Преглед на подходите и методите за проектиране на компютърни архитектури с ниска енергийна консумация

Станислав Георгиев, Димитър Тянев, Веселин Йосифов

Резюме: Представено е извършеното проучване и направения анализ на методите, с помощта на които е възможно да се постигне намаляване на енергийната консумация в ново проектирани компютърни архитектури. Изявени и класифицирани са основните приниципи, върху които се основават тези методи. Разгледана е тяхната същност и вариантите им, прилагани на различните архитектурни нива и степен на детайлизация на схемите. Посочени са стойностите на оценките, с които може да бъде намалена енергийната консумация, както за отделните методи, така и върху отделните архитектурни нива. Изказани са изводите от тяхното сравнение и са посочени софтуерните инструменти, с които може да се преследва енергийната оптимизация при схемно проектиране върху програмируеми интегрални схеми от вида CPLD и FPGA.

Review of low power architectures design methodologies

Stanislav Georgiev, Dimitar Tyanev, Veselin Josiffov

Abstract: This document presents research and analysis of methods through which it is possible to decrease (to reduce) the power consumption of newly designed computer architectures. The basic principles on which these methods are based are presented and classified. The methods are reviewed in detail, as well as their variations applicable at different architecture abstraction levels. The power consumption estimations are presented for the particular methods as well as the different architectural levels. Conclusions are made from the comparison of the different methods and automated tools through which it is possible to achieve energy consumption optimization when designing programmable chips such as FPGA or CPLD are listed.

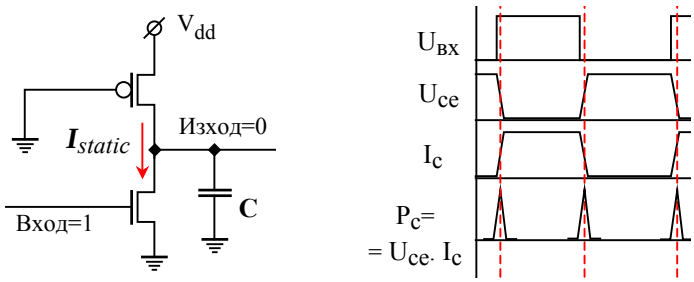
1. Увод

С напредъка в областта на минитюаризацията, числото на интегрираните в една интегрална схема транзистори непрестанно се увеличава. Това от своя страна води до намаляване размерите на електронните устройства, което от своя страна води до бума на преносимите електронни устройства – от телефони до компютри, през последното десетилетие. Едно от основните изисквания към преносимите устройства е ниското им тегло, което от своя страна поставя на преден план един въпрос, който преди това е бил в една или друга степен пренебрегван – това е въпросът с консумираната от устройствата енергия. Макар и да имат възможност за свързване към електро захранващата мрежа, основният енергиен източник на мобилните устройства са батериите. Изискването за ниска маса обаче означава, че не е разумно те да бъдат комплектовани с голяма батерия. Въпреки развитието на батерийната технология, сама по себе си тя не може да реши проблемът с консумацията. За това актуални за разработка стават така наречените компютърни архитектури с ниска енергийна консумация (*Low Power Architectures*).

2. Изложение

Преди да бъдат изложени подобренията, отнасящи се към понятието архитектура с ниска енергийна консумация, е редно първо да се обърне внимание на самия механизъм на енергийна консумация в CMOS интегралните схеми. Като цяло, енергийната консумация може да се раздели на две части – статична и динамична. Статичната консумация е с по-малко влияние върху общата консумация и се определя от явления като ток на утечката. Динамичната консумация от своя страна се определя от "превключването" на транзисторите в интегралната схема.

Динамичната консумация има няколко компонента, но с най-голямо значение са преходните процеси при зареждане и разреждане на паразитните капацитети. На фигура 1 е представен CMOS инвертор, в схемата на който паразитният капацитет е изобразен чрез кондензаторът *C*. Представени са процесите в пълен период на входния сигнал от високо ниво до ниско ниво (нула) и отново до високо ниво.



Фиг. 1 CMOS инвентор

При превключване на входа от високо към ниско ниво, NMOS транзисторът се запушва, а PMOS транзисторът се отпушва. Кондензаторът C се зарежда до ниво U_{ce}, като при този процес се черпи енергия равна на $C.U_{ce}^{\ 2}$ [1, 2]. Половината от тази енергия се разсейва чрез PMOS транзистора а другата половина се натрупва в кондензатора. При промяна на входния сигнал от ниско към високо ниво, кондензаторът се разрежда през отпушеният NMOS транзистор. Така динамичната съставна на консумираната енергия може да се изрази със следната формула:

$$P_{dyn} = \left(\frac{1}{2} C.U_{ce}^{\ 2}\right).\alpha.f \quad , \tag{1}$$

където с f е означена тактовата честота а с α е отбелязана активността на превключване на транзистор от крайния възел. Причината за наличието на тези две величини е следната. С f е отбелязана честотата, с която постъпват входните за схемата данни. В синхронни схеми се приема, че тази честота е равна на тактовата честота. С α е отбелязана така наречената активност на данните [1]. Тя изразява очаквания брой превключвания на CMOS елементите, предизвикани от ново постъпилите данни в рамките на един период.

