
بسم الله الرحمن الرحیم

دانشکده مهندسی برق و رباتی‌ک

پایان‌نامه دوره کارشناسی ارشد مهندسی برق-الکترونی‌ک

پی‌اده سازی ساختار کلاسه بند ماشی‌ن بردار پشتی‌بان بر روی FPGA

نگارش:

داود محمودی

استاد راهنما:

دکتر سی‌د علی سلی‌مانی ای‌وری

استاد مشاور:

دکتر حسی‌ن خسروی

خرداد 1390

تأییدیّه هیات داوران

اعضای هیئت داوران، نسخه نهائی پایان نامه آقای: **داود محمودی**

را با عنوان: **پی‌اده سازی ساختار کلاسه بند ماشی‌ن بردار پشتی‌بان بر روی FPGA**

از نظر فرم و محتوی بررسی نموده و پذیرش آن را برای تکمیل درجه کارشناسی ارشد تأیید می‌کند.

اعضای هیئت داوران	نام و نام خانوادگی	رتبه علمی	امضاء
1- استاد راهنما			
2- استاد مشاور			
3- استاد مشاور			
4- استاد ممتحن			
5- استاد ممتحن			
6- نماینده گروه			

تقدیم

به خانواده ام

تشکر و قدردانی

سپاس بیکران ایزد یکتا را که مهربانیش، یادش و همراهي پیوسته اش، همواره انگیزه من براي حرکت به سمت هدف بوده است.

ارج مي نهم زحمات بي دریغ استاد راهنمای بزرگوارم، دکتر سید علی سلیمانی، که بدون راهنمایی های دلسوزانه ایشان این اثر به انجام نمی رسید. ایشان با داشتن پیشینه اطلاعاتی غنی از طراحی و پیاده سازی سخت افزاری، ایده ها، نظرات و پیشنهادهای موثری در طول تحقیق به من ارائه نمودند.

همچنین از استاد مشاور گرانقدر خود دکتر حسین خسروی به خاطر کمک های ارزشمند ایشان در تهیه دیتابیس و راهنمایی های مفید و موثر در زمینه اجرای نرم افزاری ماشین بردار پشتیبان کمال تشکر دارم.

از تمامی اساتید گروه الکترونیك دانشگاه صنعتی شاهرود نیز سپاسگزاري و تشکر نموده و براي همه این بزرگواران آرزوي موفقیت و سلامت دارم.

در پایان جا دارد از حمایتها و همراهی های دلسوزانه خانواده ام که کاستی های مرا در طول مدت انجام پروژه تحمل نمودند، تشکر و قدردانی نمایم.

چکیده

ماشین بردار پشتیبان یکی از الگوریتمهای مورد اعتماد و با کارآیی فوق العاده بالا در تشخیص الگو است که در سالهای اخیر به طور وسیعی در مسائل کلاسه بندی و رگرسیون خطی و غیرخطی مورد استفاده قرار گرفته است. از آنجا که یکی از روشهای حل مسائل چند کلاس، استفاده از کلاسه بندهای دوتایی است، پیاده سازی همزمان کلاسه بندهای دوتایی ماشین بردار پشتیبان بر روی FPGA به صورت موازی، می تواند کاربرد خوبی مخصوصاً در کاربردهای بلادرنگ داشته باشد.

در این پایان نامه یک ساختار سخت افزاری ساده برای پیاده سازی کلاسه بندهای دوتایی ماشین بردار پشتیبان بر روی FPGA ارائه می شود. از بخش آموزش ماشین بردار پشتیبان که به صورت نرم افزاری اجرا می گردد، پارامترهای لازم استخراج شده و برای اجرای بخش تست بر روی سخت افزار به کار گرفته می شود. در ساختار سخت افزاری طراحی شده، عملیات ضرب برداری و همچنین کل عملیات کلاسه بندی دوتایی چند کلاس به صورت موازی و به طور همزمان انجام می شود. به منظور واقعی کردن طرح، از داده های مربوط به سیستم تشخیص ارقام دستنوشته فارسی در سه کلاس مختلف، برای آموزش و تست ماشین بردار پشتیبان استفاده شده است. شبیه ساز گرافیکی System Generator برای شبیه سازی طرح سخت افزاری مورد نظر مورد استفاده قرار گرفته است. پیاده سازی کلاسه بند ماشین بردار پشتیبان خطی و غیرخطی با توابع و بلوکهای ساده، امکان افزایش ابعاد بردارهای ویژگی، امکان تعمیم به چندین کلاسه بند دوتایی همزمان، عدم پیچیدگی در طراحی سخت افزاری و ساده بودن سایر بلوک ها و عملیات استفاده شده در طرح مورد نظر، از خصوصیات بارز این تحقیق به شمار می رود. طبق نتایج شبیه سازی بر روی FPGA، ماکزیمم فرکانس کاری 202/840MHz برای کلاسه بند خطی، و نرخ بازشناسی 98/67% برای کلاسه بند غیرخطی به دست آمده است. این نتایج نشان از عملکرد فوق العاده سیستم طراحی شده دارد.

کلید واژه: FPGA، پردازش بلادرنگ، ماشین بردار پشتیبان، System Generator.

فهرست مطالب

عنوان **صفحه**

فهرست اصطلاحات اختصاری

عنوان	علامت اختصاری
Support Vector Machine	SVM
Field Programmable Gate Array	FPGA
Digital Signal Processing/Processor	DSP
COordinate Rotation DIgital Computer	CORDIC
Karush-Kuhn-Tucker	KKT
Supprot Vectors	SVs
Global Positioning System	GPS
Fast Fourier Transform	FFT
Discrete Cosine Transform	DCT
Finite Impulse Response	FIR
Infinite Impulse Response	IIR
Clock	CLK
Most Significant Bit	MSB
Hardware Description Language	HDL
Very-High-Speed Integrated Circuits Hardware Description Languages	VHDL
Electrically Erasable Programmable Read-Only Memory	EEPROM
Institute of Electrical and Electronics Engineers	IEEE
Radial Basis Function	RBF
Magnetic Response Imaging	MRI
Mel-Frequency Cepstral Coefficient	MFCC
Linear Predictive Coding	LPC
Self Organized Feature Mapping	SOFM
System Generator	Sys Gen

فهرست جدول‌ها

عنوان **صفحه**

فهرست شکل‌ها

عنوان **صفحه**

فصل 1– مقدمه

1–1– پیشگفتار

ماشین بردار پشتیبان (SVM)[1] در دهه ی نود میلادی توسط آقای وپنیک[1] معرفی شد و تا کنون به دلیل قابلیت اعتماد، عمومیت و برتری کارآیی اش نسبت به دیگر الگوریتمهای کلاسه بند مانند شبکه های عصبی چند لایه، به طور گسترده ای در انواع کاربردهای کلاسه بندی مورد استفاده قرار گرفته است[2] .

با توجه به اینکه طراحی ماشین بردار پشتیبان شامل یک پروسه بهینه سازی درجه دوم است، بنابراین محاسبات آن به خصوص در طول مرحله آموزش[2] بسیار پیچیده خواهد بود. علاوه بر این کاربردهای عملی چنین کلاسه بندی به دلیل فضا و توان محاسباتی زیادی که از مشخصه های مدل آموزشی به دست می آید، نیز محدود خواهد شد. واضح است که کاربردهای برخط[3] مفید ماشین بردار پشتیبان نیازمند پیاده سازی سخت افزاری مناسب برای فاز آموزش[4] و تست[5] می باشد، اما باتوجه به نوع کاربرد (به عنوان مثال کاربردهایی که در آن حافظه ضروری نیست) فاز آموزش ماشین بردار پشتیبان می تواند به صورت خارج خط[6] در یک کامپیوتر انجام شود و بنابراین در این گونه موارد منحصراً فاز تست کلاسه بند ماشین بردار پشتیبان بر روی سخت افزار پیاده سازی می شود. فاز تست ماشین بردار پشتیبان فقط شامل یک تابع تصمیم برای هدف کلاسه بندی است که بسیار سریع اجرا خواهد شد.

1–2– انگیزه

امروزه الگوریتمهای زیادی در مباحث پردازش صوت و تصویر، بینایی ماشین، داده کاوی[7] و تشخیص الگو[8] به صورت توسعه یافته بر روی نرم افزار اجرا می شود و هر روز نیز پیشرفت های جدیدی در توسعه آنها حاصل می گردد. بسیاری از این الگوریتمها هنگام پیاده سازی نرم افزاری

[1] Support Vector Machine
[2] Training phase
[3] Online
[4] Training phase
[5] Testing phase
[6] Offline
[7] Data mining
[8] Pattern recognition

به دلیل پیچیدگی های محاسباتی با نرخ فریم[1] پایینی اجرا می شوند، بنابراین برای پردازش سریع و بلادرنگ[2] نیاز به اجرای این الگوریتمها بر روی سخت افزار است. با این حال پیاده سازی این الگوریتمها بر روی سخت افزارهایی مانند میکروپروسسورها، پردازنده های DSP[3] و یا FPGA[4] ها همچنان چالش مهمی محسوب می شود.

پردازش سریع اطلاعات از زمان ظهور پردازش سیگنال دیجیتال[5] به عنوان یک چالش برای محققان محسوب می شود. در این زمینه محققان سعی کرده اند با ارائه سخت افزارها و الگوریتمهای مناسب و روشهای برنامه نویسی خاص گامی در جهت توسعه پردازش بلادرنگ بردارند. یکی از سخت افزارهایی که اخیراً در پردازش بلادرنگ سیگنالهای دیجیتال کاربرد عملی پیدا کرده و نتایج قابل قبول ارائه داده است، FPGA ها هستند. در این سخت افزار ها با ارائه الگوریتم های مناسب به روش پردازش موازی، سرعت پردازش سیگنال چندین برابر می شود.

از طرفی روش ماشین بردار پشتیبان یک روش مورد اعتماد و با کارآیی فوق العاده بالا در کلاسه بندی الگو است که در سالهای اخیر به طور وسیعی در مسایل کلاسه بندی و رگرسیون[6] خطی و غیرخطی نیز مورد استفاده قرار گرفته است و نتایج بسیار بهتری نسبت به روش های مشابه ارائه می دهد. از آنجا که ماشین بردار پشتیبان ذاتاً یک کلاسه بند باینری است، برای کلاسه بندی مسائلی با چند کلاس باید از چند کلاسه بند ماشین بردار پشتیبان استفاده کرد. بنابراین به نظر می رسد پیاده سازی همزمان چند کلاسه بند ماشین بردار پشتیبان به صورت موازی بر روی FPGA، کارآیی خوبی برای کاربردهای پردازش بلادرنگ داشته باشد.

این تحقیق با بیان محدودیت های موجود در طراحی، یک ساختار مناسب مبتنی بر ترکیب نرم افزار-سخت افزار برای پیاده سازی الگوریتم ماشین بردار پشتبان ارایه و شبیه سازی می کند، به طوری که صحت عملکرد کلاسه بند را در حد نرم افزار برآورده کند. برای واقعی کردن طراحی انجام شده، با بکارگیری داده های سیستم تشخیص ارقام دستنوشته فارسی، نتایج کلاسه بندی خطی و غیرخطی ماشین بردار پشتیبان بر روی ساختار سخت افزاری، با نتایج متناظر آن صرفاً بر روی نرم افزار مقایسه خواهد شد.

---

[1] Frame rate
[2] Real-time
[3] Digital Signal Processor
[4] Field Programmable Gate Array
[5] Digital Signal Processing
[6] Regression

1-3- اهداف پایان نامه

در این پایان نامه از تکنیک ترکیبی نرم افزار- سخت افزار برای اجرای فاز تست الگوریتم ماشین بردار پشتیبان استفاده شده است، بدین صورت که یک سیستم تشخیص الگوی واقعی انتخاب می گردد و داده های آن در محیط نرم افزاری MATLAB به صورت الگو های ورودی، وارد الگوریتم آموزش ماشین بردار پشتیبان می شود. پس از اجرای مرحله آموزش الگوریتم ماشین بردار پشتیبان، مقادیر پارامترهایی به دست می آید که از آنها برای اجرای فاز تست الگوریتم ماشین بردار پشتیبان استفاده می شود.

برای پیاده سازی سخت افزاری فاز تست ماشین بردار پشتیبان بر روی FPGA ، از شبیه ساز System Generator استفاده می شود. روش کار در این مرحله بدین صورت است که مقادیر پارامترهای به دست آمده از مرحله آموزش با انتخاب تعداد بیت مناسب صحیح و اعشاری، به طوری که نویز کوانتیزاسیون[1] به حداقل برسد، از محیط نرم افزاری MATLAB خوانده شده و با استفاده از توابع و الگوریتمهای مناسب در شبیه ساز System Generator، ساختار سخت افزاری فاز تست ماشین بردار پشتیبان طراحی و اجرا می شود.

1-4- ساختار پایان نامه

پس از مقدمه، در فصل دوم این پایان نامه ماشین بردار پشتیبان معرفی می شود. روشهای SVM [2] خطی و غیرخطی، به طور مفصل شرح داده خواهد شد و در بخش غیرخطی توابع کرنل معروف، معرفی خواهند شد. عملکرد ماشین بردار پشتیبان در کلاسه بندهای چند کلاسه نیز مورد بررسی قرار می گیرد.

