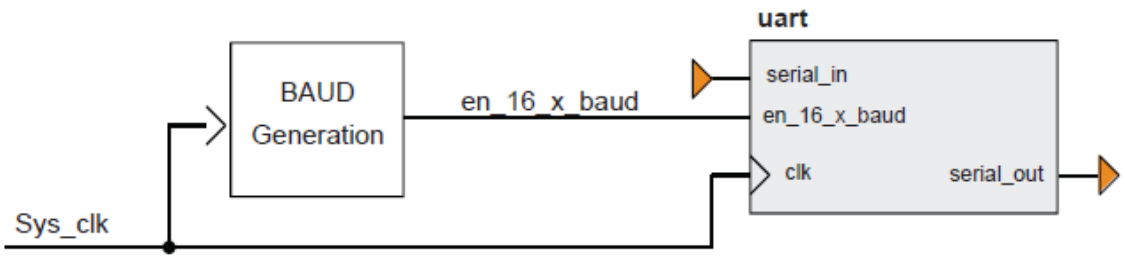
За постигане на необходима точност по стандарт интервалът на предавания бит се дискредитира с честота 16 пъти по висока от тази, с

която се предава информацията. Така при предаване със скорост 9600 бода честотата на стробиране ще бъде 9600 x 16 = 153600 хц.



Фигура 3.11. Стробиране при UART комуникация

Като основен показател за работата на един UART модул тази честота се синхронизира с тактовата честота на системата. Така общата блокова схема на UART модула, предназначен за вграждане в FPGA прибор добива вида показан на фигура 3.12. Най-често UART модула се предлага под формата на готово IP ядро, оптимизирано съобразно структурата на използвания FPGA прибор. Така цялостното изграждане на UART базирана системата за предаване на данни се свежда до разработката на модул за получаване на стробираща честота.



Фигура 3.12. Генериране на стробираща честота

При зададени стойности на системната честота и скоростта за предаване на данни коефициентът на делене на модула генериращ стробиращата честота се дава с цялата част на израза :

К = (системна честота)/(16 х скоростта за предаване на данни)

В случаите, когато стойността на К не е цяло число, задължително трябва да се направи проверка за относителното изменение на стробиращата честота, което е предпоставка за неправилно приемане на символ. Стойността на получения делител К се счита за удовлетворителна, когато величината на относителната грешка е под 0,6%.

За трите най-често срещани системни честоти, 25, 50 и 100 мхц, използвани в различни апаратни системи за развой, стойностите на делителя К за стандартната поредица от скорости на предаване варира в границите (от 2 до 4000). Точните стойности на делителя, заедно с получените грешки са показани в таблица 3.4.

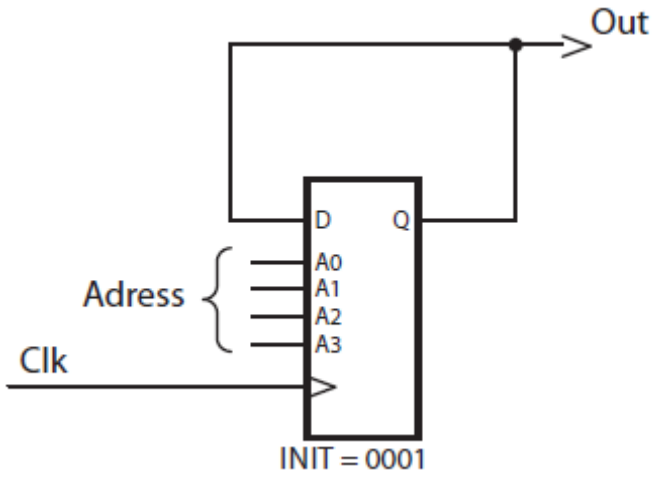
За ресурсите на съвременните FPGA прибори [4,5], реализацията на подобен коефициент на делене не е проблем. Дори и за най-малкия FPGA прибор, реализирането на 20 битов делител заема по-малко от 1% от общия брой на наличните в прибора тригери.

Класическото реализиране на тези делители се извършва в рамките на няколко CLB, което увеличава обема на необходимите системни ресурси за постигане на по-ниски скорости. Броят на необходимите CLB може да се редуцира значително, при използване на LUT таблици. Това позволява да се намали броят на връзките, необходими за изграждането на делителя и гарантира работа с честоти, близки до граничните за съответния прибор.

Таблица 3.4 Стойности на делителя, заедно с получените грешки

	Тактова честота [MHz]			Тактова честота[MHz]		
	25	50	100	25	50	100
скорост	Коефициенти на деление			Грешка в скоростта [%]		
1200	1302,00	2604,00	5208,00	-0,0064	-0,0059	-0,00295
2400	651,00	1302,00	2604,00	-0,0064	-0,0064	-0,0064
4800	325,00	651,00	1302,00	-0,16026	-0,0064	-0,0064
9600	163,00	325,00	650,00	0,146984	-0,16026	-0,16026
19200	81,00	163,00	325,00	-0,46939	-0,46939	0,146984
38400	40,00	81,00	162,00	0,755843	-0,46939	0,146984
57600	27,00	54,00	108,00	-0,46939	-0,46939	-0,46939
115200	13,50	27,00	54,00	-0,46939	-0,46939	-0,46939
230400	13,5**	13,50	27,00	-0,46939	-0,46939	-0,46939
460800	13,5**	13,5*	13,50	-0,46939	-0,46939	-0,46939
921600	13,5***	13,5*	13,5*	-0,46939	-0,46939	-0,46939

Едно от уникалните свойства на тези таблици е възможността за тяхното трансформиране в премествящи регистри от типа SRL16 и SRL32 [7,8]. С въвеждане на обратни връзки, предварителна инициализация на съдържанието на тези премествящи регистри и подходящата им адресация, с тях лесно се реализират делители с различен коефициент на делене. Идеята на този подход е показана на фигура 3.13. При него, само с промяна на адреса лесно се постига произволен коефициент на делене от 1 до 16.



Фигура 3.13. Реализиране на делител с различен коефициент на деление

В случаите когато се налага реализирането на по-голям коефициент на делене, тези делители могат да се свързват последователно.

Стойностите на коефициентите на делене, покриващи стандартните скорости за предаване на данни и системните честоти и тяхното възможно представяне под формата на произведение от числа, са показани в таблица 3.5.

С малки изключения, като се вземе предвид, че с ресурсите на един слайс от типа SLISEM може да се реализира произволен коефициент на деление в диапазона от 2 до 128, всички делители от таблица 3.4 без проблеми могат да се реализират в рамките на 2 CLB блока.

Пример, показващ VHDL описанията на делители с коефициент на делене съответно 54, и 13.5 [84] са дадени в следващите листсинги.

Таблица 3.5 Коефициентите на делене покриващи стандартните скорости за предаване

	Тактова честота [MHz]			Тактова честота [MHz]		
	25	50	100	25	50	100
скорост	Коефициенти на деление			Предсавяне на коефициентите		
1200	1302	2604	5208	2*21*31	4*21*31	8*21*31
2400	651	1302	2604	31*21	2*21*31	4*21*31
4800	325	651	1302	13*25	31*21	2*21*31
9600	163	325	650	18*9	13*25	13*5*10
19200	81	163	325	9*9	18*9	13*25
38400	40	81	162	8*5	9*9	18*9
57600	27	54	108	1*27	6*9	12*9
115200	13,5	27	54	12+13	1*27	6*9
230400	13,5**	13,5	27	12+13	12+13	1*27
460800	13,5**	13,5*	13,5	12+13	12+13	12+13
921600	13,5***	13,5*	13,5*	12+13	12+13	12+13

*- използва се удвояване на тактова честота с DCM
** - използва се учетворяване на тактова честота с DCM
*** - използва се 8 кратна тактова честота (генерирана с DCM)

```
-- делител с коефициент на делене 54
entity Baud_rate_Timer is
    Port (  En_16_x_baud : out std_logic;
                    clk : in std_logic);
    end Baud_rate_Timer;
--
-- Start of test architecture
--
architecture Low_level of Baud_rate_Timer is
--
signal a1,a2, a3 : STD_LOGIC := '0';
```

```
--
-- Start of circuit description
--
begin
--
SRLC32E_inst1 : SRLC32E
generic map (INIT => X"00000001")
port map (Q31 => a1, -- SRL cascade output pin
                     A => "11111", -- 5-bit shift depth select input
       CE => '1', -- Clock enable input
    CLK => clk, -- Clock input
         D => a2 -- SRL data input);
--
SRLC32E_inst2 : SRLC32E
generic map (INIT => X"00000000")
port map (Q => a2, -- SRL data output
     A => "10100", -- 5-bit shift depth select input
   CE => '1', -- Clock enable input
CLK => clk, -- Clock input
     D => a1 -- SRL data input);
--
FDCE_inst : FDCE
generic map (INIT => '0')
--
port map (Q => En_16_x_baud,
           C => clk,
        CE => '1',
           CLR => '0',
          D => a2 );
--
end Low_level;

-- делител с коефициент на делене 13.5
entity Baud_rate_Timer is
    Port (  En_16_x_baud : out std_logic;
                                  clk : in std_logic);
end Baud_rate_Timer;
--
architecture Low_level of Baud_rate_Timer is
--
signal a1,a2, a3 : STD_LOGIC := '0';
--
--
-- Start of circuit description
begin
--
SRL16E_inst1 : SRL16E
generic map (INIT => X"0001")
port map (Q => a1,
         A0 => '0',
         A1 => '0',
         A2 => '1',
         A3 => '1',
        CE => '1',
      CLK => clk,
           D => a2 );
--
SRL16E_inst2 : SRL16E
generic map (INIT => X"0000")
--
port map (Q => a2,
```

```
         A0 => '1',
         A1 => '0',
         A2 => '1',
         A3 => '1',
         CE => '1',
        CLK => clk,
            D => a1 );
--
LUT2_inst : LUT2
generic map (INIT => X"E")
port map (O => a3,
          I0 => a1,
          I1 => a2 );
--
FDCE_inst : FDCE
generic map (INIT => '0')
--
port map (Q => En_16_x_baud,
           C => clk,
         CE => '1',
       CLR => '0',
            D => a3 );
--
end Low_level;
```

Изследване на канала за обмен на данни

В по-голямата си част, всички приложения, реализирани на базата на вградени процесори работят в режим на реално време. В тази насока интерес представлява провеждането на изследване, поставящо си за цел да определи максималното бързодействие на реална PicoBlaze базирана система, ползваща функциите на терминалната програма PicoTerm. Наличието на подобна оценка е от особена важност за правилната работа на конкретната система, тъй като ускорява значително процеса на съвместната разработка на средствата на програмното и апаратното осигуряване на вградените процесори и значително съкращава времето за обучение при преход от проектиране на една целева архитектура в друга.

Основна роля за постигането на пълен и бърз апаратно програмен контрол и настройка на разработвания проект с помощта на специализираната апаратна среда за работа с CPLD и FPGA прибори играят вграденият в нея апаратен двуканален UART-USB мост,

реализиран на базата на интегрална схема FT 2232D на фирмата FTDI и програмният продукт PicoTerm[45].

PicoTerm представлява на PC базиран терминал, предназначен за комуникация с PicoBlaze системи, които използват UART макроси, свързани към USB / UART порт на произволна развойна система или платка. PicoTerm може да комуникира с всеки виртуален или апаратен COM порт на произволен хардуер или проект. Въпреки че подобна връзка може да се осъществи с всяка една терминална програма (Hyper terminal, PuTTY, SecureCRT и др.), правилното определяне на всичките им комуникационни и ASCII опции често се явява предизвикателство. В това отношение настройките за комуникация на PicoTerm са предварително фиксирани както следва 8-бита, 1 Stop Bit, No Parity and No Handshake и скорост на обмен 115200 бода. Така само номерът на COM порт трябва да бъде зададен, което го прави съвместим с всички UART макроси, предвидени за работа с PicoBlaze. PicoTerm поддържа всички стандартни скорости на обмен от 1200 до 921600 бода.

PicoTerm се стартира от DOS прозорец или от батч файл, посредством команден ред, в който трябва да са указани номерът на COM порта и скоростта на предаване.

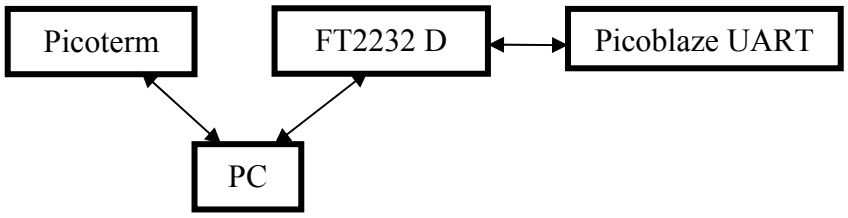
PicoTerm -c13-b9600 (отваря COM13 със скорост от 9600бода)

След въвеждане на номера на COM порта, PicoTerm, прави опит да отвори този порт. Ако той не бъде открит, или други отворени приложения вече ползват този порт, PicoTerm извежда съобщение за грешка. При успешно отворен сом порт PicoTerm извежда съобщение от вида

COM № is open for communication at .xxxx.. baud.

PicoTerm поддържа стандартните кодове за контрол, изпълнява управляващи последователности (Device Control String DCS) и предлага набор от графични функции.

В класическия си вариант структурата на канала за обмен на данни може да се представи по начин показан на фигура 3.14.



Фигура 3.14. Канал за обмен за дани

С оглед да се осигури стабилна работа в комбинация с други обекти, всеки един от елементите, изграждащи тази система притежава по два собствени буфера за данни, които гарантират неговата коректна работа в режим на предаване и приемане на данни от Picoblaze към Picoterm и обратно. Всеки един от буферите за данни на Picoblaze е апаратен и може да съхранява до 16 байта. Апаратни, но с различен капацитет, надвишаваш този на буферите за данни на Picoblaze са и приемо-предавателните буфери на FT2232 D. Буферите на Picoterm и тези на виртуалния сом порт създаден в PC за обслужване на USB-UART моста FT2232 D са програмни с капацитет не надхвърлящ 4096 байта.