При разглеждане на възможните подходи за оптимизация на енергийната консумация е въведена представената по-долу класификация, в която всеки метод е “прикачен” към дадено ниво на абстракция – например гейтово ниво, схемно ниво, архитектурно ниво. Но независимо за кое ниво на подобрения се говори, в основата си те следват три основни идеи, които са:

1. “Търгуване” на производителност срещу площ. Един от основните подходи за намаляване на енергийната консумация е мащабирането на захранващото или праговото напрежение в схемите. Като цяло обаче това от своя страна води до намаляване на производителността. Друг подход е въвеждането на суперскаларност в структурите, като по този начин дизайнът може да има същата прозводителност, но при по-ниско захранващо напрежение. В резултат на това обаче се увеличава заеманата от схемата площ в чипа.

2. Избягване на излишни разходи. Типичен пример за излишен разход е подаването на тактов сигнал към структурни модули, които не са ангажирани с получаването на резултат от текущата машинна команда. Основните подходи за избягване на този излишък са няколко. На първо място това е техниката *Clock Gating*, същността на която се изразява в насочване на тактовите последователности само към текущо работещите модули. На второ място е подходът към реализация на регулярни алгоритми, а на трето място е използването на специализиран (посветен) хардуер, вместо програмируем такъв.

3. Максимално прилагане на принципа на локалността. Обръщенията извън чипа по време на изчислителния процес са изключително скъпи от енергийна гледна точка. Това се дължи на големите паразитни капацитети на свързващите чипа шини с външните компоненти. Зле разпределен проект, в който данните са “разхвърляни” на много места, е също силно енергоемка ситуация. Целта на принципа на локалността е да се минимизират глобалните обръщения. Този подход е особено популярен при DSP архитектурите.

По-надолу ще бъдат представени изявените подходи в отделните абстрактни нива.

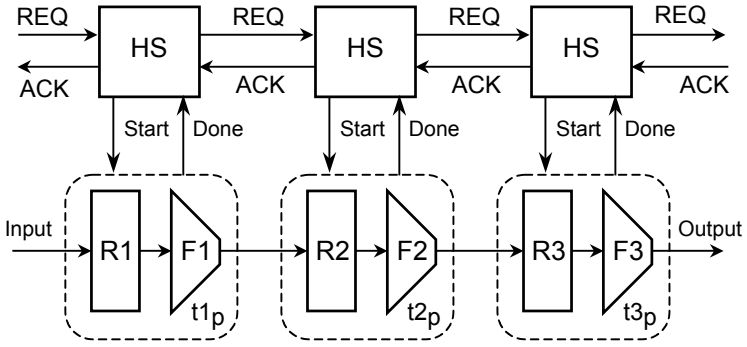
А) Технологично ниво

Най-ниското ниво в абстрактното разглеждане на архитектурата е технологичното. Тук става въпрос за технологията на изработка на чипа и на корпуса. Видът на корпуса до голяма степен влияе на консумираната енергия, тъй като генерира определена стойност на капацитета. Разликата в капацитетите вътре в чипа спрямо този в изводите е от три порядъка (от десетки фетмофаради вътре в чипа до 13-14 пикофарада в извод), както е посочено в [6]. Намалявайки този капацитет може да се постигне драстично намаляване на консумиранта енергия. Пример за това е технологията MCM (*multi-chip modules*), използвана от фирмата от *Rockwell International* в процесори, предназначени за употреба в самолети. Изследвания [7] са установили, че чрез MCM технологията капацитетите на изводите на чипа може да бъде намален до порядъка на капацититете вътре в чипа. Минимизирането на капацитета се обеснява с мащабиране на технологията. Например, при преминаване от 45 [nm] към 32 [nm] технология за съвремените процесори, на което сме свидетели, последва намаляване на енергийната консумация. Като пример за това може да се цитира изявлението на *MIPS Technologies* на международната конференция за *Solid State Circuits* през 1994 година, че с преминаване на 0,64 [μм] технология от 0,8 [μм], се е намалила енергийната консумация на техния 64 битов RISC процесор с 25%. Но само по себе си мащабирането на технологията не може да реши проблема с консумацията. Съпротивлението на връзките в чипа зависи правопропорционално от дължината им и обратно пропорционално от ширината и дебелената им. Мащабирането на технологията означава, че и трите показателя ще намалеят с един и същи коефициент, което от своя страна означава че съпротивлението на връзките ще нарастне със стойност равна на коефициента, с които намаляват физическите им размери. Влиянието на връзките в чипа върху производителнсотта и енергийната консумация е разгледано подробно в [26]. С продължителното мащабиране на технологията, влиянието на връзките в чипа върху консумираната енергия ще нараства, до момент в който по-нататъшното мащабиране няма да има влияние върху консумираната енергия или дори ще я увеличи.

Б) Схемно ниво

Следващото ниво е схемното. Тук трябва да се вземе предвид, че освен енергийна консумация, други фактори, на които трябва да се обърне внимание, са сложността и надеждността на дизайна. На това ниво вниманието е насочено към употребата на асинхронна логика, към оразмеряването на транзисторите и към стила на проектиране.

Фигура 2 илюстрира представата за асинхронна логика.