در فصل سوم کاربرد FPGAها در پردازش سیگنال دیجیتال مرور می شود. در این فصل تکنیکهای بهینه سازی طرح های سخت افزاری بوسیله FPGA و الگوریتم مهم CORDIC [3] برای پیاده سازی توابع مثلثاتی تشریح می شود.

فصل چهارم مطالعات و تحقیقات انجام شده در زمینه پیاده سازی ماشین بردار پشتیبان بر روی FPGA را مرور می کند.

در فصل پنجم ابتدا ماشین بردار پشتیبان در محیط نرم افزاری اجرا می شود و سپس ساختار طراحی شده برای پیاده

[1] Quantization
[2] Support Vector Machine
[3] COrdinate Rotation DIgital Computer

سازی سخت افزاری فاز تست ماشین بردار پشتیبان خطی و غیرخطی با ابعاد بردارهای ویژگی متفات ارائه می شود.

فصل ششم نتایج شبیه سازی ساختار طراحی شده در فصل پنجم را برای تشخیص ارقام دستنوشته فارسی نشان می دهد و نتایج به دست آمده را با اجرای نرم افزاری ماشین بردار پشتیبان مقایسه می کند.

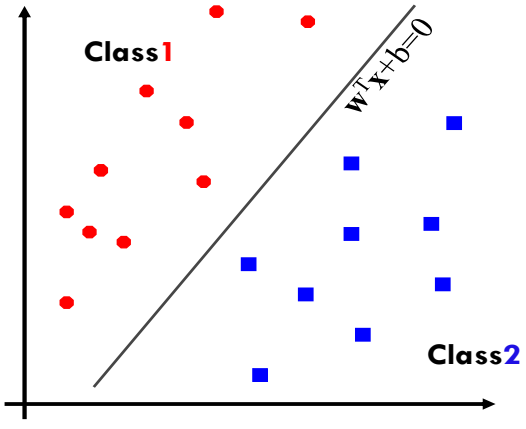
فصل هفتم به جمع بندی و نتیجه گیری از کار انجام شده، به همراه پیشنهاد برای ادامه کار و محدویت های موجود اختصاص دارد.

فصل 2– ماشین بردار پشتیبان

ماشین بردار پشتیبان یک تکنیک کلاسه بندي و رگرسیون است که اولین بار در سال 1995 توسط وپنیک [1] و گروهش در آزمایشگاه AT&T Bell پیشنهاد شد و در حال حاضر در بسیاري از زمینه های پردازش صوت و تصویر کاربرد دارد و نتایج بسیار خوبی ارائه می دهد. این کلاسه بند در واقع یک کلاسه بند دیجیتال (دوتایی) است که توانایی کلاسه بندی نمونه های چند کلاس را نیز داراست، و برای حل مسائل خطی و غیرخطی به کار می رود. ویژگی اصلی ماشین بردار پشتیبان در یافتن ابر صفحه بهینه برای کلاسه بندی داده های دو کلاس می باشد.

2–1– ماشین بردار پشتیبان خطی

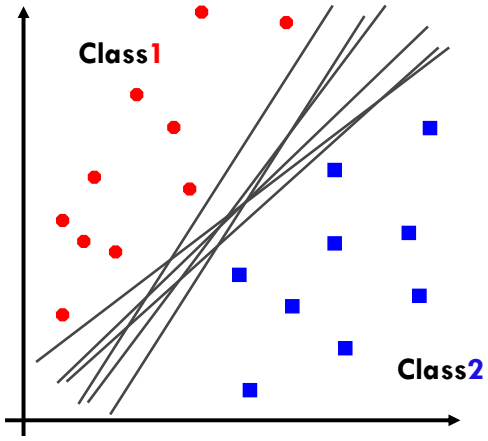
یک مسئله کلاسه بندی خطی برای جدا سازی داده های آموزشی دو کلاس در نظر بگیرید. فرض کنید $\mathbf{x}_i \ , i=1,2,\ldots,N$ مجموعه بردارهای ویژگی مربوط به داده های آموزشی باشد که به صورت خطی از هم جداپذیرند و در دو کلاس 1 و 2 به صورت شکل 2-1 طبقه بندی شده اند.



شکل 2-1: داده های آموزشی در دو کلاس جدایی پذیر خطی

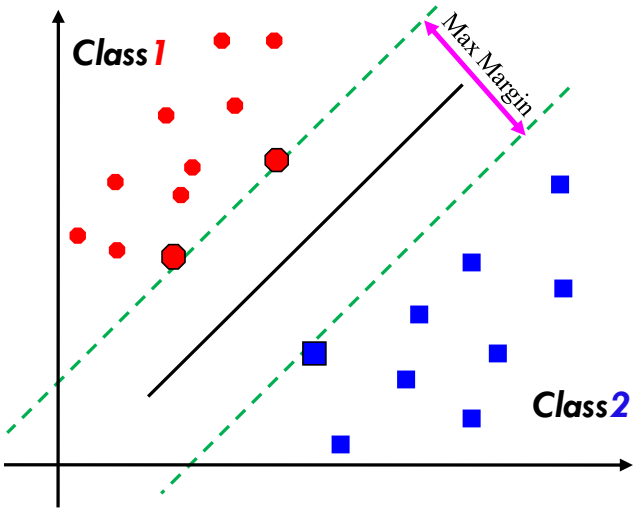
هدف بدست آوردن ابر صفحه[1] $g(\mathbf{x})=\mathbf{w}^T\mathbf{x}+b=0$ به نحوی است که تمامی داده های آموزشی به طور صحیح کلاسه بندی شود. به صورت کلی این ابر صفحه یکتا نبوده و می توان مقادیر مختلفی برای $\mathbf{w}$ و b بدست آورد. شکل 2-2 چند نمونه از ابر صفحه هایی را که می توان به منظور کلاسه بندی صحیح خطی داده های دو کلاس در نظرگرفت، نشان می دهد.

[1] Hyperplane



شکل 2-2: ابر صفحه های مجزاساز

همه ابر صفحه های نشان داده شده در شکل 2-2 عمل جداسازی را به درستی انجام می دهند، اما تنها یک ابر صفحه وجود دارد که بیشترین فاصله نسبت به داده های هر دو کلاس دارد، که به آن ابرصفحه مجزاساز بهینه[1] گفته می شود و انتظار می رود بتواند مرز به دست آمده را به تمام محدوده های ممکن تعمیم دهد[3]. ابر صفحه مجزاساز بهینه در شکل 2-3 مشاهده می شود. این ابر صفحه به دلیل اینکه بیشترین فضا تا نزدیک ترین داده از هر کلاس را در دو طرف خود ایجاد می کند، بهترین کلاسه بندی با کمترین میزان خطا را ارائه می دهد. بنابراین این نوع کلاسه بندی، در زمان اجرا بر روی نمونه های آزمایشی نتایج بهتری را از خود نشان می دهد. این نتیجه موضوعی مهم در طراحی کلاسه بند ها می باشد که آنرا قدرت تعمیم کلاسه بند می نامند.



شکل 2-3: ابر صفحه مجزاساز بهینه با حداکثر مقدار حاشیه.

[1] Optimal separating hyperplane

فرض می شود مسئله ای برای جداسازی مجموعه نمونه های آموزشی که متعلق به دو کلاس جداگانه هستند، وجود دارد. بردار شاخص $\mathbf{y}_i$ شامل مقادیر +1 و −1 را به نحوی تعریف می کنیم، که برای کلاس 1 مقدار +1 و برای کلاس 2 مقدار −1 داشته باشد، یعنی:

$$y_i = \begin{cases} 1 & if\ x_i\ in\ class\ 1 \\ -1 & if\ x_i\ in\ class\ 2 \end{cases} \qquad (1\text{-}2)$$

هر ابر صفحه دقیقاً به وسیله جهت و مکانش در فضا مشخص می شود، که $\mathbf{w}$ جهت ابر صفحه و b مکان آن را در فضا مشخص می کند.
تابع تصمیم گیری را می توان به صورت زیر تعریف کرد:

$$d(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x} + b) \qquad (2\text{-}2)$$

یعنی:

$$\begin{cases} (\mathbf{w}^T\mathbf{x_i} + b) > 0 & if\ \ y_i = +1 \\ (\mathbf{w}^T\mathbf{x_i} + b) < 0 & if\ \ y_i = -1 \end{cases} \qquad (3\text{-}2)$$

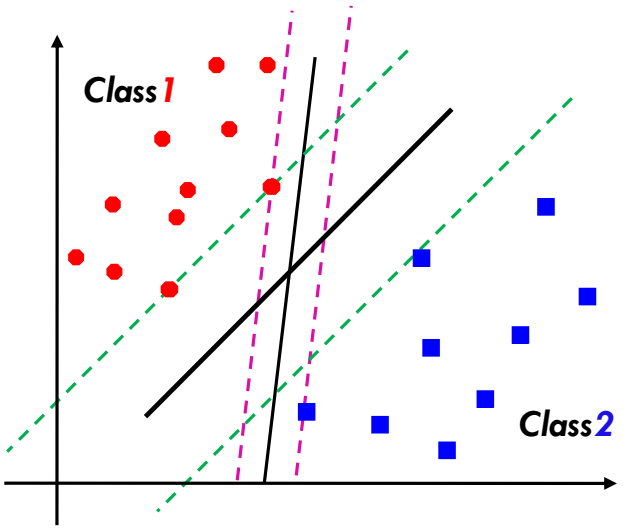
ابر صفحه ای که در این گونه از مسائل برای طبقه بندی داده ها استفاده می شود باید دارای دو ویژگی خاص باشد: اول اینکه دارای کمترین میزان خطای ممکن باشد، و از سویی دیگر از داده های هر کلاس بیشترین فاصله ممکن را داشته باشد. در این حالت اگر رابطه (2-3) برای صفحه مجزاساز در نظر گرفته شود، داده های آموزشی در بالا و پایین این صفحه قرار خواهند گرفت. که به ترتیب برای y_i=+1 ، $(\mathbf{w}^T\mathbf{x_i} + b) > 0$ و برای y_i=-1، $(\mathbf{w}^T\mathbf{x_i} + b) < 0$ خواهد بود.
بر اساس شرایط بیان شده، زمانی مجموعه ای از نقاط به صورت بهینه با یک صفحه جداسازی می شوند که:

1- بدون اشتباه در کلاس مربوط به خود قرار گرفته باشند.

2- فاصله بین نزدیکترین نقاط هر کلاس داده تا صفحه جدا کننده بیشینه باشد.

بر این اساس، پارامترهای $\mathbf{w}$ و b باید به گونه ای محاسبه گردند که دو شرط ذکر شده برقرار باشد.

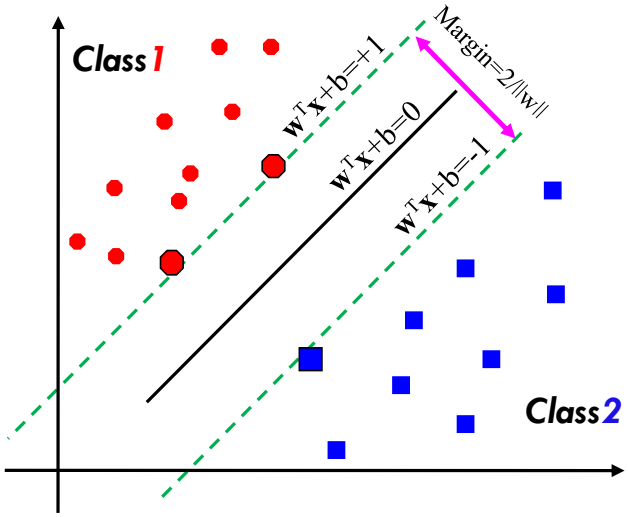
بنابراین در کلاسه بند ماشین های بردار پشتیبان خطی قصد بر این است که ابر صفحه ای را به دست آوریم که علاوه بر کلاسه بندی صحیح داده های آموزشی، حاشیه بین دو کلاس را بیشینه کند. شکل 2-4 تفاوت بین دو ابر صفحه مجزاساز را نشان می دهد.



شکل 2-4: ابر صفحه های مجزاساز: حاشیه ایجاد شده توسط خط ضخیم تر بیشتر از حاشیه ایجاد شده توسط خط باریک است.

ابر صفحه ای که در شکل 2-4 با خط ضخیم تر نشان داده شده است ابر صفحه مورد نظر جهت جداسازی دو کلاس می باشد، زیرا علاوه بر جداسازی دو کلاس، حاشیه بین ابر صفحه و دو کلاس را هم بیشینه کرده است.

فرض کنیم معادله ابر صفحه مجزاساز بهینه به صورت $\mathbf{w}^T\mathbf{x}_i+b=0$ باشد، بنابراین معادله ابرصفحه های حاشیه ای در دو طرف ابر صفحه مجزاساز به صورت $\mathbf{w}^T\mathbf{x}_i+b=+1$ و $\mathbf{w}^T\mathbf{x}_i+b=-1$ خواهد بود. شکل 2-5 معادلات در نظر گرفته شده برای حاشیه ها و صفحه مجزاساز بهینه را نشان می دهد.