Наличието на разлики в размера на буферите, отсъствието на синхронизация между тях подсказва за възможни проблеми при предаването на данни по канала, формиран от тези устройства. Това е повод за провеждане на изследване, целящо да се определят параметрите на канала за предаване на данни. В съответствие с

поддържаните от Picoterm режими на работа, данните към него могат да се подават под формата на поток или поединично.

Изпращането на единични данни от и към Picoterm се налага при извеждане на цветен текст, информация за дата, час и използване на графичен или виртуален дисплей. В този случай скоростта на предаване на данни не е от особено значение, тъй като тя се определя от изискванията за обновяване на информацията и в общия случай не бива да пада под величина, гарантираща стабилно изображение. В този случай, предвид по-големия капацитет на буферите на Picoterm и тези на моста FT2232 D, скоростта на предаване се определя еднозначно от временните интервали, през които информацията постъпва на входа на FT2232 от изхода на UART буфера към Picoblaze. По подразбиране типичната скорост, с която работи Picoblaze е 115к бода. И понеже USB-UART мостът поддържа скорости на обмен близки до 5Мбита и капацитет на входния апаратен буфер от порядъка на 128 байта, то вероятността за появата на грешка при предаване на информация по този канал е минимална, близка до нула.

От тази гледна точка по-интересен се оказва случаят, когато по канала се предават големи потоци от данни. В този случаи може да се очаква препълване на виртуалните буфери, което да предизвика загуба на данни.

С оглед да се получи вярна представа за поведението на разглежданата система, бяха проведени серия от тестове. Всеки един от тестовете се провеждаше в обем, позволяващ постигане на 95% достоверност.

Изследванията са проведени за следните скорости на предаване 115200, 57000, 32000,19200, и 9600 бода при размера на блоковете от 4096 до 32 байта.

Резултатите от проведените експерименти показват, че времето, гарантиращо отсъствието на загуба на данни при дължина на блока от

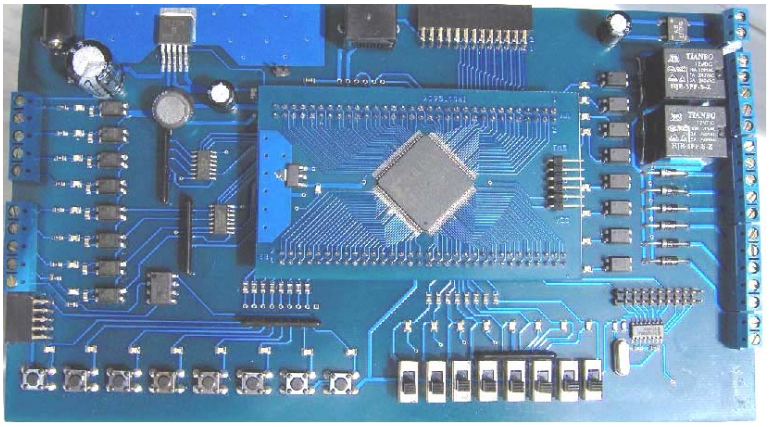
порядъка на 128 байта, е в границите от няколко десетки ms. При дължина на блока от 4096 байта това време нараства до 2s.

3.2. Апаратна среда за работа с препрограмируеми прибори

За подобряване времето за верификация към средата за автоматизирано генериране на вградени PicoBlase базирани системи е реализирана апаратна среда за проверка и настройка на апаратното и програмното осигуряване на разработваните системи.

Средата поддържа работа с CPLD и FPGA прибори от сериите XC9500XL, CoolRunner-II и Spartan-6 на фирмата Xilinx. Тя притежава възможности за въвеждане, извеждане и визуализиране на информация и подпомага процеса на разработка.

Разработената апаратна среда е показана на фигура 3.15. Тя се състои от дънна платка и присъединими програмируеми модули.



Фигура 3.15. Общ вид на универсална апаратна развойната среда

С оглед да се осигурят всички необходими и достатъчни условия за правилната и надеждна работа на всеки един от присъединените програмируеми модули на дънната платка на апаратна развойната среда са обособени следните няколко блока:

1. захранващ модул
2. блок за генериране на кварцово стабилизирани сигнали

3. блок бутони за генериране на входни въздействия
4. блок ключета за генериране на входни въздействия
5. блок оптически изолирани входове
6. блок оптически изолирани изходи
7. блок непосредствено изведени изводи с общо предназначение
8. блок за комуникация с PC

Благодарение на тези блокове съчетанието от дънна платка с програмируем модул се превръща в надеждно средство за верификация на различни проекти.

Всички модули на развойната среда се захранват със стабилизирано напрежение 3.3 волта. То се генерира от модул, възприемащ входни променливо токови или постоянни напрежения в диапазона от 7 до 15 волта.

Блокът за генериране на кварцово стабилизирани сигнали работи на честота 25MHz. От тази честота с помощта на делители, се формира набор от субхармонични честоти. С помощта на джъмпери тези честоти могат да се подават към програмируемия модул.

Блоковете за генериране на входни въздействия са реализирани съответно с 8 бутона и 8 ключета. Във веригата на всеки един от тях има включен свето диод, който показва момента на тяхното активиране.

Блокът оптически изолирани входове съдържа 8 канала, които също притежават светодиодна индикация. Изходът на всеки канал е буфериран с инвертор и тригер на Шмид. Максималната работна честота на всеки канал е 5 kHz.

Освен тези канали, блокът оптически изолирани входове съдържа още един канал, който приема входни сигнали с честота до 1 MHz. Този канал не притежава светодиодна индикация.

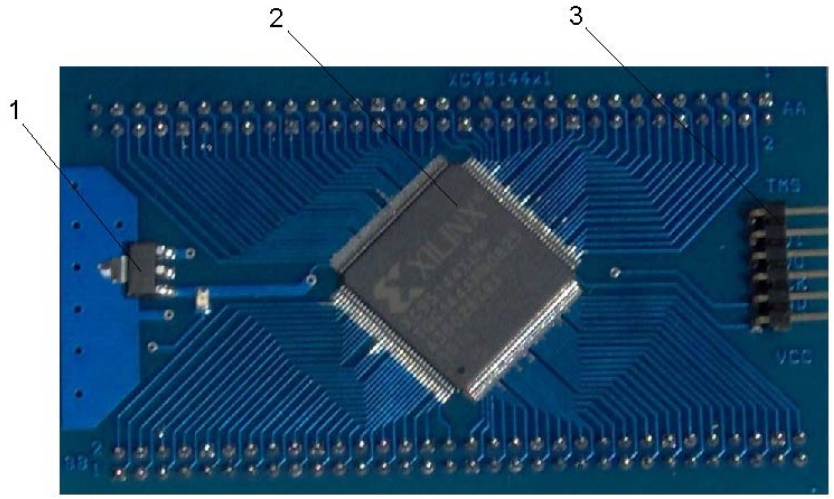
Блокът оптически изолирани изходи е 8- канален. Те са тип „отворен колектор” и позволяват комутирането на ток до 0,5A. Могат да

управляват и индуктивни товари. Всички оптически изолирани входове и изходи изискват външно захранващо напрежение 12 волта.

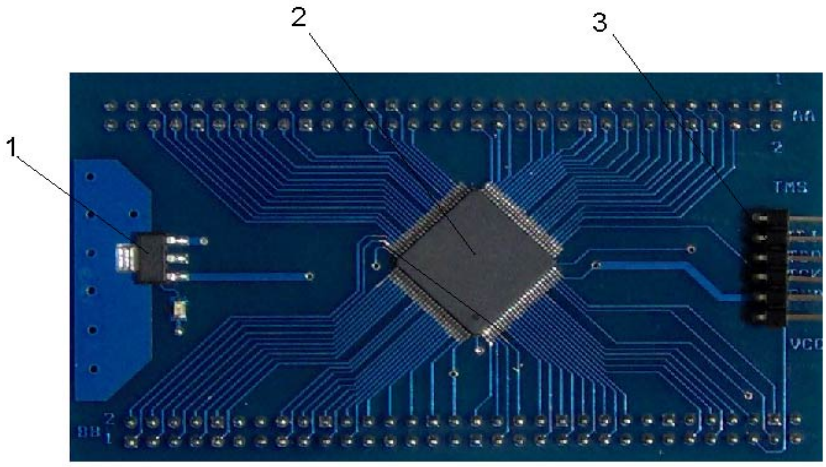
Блокът за комуникация е апаратен мост, който позволява прехвърляне на информация между апаратната развойна среда и PC по USB. Той е реализиран на базата на интегрална схема FT2232D на фирмата FTDI. При подключването му към PC, той се разпознава автоматично от операционната система, която генерира за него два независими един от друг виртуални серийни порта. Всеки един от тези портове може да поддържа всички стандартни скорости за обмен на данни по сериен канал до 921600 бода. По подразбиране скоростта на обмен е фиксирана на 115200 бода.

Характерна особеност на разработената развойна среда представлява наличието на две групи изводи с общо предназначение, които осигуряват непосредствен достъп до изводите на препрограмируемия чип. Общият брой на тези изводи е 42. Към тях може да се подключват периферийни модули предлагани като опция към апаратната развойната среда или такива, разработени от потребителя.

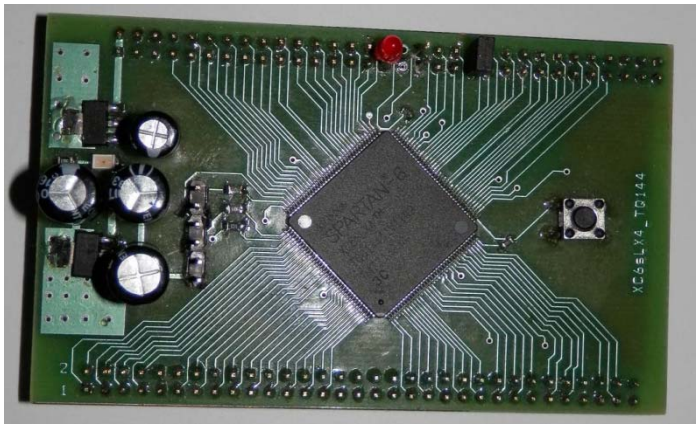
Програмируемите модули, с които е окомплектована развойната среда са показани на фигури 3.16, 3.17, 3.18:



Фигура 3.16. Общ вид на програмируем модул XC95144XL



Фигура 3.17. Общ вид на модул XC2C128



Фигура 3.18. Общ вид на модул XC6SLX4

Всеки един от програмируемите модули съдържа захранващ модул (1), програмируем чип (2) и извод за вътрешно системно програмиране (3). За присъединяване на програмируемите модули към дънната платка се използват две мъжки рейки.

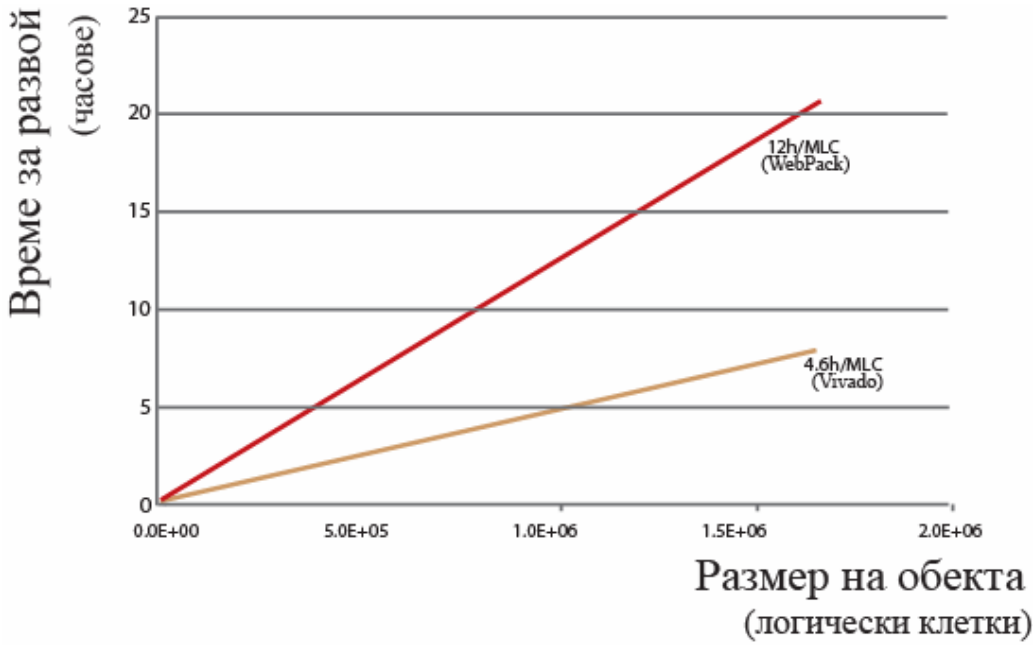
За разширяване на функционалните възможности на апаратната среда, освен програмируемите модули към нея в дисертацията труд са разработени набор от апаратни средства. Всеки модул е окомплектован със съответен драйвер и подходяща документация.

3.3 Оценка на ефективността на разработените програмни средства

Основна цел на създадената автоматизирана среда за проектиране на вградени PicoBlase системи е съкращаване на общото време за тяхното реализиране. Опростеният алгоритъм за разработка на вградени PicoBlase системи, показан на фигура 2.1 позволява разделянето на процеса на генериране на две части: специфична и тривиална.

Специфичната част обхваща в себе си всички дейности, свързани с изготвянето на конфигурационния файл на системата и програмата на вградения процесор. Втората част обхваща тривиалните дейности по синтез, имплементация и генериране на бинарен файл, извършвани от съответната развойна среда (webpack, ISE или VIVADO). Времето за изпълнение на тези дейности е важен показател за използваната развойна среда и е в пряка зависимост от обема на проекта. То се измерва с величина, чиято дименсия е [време/милион логически клетки(MLC)]. Типичната производителност на развойните среди ISE и VIVADO е показана на фигура фигура.3.19 [136].

При вградените PicoBlase базирани системи, количеството на логическите клетки, необходимо за тяхното изграждане, рядко надхвърля стойността 5к. Вследствие на това, времето за тяхното създаване се оказва достатъчно малко и може да се пренебрегне. По тази причина, времето, необходимо за реализиране на разработваната вградена PicoBlase базирана система се определя с времето за извършване на всички операции от специфичната част на процеса на проектиране.