Фиг. 2 Представа за асинхронна логика

В асинхронната логика липсва тактовият сигнал. Синхронизация се постига чрез така наречният *handshake* (HS, ръкостискане), което се осигурява от допълнителен хардуер. Поради тази причина, асихнронната логика често е пренебрегвана като опция, но имплементацията й е полезна от гледна точка на енергийната консумация. Предимствата на асинхронната логика могат да се изразят с идеята за избягване на излишните разходи. Липсата на тактов сигнал означава че я няма и често сложната мрежа за неговото разпространяване. Също така използването на сигнали за начало и завършване на операцията (сигналите *start* и *done* на схемата) спомагат за избягването на гличовете. На

последно място фактът, че процесите започват при наличието на данни, означава, че асинхронната логика има вграден механизъм за "изключване" на неактивни модули. Но освен предимства, асинхронната логика си има и недостатъци. Допълнителният хардуер, който осъществява ръкостискането също така консумира енергия, и в зависимост от размера на логиката, която изгражда този хардуер, могат да бъдат загубени предимствата както и в размера на проекта така и като консумация. Логиката за ръкостискане няма пряко отношение към изчислителния процес, така че донякъде тя може да се сравни с логиката по разпространие на тактовия сигнал, въпреки че консумацията й е по-малка поради по ниската активност на сигналите за ръкостискане. Друг недостатък, който вече не с такава тежест, която е имал преди няколко години е, че по-малък брой развойни среди поддържат проектирането на асинхронна логика. На приципите на асинхронната логика се базира техниката *clock gating*. Пример за приложението на тази техника е даден в [23].

Независимо от метода за управление (синхронен или асинхронен), размерът на транзисторите е фактор при оптимизиране с цел постигане на ниско енергийна консумация. Тук проблемът се състои в намиране на баланс между производителност и цена, като под цена в случая се има предвид площта в чипа с ниска енергийна консумация. Транзистори с по-големи размери могат да провеждат по-големи токове, което спомага за по-голяма производителност. Но също така транзистори с по-големи размери превключват по-голям ток на късо съединение. Те също така внасят и по-големи капацитивностти в схемата, което от своя страна води до повишена консумация. В този случай е необходимо да се намери баланс между двете изисквания. Една добра стратегия е използването на транзистори с минимални възможни рамери, като в най-важните части на схемата, така наречените *critical paths*, се използват транзистори с по-големи размери за постигане на необходимата производителност. Изследвания [8] показват, че при реализиране на суматори по тази стратегия, се наблюдава подобрение в консумация с до 36%.

Също така влияние върху консумацията има и стилът на проектиране, т.е. дали в проекта ще бъдат използвани специално проектирани логически клетки, стандартни логически клетки или масиви от гейтове. Най-добри от гледна точка на енергийната консумация са специално прокетираните клетки, понеже при тях принципите на ниската енергийна консумация могат да бъдат приложени най-пълно и за всеки отделен елемент. Специално проектираните логически клетки съдържат само необходимите за конкретната задача логически елементи и по този начин не съдържат излишни такива. Излишни логически елементи обикновено остават, когато реализацията използва стандартни клетки. Голямата цена на такива специални устройства, както и дългото време, нужно за тяхното проектиране, правят този стил на проектиране скъп и приложим в редки случаи.

Стандартните клетки от друга страна, предлагат ниско време за проектиране и цена на проекта, но за тях може да се каже, че са анти-тезата на ниската енергийна консумация. Повечете стандартни клетки са оптимизирани за производителност, и често са с излишно големи размери, което води до увеличена енергийна консумация.

Масивите от гейтове се явяват като компромис между посочените по-горе стилове на проектиране. Макар и да нямат гъвкавостта на специализираните клетки, те предлагат възможност за оразмеряване на транзисторите и връзките между тях.

В) Гейтово ниво

Техниките, приложими на следващото, гейтовото ниво, са базирани на трите основни принципа, които са изведени по-горе. На това ниво техниките, предлагащи оптимизация от гледна точка на енергийната консумация се изразяват в манипулиран мапинг на технологията, стремеж към намаляване на нивата на активност и борба с гличовете.

Под мапинг се разбира процесът на преобразуване на двоичен файл в реални варианти за имплементацията на проекта в чип, като за всяка схема има няколко варианта за реализация. Например, три входов И-НЕ елемент може да бъде реализиран като един сложен CMOS гейт или като два каскадно свързани двувходови гейта. Всеки вариант предлага различни нива на активност и внася различни капацитети. Като цяло, сложните гейтове внасят по ниски паразитни капацитети, понеже повечето сигнали са "скрити" във вътрешните възли на гейта.

Мапинг, целящ подобряване на енергината консумация, е мапинг, при който активността на превключване е намалена до минимум, а елементи с високо ниво на активност са "скрити" в сложни CMOS гейтове. По този начин сигнали с високо ниво на активност са мапирани във вътрешни възли с ниски капацитивни товари, което намалява енергийната консумация. Ако обаче бъдат проектирани прекалено усложнени гейтове, същите ще имат прекалено голямо закъснение, което ще бъде заплатено с намалена производителност на схемата като цяло, т.е. за подобрената енергийна консумация е платено със загубената производителност. Разбработени са няколко алгоритми за технологичен мапинг, които според [9] и [10] намаляват консумацията с от 12% до 21%.

Техниката за намаляване на нивото на активност се изразява в подходящо преподреждане на входовете. В този случай е желателно сигнали с висока активност да бъдат изместени назад в реда на постъпване на входовете, като по този начин по-малък брой гейтове ще бъдат засегнати (манипулирани) от тях.

Техниката за намаляване на глитчовете е свързана с намаляване на нежеланите (непроизводителните или още междинните) превключвания. Примерните схеми от фигура 3 илюстрират тази техника. Представени са два варианта на реализиране на една и съща логическа функция, като едната реализация представлява балансирана дървовидна структура, а втората – каскадна.