شکل 2-5: صفحه مجزاساز بهینه و حاشیه های آن

چون داده ها به صورت خطی جدایی پذیر هستند پس هیچ کدام از داده ها روی صفحه $\mathbf{w}^T\mathbf{x}+b=0$ قرار نمی گیرد.

بنابراین می توان به جای رابطه (2-3)، از رابطه زیر استفاده کرد:

$$(\mathbf{w}^T \mathbf{x_i} + b)\begin{cases} \geq 1 & if \quad y_i = +1 \\ \leq -1 & if \quad y_i = -1 \end{cases} \tag{4-2}$$

البته به جای 1 و 1- در سمت راست نامساوی می توان به ترتیب a و a- قرار داد، اما در هر صورت با تقسیم طرفین رابطه بر a باز هم به همان رابطه (2-4) خواهیم رسید. رابطه (2-4) را می توان به فرم ساده تر زیر تبدیل کرد:

$$y_i(\mathbf{w}^T \mathbf{x_i} + b) \geq 1 \qquad , i = 1,2,...,N \tag{5-2}$$

جهت معرفی صفحه مجزاسازی که از بیشترین حاشیه ممکن برخوردار باشد، باید فاصله بین دو حاشیه در نظر گرفته شده، ماکزیمم گردد. فاصله بین این دو حاشیه به صورت زیر به دست می آید:

$$\frac{\left|(\mathbf{w}^T \mathbf{x} + b - 1) - (\mathbf{w}^T \mathbf{x} + b + 1)\right|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \tag{6-2}$$

بر اساس خروجی محاسبه شده از رابطه (2-6)، اگر $2/\|\mathbf{w}\|$ ماکزیمم گردد، حاشیه مورد نظر ماکزیمم خواهد شد. اما برای سادگی کار به جای ماکزیمم کردن $2/\|\mathbf{w}\|$، می توان مربع مخرج آن یعنی $\|\mathbf{w}\|^2$ را مینیمم کرد. یا به عبارتی می توان جمله را به صورت $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ نوشت.

بنابراین بر اساس شرایط بیان شده، مسئله ماشین بردار پشتبان به فرم زیر تبدیل می شود که به آن یک مسئله درجه دو[1] گفته می شود.

$$\begin{aligned} &\text{minimize} \quad \mathrm{J}(\mathbf{w}) \equiv \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ &\text{subject to} \quad \mathrm{y_i}\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1, \quad i = 1,2,\ldots,N \end{aligned} \tag{7-2}$$

چون فرم رابطه (2-7) یک مسئله درجه 2 با شرط نامساوی است، بنابراین مقدار تابع هدف[2] یکتا خواهد بود، یعنی فقط یک نقطه اکسترمم دارد. این یکی از ویژگیهای مهم ماشین بردار پشتیبان نسبت به شبکه های عصبی چند لایه[3]

[1] Quadratic problem
[2] Objective function
[3] Multi-layer Neural Networks

است که در آن تعداد زیادی نقاط مینیمم محلی وجود دارد [3].

سوالی که پس از معرفی معادله (2-7) مطرح می شود، چگونگی مینیمم سازی مسئله درجه دو فوق و به دست آوردن مقادیر بهینه **w** و b با توجه به قید نامساوی موجود است. راه کار پیشنهادی استفاده از ضریب لاگرانژ[1] است.

ضریب لاگرانژ که گاهی ضریب نامعین نیز خوانده می شود، جهت شناسایی نقاط خاص تابعی که دارای چندین متغیر و محدودیت است، مورد استفاده قرار می گیرد. در واقع اگر هدف مینیمم سازی تابع **f(x)** با توجه به محدودیت **g(x)≥0** باشد، باید تابع لاگرانژ که به صورت زیر تعریف می شود، مینیمم شود:

$$L(\mathbf{x},\alpha) \equiv f(\mathbf{x}) + \alpha g(\mathbf{x}) \qquad (2\text{-}8)$$

که در آن $\alpha \geq 0$ ضریب لاگرانژ خوانده می شود.

با در نظر گرفتن تابع لاگرانژ، فرم با قید نامساوی مسئله درجه دوم (2-7) به فرم بدون قید رابطه (2-9) تبدیل می شود که به آن تابع لاگرانژ اولیه[2] گفته می شود.

$$L_p(\mathbf{w},b,\boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N}\alpha_i\left[y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1\right] \qquad (2\text{-}9)$$

برای مینیمم کردن معادله (2-9) باید نقاط ایستان[3] تابع لاگرانژ را به دست آورد. برای رسیدن به این هدف شرایط KKT [4] اعمال می شود که به صورت زیر است:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}} = 0 &\Rightarrow \mathbf{w}_0 = \sum_{i=1}^{N}\alpha_i\mathbf{x}_i y_i \\ \frac{\partial L}{\partial b} = 0 &\Rightarrow \sum_{i=1}^{N}\alpha_i y_i = 0 \end{aligned} \qquad (2\text{-}10)$$

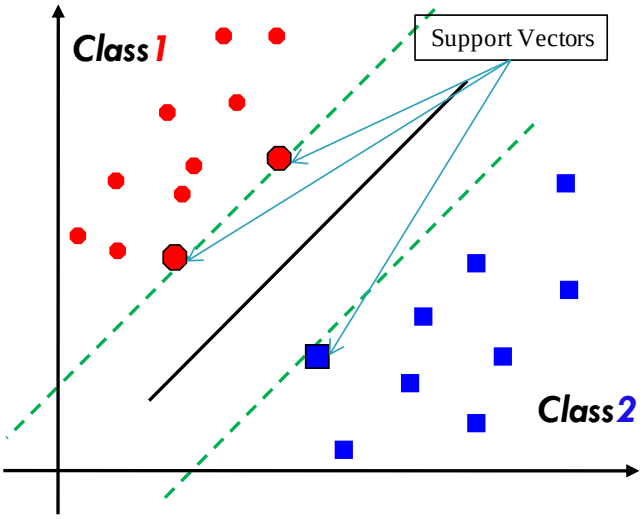
حال اگر مقدار **w** به دست آمده از مشتقات جزئی رابطه (2-10) در خود رابطه (2-9) قرار داده شود، معادله اساسی ماشین های بردار به صورت رابطه (2-11) معرفی می شود که فرم دوگان معادله لاگرانژ[5] (2-9) است.

[1] Lagrange coefficient
[2] Primal Lagrange Function
[3] Stationary points
[4] Karush-Kuhn-Tucker conditions
[5] Dual Lagrange Function

$$\text{Max} \quad L_d(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{S.T.} \quad \begin{cases} \alpha_i \geq 0 \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \qquad (11\text{-}2)$$

با حل معادله بالا مقادیر ضرایب لاگرانژ یعنی $\boldsymbol{\alpha}$ برای هر یک از بردارهای آموزشی به دست خواهد آمد، که باتوجه به محدودیت ذکر شده در رابطه (11-2) مقداری بزرگتر یا مساوی صفر خواهد بود. از میان بردارهای آموزشی کلاس 1 و 2، آنهایی که مقدار α متناظر با آنها بزرگتر از صفر است، بردارهای پشتیبان[1] نامیده می شوند. در محاسبات برای بدست آوردن ابر صفحه بهینه فقط این بردارها استفاده می گردند، بنابراین حجم محاسبات تا حد زیادی کاهش می یابد. بردارهای پشتیبان درست بر روی دو ابر صفحه $\mathbf{w}^T\mathbf{x}+b=\pm 1$ قرار می گیرند. این بردارها در شکل 6-2 برجسته تر نشان داده شده اند.



شکل 6-2: بردارهای پشتیبان کلاسهای 1 و 2 بر روی ابر صفحه های حاشیه ای

از رابطه (10-2) مقدار مطلوب $\mathbf{w}_0$ به دست آمد. مقدار بهینه b نیز از طریق رابطه $b = y_i - \mathbf{w}^T \mathbf{x}_i$ و میانگین گیری از تمامی مقادیر به دست آمده، محاسبه می شود. معادله کلی محاسبه مقدار بهینه b را می توان به صورت رابطه (12-2) بیان نمود.

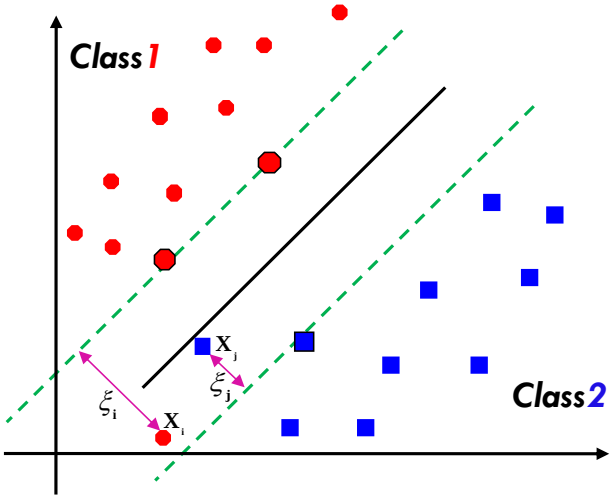
[1] Support Vectors

$$b_0 = \frac{1}{SV}\sum_{i=1}^{SV}\left(y_i - \mathbf{w}^T \mathbf{x}_i\right) \qquad (12\text{-}2)$$

که در آن SV مشخص کننده تعداد بردارهای پشتیبان[1] است. با به دست آوردن مقادیر $\mathbf{w}_0$ و b_0 می توان به ابر صفحه مجزاساز بهینه دست یافت. با داشتن این مقادیر و با توجه به رابطه تابع تصمیم ماشین بردار پشتیبان، یعنی رابطه (2-2)، برای کلاسه بندی یک داده ناشناخته مثل $\mathbf{x}$، اگر مقدار تابع تصمیم بزرگتر از صفر باشد، داده در کلاس 1 طبقه بندی خواهد شد، و اگر مقدار تابع تصمیم کوچکتر از صفر باشد، داده در کلاس 2 دسته بندی می شود.

2-2- ماشین بردار پشتیبان در سیستم های خطی جداناپذیر

گاهی در سیستم های خطی شرایطی ایجاد می شود که داده ها به صورت خطی جداپذیر نیستند اما می توان با پذیرش مقدار کمی خطا، داده ها را به صورت خطی کلاسه بندی کرد. در این حالت تعدادی از داده ها در کلاس مربوط به خود قرار نمی گیرند. همان گونه که در شکل 2-7 مشخص می باشد نمی توان با یک کلاسه بند خطی داده های دو کلاس را کاملاً از هم جدا کرد. هر گونه تلاشی به منظور جداسازی داده ها با یک ابر صفحه خطی به نتیجه نمی رسد مگر آنکه یکسری از داده ها در داخل باند جداسازی قرار گیرد. در چنین شرایطی برای حل مسئله متغیر $\xi \geq 0$ به نام متغیر کساد[2] تعریف می شود، که به نوعی میزان خطای کلاسه بندی را نشان می دهد.



شکل 2-7: سیستم های خطی جدا ناپذیر با میزان خطای ξ_i

[1] Support Vectors
[2] Slack variable

سه حالت برای بردارهای ویژگی نمونه های آموزشی پیش می آید:

1. بردارهایی که خارج از حاشیه ها قرار می گیرند و صحیح کلاسه بندی شده اند. این بردارها شرایط در نظر گرفته شده در معادله (2-5) را برقرار می سازند.
2. بردارهایی که در فاصله بین حاشیه ها قرار می گیرند و صحیح کلاسه بندی شده اند. این بردارها در شرایط زیر صدق می کنند.

$$0 \le y_i \left(\mathbf{w}^T \mathbf{x} + b \right) < 1 \qquad (13\text{-}2)$$

3. بردارهایی که به اشتباه کلاسه بندی شده اند. این بردارها از شریط زیر تبعیت می کنند.

$$y_i \left(\mathbf{w}^T \mathbf{x} + b \right) < 0 \qquad (14\text{-}2)$$

تمامی سه حالت فوق را می توان با اضافه کردن یک محدودیت جدید به قید نامساوی معادله (2-5) به صورت زیر در نظر گرفت:

$$y_i \left(\mathbf{w}^T \mathbf{x} + b \right) \ge 1 - \xi_i \qquad (15\text{-}2)$$

گروه اول داده ها با در نظر گرفتن $\xi_i = 0$ ، گروه دوم با در نظر گرفتن $0 < \xi_i \le 1$ و گروه سوم با در نظر گرفتن $\xi_i > 1$ بدست می آیند. در این شرایط هدف این است که بین ماکزیمم شدن فاصله حاشیه و مینیمم شدن خطا یک حالت موازنه[1] ایجاد شود.

در فرم ریاضی این عمل برابر است با مینمم کردن تابع هزینه زیر:

$$J\left(\mathbf{w}, b, \boldsymbol{\xi}\right) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} l\left(\xi_i\right) \qquad (16\text{-}2)$$

که در آن $\boldsymbol{\xi}$ بردار پارامترهای ξ_i است و C پارامتری جهت برقرار شدن موازنه بین مقدار حاشیه و میزان خطاست که با سعی و خطا توسط کاربر به صورت عدد تعریف مشخص می گردد.