Фигура.3.19. Производителност на развойните среди ISE и VIVADO

Анализът на тези операции в рамките на алгоритъма за разработка на вградени PicoBlase базирани системи от фигура 3.7, позволява да се направи изводът, че най-много време отнемат операциите, свързани с изготвянето на файла, съдържащ описанието на системата. По тази причина, тези операции се оказват удобен критерий за определяне на ефективността на разработената програмна среда за автоматизирано проектиране на вградена PicoBlase базирана система.

С оглед да се получи достоверна представа за нейното бързодействие, в дисертацията труд са проведени две серии от експерименти. Те позволяват да се получи представа за времената, необходими за съставяне на конфигурационния файл на системата. Във всяка серия взеха участие по 15 човека разпределени в три категории : начинаещи, хора с опит и експерти. Задачата на участниците в първата серия бе да въведат файла с описанието на системата, представен на листинг 1

от част Приложения. Резултатите на участниците в първата серия са дадени в таблица 3.6.

Задачата на участниците във втората серия бе да съставят файла с описанието на системата по блоковата схема, представена на фигура 2.6. Резултатите на участниците от втората серия са приведени в таблица 3.7. Представени като графика, тези резултати са показани на фигура 3.20. За сравнение, на същата графика е показано и усредненото време за съставяне на файла с описанието на системата, реализирано с помощта на разработената в дисертациония труд програмна среда от двама потребители, обучени за работа с нея.

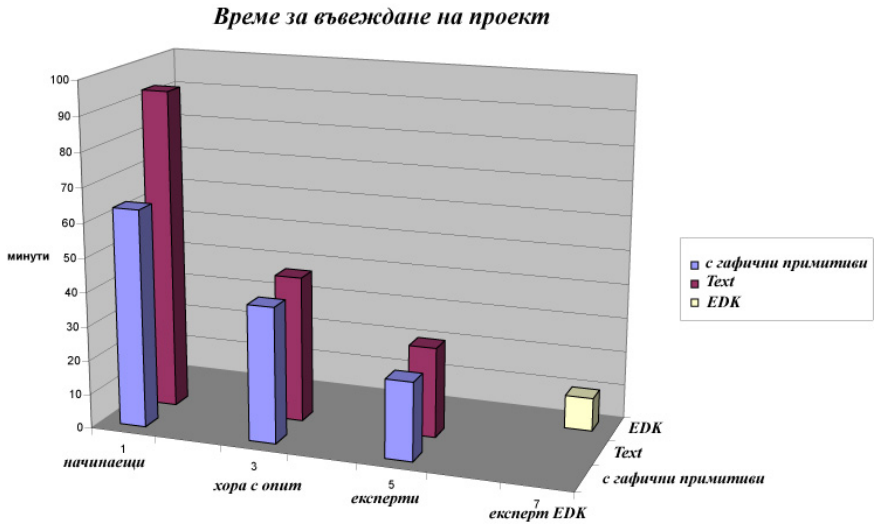
Отношението между времената за въвеждане на файла с описанието на системата и това, получено с използването на разработената програма, показващо постиганото с нейното използване ускорение, е показано на фигура 3.21.

Таблица 3.6. Средно време за въвеждане на проект по текст

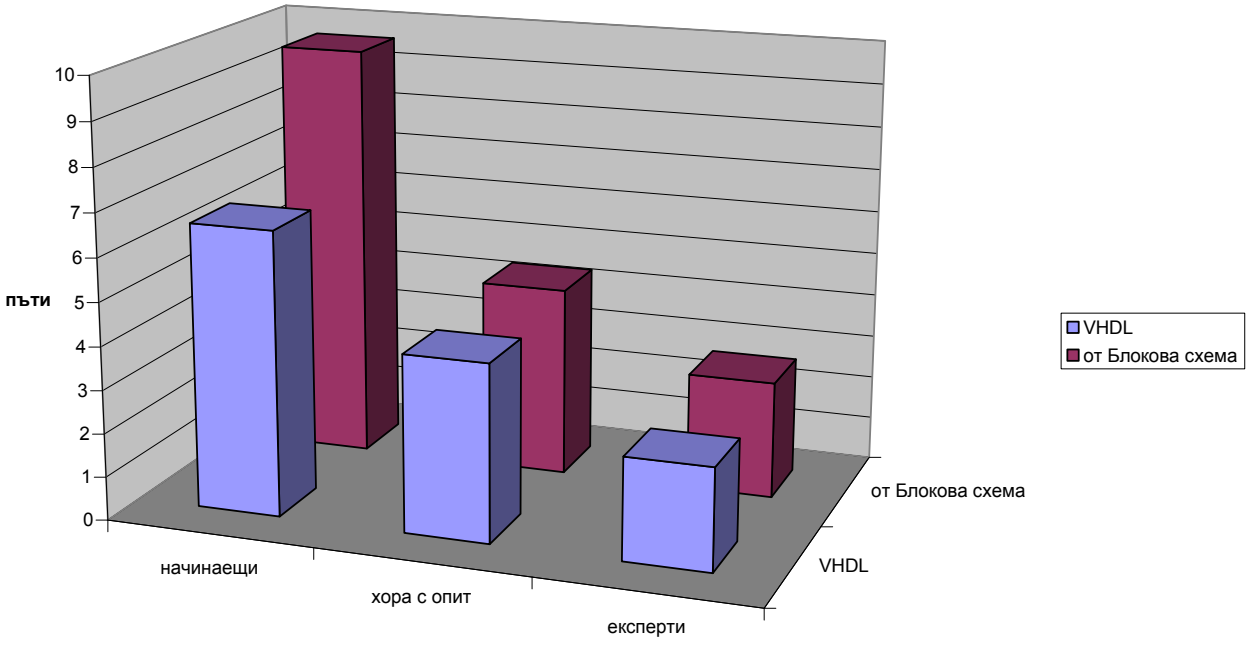
	Средно време за въвеждане на проект по текст [min/стр]					
	Начинаещи		хора с опит		експерти	
	[min]	брой грешки	[min]	брой грешки	[min]	брой грешки
	87	11	35	7	27	11
	97	7	40	5	21	9
	71	13	37	9	33	5
	101	8	53	3	31	6
	111	5	48	11	19	12
средно време	93,4		42,6		26,2	

Таблица 3.7. Средно време за въвеждане на проект по графични примитиви

	Средно време за въвеждане на проект по графични примитиви [min]					
	Начинаещи		хора с опит		експерти	
	[min]	брой грешки	[min]	брой грешки	[min]	брой грешки
	55	1	30	1	27	0
	63	3	40	0	21	0
	71	3	37	0	17	0
	68	0	43	0	31	0
	61	0	48	1	18	1
средно време	63,6		39,6		22,8	



Фигура 3.20. Средно време за съставяне на файла с описание на системата.



Фигура 3.21. Отношение на времената за съставяне на конфигурационният файла с описанието на системата по класическия начин и с използването на разработената в дисертациония труд програма.

Тези графики показват, че разработената в дисертациония труд програмна среда за автоматично генериране на вградени PicoBlase базирани системи позволява да се постигне ускоряване на процеса на проектиране, което в зависимост от квалификацията на проектанта варира в границите от 2 до 10 пъти.

3.4 Проектиране на вграден микроконтролер за система за светлинни ефекти в домове на бъдещето

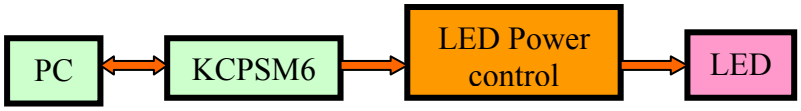
Една от задължителните черти, характеризиращи качествата на домовете на бъдещето е наличието на система за светлинни ефекти. Нейното присъствие в домовете на бъдещето се базира на съществуващото мнение, че цветотерапията е един от перспективните методи за подобряване на здравния фактор на човека. Цветотерапията се е зародила още в древна Гърция. С цвят са лекували в Египет, Китай, Индия и Персия. В египетски храмове съвременните археолози са открили стаи, чиято конструкция пречупвала слънчевите лъчи в разнообразни цветове. По този начин, пациентите са се “къпели” в оздравяващи потоци на целебни лъчи. Може да се каже, че без цветове нашето съществуване е немислимо. Основно ние възприемаме цвета посредством зрението, чрез кожата, мускулите. При проникването на цвета в нашия организъм, той предизвиква различни биохимически реакции в тъканите, стимулира съответните жлези и се оказва необходимо средство за възстановяване на дейностите на организма [86].

Реализацията на система за светлинни ефекти в домовете на бъдещето е изключително удобно да се извърши в рамките на един FPGA прибор. Този подход елиминира всички ограничения от типа на бързодействие и ресурси и позволява използването на вградени процесори. Те се

оказват най-ефективното средство за реализиране на апаратни ресурси, изпълняващи няколко задачи в режим на време деление.

Архитектура

Структурната схема на системата за светлинни ефекти в домове на бъдещето, разработена в дисертациония труд на базата на KCPSM6 е показана на фигура 3.21.



Фигура 3.21. Структурна схема на системата за светлинни ефекти

Това представлява класическа схема, при която главното управление е поверено на PC, а ролята на вградения процесор е сведена до изпълняване на получаваните от PC команди и поддържане на диалог с потребителя.

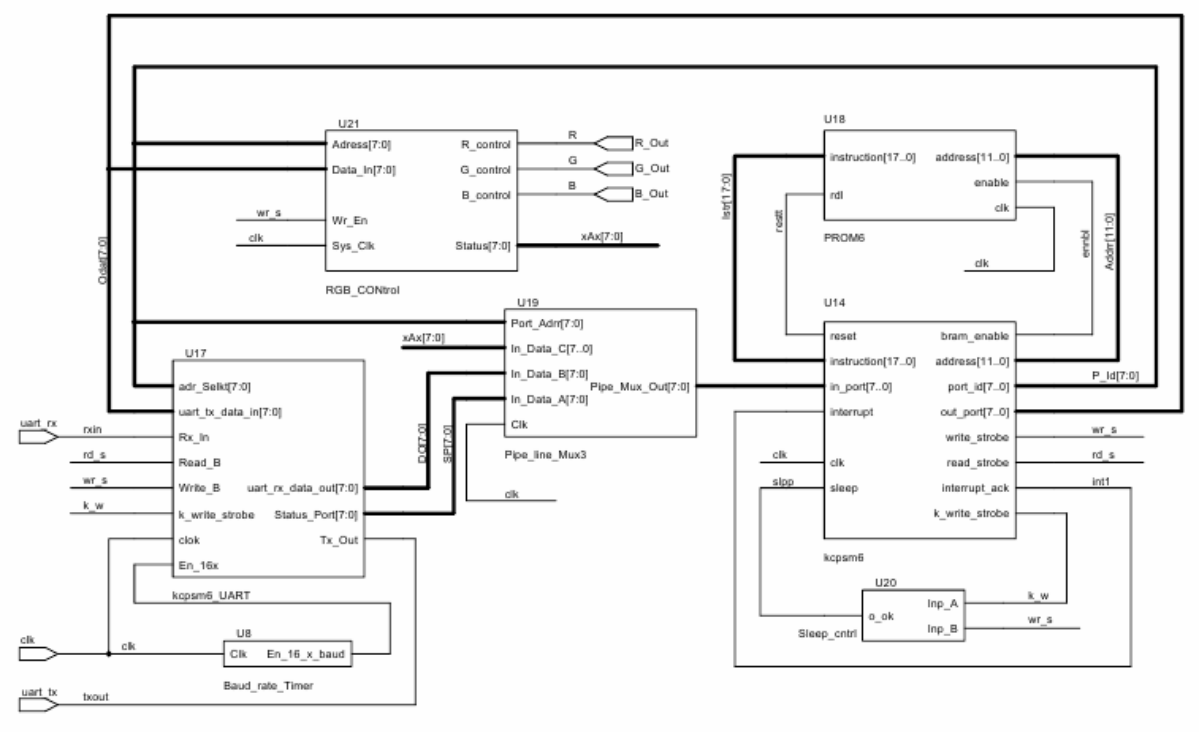
Основното съображение за използването на PC в рамките на разработената система е възможността то да се използва в качеството на терминал в процеса на диагностика и настройка.

За осъществяване на диалога с PC, в системата на вградения процесор е включен апаратен USB – UART мост. Този мост осигурява апаратната връзка с PC при скорост на обмен 115200 бода. Тази скорост се оказва напълно достатъчна за управление на разработената система и позволява нейната синхронизация със звукови въздействия.

Практическа реализация

Апаратната част, която управлява генерирането на едно или друго светлинно въздействие е реализирана на базата на вградения процесор PicoBlaze вариант 6.

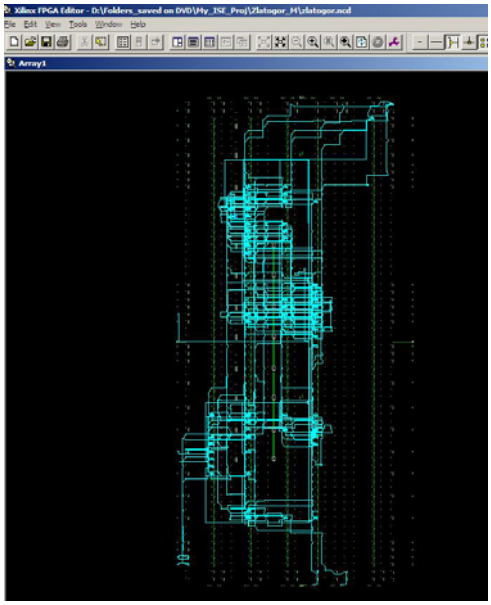
Структурната схема на контролера, управляващ системата за светлинни ефекти в домове на бъдещето, разработена на базата на този процесор е показана на фигура 3.22.



Фигура 3.22. Структурна схема на контролера на системата за светлинни ефекти.

Той е реализиран в FPGA прибора XC6SLX4 от серията Spartan-6 и заема 16% от ресурсите на чипа. Неговото разположение в FPGA е показано на фигура 3.23.