А) *балансирана дървовидна структура*

Б) *каскадна структура*

Фиг. 3 Намаляване на гличовете

Ако се приеме, че входните променливи постъпват и за двете схеми по едно и също време и че закъсненията в елементите са еднакви, ще се установи, че при каскадната структура има по-голям брой последователни превключвания. На практика постъпване на входен сигнал на вход на който и да е от елементите, предизвика превключване на изходите на всички елементи след него в каскадата. Докато в дървовидната структура всеки елемент се превключва само по веднъж при промяна на входовете му, поради балансираната му структура. Друг вариант за намаляване на глитчовете е като се намали сложността (дълбочината) на логиката, понеже броят на глитчовете е квадратично зависим от сложността на логиката [1]. Изследвания [11], показват, че чрез премахване на гличовете са възможни подобрения в консумацията на схемата с до 20%.

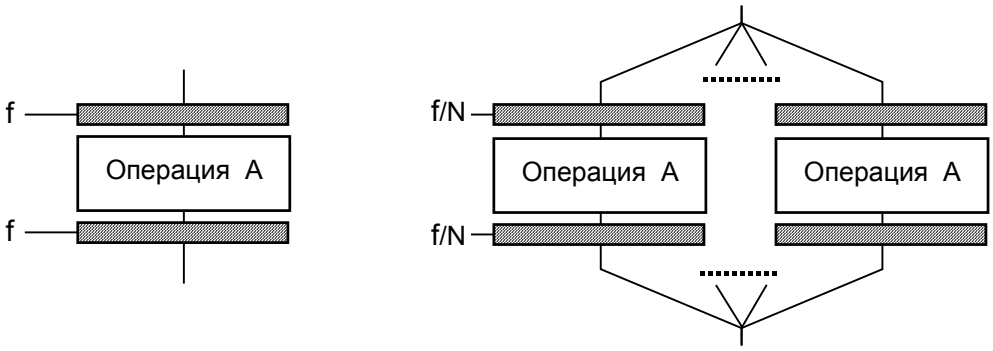
Г) Архитектурно и системно ниво

Разгледаните дотук нива се приемат като ниски. Архитектурно, системно и алгоритмово нива се приемат за високи. На тях ефектът от оптимизациите е значително по-висок. Докато оптимизациите на ниско ниво водят до двукратно подобрение, то при оптимизации на високо ниво подобрението може да е с няколко порядъка. Следващото ниво, което ще разгледаме, е архитектурно-системното ниво, като под архитектурно се имат предвид оптимизациите на RTL (*Register Transfer Level*) ниво на абстракция. Както и на ниските нива, техниките тук също се базират на трите основни принципа, дефинирани в началото на този документ.

На архитектурно ниво възможните техники са следните – паралелно изпълнение на изчислителните процеси, програмируемост, партишънинг, избор на формат на представяне на данните и адаптивни стратегии за захранване.

Паралелното изпълнение на изчислителните процеси (*concurent processing*), е може би най-важната съвременна стратегия за намаляване на консумацията. Това е директна размяна на производителност за ниска консумация. Прилагат се добре известни техники като суперскаларност и конвейерна организация на изчислителния процес и придобитата по този начин производителност се "заменя" за ниска консумация чрез намаляне на захранващото напрежение.

Фигура 4 представя два изчислителни процеса: рисунка 4а илюстрира операционна структура без приложена суперскаларност, а рисунка 4б – с приложена суперскаларност за операция A.



А) Без паралелизъм
Капацитивен товар – C. Честота – f .
Напрежение – U . Консумирана енергия $P \sim C.U^2.f$

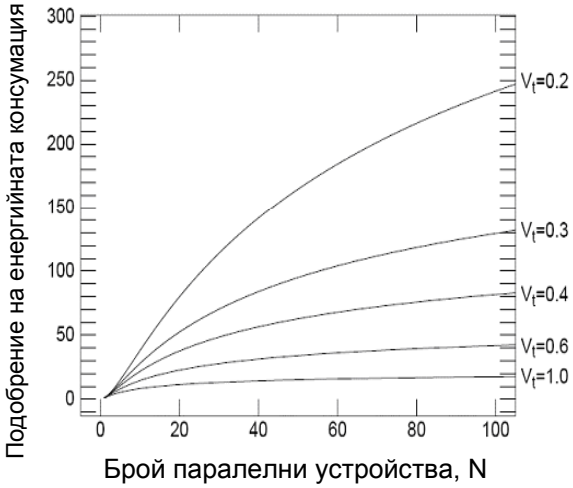
Б) С паралелизъм
Капацитивен товар – N.C. Честота – f/N .
Напрежение – U/N . Консумирана енергия P/N^2 .