تابع $l\left(\xi_i\right)$ نیز به صورت زیر تعریف می شود:

$$l\left(\xi_i\right) = \begin{cases} 1 & \xi_i > 0 \\ 0 & \xi_i = 0 \end{cases} \qquad (17\text{-}2)$$

[1] Trade off

بهینه سازی رابطه (2-16) مشکل است زیرا شامل تابع ناپیوسته $l(\cdot)$ می باشد. بنابراین فرم اولیه مسئله به صورت زیر تغییر می‌کند:

$$\begin{aligned} &\text{Minimize} \quad J(\mathbf{w},b,\boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{Subject to} \quad \begin{cases} y_i\left[\mathbf{w}^T\mathbf{x}_i + b\right] \geq 1-\xi_i, & i=1,2,\ldots,N \\ \xi_i \geq 0, & i=1,2,\ldots,N \end{cases} \end{aligned} \tag{2-18}$$

برای حل معادله بالا یک بار دیگر با استفاده از ضرایب لاگرانژ، به معادله لاگرانژ اولیه به فرم زیر می رسیم.

$$L_p(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{N}\beta_i\xi_i \tag{2-19}$$

که در آن $\boldsymbol{\alpha}$ و $\boldsymbol{\beta}$ بردارهای ضرایب لاگرانژ هستند.

باز هم برای حل معادله فوق، شرایط KKT که در واقع اعمال مشتقات جزیی از رابطه فوق نسبت به $\mathbf{w}$، b و $\boldsymbol{\xi}$ و مساوی صفر قرار دادن آن است، به صورت زیر اعمال می شود:

$$\begin{aligned} &\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i \\ &\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N}\alpha_i y_i = 0 \\ &\frac{\partial L}{\partial \boldsymbol{\xi}} = 0 \Rightarrow \alpha_i + \beta_i = C \end{aligned} \tag{2-20}$$

با قرار دادن این روابط در معادله (2-19)، به فرم دوگان معادله لاگرانژ می رسیم که معادله اساسی ماشین بردار پشتیبان در حالت خطی جداناپذیر است و به صورت رابطه زیر می باشد:

$$\begin{aligned} &\text{Maximize} \quad L_d(\boldsymbol{\alpha}) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ &\text{Subject to} \quad \begin{cases} 0 \leq \alpha_i \leq C \\ \sum_{i=1}^{N}\alpha_i y_i = 0 \end{cases} \end{aligned} \tag{2-21}$$

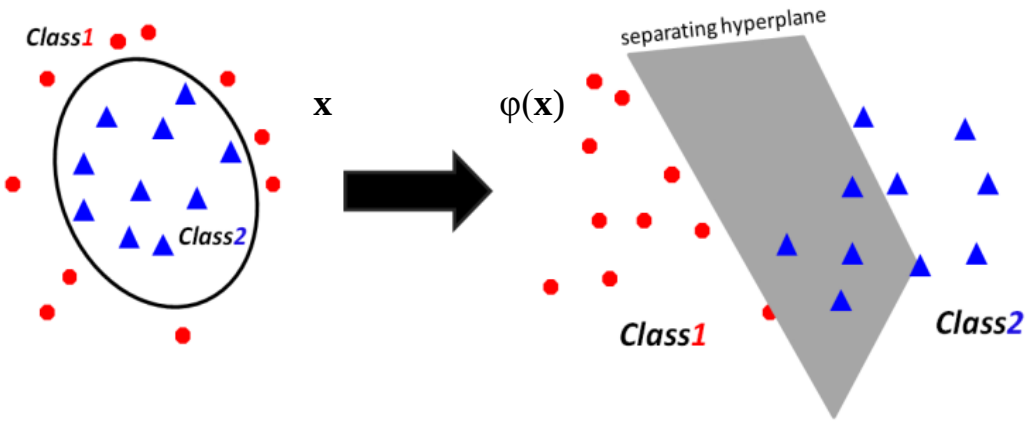
همان گونه که مشاهده می شود، فرم نهایی صورت مسئله ماشین بردار پشتیبان جداناپذیر خطی مشابه با سیستم های

جداپذیر خطی است و تنها تفاوت این دو در کران بالای ضرایب لاگرانژ می باشد.

2-3- ماشین بردار پشتیبان در سیستم های غیر خطی

در بخش‌های پیشین ماشین‌های بردار پشتیبان را به صورت یک کلاسه بند خطی بهینه معرفی کردیم. سوالی که در اینجا مطرح می شود این است که اگر داده ها به صورت خطی جداپذیر نباشند چه باید کرد؟ آیا می توان ایده ماشین بردار پشتیبان خطی را به سیستم های غیرخطی تعمیم داد؟

ماشین بردار پشتیبان در ابتدای کار، تنها برای سیستم های خطی به کارگرفته می شد و صفحه مجزاساز بهینه تنها برای حالت خطی وجود داشت. این در حالی بود که در بسیاری از مسائل طبقه بندی و رگرسیون، راه حل خطی جواب مناسبی را ارائه نمی کرد. مطالعات گسترده انجام شده در این زمینه، تئوری مرسر[1] را جهت حل این مشکل ارائه کرد تا ماشین بردار پشتیبان بتواند در سیستم های غیرخطی گسترش یابد [4]. ایده اصلی این تئوری انتقال برداری مانند $\mathbf{x}$ از فضای محدود (فضای ورودی) به فضای بالاتر (فضای ویژگی یا فضای هیلبرت[2]) با استفاده از تبدیل هیلبرت و طبقه بندی آن در فضای بالا بود. در این شرایط برداری مانند $\mathbf{x}$ در فضای بالاتر به صورت $\varphi(\mathbf{x})$ نوشته می شود. شکل 2-8 نحوه انتقال از فضای ورودی به فضای ویژگی را نشان می دهد.



شکل 2-8: انتقال فضای ورودی (سمت چپ) به فضای ویژگی (سمت راست) با تبدیل هیلبرت

با استفاده از این قضیه، با تبدیل $\mathbf{x}$ به $\varphi(\mathbf{x})$ در معادلات (2-10) و (2-12) ، به راحتی می توان روابط بردار

[1] Mercer
[2] Hilbert

وزن و بایاس ماشین بردار پشتیبان غیرخطی را نیز به صورت زیر به دست آورد.

$$\mathbf{w}_0 = \sum_{i=1}^{N} \alpha_i \varphi(\mathbf{x}_i) y_i \qquad (22\text{-}2)$$

$$b_0 = \frac{1}{SV} \sum_{i=1}^{SV} \left(y_i - \mathbf{w}^T \varphi(\mathbf{x}_i) \right) \qquad (23\text{-}2)$$

و در نهایت با به کارگیری روابط بالا، تابع تصمیم ماشین بردار پشتیبان غیرخطی به صورت زیر خواهد بود:

$$d(\mathbf{x}) = sign(\mathbf{w}^T \varphi(\mathbf{x}) + b) \qquad (24\text{-}2)$$

2-3-1- حیله کرنل

در سیستم های غیرخطی برای بردن داده ها به فضای بالاتر، برداری مانند x به صورت $\varphi(\mathbf{x})$ نوشته می شود، در نتیجه برای محاسبه تابع تصمیم ماشین بردار پشتیبان غیرخطی از رابطه (2-24) استفاده خواهد شد. مسئله ای که اکنون وجود دارد، محاسبه بردار $\varphi(x)$ و ضرب داخلی آن با $\mathbf{w}$ است، چرا که بردار $\varphi(\mathbf{x})$ اکنون در فضای هیلبرت با ابعاد ویژگی بسیار بالا قرار داد و محاسبات آن در این حالت بسیار پیچیده و وقت گیر خواهد شد. علاوه بر این، در این شرایط نمی توان از رابطه (2-22) مستقیماً برای محاسبه بردار وزن $\mathbf{w}$ استفاده نمود زیرا محاسبه بردار $\varphi(\mathbf{x})$ در فضای هیلبرت مبهم و پیچیده است. برای حل این مشکل با استفاده از حیله کرنل[1] به جای محاسبه مستقیم معادله (2-22)، رابطه تابع تصمیم ماشین بردار پشتیبان غیرخطی به صورت زیر نوشته می شود:

$$d(\mathbf{x}) = sign(\sum_{i=1}^{SV} \alpha_i y_i \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}) + b) \qquad (25\text{-}2)$$

در این حالت به جای محاسبه ضرب داخلی $\varphi(\mathbf{x}_i)^T\varphi(\mathbf{x})$ در فضای هیلبرت، که به دلیل ابعاد بسیار بالا در این فضا منجر به محاسبات پیچیده ای می شود، با استفاده از توابع کرنل[2] غیرخطی به صورت:

$$K(\mathbf{x}_i, \mathbf{x}) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}) \qquad (26\text{-}2)$$

این ضرب داخلی را در فضای ورودی انجام می دهیم که هم منجر به پیچیدگی محاسباتی کمتر و سرعت بالاتر در انجام محاسبات می شود و هم اینکه دیگر نیازی به محاسبه تابع

[1] Kernel trick
[2] Kernel function

ناشناخته $\varphi(\mathbf{x})$ در فضای هیلبرت نیست [5]. در واقع تابع کرنل برای محاسبه ضرب داخلی با استفاده از داده های ورودی اصلی به کار می رود. بنابراین هر جا که ضرب داخلی به صورت $\varphi(\mathbf{x}_i)^T\varphi(\mathbf{x})$ مشاهده شود، کافیست به جای آن $K(\mathbf{x}_i,\mathbf{x})$ قرار دهیم.

رابطه نهایی تابع تصمیم ماشین بردار پشتیبان خطی در حالت کلی به صورت زیر خواهد بود:

$$d(\mathbf{x}) = sign(\sum_{i=1}^{SV} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b) \qquad (27\text{-}2)$$

$$b_0 = \frac{1}{SV}\sum_{i=1}^{SV}(\alpha_i y_i K(\mathbf{x}_i, \mathbf{x})) \qquad (28\text{-}2)$$

2-3-2- توابع کرنل

کرنل های مختلفی برای استفاده در فضای ویژگی معرفی شده اند که بسته به شرایط و کاربردهای مختلف مورد استفاده قرار می گیرند. جدول 2-1 توابع کرنل مختلف را نشان می دهد.

جدول 2-1: توابع کرنل

تابع کرنل	نوع کلاسه بند
$K(\mathbf{x}_i, \mathbf{x}) = \mathbf{x}_i^T\mathbf{x}$	خطی
$K(\mathbf{x}_i, \mathbf{x}) = (\mathbf{x}_i^T\mathbf{x}+1)^d$	چند جمله ای از درجه d
$K(\mathbf{x}_i, \mathbf{x}) = \exp(-\frac{\|\mathbf{x}_i-\mathbf{x}\|^2}{2\sigma^2})$	تابع گوسی اساس شعاعی[1]
$K(\mathbf{x}_i, \mathbf{x}) = \exp(-\frac{\|\mathbf{x}_i-\mathbf{x}\|}{\sigma})$	لاپلاسی[2]
$K(\mathbf{x}_i, \mathbf{x}) = \tanh(a\mathbf{x}_i^T\mathbf{x}+b)$	سیگموید[3]

تابع سیگموید اساساً یک تابع کرنل نیست، در واقع این تابع شرایط مرسر را فقط برای مقادیر خاص a و b برآورده می کند. اما این تابع به عنوان یک تابع کرنل بسیار

[1] Gaussian Radial Basis Function
[2] Laplacian
[3] Sigmoid

محبوب در بعضی شرایط خاص، مانند شبکه های عصبی دولایه، به کار می رود. توابع کرنل گوسی و لاپلاسی از نوع کرنلهای ایستان[1] هستند که فرم کلی $K(\mathbf{x}_i,\mathbf{x})=K(\mathbf{x}_i-\mathbf{x})$ را دارند، و تابع کرنل چندجمله ای از نوع کرنل غیرایستان[2] است.

نکته ای که باید در مورد توابع کرنل و نوع استفاده از هر کدام در نظر داشت این است همیشه دقیقاً از قبل مشخص نیست که کدام تابع کرنل برای چه کاربردی باید انتخاب شود و پارامترهای توابع کرنل چه مقدار باید باشد. با این حال مطالعات نشان می دهد که تابع کرنل گوسی که فقط یک پارامتر دارد و همچنین تابع چند جمله ای با درجه 1 یا 2 بهترین انتخاب است. اما در صورت لزوم می توان از توابع کرنل دیگر با پیچیدگی بیشتر نیز استفاده کرد [5].