Вграденият процесор, съпътстващата го логистична среда и изпълнителни модули са описани на език VHDL, с прилагане на методиката, описана в публикации [6,7] от списъка на трудовете свързани с дисертацията. Тази методика позволява автоматично получаване на конфигурационен файл на изграждания апаратен контролер и шаблон за управляващата програма на вградения процесор непосредствено от блоковата му схема. Управляващата програма е написана на асемблер и заедно с мониторната и комуникационната програма не надхвърля 300 команди.



Фигура 3.23. Разположение на проекта в FPGA чипа.

Това позволява в паметта на процесора да се съхранява информация за различни светлинни ефекти, които потребителя избира по едни или други съображения. Двупортова структура на тази памет позволява презареждане на управляващото устройство в процеса на работа, без това да се отрази на изпълнението на основните му функции.

Програмно осигуряване

Програмното осигуряване на системата за светлинни ефекти се състои от два програмни модула. Първият управлява поведението на вградения процесор, а вторият-терминалната програма, отговорна за поддържане на диалога с потребителя през PC. Програмата, определяща работата на вградения процесор е написана на асемблер и осигурява изпълнението на следните функции:

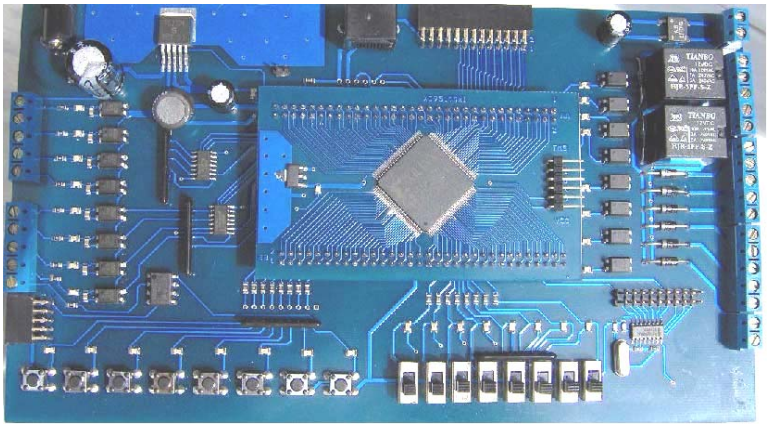
- Инициализира устройството в режим, предварително определен от потребителя.
- Инициализира регистрите, отговорни за генерирането на цвят съобразно получените от PC команди.

- Поддържа обмен на командни последователности и скриптове с терминалната програма.

Програмата за реализиране на светлинните ефекти е направена в средата на Matlab. За реализиране на информационния обмен с PC, системата ползва програмата PicoTerm. Кодът на програмата е показан на листинг 3.

Експериментални резултати

Системата за генериране на светлинни ефекти е реализирана в апаратна среда за развой с препрограмируеми прибори [133]. Тя е показана на фигура 3.24.



Фигура 3.24. Общ вид на управляващото устройство

Заключение и бъдещо развитие

Практически изпитания на устройството за генериране на светлинни ефекти на базата на вграден процесор PicoBlaze вариант KCPSM 6 показва висока надеждност. Неговите възможности позволяват използването му за постигане на различни ефекти в светодиодни табла за информация и реклама.

3.5 Проектиране на генератор на случайни числа

Независимо от голямото разнообразие на програми за генериране на случайни числа, въпросът за тяхното апаратно реализиране не е загубил своята актуалност и заема особено място в процеса на разработка на прецизни математически модели и провеждането на сложни експерименти и обработката на получените от тях резултати.

Необходимостта от създаването на такъв клас прибори се обуславя от факта, че бързодействието, големите функционални възможности на съвременните градивни елементи в лицето на FPGA приборите позволяват изграждането на устройства, откриващи възможности за успешно решаване в реално време на широк кръг от научно приложни задачи.

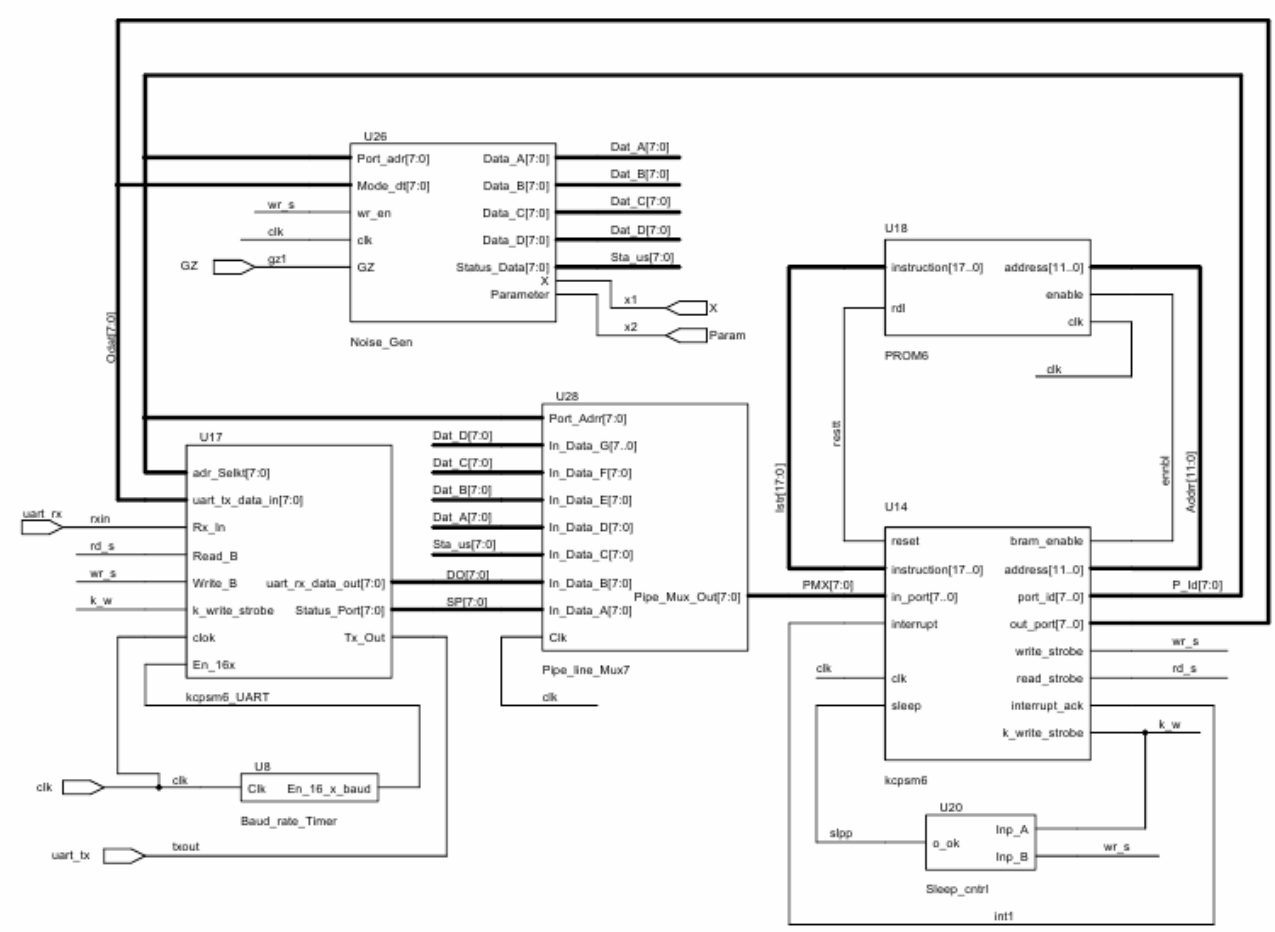
В рамките на дисертационния труд, са извършени различни по вид и обем дейности имащи отношение към създаването на генератор на случайни числа от ново поколение на базата на FPGA прибори:

Архитектура

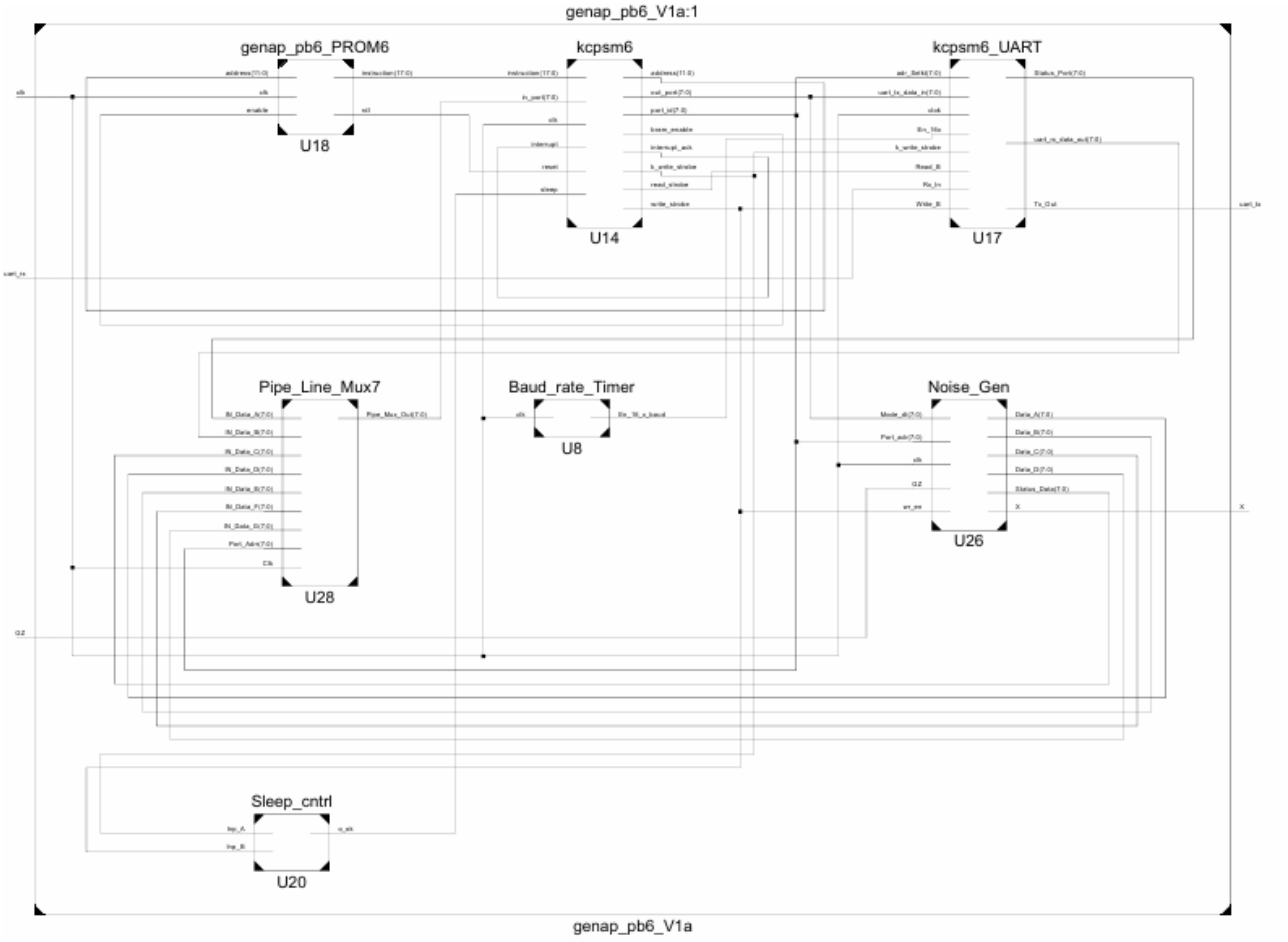
За разработване на генератора на случайни числа е използван FPGA приборът XC6SLX4 от серията Spartan-6. Това позволява да се извърши моделиране и изследване на свойствата и показателите на различни достъпни апаратни и програмни реализации за генериране на случайни числа. Получените резултати позволяват да се направи извода, че използването на структурата на генератора тип GENAP [135], е най подходяща за основа на разработвания генератор на случайни числа, тъй като след добавяне на апаратни разширения позволява генерирането на случайни числа със свойства, практически неотличими от идеалните.

Структурната схема на разработения подобрен генератор на случайни числа е показана на фигура 3.25.

За да се осигури мобилност на разработвания проект, спрямо неговото имплементиране в настоящи и бъдещи модификации на FPGA прибори проектирането се извършва със средствата на езика VHDL. Структурната схема на така разработения генератор е показана на фигура 3.26.