Фиг. 4 Суперскларност

В първата рисунка има един регистър, който е тактуван с честота f. В схемата 4б хардуерът от схема 4а е дуплициран N пъти. Поради това са налични N на брой процесори, като всеки от тях може да бъде тактуван с честота равна на f/N и производителноста на схемата със суперскаларност ще бъде същата както тази на схемата без паралелизъм, при положение, че няма зависимости, които да пречат на паралелното изпълнение на операция A. Привлекателността на суперскаларната архитектура, от гледна точка на енергийната консумация, се състои в това, че всеки процесор е захранван с напрежение приблизително равно на V/N. Следва да се отчете, че поради наличието на N на брой процесори, капацитивният товар на схемата с паралелизъм е равен на произведението C.N. Ако се приложи зависимостта (1), то консумираната енергия ще се изрази както следва:

$$P_{par} = \frac{P}{N^2} \quad , \tag{2}$$

където с P е означена енергийната консумация на схемата без супер скаларност. Формулата е валидна за идеалния случай. В този случай не се взема предвид, че алгоритмите рядко позволяват голяма степен на паралелизъм. Също така не се взема предвид и влиянието на праговото напрежение, но за разлика от влиянието на алгоритъма, то е положително. Влиянието на праговото напрежение е илюстрирано на фигура 5.



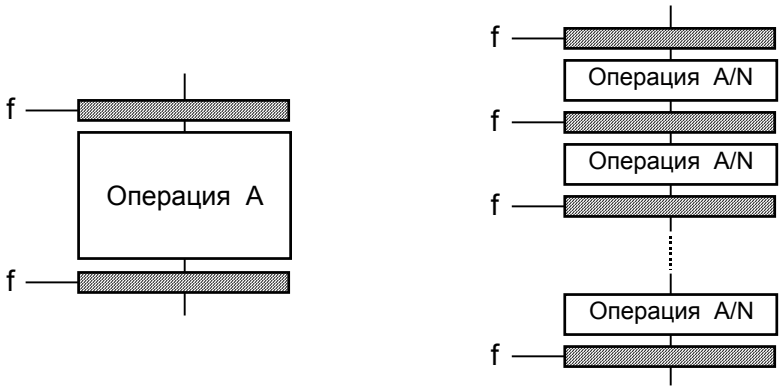
Фиг. 5 Влияние на праговото напрежение

Недостатъците на суперскаларността са следните. На първо място N-кратното дуплициране на процесорната структура води до огромно нарастане на заеманата върху кристала площ. Второ, нараства допълнителният хардуер и връзките за реализиране на разпределнието на сигналите по процесите, както и тези за събирането на резултатите. Това

допринася за увеличаване на консумацията и е фактор за ограничаване максималния брой на процесорите.

Друг начин за реализация на паралелизъм е съвместеното изпълнение чрез реализиране на конвейерна организация. На фигура 6 е показано сравнение на схема със и без конвейерна организация.

Тук за разлика от суперскаларната архитектура няма дуплициране на хардуера, а наместо това операцията се разбива на отделни части като за всяка част се посвещава част от хардуера. Работната честота остава непроменена, както и капацитивният товар. Предимството в този случай идва от намалената работа по необходимите изчисления във всяка отделна степен на конвейера. Вместо цялата операция , за един цикъл се извършва 1/N-та част от нея. Това от своя страна позволява намаляне на захранващото напрежение до V/N. Прилагайки (1) излиза че консумираната енергия в този случай е същата както при прилагане на суперскаларност. Както при суперскаларността, конвейерната органи-зация се характеризира с увеличена площ, понеже за всяка степен на конвейера се заделя регистър. Това от своя страна добавя капацитивен това към мрежата за разпространение на тактовия сигнал. С увеличаване на дълбочината на конвейера този капацитивен товар нараства до степен, в която по нататъшно разширение на конвейера става безмислено. Но като цяло "преразходите" на конвейерната организация са по-малки от тези на супер скаларността, което го прави предпочитана алтернатива за оптимизации с цел намаляване на енергийната консумация.



А) Без конвейерна организация
Капацитивен товар – C.
Честота – f .
Напрежение – U .
Консумирана енергия $P \sim C.U^2.f$

Б) С конвейерна организация
Капацитивен товар – C.
Честота – f .
Напрежение – U/N .
Консумирана енергия $P \sim P/N^2$.

Фиг. 6 Конвейерна организация

Конвейерната организация има същите недостатъци както и суперскаларната архитектура – не всички алгоритми могат да бъдат оптимизирани за конвейерна организация на изчислителния процес. Освен това си има и уникални за себе си недостатъци с голямо значение за дълбоки конвейери. Такъв недостатък представлява извличането и декодирането на машинна команда №(i), което става преди да бъдат завършени изчисленията заповядани от команда №(i-1). Съществува вероятност изпълнението на i-тата команда да зависи от резултата на предидущата (i-1)-ва команда. Конвейерните зависимости и методите за тяхното разрешаване са подробно изяснени в [14]. Пример за приложението на този метод е даден в [23].

Целта на адаптивните стратегии за захранването е елиминирането на излишните разходи. Съществуват два основни варианта: селективно деактивиране и режим на сън. Селективно деактивиране е известно още под името *clock gating*. Селективното деактивиране се състои в спиране на тактовия импулс към модули, които нямат отношение към текущото изчисление. Стратегията изисква допълнителна логика, която да следи активността на отделните модули в система. За целта обаче тя трябва да се прилага с малки измениния при динамична логика, където тактовият сигнал се използва за опресняване на информацията. В този случай е препоръчително вместо да се спира тактуването към този модул, същото да се забавя. Така