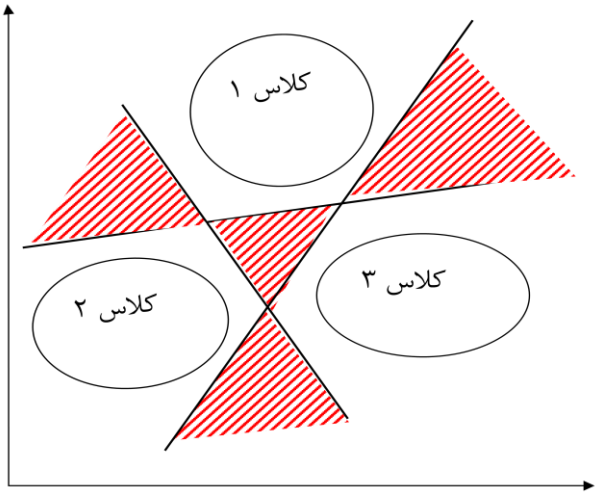
2-4- ماشین‌های بردار پشتیبان چند کلاس

ماشین بردار پشتیبان اساساً یک کلاسه بند دوتایی است، اما می توان با روشهای مختلفی از این کلاسه بند برای حل مسائل چند کلاس نیز استفاده کرد. چهار روش کلاسه بندی ماشین بردار پشتیبان چند کلاس[3] وجود دارد که عبارتند از:

1. روش یک کلاس در برابر همه[4]
2. روش کلاسه بندهای دوتایی[5]
3. روش کد خروجی تصحیح کننده خطا[6]
4. روش همه کلاسها در یک بار[7]

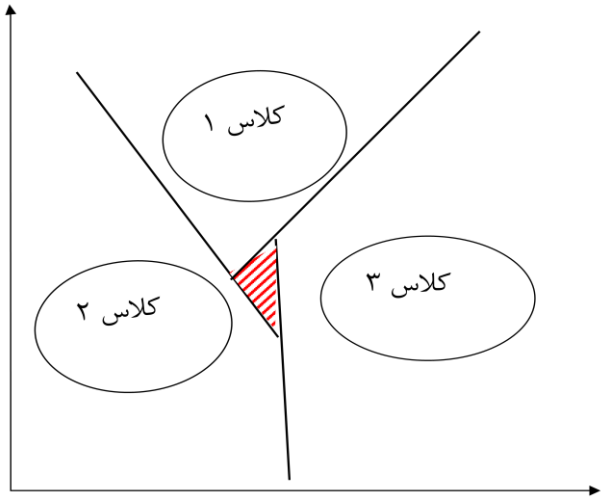
در روش یک کلاس در برابر همه، اگر n کلاس داشته باشیم، n تا کلاسه بند ماشین بردار پشتیبان دوتایی تشکیل خواهد شد به طوری که برای کلاسه بند دوتایی iام، کلاس i از بقیه کلاسها جدا می شود. اما با این روش اگر از توابع تصمیم گسسته استفاده کنیم، نواحی طبقه بندی نشده ای[8] طبق شکل 2-9 به وجود خواهد آمد [3].

[1] Stationary
[2] NonStationary
[3] Multi-class SVM
[4] One against all
[5] Pairwise
[6] Error-correcting output code (ECOC)
[7] All-at-once
[8] Unclassified regions



شکل 2-9: نواحی طبقه بندی نشده در روش کلاسه بندی یک کلاس در برابر همه

برای حل این مشکل، کرِبل [6] روش کلاسه بندهای دوتایی را ارائه کرد که در آن برای حل مسئله کلاسه بندی n کلاس، از $n(n-1)/2$ مسئله کلاسه بندی دوتایی استفاده می شود. این روش نسبت به روش یک کلاس در برابر همه روش بهتری است اما باز هم نواحی طبقه بندی نشده ای طبق شکل 2-10 وجود خواهد داشت.



شکل 2-10: ناحیه طبقه بندی نشده در کلاسه بندی دوتایی

البته نواحی کلاسه بندی نشده با استفاده از یکی از روشهای: توابع عضویت، درخت های تصمیم، روش کد خروجی تصحیح کننده خطا و یا روش همه کلاسها در یکبار اصلاح خواهد شد.

در این تحقیق از روش کلاسه بندهای ماشین بردار پشتیبان دوتایی استفاده شده است. در روش کلاسه بندهای دوتایی، در فاز آموزش آن، برای همه ترکیبهای دوتایی از کلاسهای i و j , $i \neq j$ مقادیر $\mathbf{w_{ij}}$ و b_{ij} متناظر استخراج شده و

برای تشکیل توابع تصمیم دوتایی $d_{ij}(\mathbf{x})$ مورد استفاده قرار می گیرد. در هر کلاسه بند دوتایی داده های آموزشی کلاسهای متناظر در تشکیل تابع تصمیم بین آن دو کلاس مورد استفاده قرار می گیرد، بنابراین تعداد داده های آموزشی نسبت به روش یک کلاس در برابر همه کلاسها، که از کل داده های آموزشی برای هر کلاسه بند استفاده می کند، بسیار کمتر خواهد شد. اما تعداد توابع تصمیم در روش کلاسه بندهای دوتایی، که برابر با $n(n-1)/2$ است، در مقایسه با روش یک کلاس در برابر همه، که برابر n می باشد، بیشتر است.

برای کلاسه بندی داده ورودی $\mathbf{x}$، در روش کلاسه بندهای دوتایی، در هر تابع تصمیم دوتایی $d_{ij}(\mathbf{x})$ کلاس متناظر استخراج می شود و در نهایت کلاسی که بیشترین رای را داشته باشد به عنوان کلاس مورد نظر انتخاب می شود.

فصل 3- کاربرد FPGAها در پردازش سی‌گنال دی‌جی‌تال

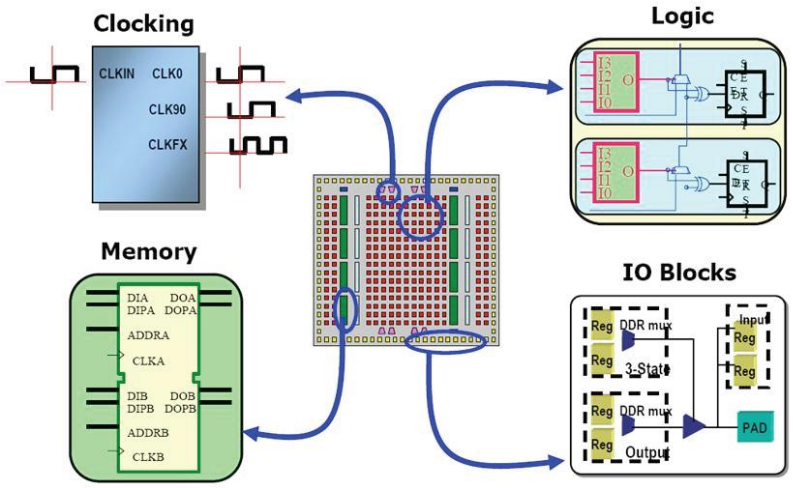
امروزه پردازش سیگنال دیجیتال در رنج بسیار وسیعی از کاربردها مانند پردازش صوت و تصویر، رادیوهای دیجیتال، رادار، موبایل، سیستم موقعیت یاب جهانی[1] و ... استفاده می شود. توسعه این کاربردها بر روی سخت افزار یک موضوع فعال در سه دهه اخیر بوده است، بنابراین یک سری از پردازنده های خاص برای پردازش سیگنال دیجیتال به نام پردازنده های DSP پدید آمد. این پردازنده های DSP در بسیاری از کاربردهای پردازش سیگنال دیجیتال مورد استفاده قرار می گرفت اما در بعضی موارد به خوبی پاسخگوی مصرف توان، فضای سخت افزاری و سرعت پردازش نبود. در سالهای اخیر FPGAها به عنوان مناسب ترین ساختار مداری برای اجرای الگوریتم های پردازش سیگنال دیجیتال به ویژه در کاربردهای بلادرنگ معرفی شدند [7]. در واقع ویژگی موازی بودن عملیات در FPGA است که در حال حاضر آن را به عنوان بهترین انتخاب در پردازش سیگنال دیجیتال با حجم محاسبات زیاد و پیچیده معرفی کرده است. FPGA به طراح اجازه می دهد تا طرح دیجیتال خود را آنچنان که می خواهد و با هر حجم و پیچیدگی لازم، طراحی کند. FPGAها مجموعه ای از بلوک های دیجیتال ساده تا پیشرفته است که با طراحی مناسب می توانند الگوریتم های پیچیده را به خوبی حل نماید. اگر عملیات الگوریتم های اصلی پردازش سیگنال دیجیتال مانند FFT[2]، DCT[3]، فیلترهای FIR[4] و IIR[5] و ... را به طور دقیق مطالعه کنیم، ملاحظه می شود که امکان اجرای این الگوریتم ها بر روی سخت افزار FPGA به منظور به دست آوردن سرعت محاسبات بیشتر به راحتی امکان پذیر است.

3-1- ساختار FPGA

ساختار درونی یک FPGA پیشرفته به صورت شکل 3-1 می باشد. یک FPGA در ساختار داخلی خود، بلوک های دیجیتالی قابل برنامه ریزی را برای طراح فراهم می کند که شامل هزاران بلوک ترکیبی و فلیپ فلاپ به همراه بلوک های I/O [6] برای ارتباط با دنیای خارج، و یک مبنع کلاک مستقل است.

---

[1] Global positioning system
[2] Fast Fourier Transform
[3] Discrete cosine transform
[4] Finite Impulse Response
[5] Infinite Impulse Response
[6] Input/Output



شکل 3-1: ساختار داخلی یک FPGA نوعی

یک تراشه FPGA به تنهایی و جدای از ارتباطات با دنیای خارج معنی پیدا نمی کند و لازم است که به راحتی با تراشه های دیگر و یا سیگنال های خارجی ارتباط داشته باشد. به منظور دستیابی به این هدف، سازندگان FPGA اند تلاش و سرمایه گذاری بسیار وسیعی برای بالا بردن انعطاف پذیری بلوک های I/O تراشه های خود انجام داده اند. در حال حاضر پورت های FPGA می توانند به عنوان ورودی، خروجی یا هم ورودی و هم خروجی مورد استفاده قرار گیرد. همچنین تکنیکهای جدیدی برای افزایش پهنای باند در FPGAها به کار می رود، مثلاً کلاک کردن مدار در هر دو لبه بالارونده و پایین رونده آن.

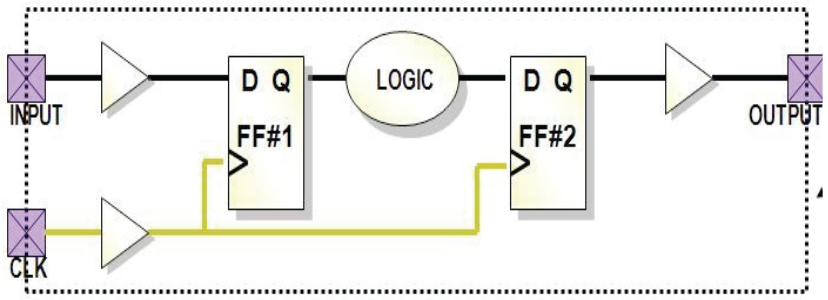
تمام عناصری که در شکل 3-1 نشان داده شده است، کمتر از 20% فضای سیلیکون درون چیپ FPGA را شامل می شود. چیزی که در شکل نشان داده نشده، حجم وسیعی از اتصالات داخلی قابل برنامه ریزی و مدارهای کمکی است که بلوک های دیجیتال را برنامه ریزی می کند تا به یک قطعه دیجیتال قابل استفاده تبدیل شود.

برای حل مشکلات و محدودیت های طراحی بر روی قطعات سیلیکونی، سازندگان FPGAها همچنین از هسته های هوشمند[1] با اتصالات سخت افزاری در درون تراشه ها برای ساخت توابعی که در کاربردهای پردازش سیگنال دیجیتال زیاد کاربرد دارد، استفاده می کنند. این بلوک های غیر قابل برنامه ریزی شامل پردازنده های همه منظوره[2]، واسط های سریال[3] با سرعت بالا، بلوک های محاسباتی ریاضی و واحدهای کنترل دسترسی میانی اترنت[4] است.

[1] Intellectual Property (IP) core
[2] General purpose processors
[3] Serial Interface
[4] Ethernet Medium Access Control (MAC) unit

3-2- اصول یک طراحی دیجیتال

شکل 3-2 ساختار پایه ای یک طراحی دیجیتال نمونه در درون FPGA، که از بلوک های دیجیتال ترکیبی در بین آرایه ای از فلیپ فلاپها تشکیل شده است، را نشان می دهد. این ساختار یک ساختار اصولی است که در بسیاری از کاربردهای پردازش سیگنال دیجیتال از آن استفاده می شود.



شکل 3-2: ساختار پایه یک طراحی دیجیتال در FPGA

در شکل 3-2 بلوک دیجیتال می تواند هر زیر مجموعه ای از مدارات دیجیتال باشد، که در آن وضعیت فعلی خروجی تنها به وضعیت فعلی ورودی و کلاک مدار بستگی دارد. بنابراین تمام توابع دیجیتال مانند AND، OR و هر ترکیب پیچیده ای از آنها (مانند مالتی پلکسرها، دیکدرها، انکدرها و...) می تواند در قسمت بلوک دیجیتال قرار بگیرد. به این نکته باید دقت شود که توابع دیجیتال با هر پیچیدگی دلخواه، مانند مالتی پلکسرها، دیکدرها، انکدرها و ... ، با استفاده از این بلوکهای پایه قابل ساخت است. قسمت INPUT می تواند یک یا چند بیتی باشد. در این مدار همچنین یک بخش کلاک (CLK)[1] وجود دارد که شامل یک نوسانساز موج مربعی با فرکانس ثابت است. هر دو بلوک فلیپ فلاپ که می تواند شامل چندین هزار فلیپ فلاپ باشد، توسط یک منبع کلاک تغذیه می شود و هر زمان که سیگنال کلاک در لبه بالارونده باشد، سیگنال را از ورودی D به خروجی Q انتقال می دهد.