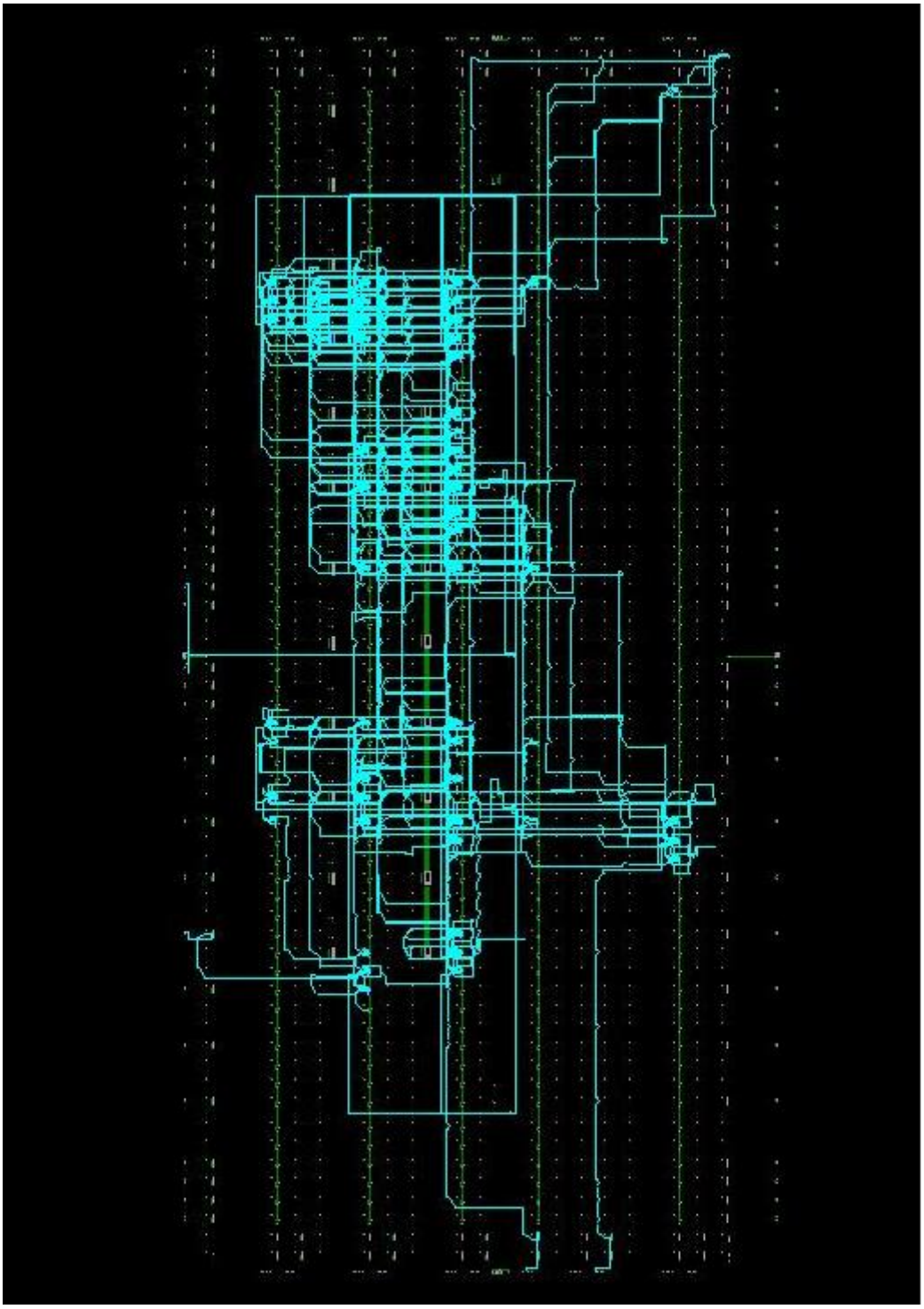


Фигура 3.25. Структурна схема на генератор на случайни числа



Фигура 3.26. Структурна схема на генератора в средата WebPack 14.4.

За всеки един от блоковете, изграждащи структурната схема на генератора, са разработени техническите изисквания, които след физическото им моделиране и проектиране са подложени на симулация в средата WebPack. За апаратната реализация на базовите и специализираните модули на генератора са използвани LUT таблиците на CLB блоковете от вътрешната структура на FPGA прибора в режим SRL16 и SRL32. Разположението на генератора в ресурсите на използвания FPGA прибор е показано на фигура 3.27.



Фигура 3.27. Разпределение на генератора в ресурсите на използвания FPGA прибор.

Този проект е имплементиран в апаратна развойна среда [133]. Неговото практическо изследване в условия максимално близки до

реалните е извършено в ИТИА, (Институт по теория на информацията и автоматизацията) Прага. Проведени са пълни функционални изпитания на генератора. Показателите на генерираните от него случайни числа удовлетворяват предварително зададените параметри на генерираните случайни числа.

3.6. Обобщения и изводи

При генериране на приложения, реализирани с помощта на PicoBlaze базирани системи, е възможно възникването на необходимост от доработка на управляващата програма на процесора или създаване на не съществуващ в библиотеката на разработената среда МПО модул. За подпомагане реализирането на тези процедури, в рамките на тази глава е представена методика за изграждане на МПО модул и доработка на управляващата програма на процесора.

Синтезирано е устройство „генератор на случайни числа” като е използвана разработената програмна система за автоматизирано пректиране с PicoBlaze елементна база.

Извършена е оценка на бързодейдтвието на процеса на проектиране като е постигнато неколкократно съкращаване на времето, необходимо за практическата реализация на проекта.

Приведени са примери за цялостна разработка на вградена PicoBlase базирана система за светлинни ефекти в домове на бъдещето и генератор на случайни числа.

За основа при съставяне на МПО модула съдържащ управляващата програма на процесора, се използва автоматично генерираният в рамките на средата асемблерски файл, към който се добавят описание на специфичните дейности, подлежащи на изпълнение от процесора.

Разработени са правила за използването на средата pBlazIDE на фирма Mediatronix за симулиране на разработената програма.

В качеството на пример за разработка на МПО модул е представено създаването на един апаратен и два програмни модула, осъществяващи комуникация по протокола SPI. Показано е как с въвеждането на незначителни усложнения програмните модули могат да поддържат паралелна работа на няколко устройства по един канал със скорост на обмен 3,5 Mb/sec, при тактова честота на процесора 100 Мхц.

Синтезирано е второ устройство за управление на светлинни ефекти. С разработената програмна система е съставен модул генераща стробираща честота за UART. Параметрите на този модул за трите най-често срещани системни честоти, 25, 50 и 100 мхц, използвани при неговата практическа реализация, са показани в таблица. Показана е възможността за реализирането на модула в рамките на един CLB блок.

Направено е изследване на канала за обмен на данни, реализиран на базата на UART модул.

За подобряване времето за верификация, към програмата за автоматично генериране на вградени PicoBlase базирани системи в тази глава е описана разработената в рамките на дисертационния труд специализирана апаратна среда, която предоставя възможност за проверка на апаратното и програмното осигуряване на всички етапи на разработката и съкращава значително времето за изготвяне на готовото крайно изделие.

В тази глава на дисертационната работа е приложена системата за автоматизирано проектиране на две самостоаятелни хардуерни системи: генератор на случайни числа и система за светлинни ефекти.

Изследванията свързани с използването на разработената в дисертациония труд специализирана апаратна среда изложени в тази

глава са представени в публикация [3] от списъка на публикациите свързани с дисертацията

Приноси на дисертационния труд

1. Направен е анализ на процесите на проектиране на хардуерни устройства с използване на обзор на съвременните FPGA прибори и вградени процесори като е дефинирана необходимостта от разработване на автоматизирани средства за подпомагане процесите на синтез и верификация на функционалностите на хардуерните системи.
2. Разработена е програмна среда позволяваща автоматично проектиране на хардуерни устройства прилагаща вградена PicoBlaze елементна база. В програмната среда са включени нови три типа системни библиотеки:
    - Symbol library
    - PSM libary
    - VHDL libary

Чрез тези нови библиотеки се постига значително повишаване функционалността на автоматизираната среда за проектиране на вградени системи.

3. Програмната среда е приложена за проектиране на реални хардуерни устройства като:

- Генератор на случайни числа.

-Вграден микроконтролер за управление на система за светлинни ефекти в домове на бъдещето

4. Оценена е ефективнастта на работа на програмната система за автоматизирано проектиране. За целта е създаден алгоритъм за количествена оценка на ефективността от използване на автоматизирани средства за проектиране на FPGA прибори. Алгоритъмът е приложен за случаите на разработените уреди в дисертационната работа.

Бъдещи насоки за работа

1. Доработка на разработената програмна среда с оглед на нейното използване за автоматично генериране на многопроцесорни вградени PicoBlaze базирани системи.
2. Доработка на системните библиотеки на разработената програма, позволяващи въвеждането на атрибути за автоматичната им идентификация при генериране на многопроцесорни вградени PicoBlaze базирани системи.
3. Доработка на системните библиотеки на разработената програма, позволяващи тяхното вграждане в средата WebPack.
4. Разработка на програмируем модул за разработената апаратна среда, осигуряващ необходимите и достатъчни условия за работата с FPGA прибори от 7-а серия на Xilinx.

Декларация за оригиналност на резултатите

Декларирам, че настоящата дисертация съдържа оригинални резултати, получени при проведени от мен научни изследвания с подкрепата и съдействието на научния ми ръководител. Резултатите, които са получени, описани и/или публикувани от други учени, са надлежно и подробно цитирани в библиографията.
Настоящата дисертация не е прилагана за придобиване на научна степен в друго висше училище, университет или научен институт.

Подпис:

Благодарности

Бих искал да изразя благодарност на моя научен ръководител Професор Тодор Стоилов, за неговото експертно ръководство и консултации по време на моят PhD курс. Аз научих много от него и през цялото време се чувствах привилегирован от удоволствието на съвместната ни работа, добре подплатена с елегантно чувство за хумор и висок професионализъм.

Също така бих искал да благодаря на всички колеги от секция ЙС при ИИКТ БАН за съветите и интересните провокативни дискусии в интересуващите ме научни области.

И на края големи благодарности на семейството ми, което ме подкрепяше и мотивираше през моето PhD обучение.

Списък на публикациите свързани с дисертацията

На Маг. Инж. Владимир Иванов

1. Иванов. Вл., Иванов. Н.“Етапи при проектирането на picoblaze системи Сб. доклади от Научна конференция 2013 „25 години от полета на втория български космонавт” „Д.Митрополия, 10-11 Октомври 2013, стр90-94 ISBN 978-954-753-177-2

2. Иванов. Вл.“ХАРАКТЕРИСТИКИ И ОСОБЕНОСТИ НА FPGA СЕМЕЙСТВА SPARTAN 6 VIRTEX 6 И СЕРИЯ 7“, *proc. of AUTOMATICS AND INFORMATICS conf., Sofia,3-5 October, 2012, ISSN 1313-1869 page 95-99*

3. Иванов .Вл.“Управление на Стендове с Препрограмируема Логика”, *сб. доклади от Научна конференция „Сервизна роботика и интелигентни системи 2012”, ЦУ БАН, София, 30 ноември, 2012,.ISSN1310-8255, pp85-92*

4. Stoilov T., N. Ivanov, V. Ivanov “The Contemporary FPGA Devices Offered by Xilinx” , Proc. Of International Conference AUTOMATICS AND INFORMATICS ’ 11, 3-7 October 2011, Sofia, ISSN 1313-1850 pp B-377-B382

5. *Ivanov Vladimir*“A program for an automatic PicoBlaze type embedded system generation ” 14-th International Conference on Computer Systems and Technologies , 28-29 June 2013, Ruse, Bulgaria pp 91-97, ACM ISBN: 978-1-4503-2021-4

6. Ivanov. Vl. „On the approach for automatic generation of small embedded PicoBlaze system”, proc of the 13th International Conference on ACSD, 8-10 July Barcelona, Spain,2013 pp257-260, ISBN 978-0-7695-5035-0, ISSN 1550-4808. https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6597648

7. Ivanov. Vladimir “An approach for a PicoBlaze system generation ” proc of DISTRIBUTED COMPUTER AND COMMUNICATION NETWORKS (DCCN-2013): 7-10 October DCCN-2013 pp 233-237 ISBN 978-5-94836-366-0

Научноизследователски проекти

1.Анализиране и идентификация на зависимости в големи масиви от данни – приложение за икономически и технологични анализи" по договор № BG161PO003-1.1.06-0023-C0001. „Подкрепа за научноизследователската и развойна дейност на българските предприятия" от ОП „Развитие на конкурентоспособността на българската икономика" 2007-2013г. р-л. проф.дтн. Т.Стоилов (финансиран от МОН)

2. Проект : "Изследване на Възможностите за идентифициране на кибер-заплахи и тяхната връзка с поведенческата динамика на потребителите в домовете на бъдещето " FFNNIPO_12_00329 р-л. доц. Л. Боянов. (финансиран от МОН)

3.Проектиране и разработване на йерархични информационно-управляващи системи
р-л. проф.дтн. Т.Стоилов(финансиран от ИИКТ)

4. " Използване на съвременни препрограмируеми прибори за разработка на генератори на случайни числа от типа GENAP предназначени за Monte Carlo симулации във финансови изследвания работещи на PC", договор по ЕБР между ИИКТ БАН и Институт по теория на информацията и автоматизация (ІІТА) ЧАН – Прага2011-2013г.

Библиография

1. С.И.Перницкий Многофункциональность. Направления исследований и некоторые способы реализации. TRIZ-fest-2007 www.pentax.ru
2. Earnings & Estimates Summary - XILINX INC (XLNX), Bloomberg Businessweek, Jun 3 2013.
3. Blaza D., A. Wolfe, 2013 Embedded Market Study , proc of the DesignWest conference, April,2013 San Jose CA].
4. Н.Иванов „Алгебра на препрограмируемите прибори - част I", „Атика", София, 2004.
5. www.Xilinx.com
6. www.Xilinx.com Spartan3AN.
7. www.Xilinx.com Virtex 4.
8. www.Xilinx.com Virtex5.
9. Royer M., Goode M. FPGA to ASIC migration // Integrated Systems Design Magazine. – November, 2000.
10. www.Xilinx.com /publications/XCELL issue 51.
11. Solutions for a programmable world Xcell journal 2010 customer inovation issue p11
12. Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers 10193.pdf
13. www.artemis.eu
14. CZ80CPU - 8-bit Microprocessor Core, CAST, Inc. April 2003.
15. B: Roman L., B.Fayette "Emulate 8051 Microprocessor in PicoBlaze
16. Spartan_3E FPGA Family: Complete Data Sheet.
17. Spartan_3/3E Starter Kit Resource CD.
18. Modeling Kinematic Cellular Automata NASA Inst. for Adv. Concepts April 30, 2004
19. A.Janusz,Y. Starzyk,, Z. Zhineng "Dynamically Reconfigurable Neuron Architecture for the Implementation of Self-Organizing Learning Array" Proc of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04)
20. Mostafa Abd-El-Barr, Sadiq M. Sait, Bambang A. B. Sarif, Uthman AI-Saiari, "A Modified Ant Colony Algorithm for Evolutionary Design of Digital Circuits", 0-7803-7804.0 /0317.00 t2 2003 EEE
21. Bahreininejad, "A hybrid ant colony optimization approach for .nite element mesh decomposition" DOI 10.1007/s00158-004-0432-x, Struct Multidisc Optim 28, 2004.
22. KLEINROCK L., "DISTRIBUTED SYSTEMS", Communications of the ACM, November 1985 Volume 28 Number 11
23. J.A. Reggia, H.-H. Chou, and J.D. Lohn. ``Cellular Automata Models of Self-replicating Systems,'' in Advances in Computers, M. I
24. B. Fletcher, FPGA Embedded Processors,Embedded Systems Conference, San Francisco, 2005
25. Н. Иванов „Микропроцесорни ядра за вграждане в FPGA", Автоматика и Информатика,№1, 2007.
26. Altera Corporation, The ARM992T Escalibur Development Handbook.
27. PowerPC Processor Reference Guide. Xilinx Inc. 2003.