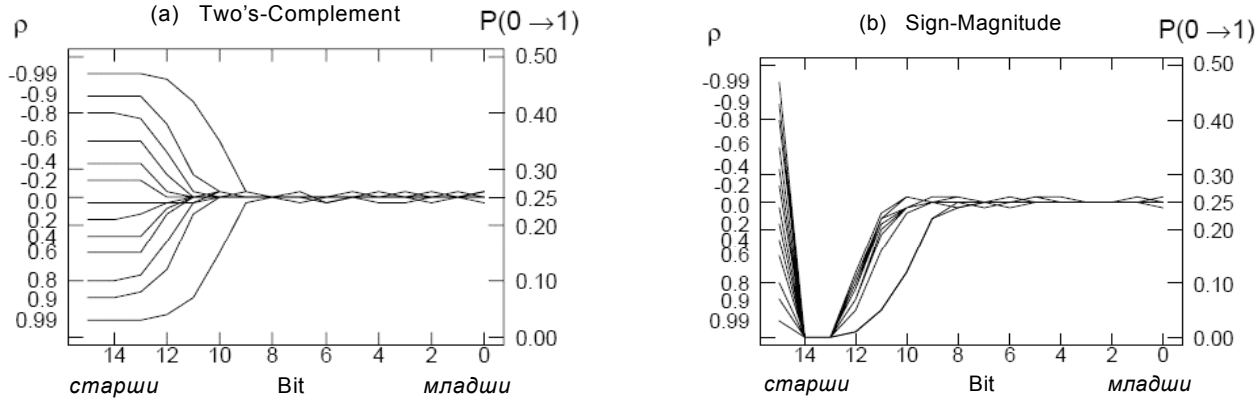
информацията в динамичните възли ще бъде опреснявана периодично, но с по-ниска честота. Пример за селективно деактивиране е разгледан подробно в [28].

Режимът на сън представлява разширение на стратегията за селективно деактивиране. Разликата е в това, че вместо наблюдение на всеки отделен модул, активността се наблюдава в цялата система. Ако системата стои "без работа" за определен срок от време, то тя или се изключва или навлиза в режим на сън. В режим на сън се наблюдават входовете на системата за необходимо активизиране, което ще "събуди" системата и ще я въведе в работен режим. Трябва да се отбележи, че превключването на системата от работен в спящ режим е свързано с преразход на енергия, така че е препоръчително да се установи точно продължителността на периода, в които системата няма да работи, за да премине в спящ режим. Тези техники са особено популярни при преносимите компютри, където намират широко приложение, и според данни на IBM и *Intel* спомагат за намаляване на консумацията с до 30%.

Така наречения партишънинг представлява разпределение на ресурсите на изчислителната система. Това се прави с цел увеличаване на локалността и намаляване на глобалните обръщения. Същността на идеята се състои в това, че вместо един процесор, изпълняващ целия алгоритъм, изчислителната система може да се рапредели така, че всяка нейна част да се специализира за изпълнението на определена част от алгоритъма. Паметта също подлежи на подобно разпределение. От енергийна гледната точка е много по-ефективно към всеки процесор да има малка локална памет отколкото голяма по размер обща памет. Това разпределение се отнася и за управляващите автомати [5]. Вместо един краен автомат, който да генерира всички управляващи сигнали и да ги разпространява, при разпределен краен автомат само малка част от сигналите трябва да се генерират и разпределят глобално, в смисъл извън границите на чипа, което се отразява силно положително на енергийната консумация. Като част от темата за разпределен и централизиран изчислител процес може да се погледне и върху въпроса за програмируем и специлизиран (наричан още посветен) хардуер. Специализираният хардуер е по-изгоден от гледна точка на енергийната консумация, понеже се избягват изшлините разходи на програмируемия хардуер, който трябва да е способен да изпълнява много задачи, а не само една. От своя страна специализираният хардуер заема повече площ върху чипа, а също така той не притежава важното удобство на програмируемия хардуер – лесната промяна на дизайна като цяло. При програмируемият хардуер това изисква само промяна в микрокомандите, издавани от софтуера или фирмуера, докато при специализираният хардуер се налага на практика разработка на нова схема.

Последната стратегия на това ниво е изборът на формат за представяне на данните. Тук дизайнерът има голям избор – фискирана запетая или плаваща, кодирано представяне или не, *sign-magnitude* или *two's-complement* и така нататък. Изборът на формат за представяне на данните е свързан с компромис между точност на представяне, сложност на дизайна, производителност и енерегийна консумация. Ако започнем анализа с първия избор – фикисирана или плаваща запетая, то ще видим че фиксираната запетая има най-малки изисквания към хардуера и съответно с най-ниска консумация, но страда от ограничения на обхвата. При плаваща запетая тези ограничения не важат, но това е на цената на допълнителен хардуер, което се транслира в допълнителна площ и консумация. Също така трябва да се вземе предвид и дължината на разрядната мрежа. Често дизайнерите се презастраховат като използват по-голяма дължина от необходимата, като за тази сигурност се заплаща с производителност и консумация. За това е препоръчително този метод да се избягва и да се прави точен анализ на необходимата, а не на желаната точност на представяне за проекти с ниска консумация. Освен точността на представяне и дължината на думата, дизайнерът трябва да има в предвид и артиметичното представяне на данните. Например, машинните кодове прав, обратен (*sign-magnitude*) и допълнителен (*two's-complement*), както и техните модифицирани варианти. Допълнителният код е най-подходящ за аритметични изчисления, затова и се ползва най-често. В допълнителен код старшите незначещи цифри в изображението на числата съвпадат със знаковата цифра [14]. В

резултат, в старшата част на разрядната мрежа тези битове могат да се приемат за излишна информация. Освен това те са източник на възможни и излишни превключвания, а от там и на излишна консумация при изчисления със смяна на знака. Правият код (*Sign-magnitude*) използва един бит за означаване на знака, така че от енергийна гледна точка тази форма на презентация на данните се счита за по-ефективна [13]. За съжаление, за аритметични изчисления в този формат се налагат допълнителни преобразувания, които елиминират ефекта на спестената консумация от по-малкия брой превключвания. Вероятността за активни превключвания по причина на възникнали преноси по дължината на разрядната мрежа е представена на фигура 7 [13].