نکته مهمی که طراح باید آن را مد نظر قرار دهد، این است که تاخیر بین هر کدام از ورودیها به سمت بلوک دیجیتال و خروجی آن، کمتر از یک پریود کلاک مدار باشد. اگر این تاخیر بیش از مقدار یادشده باشد، مقدار قبلی در ورودی فلیپ فلاپ به خروجی آن انتقال داده می شود که باعث اختلال در پردازش زمانی سیستم می گردد. همانطور که

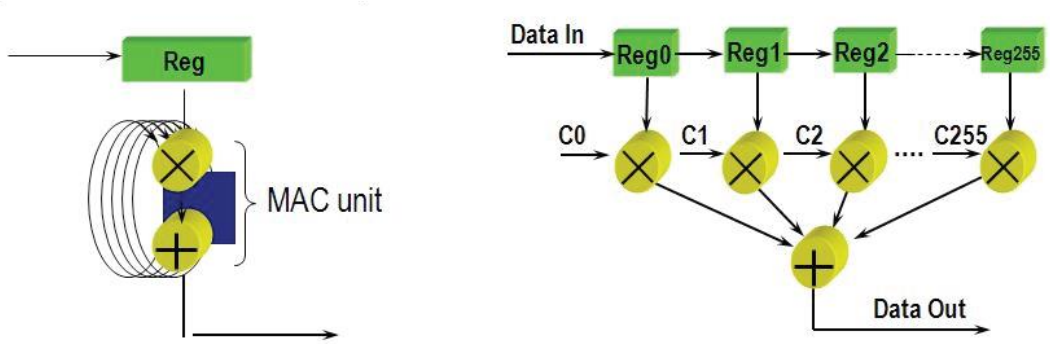
[1] Clock

بعداً نشان خواهیم داد این روند به صورت اتوماتیک توسط نرم افزارهای شبیه سازی انجام می شود و طراح فقط باید مشخصات رفتاری و ساختاری مدار را مد نظر قرار دهد.

در بعضی موارد ممکن است طراح بخواهد کلاک ورودی فلیپ فلاپ را از خروجی یک کلاک ترکیبی تغذیه کند، و ممکن است این طراحی نتایج خوبی هم در طول شبیه سازی بدهد. اما بعداً با ایجاد تغییرات کوچک در تاخیرهای مختلف سیگنال ها، ممکن است زمان بندی مدار دچار اغتشاش شود. بنابراین باید در مدارات همزمان جهت تغییرات زمانی تاخیرها، برای ماکزیمم زمان ممکن هر سیگنال یک مرز در نظر گرفته شود.

3-3- ویژگی‌های برتر FPGA ها در پردازش سیگنال دیجیتال

مهمترین ویژگی FPGA ها در مقابل پردازنده های DSP اجرای عملیات پردازش سیگنال دیجیتال به صورت موازی است، یعنی ترکیب توابع سخت افزاری که می توانند به صورت همزمان در قسمتهای مختلف چیپ عمل کنند. شکل 3-3 نشان می دهد که چگونه یک فیلتر FIR با 256 ضریب توسط هر کدام از این سخت افزارها می تواند پیاده سازی شود.

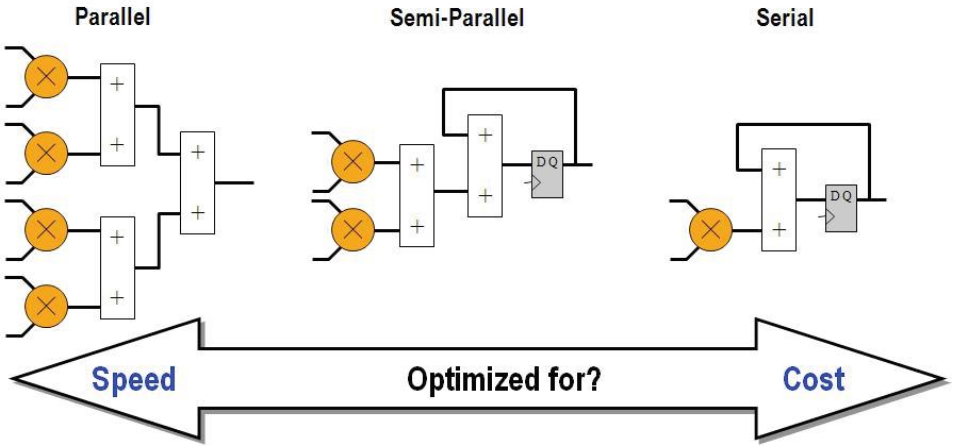


شکل 3-3: مقاسیه پیاده سازی یک فیلتر FIR بر روی دو سخت افزار:
سمت راست: FPGA ، سمت چپ: DSP

برای به دست آوردن خروجی فیلتر با استفاده از پردازنده DSP به 256 کلاک نیاز است، در حالیکه FPGA با هر کلاک ورودی، خروجی جدید را تولید می کند. هر چند که سرعت کلاک پردازنده های DSP نسبتاً سریع تر از FPGA ها است، اما اگر بخواهیم تعداد زیادی از فیلترهای FIR با ضرایب طولانی را پیاده کنیم و همزمان عملیات جمع ها و ضرب ها انجام شود، آنگاه خواهیم دید که DSP ها قابل مقایسه با FPGA ها نیستند [8].

ویژگی مهم دیگر FPGAها، انعطاف پذیر بودن آنها برای انتخاب بین سرعت و فضای سخت افزاری در فرآیند طراحی

است. شکل 3-4 سه ساختار متفاوت از پیاده سازی مجموع حاصلضربها روی FPGA را نشان می دهد.



شکل 3-4: انعطاف پذیری FPGA در انتخاب بین سرعت و هزینه

همانگونه که در شکل مشاهده می شود، با توجه به نوع سیستم مورد طراحی، نوع FPGA و میزان اهمیت بین سرعت و فضا در فرآیند طراحی، می توان از حالت سمت چپ، که یک طراحی کاملا موازی است و بیشترین سرعت را دارد، تا حالت سمت راست، که کمترین فضا را اشغال می کند، در حالیکه دارای محدودیت سرعت است، مناسب ترین حالت را انتخاب کرد [9].

3-4- روند طراحی

هر طراح که با یک مسئله طراحی مواجه می شود، باید یک سری از مراحل انجام طرح از ایده ی اولیه تا سخت افزار نهایی را مرور کند و در مورد آن بحث و بررسی نماید. این مراحل معمولاً به عنوان "روند طراحی"[1] شناخته می شود. ابتدا بعد از اینکه تمام نیازها مشخص شد، یک طراحی دیجیتال مناسب باید پیاده شود. می بایست توجه شود که ابزارهای طراحی که توسط سازندگان مختلف FPGA ارائه می شود، در این مرحله به طراح کمکی نمی کند. این ابزارها فقط زمانی که طراح آماده تبدیل یک طراحی مشخص به روی سخت افزار است، می توانند به او کمک کنند.

امروزه معمول ترین روند طراحی که در FPGA ها مورد استفاده قرار می گیرد، شامل مراحل زیر است:

[1] Design flow

3-4-1- ورود طرح

ورود طرح[1] شامل تبدیل کردن ایده های طراحی به فرم نمایش کامپیوتری است، که توسط زبانهای توصیف سخت افزاری[2] انجام می شود. مقبول ترین زبانهای توصیف سخت افزاری، دو زبان Verilog و VHDL [3] است[10]. باید به این نکته توجه کرد که زبان توصیف سخت افزاری همانطور که از نامش پیداست، فقط یک ابزار برای توصیف طرحی است که قبلاً در ذهن طراح بوده است، و یک زبان برای طراحی مدارهای الکترونیک نیست.

3-4-2- سنتز

ابزار سنتز[4]، HDL و نوع FPGA را دریافت می کند و از این اطلاعات یک نتلیست[5] ای می سازد که رفتار بلوکهای لاجیک را متناسب با فایل HDL مشخص می کند. بسیاری از ابزارهای سنتز، مراحل سنتی طراحی دیجیتال مانند بهینه سازی لاجیک، متعادل کردن رجیستر و تکنیکهای افزایش کارآیی زمانی را به طور اتوماتیک انجام می دهند، بنابراین نتلیست به دست آمده می تواند به عنوان موثرترین مرحله پیاده سازی طرح HDL باشد.

3-4-3- مکان یابی و سیم کشی

مکان یابی و سیم کشی[6] شامل دو مرحله است: ابتدا در مرحله مکان یابی، نتلیست به دست آمده در مرحله سنتز دریافت شده و یک مکان مناسب برای هر کدام از عناصر درون چیپ انتخاب می شود. سپس در مرحله سیم کشی، اتصالات داخلی تمام این عناصر با هم انجام می شود، به طوریکه از محدودیت های زمانی اجتناب شود. مهمترین محدودیتی که برای طراح به وجود می آید، فرکانس کلاک سیستم است. اما در طراحی به وسیله ابزارها و نرم افزارهایی که توسط سازندگان ارائه می شود، مشکلات و محدودیت های بیشتری نیز وجود دارد.

3-4-4- تولید BitStream

بعد از اینکه پروسه مکان یابی و سیم کشی تمام شد، یک فایل bitstream تولید می شود. این فایل زمانی که درون یک FPGA بارگذاری می شود، با استفاده از حافظه ها، دروازه

---

[1] Design entry
[2] Hardware Description Languages (HDL)
[3] VHSIC (Very-High-Speed Integrated Circuits) Hardware Description Languages
[4] Synthesis
[5] Netlist
[6] Place & route

های عبور سیگنال[1] و سوئیچهای مسیریاب[2]، اتصالات بین تمام عناصر را برقرار می کند. bitstream همچنین مشخصات بلوکها و توابع دیجیتالی که قبلاً در زبان توصیف سخت افزاری تعریف شده بود را تنظیم و اجرا می کند [11]. بعضی ازFPGA ها در زمان روشن شدن، توسط یک حافظه دائم خارجی مانند EEPROM یا حافظه فلش[3] ، bitstream را دریافت می کنند.

از بین چهار مرحله فوق فقط مرحله اول توسط طراح انجام می شود و بقیه مراحل می تواند بهینه شود. برای طراحی بسیاری از سیستمهای پردازش سیگنال دیجیتال با استفاده از زبانهای توصیف سخت افزار، طراح مجبور است کدهای برنامه نویسی را که ممکن است خسته کننده و همراه با خطای زیاد باشد، تایپ کند. اخیراً ابزارهایی وجود دارد که به وسیله آن طراح می تواند بلوک های دیجیتال را در سطح بسیار فشرده به صورت شماتیک به کار ببرد، سیستم خود را طراحی کند و سپس کد HDL متناظر را با ابزارهای نرم افزاری تولید کند. بعضی از این ابزارها دارای بلوک های شبیه سازی نیز می باشد، بنابراین راهکار شبیه سازی سخت افزاری بیشتر مورد استفاده قرار می گیرد. ویژگی مهم شبیه سازی سخت افزاری این است که می توان قبل از مرحله پیاده سازی واقعی، سیستم را مورد تست قرار داد و تا حدودی نتایج را با دنیای واقعی برابر دانست.

3-5- محاسبات ممیز ثابت

در طراحی FPGA، معمولاً از روش ممیز ثابت[4] مکمل دو برای نمایش اعداد استفاده می شود. البته نمایش ممیز شناور[5] نیز کاملاً قابل انجام است اما در کاربردهای با کارآیی بالا، با بکارگیری تعداد بیت محدود و مناسب از نمایش ممیز ثابت استفاده می شود، که این نیز از دیگر ویژگیهای FPGA محسوب می شود؛ یعنی امکان انتخاب مناسب طول کلمه در بخشهای مختلف طراحی به منظور دست یافتن به دقت مورد نیاز.

شکل 3-5 مثالی از نمایش مکمل دو ممیز ثابت را نشان می دهد. طبق شکل، 3 بیت برای قسمت صحیح و 5 بیت برای بخش اعشاری در نظر گرفته شده است.