28. PowerPC™ 405 Processor Block Reference Guide. Xilinx Inc. 2004.
29. Processor IP Reference Guide. Xilinx Inc.2005.
30. PPC440x4 CPU Core, www.ibm.com/chips/techlib/techlib.nsf/techdocs/ppc440.pdf .
31. EDK Concepts, Tools, and Techniques UG683 April 13, 2011,www.xilinx.com
32. Altera Corporation, Nios® II Processor Reference Handbook.
33. LatticeMico8 Core Source Code Revision 2.3 VHDL, Lattice Semiconductor Corp.
34. MicroBlaze Microcontroller Reference Design UG133 v1.3.1, 2005.
35. MicroBlaze Processor Reference Guide Embedded Development Kit EDK 7.1i UG081
36. Chapman PicoBlaze 8-Bit Microcontroller for Virtex Devices XAPP213, 2002
37. XCELL journal No 64
38. B: Roman L., B.Fayette “Emulate 8051 Microprocessor in PicoBlaze IP Core”, Xilinx Embedded magazine March 2005.
39. PicoBlaze 8-bit Embedded Microcontroller User Guide, for Spartan-3, Virtex-II, and Virtex-II Pro FPGAs UG129, 2004.
40. Зотов В. Особенности микропроцессорного ядра PicoBlaze, предназначенного для применения в проектах, реализуемых на основе ПЛИС семейства CoolRunner-II // Компоненты и технологии. 2003. № 7.
41. kcpsm3 for fun
42. Device Reliability Report UG116 Nov. 2013,www.xilinx.com
43. UG470 Oct. 2013,www.xilinx.com
44. Н.Иванов „Алгебра на препрограмируемите прибори - част II”, „Атика”, София, 2008.
45. Chapman KCPSM6 relise 7, www.xilinx.com
46. Кузьмин Л. Т. 'Основы кибернетики. Т. 1. Москва: Энергия, 1973 - с.500
47. Вунш Г. Теория систем, Москва, Сов. радио, 1978
48. Вариченко Л.В. Абстрактные алгебраические системы и цифровая обработка сигналов, Киев, Наукова думка, 1986
49. Voutsadakis G. Categorical Abstract Algebraic Logic:(I,N)-Algebraic Systems,Applied Categorical Structures Springer 2005
50. Kimura T. “An Algebraic System For Process Structuri~G And Interprocess Communication”, University of Delaware Newark, 1971
51. Малцев А.И. Алгебраические системы М., Наука, 1970.
52. Ленг С. Алгебра М., Мир, 1968.
53. Постников М.М. Введение в теорию алгебраических чисел М., Наука, 1982
54. Акошинский И., Д. Юдитский „Машинная арифметика в остаточных классах”, М. Сов. Радио, 1968.
55. Блейхут Р. Быстрые алгоритмы цифровой обработки сигналов, М., Мир 1989
56. Аркитас А. Основы компьютерной алгебры с приложениями, М., Мир, 1994
57. Gregory R.T., E.V. Krishnamurthy Methods and Applications of Error-Free Computation, Springer-Verlagp New York 1984
58. Коблиц Н. p- адические числа, p-адический анализ и дзета-функции, М., Мир, 1982
59. Ейтс С. Репюниты и десятичные периоды, М., Мир 1992.
60. Morris K. Join the cool club, XCELL Journal No 54.

61. Clarke P, Xilinx launches ESL initiative, EE Times , 03/13/2006.
62. The Missing Link: In Search of an ESL-to-RTL Design Flow, 5th Annual ESL Symposium at DAC2007, San Diego, Convention Center
63. Wilson R., “Is there fire beneath the smoke”, EDN No34, Aug.2008.
64. Modeling Kinematic Cellular Automata NASA Inst. for Adv. Concepts April 30, 2004
65. www.Mentor.com
66. Anderson T., Bhagat R. Tackling Functional Verification for Virtual Components Integrated Systems Design Magazine. – November, 2000.
67. Файн В.С. Распознавание образов и машинное понимание естественного языка М.: Наука, 1987.
68. www.Fine_System.org
69. Labunets,V.G. Algebraic Theory of signals and Systems (in Russian). Krasnojarsk State University Press 1984
70. Chapman UART macros KCPSM3 Reference Design Xilinx Ltd, Jan 2003
71. K. Chapman “UART Real Time Clock” KCPSM3 Reference Design Xilinx Ltd Sept.2003.
72. Chapman “KCPSM3 for Spartan 3” Reference Design Xilinx Ltd Oct.2003.
73. Ken Chapman DS2432 Communicator, Xilinx, April,2006
74. Ken Chapman UART Real Time Clock Ref. Design Xilinx, Sept. 2003
75. www.xilinx.com/picoblase
76. www.xilinx.com/ipcenter/processor_central/picoblaze
77. TRACE-32 - In-circuit Emulators&Debuggers, Embedded Systems, 2000, Vol. 4 №31.
78. Maintain control with IAR visualSTATE, Embedded Systems. - 2000. – Vol.4. - №31.
79. www.xilinx.com/products/intellectual roperty ipdesignflow.htm
80. wp276, www.xilinx.com
81. www.xilinx.com/products/intellectual-property/ipdesignflow.htm
82. www.mediatronix.com/pages/pBlazIDE
83. Nexys 3,www.digilentinc.com
84. http://forums.xilinx.com/t5/PicoBlaze/divide-clock-by-13-5-circuit/td-p/328621
85. www.ftdichip.com
86. http://larisa-love.com/cvetoterapiya-lechenie-cvetom].
87. Философский энциклопедический словарь. М. СЭ. 1983г
88. С.В. Гусс Предметно –ориентированныые проектные решения для тематической области обучающих программных средств на основе лингвистических игр, Математические структуры и моделирование 2011, вып. 24, с. 55–68
89. Д.А. Губанов, Макаренко А.В., Новиков Д.А. Методы анализа терминологической структуры предметной области сб.тр.ИПУ РАН, 2010, Москва.
90. А . Б . Кунгурцев, Бородавкин С . Н . Метод построения словарей предметных областей для извлечения фактов из текстов на естественном языке, Восточно-Европейский журнал передовых технологий 1/4 (43) 2010
91. Todor Stefanov Ed Deprettere Hristo Nikolov Marin Marinov Angel Popov : Embedded System:components,modeling,design and case study Technical University of Sofia 2012 ISBN 978-954-438-975-8
92. Г.С.Альтшуллер. Найти идею. Новосибирск ,Наука, 1989 г.

93. Г.С.Альтшуллер, Б.Л.Злотин, А.В.Зусман, В.И.Филатов Поиск новых идей: от озарения к технологии. Кишинев. Картя Молдовеняскэ. 1989г. .
94. А.В.Федухин,А.А.Муха,А.А.Муха, Плис-системы как средство повышения отказоустойчивости Математичні машини і системи , 2010, No 1 ISSN 1028-9763. стр198-204
95. Сравнительный анализ применения ПЛИС и микропроцессоров при разработке информационно-управляющих систем, важных для безопасности АЭС // Научно-технический отчет. НАУ им. Н.Е. Жуковского«ХАИ», НТСКБ «Полисвит», ИПМЭ им. Г.Е. Пухова НАН Украины, ИПММС НАН Украины. – 2005.
96. Палагин А.В. Системы верификации на основе реконфигурируемых устройств Математичні машини і системи. – 2004. – № 2. стр. 100 – 113
97. Clive Maxfield What's the number of ASIC versus FPGA design starts EDT 3/20/2011 04:32 PM
98. И. Шагурин Системы на кристалле особенности реализации и переспективы применения. Сп. Электронные компоненты №1 2009 стр. 58
99. Немудров В., Мартин Г. Системы на кристалле. Проектирование и развитие. — М.: Техносфера, 2004, с. 216.
100. http://www.ehow.com/facts_6878538_embedded-processor definition.htm
101. И. Тарасов, Проектирование конфигурируемых процессоров на базе ПЛИС ж. Компоненты и технологии • № 2 '2006 стр. 78
102. Александр Калачев Мультиядерные процессоры arm-архитектуры, Компоненты и технологии • № 3 '2010 стр.66.
103. Ал. Калачёв многоядерная конфигурируемая вычислительная платформа zynqq7000 ж.современная электроника № 1 2013 стр.22
104. Why are Altera's Utilization Benchmarking Results Different,xilinx.com/wp284
105. А.В. Палагин, Ю.С. Яковлев, Е.В. Елисеева, Особенности подхода к выбору ПЛИС для проектирования PIM-систем . математичні машини і системи, 2012, № 3 issn 1028-9763
106. Bryan H. Fletcher FPGA Embedded Processors Embedded Training Program Embedded Systems Conference San Francisco 2005 ETP-367 pp18
107. Platform Specification Format Reference Manual UG1044 (v2014.1) May 15, 2014 www.xilinx.com
108. Embedded System Tools Reference Manual UG1043 (v2014.1) May 15, 2014 www.xilinx.com
109. О.А. Ильяшенко , В.С. Харченко , Г. Ерван, Информационная безопасностъ индустриальных иус на FPGA: нормативная база и SIS подход , ISSN 1814-4225. Радіоелектронні і комп'ютерні системи , 2013, №3,pp86
110. M. Klein, Kolluri S., Leveraging Power Leadership at 28 nm with Xilinx 7 Series FPGAs WP436 ,pp12.
111. А.Е. Платунов, Н.П. Постников , Высокоуровневое проектирование встраиваемых системи Учебное пособие ИТПД Санкт-Петербург 2011
112. http://www.xilinx.com/publications/prod_mktg/low-end-portfolio-brief.pdf
113. http://www.klabs.org
114. Харченко В.С. Парадигмы и принципы гарантоспособных вычислений: состояние и перспективы развития Радіоелектронні і комп'ютерні системи. – 2009. – № 2. стр.91 – 100.

115. Попович А.В. ПЛИС Actel – платформа для «систем на кристалле» бортовой аппаратуры Электроника : Наука, Технология, Бизнес. – 2004. – № 4.стр. 34 – 37.
116. ГОСТ МЭК 61508 «Функциональная безопасность электрических / электронных / программируемых электронных схем, относящихся к безопасности» (в семи частях). – 2007.
117. Avizienis A. Fundamental Concepts of Dependability Technical Report: UCLA CSD Report no. 010028, LAAS Report no. 01-145, Newcastle Report no. CS-TR-739. – 2002.
118. Avizienis A. A Fault Tolerance Infrastructure for Dependable Computing with High-Performance COTS Components. USA, 2000.
119. Randell B. Reliability Issues in Computing System Computing Laboratory, University of Newcastle upon Tyne, NE1 7R U UK. – 1978.
120. ЄфімоваТ.І. Відмовостійкість програмного забезпечення гарантоздатних комп'ютерних систем Математичні машини і системи, № 4.2009 – стр. 200 – 209.
121. http://www.actel.com.
122. http://www.fastwel.ru.
123. http://forums.xilinx.com/t5/PicoBlaze/PicoBlaze FAQ.doc
124. Сперанский В. С. Сигнальные микропроцессоры и их применение в системах телекоммуникаций и электроники — М.: Горячая линия–Телеком, 2008.
125. Литюк В. И., Литюк Л. В. Методы цифровой многопроцессорной обработки ансамблей радиосигналов — М.: Салон-Пресс, 2007.].
126. Aycinena P. Programmable Logic Devices: The Wave of Future, Integrated Systems Design Magazine. January, 2001. http://www.isdmag.com/editorial/2001/focusreport0101.html
127. B.Hu Custom FPGA-Based Emulators Accelerate IC Design Verification Integrated Systems Design Magazine. - January, 2001.
128. Mackenzie W., Demaine J. HW/SW Co-Verification Tools Make Some Progress // Integrated Systems Design Magazine. - January, 2001.
129. Smith G. The Dream - Communications/Core-Based Design // Integrated Systems Design Magazine. – December, 2000.
130. . Goering R. 21-st Century EDA: Looking Beyond ASICs // Integrated Systems Design Magazine. – December, 2000.
131. www.UML.org
132. Schulz S. Focus Report: System Design Tools // Integrated Systems Design Magazine. – November, 2000.
133. Вл. Иванов, Н. Иванов "Разширяване на възможностите на Специализирана среда за работа с препрограмируеми прибори" Сб. доклади от Научна конференция 2011 „50 години от полета на първия човек в космоса" ,Д.Митрополия, Април2011, стр158-161
134. Ababei C. PicoBlaze – an embedded microcontroller EE-459/500 HDL Based Digital Design with Programmable Logic Electrical Engineering Department, University at Buffalo, November 2012.
135. J. Havel The generation of truly random binary digits *J. Phys. E: Sci. Instrum*1973, vol 6.
136. www.xilinx.com/xcell 83

Приложения

Листинг 1.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY SCHEMATIC1 IS PORT (
uart_tx : OUT std_logic;
X : OUT std_logic;
Param : OUT std_logic;
GZ : IN std_logic;
clk : IN std_logic;
uart_rx : IN std_logic);

END SCHEMATIC1;

ARCHITECTURE STRUCTURE OF SCHEMATIC1 IS

-- COMPONENTS

COMPONENT kcpsm6
PORT (
write_strobe : OUT std_logic;
out_port : OUT std_logic_vector(7 DOWNTO 0);
address : OUT std_logic_vector(11 DOWNTO 0);
instruction : IN std_logic_vector(17 DOWNTO 0);
port_id : OUT std_logic_vector(7 DOWNTO 0);
in_port : IN std_logic_vector(7 DOWNTO 0);
read_strobe : OUT std_logic;
interrupt : IN std_logic;
reset : IN std_logic;
clk : IN std_logic;
interrupt_ack : OUT std_logic;
bram_enable : OUT std_logic;
k_write_strobe : OUT std_logic;
sleep : IN std_logic);
 END COMPONENT;