Фиг. 7 Активност на превключване при *sign-magnitude* и *two's-complement*

Д) Алгоритмово ниво

Последното и най-високо ниво е алгоритмовото ниво. Алгоритмите имат както директно така и индиректно влияние върху енергийната консумация. Например сложността и броят операции в един алгоритъм имат директно влияние, докато степента на паралелизъм, която определя колко ефективно алгоритъмът може да се приложи върху архитектура, оптимизирана за ниска енергийна консумация, има индиректно влияние. На това ниво се разглеждат качествата на алгоритмите и какви промени в тях могат да се направят за да бъдат приложими за архитектури с ниска енергийна консумация. Това разглеждане започва с качествата, които имат директно влияние върху енергийната консумация, като например сложността на алгоритъма. Сложността може да бъде измерена по няколко начина, като един прост начин за това е да приемем като оценка броя на командите. Това е полезна метрика понеже за изпълнението на всяка команда се консумира определено количество енергия. Съответно по-малък брой команди означава по-ниска консумация. Алгоритмите с голяма производителност често са с голям брой команди. Не само броят команди има влияние върху консумацията. Някой команди имат по-голяма консумация от други. Например, изпълнението на командите за умножение имат по-голяма консумация от изпълнението на тези за събиране. Също така изпълнението на команда за обръщение към паметта има голяма цена като енергийна консумация. Поради тази причина, броят на тези команди в алгоритъма, предназначен за работа върху ниско енергийна архитектура, трябва да е минимизиран. Докато сложността на алгоритъма влияе на консумацията в комуникационната среда, то регулярността влияе на управляващия автомат. Силно регулярен алгоритъм се нуждае от по-малко състояния на крайния автомат за да бъде той описан. Това означава по-просто устроен краен автомат. За програмируем процесор, регулярният алгоритъм означава по-малко команди за преход, а в случай на цикли – по-голям шанс за правилно предсказване на изхода от цикъла. Това води до намалена консумация. Влиянието на дължината на разрядната мрежа бе разгледано на архитектурното ниво на абстракция. При определянето на дължината й трябва да се вземе предвид точността на алгоритъма. Дори алгоритми, които изпълняват една и съща функция, могат да има различна точност, което води до различна дължина на разрядната мрежа на процесора.

Други характеристики на алгоритъма имат индиректно влияние върху консумираната енергия. Такиви характеристика са степента на паралелизъм и модуларността. Параленото

изпълнение на изчислителните процеси позволява да се намали консумацията, като за това се заплати с допълнителна производителност, евентуално спечелена от въвеждането на паралеността. Тази стратегия обаче е безполезна, ако алогиртъмът, използван за тази архитектура, не предлага достатъчно паралелизъм. Проблем тук се явяват последоватилните и рекурсивните команди, т.е. командите, чието изпълние зависи от резултата на предидущи команди. За да се преодолее този проблем се прилагат техники като разгъване. Същата техника може да се приложи и при циклите, предоставяйки допълнителна степен на паралелизъм [14]. Други техники, които могат да се приложат, са софтуерен конвейер, алгебрични трансформации и други. Модуларността на алгоритъма също е особено важна, ако архитектурата върху която трябва да бъде приложен алгоритъма използва разпределени изчислителни ресурси, памети и други. Особено полезни в този случай за графовете.

3. Енергиен анализ

От оптимизациите разгледани до тук само тези на архитектурно и алгоритмово ниво са в сферата на компетентност на компютърния потребител. При проектирането на компютърна система с понижена енергийна консумация, задачата на проектанта е да намери точния баланс между оптимизациите с цел подобряване на енергийната консумация и на производителността. До относително неотдавна не беше възможно да се разбере точната консумация на дадена схема или чип без измервания на реален негов образец. Ситуацията започна да се променя с въвеждането на програмируемите чипове (CPLD, FPGA), чиято консумация и спецификации по охлаждането, както е отбелязано в [15], трябва да бъдат определени в ранен стадий на процеса по проектирането им. Често това става дори преди проектирането на логиката.

С цел извършване на енергиен анализ по време на процеса на проектиране, са създадени различни инструменти, някои от които са интегрирани в развойните среди или се предоставят допълнително от фирмите производители. Като пример ще спомена предлаганите инструменти за енергиен анализ в две развойни среди – *ISE* на фирмата *Xilinx* и *Galaxy Design* на фирмата *Synopsis*, без да се спираме подробно на тях. Фирмата *Xilinx* предлага за развойната среда ISE продуктите XPE (*Xilinx Power Estimator*) и XPA (*Xilinx Power Analyzer*), а фирмата *Synopsis* предлага за развойната среда *Galaxy Design* анализатора *Prime Time* PX и допълнението към него *Prime Rail*, а така също и автоматичната система за оптимизация на енергийната консумация *Power Compiler*.