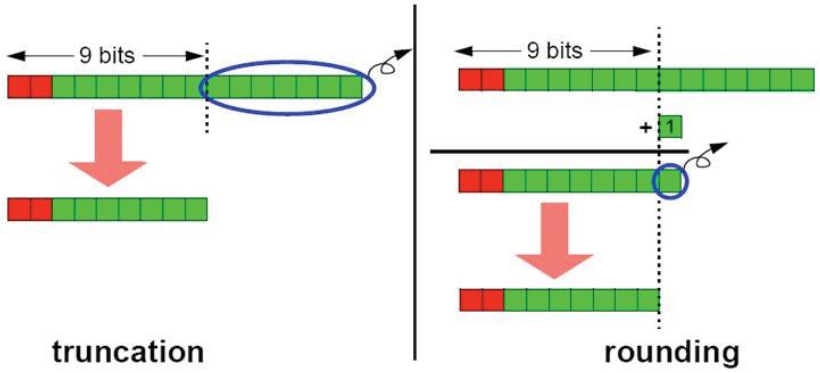
[1] Pass gates
[2] Routing switches
[3] Flash memory
[4] Fixed point
[5] Floating point

digit worth								decimal value
$-(2^2)$	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	
-4	**2**	**1**	**0.5**	**0.25**	**0.125**	**0.0625**	**0.03125**	
0	0	0	0	0	0	0	1	0.03125
0	0	0	0	0	0	1	0	0.0625
1	0	1	0	0	0	0	0	-3.0
1	1	0	0	0	1	1	1	-1.78125
1	1	1	1	1	1	1	1	-0.03125

شکل 3-5: نمایش مکمل دو ممیز ثابت اعداد علامت دار با 3 بیت صحیح و 5 بیت اعشاری

سوالی که در اینجا مطرح می شود این است که در صورتی که طول کلمه با انجام محاسبات پشت سر هم افزایش پیدا کند، چه باید کرد؟

در این حالت به دلیل افزایش طول کلمه، محاسبات مسیرهای بلوک های لاجیک وقت گیرتر می شود و بنابراین ممکن است زمان طولانی ترین مسیر[1] از زمان یک پریود کلاک سیستم بیشتر شود که باعث اختلال می گردد.
راه حلی که برای رفع این مشکل پیشنهاد می شود این است که بعد از انجام هر عملیات، طول کلمه به مقدار مناسب محدود شود. در شکل 3-6 دو روش برای انجام این عمل وجود دارد: قطع کردن[2] و روندکردن[3].



شکل 3-6: قطع کردن و روند کردن در نمایش ممیز ثابت

در روش قطع کردن، تمام بیتهایی که از سمت راست بعد از تعداد بیت های مطلوب در هر مرحله از انجام عملیات اضافه می شود، حذف خواهد شد. اما در روش روند کردن، بیت MSB از بیتهایی که در روش قبل باید حذف شوند،

[1] Combinatorial path
[2] Truncation
[3] Rounding

ابتدا با 1 جمع کرده و سپس عمل قطع کردن مانند روش قبل انجام می شود.
در مباحث احتمالات، عمل های قطع کردن و روند کردن را می توان با منبع نویز سفید که دامنه آن متناسب با تعداد بیتهای حذف شده است، مدل کرد [12].

3-6- محاسبات ممیز شناور

ممیز شناور ابزار بسیار گسترده تری برای نمایش واقعی اعداد فراهم می کند و تمایل به استفاده از آن در کاربردهای محاسبات عددی و خصوصاً در کاربردهای پردازش سیگنال دیجیتال بسیار بیشتر است[7]. در ممیز شناور هدف این است که عدد واقعی را با استفاده از یک علامت، توان[1] و رقم اعشاری[2] مانند شکل 3-7 نشان دهیم. در این شکل، s نشان دهنده علامت، exp نمایانگر توان و Fraction بیان کننده رقم اعشار می باشد.

علامت (s)	توان (exp)	اعشار (Fraction)

شکل 3-7: نمایش ممیز شناور

پر استفاده ترین فرم ممیز شناور، استاندارد IEEE754 برای محاسبات ممیز شناور باینری است، که به طور گسترده ای در زبانهای C و MATLAB استفاده می شود. در فرم 32 بیتی تک دقتی[3]، 23 بیت برای بخش اعشاری، 1 بیت برای علامت و 8 بیت برای مقدار توان علامت دار در نظر گرفته می شود. به صورت جبری برای نمایش یک عدد ممیز شناور از رابطه زیر استفاده می شود [8]:

$$X = (-1)^s \times 1.\text{Fraction} \times 2^{\exp - bias} \qquad (1\text{-}3)$$

که در آن:

$$bias = 2^{\exp - 1} - 1 \qquad (2\text{-}3)$$

بنابراین 8 بیت اختصاص داده شده برای توان، از 2^{-126} تا 2^{127} را شامل می شود. بخش اعشاری معمولاً به دلیل قرار گرفتن نقطه اعشار در سمت راست با ارزش ترین بیت 1، نرمالیزه می شود[13]. پس با فرض قرار گرفتن یک بیت در

[1] Exponent
[2] Mantissa or Fraction
[3] Single precision

سمت چپ نقطه اعشاری، 24 بیت موثر برای بخش اعشاری وجود خواهد داشت.

علامت (s): 1 بیت	توان (exp): 8 بیت	اعشار (Fraction): 23 بیت
1	**10001001**	**10000111010101010000000**

به عنوان مثال فرض کنید می خواهیم عدد 1082/65625- را به فرم ممیز شناور استاندارد IEEE754 نشان دهیم:
چون عدد منفی است پس S=1 . بخش صحیح یعنی عدد 1082 به معادل باینری آن تبدیل می شود:

$1082=(10000111010)_2$

معادل باینری بخش اعشاری یعنی عدد 0/65625 نیز به صورت زیر در می آید:

$0/65625=(0/10101)_2$

بنابراین با ترکیب این دو مقدار باینری، عدد زیر را خواهیم داشت:

$(10000111010/10101)_2$

ممیز اعشاری را به سمت چپ منتقل می کنیم به طوری که فقط یک بیت 1 در سمت چپ آن موجود باشد:

$1/000011101010101\times 210$

برای رسیدن به فرم اعشاری 23 بیتی می بایست تعدادی صفر به جلو عدد به دست آمده اضافه کنیم:

10000111010101010000000

بخش نمایی برابر با 10 است و با جمع با مقدار بایاس در استاندارد IEEE754 (یعنی 127) عدد 137 به دست می آید که برابر با عدد باینری 10001001 می باشد. پس در نهایت عدد مورد نظر به صورت زیر خواهد شد:

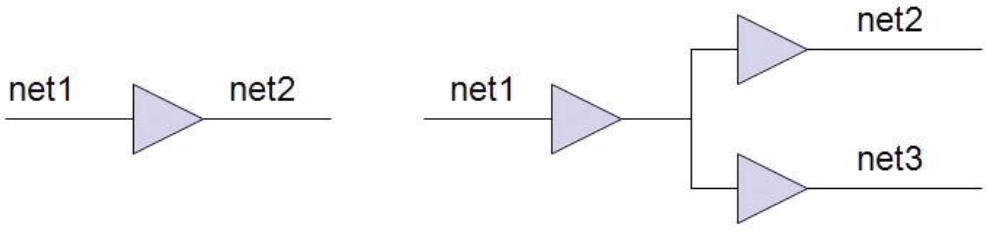
مهمترین خصوصیت محاسبات ممیز شناور این است که بخش نمایی هر عدد را می توان برای رسیدن به بیشترین میزان دقت، تنظیم کرد. بنابراین چنین سیستمی دقت بسیار خوبی برای نمایش اعداد بزرگ و اعداد کوچک خواهد داشت [13]، اما به هر حال پیاده سازی این سیستم چه بر روی سخت افزار و چه به صورت نرم افزاری مشکلاتی را به همراه خواهد داشت. مثلاً در پیاده سازی سخت افزاری ممیز شناور،

در هر مرحله از عملیات جبری، تعداد بیتها و مکان ممیز اعشاری تغییر خواهد کرد بنابراین باعث افزایش فضای سیلیکونی، زمان تاخیر گیتها و مصرف توان و کاهش سرعت می شود و در بخش پیاده سازی نرم افزاری نیز باعث کاهش سرعت پردازش خواهد شد. بنابراین معمولاً در پیاده سازی سخت افزاری بر روی FPGA از محاسبات ممیز ثابت استفاده می شود، هر چند که استفاده از ممیز شناور نیز امکان پذیر است.

3-7- تکنی‌کهای بهی‌نه سازی در طراحی سخت افزاری

3-7-1- تکنی‌ک بافرکردن[1]

همانطور که قبلاً بحث شد ابزارهای مکان یابی و سیم کشی با آنالیز کردن محدودیت های زمانی در طراحی، مکان یابی و سیم کشی مناسب را انجام می دهند. اما گاهی اوقات این ابزارها بهینه ترین طراحی را اجرا نمی کنند، به عنوان مثال اگر تاخیر بلوک های دیجیتال بیشتر از یک پریود کلاک سیستم باشد، ابزارهای مکان یابی و سیم کشی نمی توانند کاری انجام دهند.
در طراحی های پیشرفته 90% تاخیرها به دلیل سیم کشی و 10% به دلیل بلوک های دیجیتال است. بسیاری از ابزارهای سنتز به طور اتوماتیک بافرهایی را در مسیر سیم کشی های طولانی قرار می دهند تا جریان بیشتری برای تغذیه بقیه مسیر تامین کند و از اثرات خازنی سیم ها کاسته شود، بنابراین تاخیر سیم کشی کمتر می شود. به این عمل بافر کردن گفته می شود. شکل 3-8 تکنیک بافرکردن را به خوبی نشان می دهد.

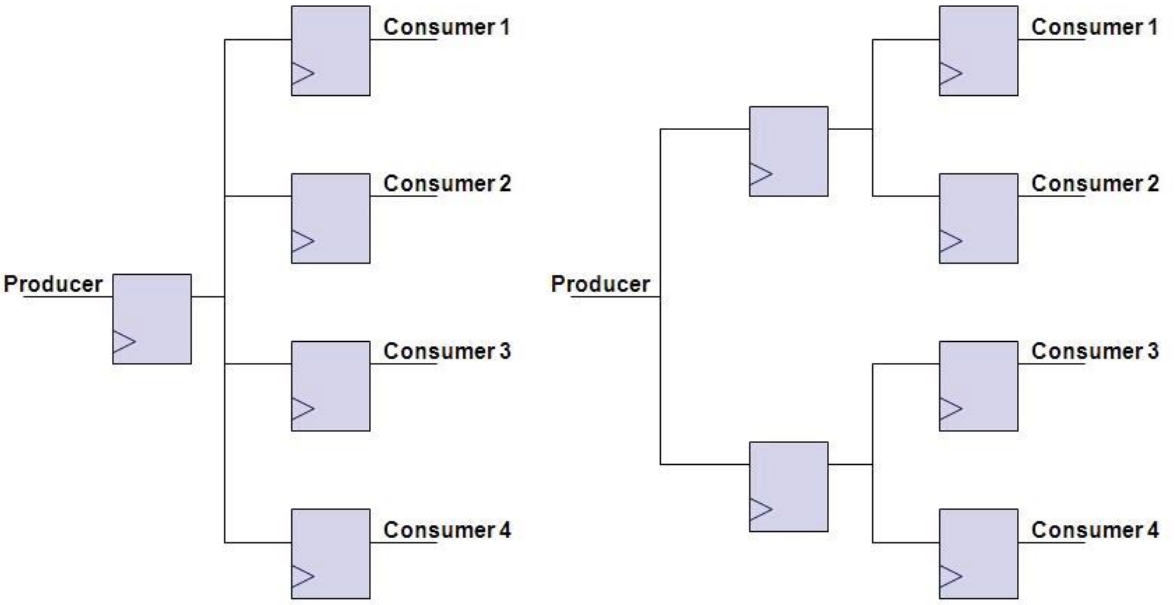


شکل 3-8: اضافه کردن اتوماتیک بافر در FPGA: سمت چپ: قبل از بافرکردن، سمت چپ: بعد از بافرکردن

[1] Buffering

3-7-2- تکنیک تکرارکردن رجی‌سترها[1]

تکرار رجیسترها معمولاً یا به وسیله ابزارهای سنتز انجام می شود و یا در سطح HDL با دست (توسط طراح) قابل تعریف است. با توجه به شکل 3-9 خروجی فلیپ فلاپ سمت چپ مستقیماً به چهار فلیپ فلاپ دیگر وارد می شود که طول زیادی را در FPGA پوشش می دهد.



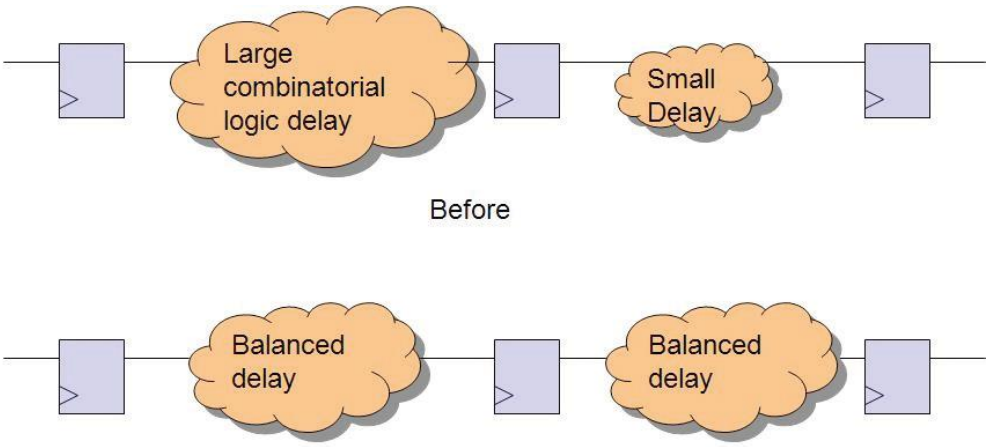
شکل 3-9: تکرار رجیسترها به طور اتوماتیک: سمت چپ: قبل از تکرار، سمت راست: بعد از تکرار

بعد از انجام عملیات تکرار رجیسترها، هر کدام از خروجیهای فلیپ فلاپ ها به دو فلیپ فلاپ دیگر وارد می شود که از نظر طول و قیدهای زمانی نسبت به حالت قبل بهتر عمل خواهد کرد. البته در اینجا تاوان اضافه شدن یک رجیستر به کل مدار را هم باید در نظر گرفت.