COMPONENT kcpsm6_UART
PORT (
uart_rx_data_out : OUT std_logic_vector(7 DOWNTO 0);
Read_B : IN std_logic;
En_16x : IN std_logic;
clok : IN std_logic;
Rx_In : IN std_logic;
Write_B : IN std_logic;
uart_tx_data_in : IN std_logic_vector(7 DOWNTO 0);
Status_Port : OUT std_logic_vector(7 DOWNTO 0);
Tx_Out : OUT std_logic;
adr_Selkt : IN std_logic_vector(7 DOWNTO 0);
k_write_strobe : IN std_logic);
END COMPONENT;

COMPONENT PROM6
PORT (
address : IN std_logic_vector(11 DOWNTO 0);
instruction : OUT std_logic_vector(17 DOWNTO 0);
clk : IN std_logic;
enable : IN std_logic;
rdl : OUT std_logic);
 END COMPONENT;

COMPONENT Sleep_cntrl
PORT (
Inp_A : IN std_logic;
o_ok : OUT std_logic;
Inp_B : IN std_logic);
 END COMPONENT;

COMPONENT Baud_rate_Timer
PORT (
Clk : IN std_logic;
En_16_x_baud : OUT std_logic);
 END COMPONENT;

COMPONENT Noise_Gen
PORT (
Data_A : OUT std_logic_vector(7 DOWNTO 0);
clk : IN std_logic;
Port_adr : IN std_logic_vector(7 DOWNTO 0);
wr_en : IN std_logic;
Mode_dt : IN std_logic_vector(7 DOWNTO 0);
Data_B : OUT std_logic_vector(7 DOWNTO 0);
Data_C : OUT std_logic_vector(7 DOWNTO 0);
Status_Data : OUT std_logic_vector(7 DOWNTO 0);
Data_D : OUT std_logic_vector(7 DOWNTO 0);
GZ : IN std_logic;
X : OUT std_logic;
Parameter : OUT std_logic);
 END COMPONENT;

COMPONENT Pipe_line_Mux7
PORT (
In_Data_G : IN std_logic_vector(7 DOWNTO 0);
Clk : IN std_logic;
Port_Adrr : IN std_logic_vector(7 DOWNTO 0);
In_Data_F : IN std_logic_vector(7 DOWNTO 0);
In_Data_E : IN std_logic_vector(7 DOWNTO 0);
In_Data_D : IN std_logic_vector(7 DOWNTO 0);
In_Data_C : IN std_logic_vector(7 DOWNTO 0);
In_Data_B : IN std_logic_vector(7 DOWNTO 0);
In_Data_A : IN std_logic_vector(7 DOWNTO 0);
Pipe_Mux_Out : OUT std_logic_vector(7 DOWNTO 0));
 END COMPONENT;

-- SIGNALS

SIGNAL N01659 : std_logic;
SIGNAL WR_S : std_logic;
SIGNAL RESTT : std_logic;
SIGNAL SLPP : std_logic;
SIGNAL K_W : std_logic;
SIGNAL ENNBL : std_logic;
SIGNAL RD_S : std_logic;
SIGNAL INT1 : std_logic;
SIGNAL SP : std_logic_vector(7 DOWNTO 0);
SIGNAL DAT_A : std_logic_vector(7 DOWNTO 0);
SIGNAL DAT_B : std_logic_vector(7 DOWNTO 0);
SIGNAL ISTR : std_logic_vector(17 DOWNTO 0);
SIGNAL P_ID : std_logic_vector(7 DOWNTO 0);
SIGNAL STA_US : std_logic_vector(7 DOWNTO 0);
SIGNAL DAT_C : std_logic_vector(7 DOWNTO 0);
SIGNAL DAT_D : std_logic_vector(7 DOWNTO 0);
SIGNAL ODAT : std_logic_vector(7 DOWNTO 0);
SIGNAL ADDRR : std_logic_vector(11 DOWNTO 0);
SIGNAL PMX : std_logic_vector(7 DOWNTO 0);
SIGNAL DO : std_logic_vector(7 DOWNTO 0);

-- GATE INSTANCES

BEGIN
U14 : kcpsm6            PORT MAP(
write_strobe => WR_S,
out_port(7) => ODAT(7),
out_port(6) => ODAT(6),
out_port(5) => ODAT(5),
out_port(4) => ODAT(4),
out_port(3) => ODAT(3),
out_port(2) => ODAT(2),
out_port(1) => ODAT(1),
out_port(0) => ODAT(0),
address(11) => ADDRR(11),
address(10) => ADDRR(10),
address(9) => ADDRR(9),
address(8) => ADDRR(8),
address(7) => ADDRR(7),
address(6) => ADDRR(6),
address(5) => ADDRR(5),
address(4) => ADDRR(4),
address(3) => ADDRR(3),
address(2) => ADDRR(2),
address(1) => ADDRR(1),
address(0) => ADDRR(0),
```

```
instruction(17) => ISTR(17),
instruction(16) => ISTR(16),
instruction(15) => ISTR(15),
instruction(14) => ISTR(14),
instruction(13) => ISTR(13),
instruction(12) => ISTR(12),
instruction(11) => ISTR(11),
instruction(10) => ISTR(10),
instruction(9) => ISTR(9),
instruction(8) => ISTR(8),
instruction(7) => ISTR(7),
instruction(6) => ISTR(6),
instruction(5) => ISTR(5),
instruction(4) => ISTR(4),
instruction(3) => ISTR(3),
instruction(2) => ISTR(2),
instruction(1) => ISTR(1),
instruction(0) => ISTR(0),
port_id(7) => P_ID(7),
port_id(6) => P_ID(6),
port_id(5) => P_ID(5),
port_id(4) => P_ID(4),
port_id(3) => P_ID(3),
port_id(2) => P_ID(2),
port_id(1) => P_ID(1),
port_id(0) => P_ID(0),
in_port(7) => PMX(7),
in_port(6) => PMX(6),
in_port(5) => PMX(5),
in_port(4) => PMX(4),
in_port(3) => PMX(3),
in_port(2) => PMX(2),
in_port(1) => PMX(1),
in_port(0) => PMX(0),
read_strobe => RD_S,
interrupt => INT1,
reset => RESTT,
clk => CLK,
interrupt_ack => INT1,
bram_enable => ENNBL,
k_write_strobe => K_W,
sleep => SLPP);

U17 : kcpsm6_UART
PORT MAP(
uart_rx_data_out(7) => DO(7),
uart_rx_data_out(6) => DO(6),
uart_rx_data_out(5) => DO(5),
uart_rx_data_out(4) => DO(4),
uart_rx_data_out(3) => DO(3),
uart_rx_data_out(2) => DO(2),
uart_rx_data_out(1) => DO(1),
uart_rx_data_out(0) => DO(0),
Read_B => RD_S,
En_16x => N01659,
clok => CLK,
Rx_In => UART_RX,
Write_B => WR_S,
uart_tx_data_in(7) => ODAT(7),
uart_tx_data_in(6) => ODAT(6),
uart_tx_data_in(5) => ODAT(5),
uart_tx_data_in(4) => ODAT(4),
uart_tx_data_in(3) => ODAT(3),
uart_tx_data_in(2) => ODAT(2),
uart_tx_data_in(1) => ODAT(1),
uart_tx_data_in(0) => ODAT(0),
Status_Port(7) => SP(7),
Status_Port(6) => SP(6),
Status_Port(5) => SP(5),
Status_Port(4) => SP(4),
Status_Port(3) => SP(3),
Status_Port(2) => SP(2),
Status_Port(1) => SP(1),
Status_Port(0) => SP(0),
Tx_Out => UART_TX,
adr_Selkt(7) => P_ID(7),
adr_Selkt(6) => P_ID(6),
adr_Selkt(5) => P_ID(5),
adr_Selkt(4) => P_ID(4),
adr_Selkt(3) => P_ID(3),
adr_Selkt(2) => P_ID(2),
adr_Selkt(1) => P_ID(1),
adr_Selkt(0) => P_ID(0),
k_write_strobe => K_W);
U18 : PROM6
PORT MAP(
address(11) => ADDRR(11),
address(10) => ADDRR(10),
address(9) => ADDRR(9),
address(8) => ADDRR(8),
address(7) => ADDRR(7),
address(6) => ADDRR(6),
address(5) => ADDRR(5),
address(4) => ADDRR(4),
address(3) => ADDRR(3),
address(2) => ADDRR(2),
address(1) => ADDRR(1),
address(0) => ADDRR(0),
instruction(17) => ISTR(17),
instruction(16) => ISTR(16),
instruction(15) => ISTR(15),
instruction(14) => ISTR(14),
instruction(13) => ISTR(13),
instruction(12) => ISTR(12),
instruction(11) => ISTR(11),
instruction(10) => ISTR(10),
instruction(9) => ISTR(9),
instruction(8) => ISTR(8),
instruction(7) => ISTR(7),
instruction(6) => ISTR(6),
instruction(5) => ISTR(5),
instruction(4) => ISTR(4),
instruction(3) => ISTR(3),
instruction(2) => ISTR(2),
instruction(1) => ISTR(1),
instruction(0) => ISTR(0),
clk => CLK,
enable => ENNBL,
rdl => RESTT);

U20 : Sleep_cntrl       PORT MAP(
Inp_A => K_W,
o_ok => SLPP,
Inp_B => WR_S);

U8 : Baud_rate_Timer
PORT MAP(
Clk => CLK,
En_16_x_baud => N01659);

U26 : Noise_Gen       PORT MAP(
Data_A(7) => DAT_A(7),
Data_A(6) => DAT_A(6),
Data_A(5) => DAT_A(5),
Data_A(4) => DAT_A(4),
Data_A(3) => DAT_A(3),
Data_A(2) => DAT_A(2),
Data_A(1) => DAT_A(1),
Data_A(0) => DAT_A(0),
clk => CLK,
Port_adr(7) => P_ID(7),
Port_adr(6) => P_ID(6),
Port_adr(5) => P_ID(5),
Port_adr(4) => P_ID(4),
Port_adr(3) => P_ID(3),
Port_adr(2) => P_ID(2),
Port_adr(1) => P_ID(1),
Port_adr(0) => P_ID(0),
wr_en => WR_S,
Mode_dt(7) => ODAT(7),
Mode_dt(6) => ODAT(6),
Mode_dt(5) => ODAT(5),
Mode_dt(4) => ODAT(4),
Mode_dt(3) => ODAT(3),
Mode_dt(2) => ODAT(2),
Mode_dt(1) => ODAT(1),
Mode_dt(0) => ODAT(0),
Data_B(7) => DAT_B(7),
```