4. Заключение

Това изследване представя извършения обзор и анализ на методи за оптимизация на енергийната консумация. Бе установено, че най-големи придобивки се получават от оптимизации на така наречените високи нива – RTL (архитектурно) и алгоритмово нива. Също така, бе изведен изводът, че поради голямата сложност на съвременната техника, все по-голяма става необходимостта от наличието на автоматизирани системи, подпомагащи на проектанта да извършва анализ и оптимизация на проектите. И докато подобни системи за анализ са разработени в голям брой варианти, то с малки изключения системите за оптимизация все още са насочени към оптимизации на ниски нива. С цел понижаване на времето за разработка на устройства с понижена енергийна консумация е необходимо разработването на автоматични системи за оптимизация на високо ниво, както и запознаване на проектантите с тях.

5. Литература

[1]. Paul E. Landman, "*Low Power Architecture Design Methodologies*", University of California 1994.

[2]. Ricardo E. Gonzalez, "*Low Power Architecture Design*", Stanford University, 1997.

[3]. Jeff Scott, Lea Hwang Lee, John Arends, Bill Moyer, "*Designing the Low- Power M-core Architecture*", Proc. IEEE Power Driven Microarchitecture Workshop, 1998.

[4]. Binu Mathew, Al Davis, Mike Parker, "*A Low Power Architecture for Embedded Perception*", University of Utah, 2004.

[5]. Shi-Yu Huang, Chien-Jyh Liu. "*A Low-power architecture for extended Finite State Machines using input gating*", National Tsing-Hua University Taiwan, 2007.
[6]. H. Bakoglu, "*Circuits, Interconnections, and Packaging for VLSI*", Addison-Wesley, 1990
[7]. D. Benson, Y. Bobra, B. McWilliams, "*Silicon Multichip Modules*", Hot chips symposium III, Santa Ana CA, 1990.
[8]. C. Nagendra, U. Mehta, R. Owens, and M. Irwin, "*A Comparison of the Power-Delay Characteristics of CMOS Adders*", Proceedings of the 1994 International Workshop on Low Power Design, Napa Valley, CA, 1994.
[9]. V. Tiwari, P. Ashar, and S. Mailk, "Technology Mapping for Low Power", Proceedings of the 30th Design Automation Conference, Anaheim, CA, 1993.
[10]. C.-Y. Tsui, M. Pedram, and A. Despain, "*Technology Decomposition and Mapping Targeting Low Power Dissipation*", Proceedings of the 30th Design Automation Conference, Anaheim, CA, 1993.
[11]. L. Benini, M. Favalli, and B. Ricco, "*Analysis of Hazard Contributions to Power Dissipation in CMOS IC's*", 1994 International Workshop on Low-Power Design, Napa Valley, CA, 1994.
[12]. http://www.intel.com/design/mobile/datashts/318914.htm
[13]. A. Chandrakasan, R. Allmon, A. Stratakos, and R. W. Brodersen, "*Design of Portable Systems*", Proceedings of CICC '94, San Diego, 1994.
[14]. Тянев Д. С., "*Организация на компютъра*", Том 2, ISBN 978-954-20-0413-4, Технически университет – Варна, 2008.
[15]. Brian Philoflsky, "*Seven steps to an accurate worst-case scenario power analysis using Xiling Power Estimator*", white paper, Xilinx, 2008.
[16]. Xilinx, "*Xilinx Power Estimator User Guide*", June 2007.
[17]. Derek Curd, "*Power Consumption in 65 mm FPGA*", February 2007.
[18]. Xilinx Power Analyzer
[19]. Xilinx Power Solutions
[20]. Synopsis, "*Synopsis PrimeTime PX Expanding the PrimeTime solution with power analysis*", Synopsis Inc, 2007.
[21]. Synopsis, "*PrimeRail*", Synopsis Inc, 2007.
[22]. Synopsis, "*Power Compiler, automatic power management within Galaxy Design platform*". Synopsis Inc, 2007.
[23]. Kyoung-Mook Lim, Seh-Woong Jeong, Yong-Chun Kim, Seung-Jae Jeong, Hong-Kyu Kim,Yang-Ho Kim, Bong-Young Chung, Hyung-Lae Roh, and H.S. Yang, *"CalmRisc: A low power Microcontroller with efficient coprocessor Interface"*, Computer Design, 1999. (ICCD '99) International Conference on 10-13 Oct. 1999 Page(s):299 – 302, Samsung Semiconductor and KAIST.
[24]. E.P. Ramo, J. Resano, D. Mozos and F. Catthoor, *"Memory hierarchy for high-performance and energy aware re-configurable systems"*, IET Computer Digital Technology, Vol.1 No 5, 2007.
[25]. Giuseppe Notarangelo, Marco Gibilaro, Gaetano Palumbo, Francesco Pappalardo, Agatino Pennisi, *"Low power strategy for a TFT controller"*, Proceedings of the Euromicro Symposium for Digital System Design, 2002.
[26]. Rajeev Balasubramonian, Naveen Muralimanohar, Karthik Ramani, Venkatanand Venkatachalapathy, "*Microarchitectural wire management for performance and power in partitioned architectures",* Proceedings of the 11th International Symposium on High-Performance Computer Architecture, 2005.
[27]. David Garrett, Mircea Stan, *"Low Power Architecture of the Soft-Output Viterbi Algorithm"*, University of Virginia – department of Electrical Engineering, 1998.
[28]. Radu M. Secareanu, Ilon Hartin, *"Low power architectures using localized non-volatile memory and selective power shut-down"*, ISCAS 2006.