```
Data_B(6) => DAT_B(6),
Data_B(5) => DAT_B(5),
Data_B(4) => DAT_B(4),
Data_B(3) => DAT_B(3),
Data_B(2) => DAT_B(2),
Data_B(1) => DAT_B(1),
Data_B(0) => DAT_B(0),
Data_C(7) => DAT_C(7),
Data_C(6) => DAT_C(6),
Data_C(5) => DAT_C(5),
Data_C(4) => DAT_C(4),
Data_C(3) => DAT_C(3),
Data_C(2) => DAT_C(2),
Data_C(1) => DAT_C(1),
Data_C(0) => DAT_C(0),
Status_Data(7) => STA_US(7),
Status_Data(6) => STA_US(6),
Status_Data(5) => STA_US(5),
Status_Data(4) => STA_US(4),
Status_Data(3) => STA_US(3),
Status_Data(2) => STA_US(2),
Status_Data(1) => STA_US(1),
Status_Data(0) => STA_US(0),
Data_D(7) => DAT_D(7),
Data_D(6) => DAT_D(6),
Data_D(5) => DAT_D(5),
Data_D(4) => DAT_D(4),
Data_D(3) => DAT_D(3),
Data_D(2) => DAT_D(2),
Data_D(1) => DAT_D(1),
Data_D(0) => DAT_D(0),
GZ => GZ,
X => X,
Parameter => PARAM);
U28 : Pipe_line_Mux7
PORT MAP(
In_Data_G(7) => DAT_D(7),
In_Data_G(6) => DAT_D(6),
In_Data_G(5) => DAT_D(5),
In_Data_G(4) => DAT_D(4),
In_Data_G(3) => DAT_D(3),
In_Data_G(2) => DAT_D(2),
In_Data_G(1) => DAT_D(1),
In_Data_G(0) => DAT_D(0),
Clk => CLK,
Port_Adrr(7) => P_ID(7),
Port_Adrr(6) => P_ID(6),
Port_Adrr(5) => P_ID(5),
Port_Adrr(4) => P_ID(4),
Port_Adrr(3) => P_ID(3),
Port_Adrr(2) => P_ID(2),
Port_Adrr(1) => P_ID(1),
Port_Adrr(0) => P_ID(0),
In_Data_F(7) => DAT_C(7),
In_Data_F(6) => DAT_C(6),
In_Data_F(5) => DAT_C(5),
In_Data_F(4) => DAT_C(4),
In_Data_F(3) => DAT_C(3),
In_Data_F(2) => DAT_C(2),
In_Data_F(1) => DAT_C(1),
In_Data_F(0) => DAT_C(0),
In_Data_E(7) => DAT_B(7),
In_Data_E(6) => DAT_B(6),
In_Data_E(5) => DAT_B(5),
In_Data_E(4) => DAT_B(4),
In_Data_E(3) => DAT_B(3),
In_Data_E(2) => DAT_B(2),
In_Data_E(1) => DAT_B(1),
In_Data_E(0) => DAT_B(0),
In_Data_D(7) => DAT_A(7),
In_Data_D(6) => DAT_A(6),
In_Data_D(5) => DAT_A(5),
In_Data_D(4) => DAT_A(4),
In_Data_D(3) => DAT_A(3),
In_Data_D(2) => DAT_A(2),
In_Data_D(1) => DAT_A(1),
In_Data_D(0) => DAT_A(0),
In_Data_C(7) => STA_US(7),
In_Data_C(6) => STA_US(6),
In_Data_C(5) => STA_US(5),
In_Data_C(4) => STA_US(4),
In_Data_C(3) => STA_US(3),
In_Data_C(2) => STA_US(2),
In_Data_C(1) => STA_US(1),
In_Data_C(0) => STA_US(0),
In_Data_B(7) => DO(7),
In_Data_B(6) => DO(6),
In_Data_B(5) => DO(5),
In_Data_B(4) => DO(4),
In_Data_B(3) => DO(3),
In_Data_B(2) => DO(2),
In_Data_B(1) => DO(1),
In_Data_B(0) => DO(0),
In_Data_A(7) => SP(7),
In_Data_A(6) => SP(6),
In_Data_A(5) => SP(5),
In_Data_A(4) => SP(4),
In_Data_A(3) => SP(3),
In_Data_A(2) => SP(2),
In_Data_A(1) => SP(1),
In_Data_A(0) => SP(0),
Pipe_Mux_Out(7) => PMX(7),
Pipe_Mux_Out(6) => PMX(6),
Pipe_Mux_Out(5) => PMX(5),
Pipe_Mux_Out(4) => PMX(4),
Pipe_Mux_Out(3) => PMX(3),
Pipe_Mux_Out(2) => PMX(2),
Pipe_Mux_Out(1) => PMX(1),
Pipe_Mux_Out(0) => PMX(0));
END STRUCTURE;
```

Листинг 2.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY SCHEMATIC1 IS PORT (
uart_tx : OUT std_logic;
X : OUT std_logic;
Param : OUT std_logic;
GZ : IN std_logic;
clk : IN std_logic;
uart_rx : IN std_logic);

END SCHEMATIC1;

ARCHITECTURE STRUCTURE OF SCHEMATIC1 IS

-- COMPONENTS

COMPONENT kcpsm6
PORT (
write_strobe : OUT std_logic;
out_port : OUT std_logic_vector(7 DOWNTO 0);
address : OUT std_logic_vector(11 DOWNTO 0);
instruction : IN std_logic_vector(17 DOWNTO 0);
port_id : OUT std_logic_vector(7 DOWNTO 0);
in_port : IN std_logic_vector(7 DOWNTO 0);
read_strobe : OUT std_logic;
interrupt : IN std_logic;
reset : IN std_logic;
clk : IN std_logic;
interrupt_ack : OUT std_logic;
bram_enable : OUT std_logic;
k_write_strobe : OUT std_logic;
sleep : IN std_logic);
 END COMPONENT;

COMPONENT kcpsm6_UART
PORT (
uart_rx_data_out : OUT std_logic_vector(7 DOWNTO 0);
Read_B : IN std_logic;
En_16x : IN std_logic;
clok : IN std_logic;
Rx_In : IN std_logic;
Write_B : IN std_logic;
uart_tx_data_in : IN std_logic_vector(7 DOWNTO 0);
```

```
Status_Port : OUT std_logic_vector(7 DOWNTO 0);
Tx_Out : OUT std_logic;
adr_Selkt : IN std_logic_vector(7 DOWNTO 0);
k_write_strobe : IN std_logic);
END COMPONENT;

COMPONENT PROM6
PORT (
address : IN std_logic_vector(11 DOWNTO 0);
instruction : OUT std_logic_vector(17 DOWNTO 0);
clk : IN std_logic;
enable : IN std_logic;
rdl : OUT std_logic);
 END COMPONENT;

COMPONENT Sleep_cntrl
PORT (
Inp_A : IN std_logic;
o_ok : OUT std_logic;
Inp_B : IN std_logic);
 END COMPONENT;

COMPONENT Baud_rate_Timer
PORT (
Clk : IN std_logic;
En_16_x_baud : OUT std_logic);
 END COMPONENT;

COMPONENT Noise_Gen
PORT (
Data_A : OUT std_logic_vector(7 DOWNTO 0);
clk : IN std_logic;
Port_adr : IN std_logic_vector(7 DOWNTO 0);
wr_en : IN std_logic;
Mode_dt : IN std_logic_vector(7 DOWNTO 0);
Data_B : OUT std_logic_vector(7 DOWNTO 0);
Data_C : OUT std_logic_vector(7 DOWNTO 0);
Status_Data : OUT std_logic_vector(7 DOWNTO 0);
Data_D : OUT std_logic_vector(7 DOWNTO 0);
GZ : IN std_logic;
X : OUT std_logic;
Parameter : OUT std_logic);
 END COMPONENT;

COMPONENT Pipe_line_Mux7
PORT (
In_Data_G : IN std_logic_vector(7 DOWNTO 0);
Clk : IN std_logic;
Port_Adrr : IN std_logic_vector(7 DOWNTO 0);
In_Data_F : IN std_logic_vector(7 DOWNTO 0);
In_Data_E : IN std_logic_vector(7 DOWNTO 0);
In_Data_D : IN std_logic_vector(7 DOWNTO 0);
In_Data_C : IN std_logic_vector(7 DOWNTO 0);
In_Data_B : IN std_logic_vector(7 DOWNTO 0);
In_Data_A : IN std_logic_vector(7 DOWNTO 0);
Pipe_Mux_Out : OUT std_logic_vector(7 DOWNTO 0));
 END COMPONENT;

-- SIGNALS

SIGNAL N01659 : std_logic;
SIGNAL WR_S : std_logic;
SIGNAL RESTT : std_logic;
SIGNAL SLPP : std_logic;
SIGNAL K_W : std_logic;
SIGNAL ENNBL : std_logic;
SIGNAL RD_S : std_logic;
SIGNAL INT1 : std_logic;
SIGNAL SP : std_logic_vector(7 DOWNTO 0);
SIGNAL DAT_A : std_logic_vector(7 DOWNTO 0);
SIGNAL DAT_B : std_logic_vector(7 DOWNTO 0);
SIGNAL ISTR : std_logic_vector(17 DOWNTO 0);
SIGNAL P_ID : std_logic_vector(7 DOWNTO 0);
SIGNAL STA_US : std_logic_vector(7 DOWNTO 0);
SIGNAL DAT_C : std_logic_vector(7 DOWNTO 0);
SIGNAL DAT_D : std_logic_vector(7 DOWNTO 0);
SIGNAL ODAT : std_logic_vector(7 DOWNTO 0);
SIGNAL ADDRR : std_logic_vector(11 DOWNTO 0);
SIGNAL PMX : std_logic_vector(7 DOWNTO 0);
SIGNAL DO : std_logic_vector(7 DOWNTO 0);

-- GATE INSTANCES

BEGIN
U14 : kcpsm6            PORT MAP(
write_strobe => WR_S,
out_port => ODAT,
address => ADDRR,
instruction => ISTR,
port_id => P_ID,
in_port => PMX,
read_strobe => RD_S,
interrupt => INT1,
reset => RESTT,
clk => CLK,
interrupt_ack => INT1,
bram_enable => ENNBL,
k_write_strobe => K_W,
sleep => SLPP);

U17 : kcpsm6_UART
PORT MAP(
uart_rx_data_out => DO,
Read_B => RD_S,
En_16x => N01659,
clok => CLK,
Rx_In => UART_RX,
Write_B => WR_S,
uart_tx_data_in => ODAT,
Status_Port => SP,
Tx_Out => UART_TX,
adr_Selkt => P_ID,
k_write_strobe => K_W);
U18 : PROM6
PORT MAP(
address => ADDRR,
instruction => ISTR,
clk => CLK,
enable => ENNBL,
rdl => RESTT);

U20 : Sleep_cntrl       PORT MAP(
Inp_A => K_W,
o_ok => SLPP,
Inp_B => WR_S);

U8 : Baud_rate_Timer
PORT MAP(
Clk => CLK,
En_16_x_baud => N01659);

U26 : Noise_Gen         PORT MAP(
Data_A => DAT_A,
clk => CLK,
Port_adr => P_ID,
wr_en => WR_S,
Mode_dt => ODAT,
Data_B => DAT_B,
Data_C => DAT_C,
Status_Data => STA_US,
Data_D => DAT_D,
GZ => GZ,
X => X,
Parameter => PARAM);
U28 : Pipe_line_Mux7
PORT MAP(
In_Data_G => DAT_D,
Clk => CLK,
Port_Adrr => P_ID,
In_Data_F => DAT_C,
In_Data_E => DAT_B,
In_Data_D => DAT_A,
In_Data_C => STA_US,
In_Data_B => DO,
In_Data_A => SP,
Pipe_Mux_Out => PMX);
```

Листинг 3

```
s=serial('COM5','baudrate',115200);
fopen(s);
set(s,'terminator','CR')
```

```
red=[
'c 01 00 00'
'c 02 00 00'
'c 03 00 00'
'c 04 00 00'
'c 05 00 00'
'c 06 00 00'
'c 07 00 00'
'c 08 00 00'
'c 09 00 00'
'c 0A 00 00'
'c 0B 00 00'
'c 0C 00 00'
'c 0D 00 00'
'c 0E 00 00'
'c 0F 00 00'
'c 10 00 00'
'c 11 00 00'
'c 12 00 00'
'c 13 00 00'
'c 14 00 00'
'c 15 00 00'
'c 16 00 00'
'c 17 00 00'
'c 18 00 00'
'c 19 00 00'
'c 1A 00 00'
'c 1B 00 00'
'c 1C 00 00'
'c 1D 00 00'
'c 1E 00 00'
'c 1F 00 00'
'c 20 00 00'
'c 21 00 00'
'c 22 00 00'
'c 23 00 00'
'c 24 00 00'
'c 25 00 00'
'c 26 00 00'
'c 27 00 00'
'c 28 00 00'
'c 29 00 00'
'c 2A 00 00'
'c 2B 00 00'
'c 2C 00 00'
'c 2D 00 00'
'c 2E 00 00'
'c 2F 00 00'
'c 30 00 00'
'c 31 00 00'
'c 32 00 00'
'c 33 00 00'];



green=[
'c 00 00 00'
'c 00 01 00'
'c 00 02 00'
'c 00 03 00'
'c 00 04 00'
'c 00 05 00'
'c 00 06 00'
'c 00 07 00'
'c 00 08 00'
'c 00 09 00'
'c 00 0A 00'
'c 00 0B 00'
'c 00 0C 00'
```

```
'c 00 0D 00'
'c 00 0E 00'
'c 00 0F 00'
'c 00 10 00'
'c 00 12 00'
'c 00 13 00'
'c 00 14 00'
'c 00 15 00'
'c 00 16 00'
'c 00 17 00'
'c 00 18 00'
'c 00 19 00'
'c 00 1A 00'
'c 00 1B 00'
'c 00 1C 00'
'c 00 1D 00'
'c 00 1E 00'
'c 00 1F 00'
'c 00 20 00'
'c 00 21 00'
'c 00 22 00'
'c 00 23 00'
'c 00 24 00'
'c 00 25 00'
'c 00 26 00'
'c 00 27 00'
'c 00 28 00'
'c 00 29 00'
'c 00 2A 00'
'c 00 2B 00'
'c 00 2C 00'
'c 00 2E 00'
'c 00 2F 00'
'c 00 30 00'
'c 00 31 00'
'c 00 32 00'
'c 00 33 00'];



blue=[
'c 00 00 00'
'c 00 00 01'
'c 00 00 02'
'c 00 00 03'
'c 00 00 04'
'c 00 00 05'
'c 00 00 06'
'c 00 00 07'
'c 00 00 08'
'c 00 00 09'
'c 00 00 0A'
'c 00 00 0B'
'c 00 00 0C'
'c 00 00 0D'
'c 00 00 0E'
'c 00 00 0F'
'c 00 00 10'
'c 00 00 12'
'c 00 00 13'
'c 00 00 14'
'c 00 00 15'
'c 00 00 16'
'c 00 00 17'
'c 00 00 18'
'c 00 00 19'
'c 00 00 1A'
'c 00 00 1B'
'c 00 00 1C'
```

```
'c 00 00 1D'
'c 00 00 1E'
'c 00 00 1F'
'c 00 00 20'
'c 00 00 21'
'c 00 00 22'
'c 00 00 23'
'c 00 00 24'
'c 00 00 25'
'c 00 00 26'
'c 00 00 27'
'c 00 00 28'
'c 00 00 29'
'c 00 00 2A'
'c 00 00 2B'
'c 00 00 2C'
'c 00 00 2E'
'c 00 00 2F'
'c 00 00 30'
'c 00 00 31'
'c 00 00 32'
'c 00 00 33'];
replay=true;
while replay==true
color = input('Enter Color (red, green or blue): ','s');
time = input('Enter time: ','s');
time = str2double(time);
if (isnan(time) || isempty(time) || time < 1)
   disp('bad info')
end
time = time*60;

switch color
   case 'blue'
      tic;
      while toc<time
            for numcolor=length(blue):-1:1
       fprintf(s,'%s\n',blue(numcolor,:));
       end
      for numcolor=1:length(blue)
      fprintf(s,'%s\n',blue(numcolor,:));
      end
      end
      fprintf(s,'%s\n','c ff ff ff');
      replay = input('Do you want more? Y/N [y]: ', 's');
      if  strcmp(replay,'y')
        replay = true;
      else
        replay=false;
      end

   case 'green'
       tic;
       while toc < time
       for numcolor=1:length(green)
      fprintf(s,'%s\n',green(numcolor,:));
       end
       for numcolor=length(green):-1:1
       fprintf(s,'%s\n',green(numcolor,:));

       end
       end
       fprintf(s,'%s\n','c ff ff ff');
       replay = input('Do you want more? Y/N [y]: ', 's');
      if  strcmp(replay,'y')
        replay = true;
      else
        replay=false;
      end
```

```
    case 'red'
        tic;
        while toc < time
        for numcolor=1:length(red)
        fprintf(s,'%s\n',red(numcolor,:));
        end
        for numcolor=length(red):-1:1
        fprintf(s,'%s\n',red(numcolor,:));
        end
        end
        fprintf(s,'%s\n','c ff ff ff');
       replay = input('Do you want more? y/n [y]: ', 's');
       if  strcmp(replay,'y')
         replay = true;
       else
         replay=false;
       end
end
end
fprintf(s,'%s\n','c 00 00 00');
fclose(s);
```