3-7-3- تکنیک توازن رجی‌سترها[2]

همانطور که قبلا نیز گفته شد، اگر تاخیر های طولانی در مدارهای دیجیتال بین فلیپ فلاپ ها اتفاق بیفتد، ابزارهای مکان یابی و سیم کشی هیچ کاری جهت بهبود کارآیی سیستم نمی توانند انجام دهند. Retiming یا توازن رجیسترها، تکنیکی است که در این موارد می تواند مورد استفاده قرار گیرد. شکل 3-10 نحوه عملکرد این تکنیک را نشان می دهد.

[1] Replication of registers
[2] Register balancing

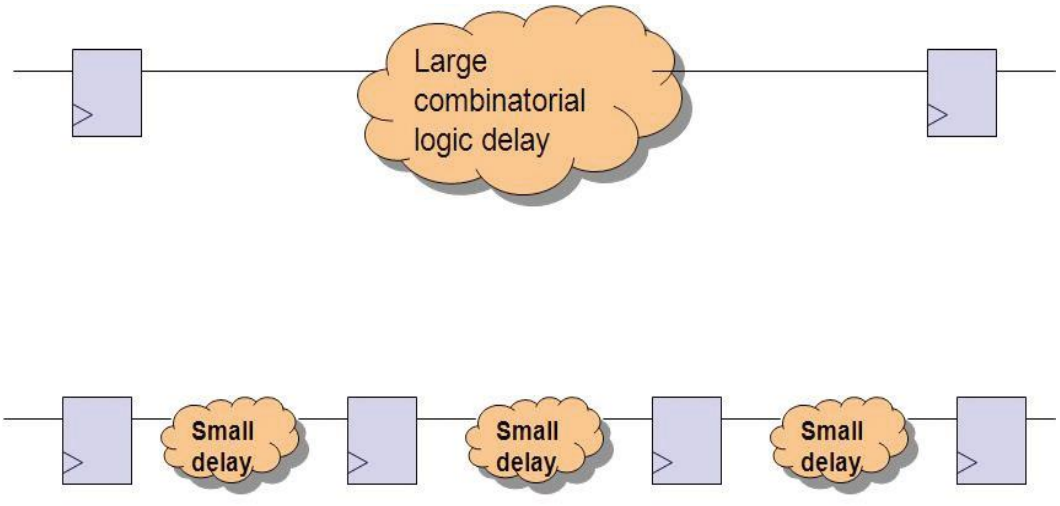


شکل 3-10: تکنیک توازن رجیسترها: بالا: قبل از توازن، پایین: بعد از توازن

با توجه به شکل 3-10، تعدادی از عناصر لاجیک درون مدار ترکیبی که تاخیر بیشتری دارد، به مدار ترکیبی با تاخیر کمتر بعد از فلیپ فلاپ انتقال داده می شود. بنابراین ماکزیمم تاخیر در هر کدام از مدارهای ترکیبی به یک اندازه متعادل می شود و در نتیجه ماکزیمم فرکانس کار سیستم افزایش خواهد یافت.

3-7-4- تکنیک Pipelining

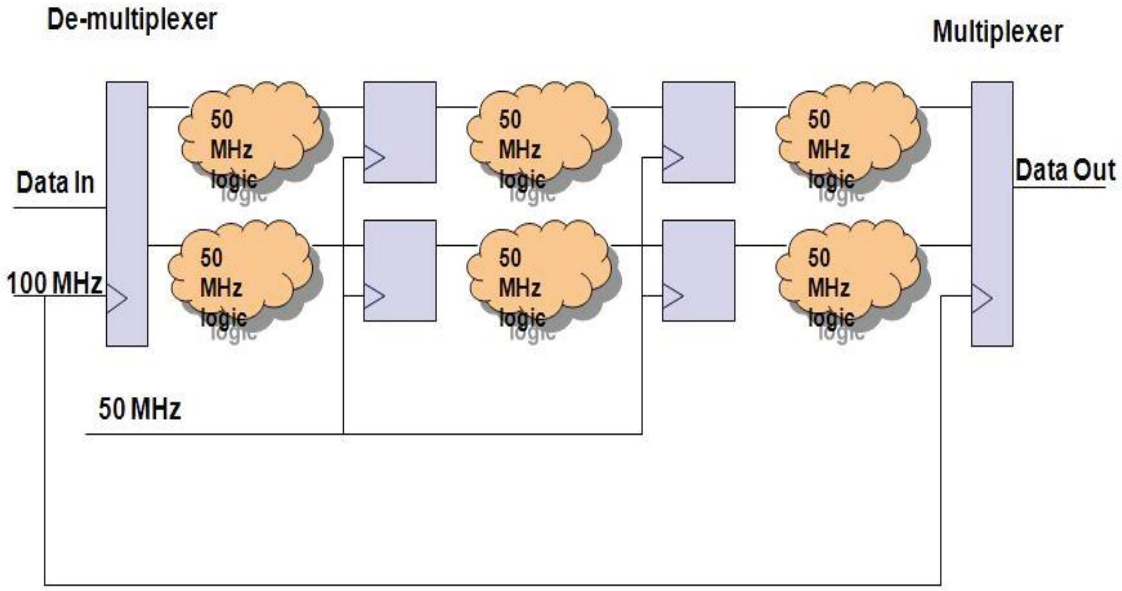
اگر روش Retiming قابل اجرا نباشد، یعنی به علت محدودیت های طراحی نتوان عناصر لاجیک را به مرحله بعد انتقال داد، از Pipelining استفاده می کنیم. در این تکنیک مدارهای دیجیتال با تاخیر طولانی به مدارهای با تاخیر کمتر شکسته می شود، به طوریکه در هر مرحله یک فلیپ فلاپ به مدار اضافه می شود. (شکل 3-11)



شکل 3-11: تکنیک Pipelining: بالا: قبل، پایین: بعد از Pipelining

3–7–5– تکنیک زمان چندگانه[1]

در روش زمان چندگانه، طبق شکل 3-12 ، به جای اینکه دیتا را از یک مسیر مثلاً 50MHz ای عبور دهیم، آن را به دو قسمت تقسیم کرده و توسط DeMux از دو مسیر موازی به صورت همزمان عبور می دهیم و در انتها نتایج را با استفاده از Mux با هم ترکیب می کنیم، بنابراین فرکانس کاری سیستم دو برابر، یعنی 100MHz، می شود.



شکل 3-12: تکنیک زمان چندگانه همراه با تکرار رجیسترها

3–8– الگوریتم CORDIC

در میان الگوریتمهای مناسب طراحی های سخت افزاری، کلاسی از راه حل های تکراری برای حل توابع مثلثاتی و غیر جبری به نام CORDIC وجود دارد که فقط از جمع کننده ها و شیفت دهنده ها برای انجام محاسبات استفاده می کند. الگوریتم CORDIC یک روش کارآمد مبتنی بر چرخش دایره ای برای حل توابع مثلثاتی sin، cos، tan، sinh، cosh، tanh بدون استفاده از ضرب کننده ها است که مناسب پیاده سازی روی سخت افزار طراحی شده است [14]. الگوریتم CORDIC اولین بار در سال 1959 توسط ولدر[15] به عنوان یک راه حل دیجیتال برای ساخت کامپیوتر مخصوص ناوبری برای استفاده در هواپیما ارائه شد. این الگوریتم ابتدا برای حل توابع مثلثاتی معرفی شد و بعداً توسط والتر[16]

[1] Time multiplexing

برای حل رنج وسیعی از توابع شامل توابع هیپربولیک و ریشه دوم عمومیت یافت. از الگوریتم CORDIC در محاسبه توابعی چون تبدیل فوریه ، تبدیل کسینوسی، تبدیل هارتلی، تبدیل z ، فیلترهای دیجیتال و حل سیستمهای خطی نیز استفاده شده است.

در کل سه روش عمده برای پیاده سازی توابع مثلثاتی وجود دارد:

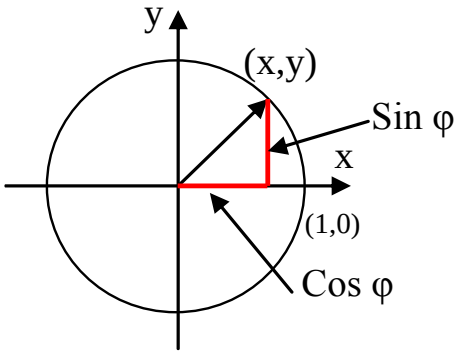
1- با استفاده از جدول مراجعه[1]
2- با استفاده از تقریبهای چندجمله ای
3- با استفاده از الگوریتم CORDIC

الگوریتم CORDIC در مقایسه با سایر روشها بسیار مفیدتر عمل خواهد کرد، هر گاه:

- بخواهیم توابع مثلثاتی را بر روی سخت افزاری پیاده سازی کنیم که در آن ضرب کننده سخت افزاری وجود نداشته باشد. (مانند میکروکنترلرها)
- بخواهیم از فضای سخت افزاری کمتری برای پیاده سازی توابع مثلثاتی استفاده کنیم، که در این حالت FPGA ها با توجه به خصوصیات ذاتی آنها بهتر گزینه است.

3-9- اصول CORDIC

فرض کنید نقطه (0و1) روی محورهای مختصات به اندازه φ درجه در جهت خلاف حرکت عقربه ساعت دوران کند (شکل 3-13).



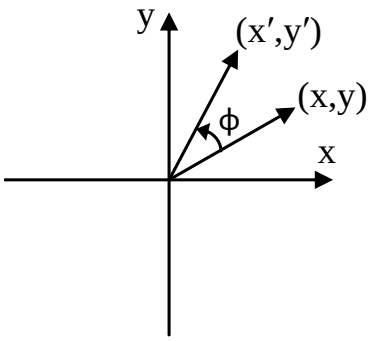
شکل 3-13: دوران بردار (0و1) به اندازه φ درجه

در این صورت مختصات بردار دوران یافته جدید (x,y)، به صورت زیر به دست خواهد آمد:

$$x = \cos(\varphi)$$
$$y = \sin(\varphi) \qquad (3\text{-}3)$$

[1] Look-up table

حال فرض کنید بردار (x,y) به دست آمده در مرحله قبل را این بار به اندازه Φ درجه دوران دهیم (شکل 3-14).



شکل 3-14: دورن بردار (x,y) به اندازه Φ درجه

مختصات بردار جدید حاصل از این دوران برای هر (x,y) به صورت زیر خواهد بود:

$$\begin{aligned} x' &= x\cos(\phi) - y\sin(\phi) \\ y' &= y\cos(\phi) + x\sin(\phi) \end{aligned} \qquad (4\text{-}3)$$

در واقع این روش، روش سنتی پیاده سازی دوران برداری دوبعدی است، که در آن (x,y) مختصات ابتدایی و (x′,y′) مختصات نهایی بردار مورد نظر است [17]. برای پیاده سازی سخت افزاری این معادلات به چهار ضرب کننده و دو جمع/ضرب کننده و دسترسی به جدول مراجعه، برای محاسبه ضرایب مثلثاتی نیاز است. الگوریتم CORDIC دوران دوبعدی را با بکارگیری معادلات بازگشتی و استفاده از فقط عملیات شیفت و جمع محاسبه می کند.

برای شروع کار، معادله (3-4) را می توان به صورت زیر باز نویسی کرد:

$$\begin{aligned} x' &= \cos(\phi)[x - y\tan(\phi)] \\ y' &= \cos(\phi)[y + x\tan(\phi)] \end{aligned} \qquad (5\text{-}3)$$

تا اینجای کار هیچ چیزی ساده نشده است. اما اگر زوایای دوران محدود شود به نحوی که $\tan(\phi) = \pm 2^{-i}$ ، آنگاه پیاده سازی آن با استفاده از جمله تانژانت به یک عملیات شیفت ساده، کاهش پیدا می کند. این اصول کار الگوریتم CORDIC است، یعنی دوران برداری به صورت سلسله ای از دوران های پی در پی کوچک تر ، هر کدام با زاویه $\tan^{-1}(2^{-i})$ ، به فرم زیر در می آید:

$$\begin{aligned} x_{i+1} &= K_i[x_i - y_i d_i 2^{-i}] \\ y_{i+1} &= K_i[y_i - x_i d_i 2^{-i}] \end{aligned} \qquad (6\text{-}3)$$

که در آن

$$K_i = \cos(\tan^{-1} 2^{-i}) = 1/\sqrt{1+2^{-2i}}$$
$$d_i = \pm 1 \qquad (7\text{-}3)$$

در معادله فوق اگر ثابت K حذف شود، الگوریتم دوران برداری فقط شامل شیف و جمع خواهد شد. ضرب K_i ها را می توان در جای دیگری از سیستم انجام داد یا اینکه آن را به عنوان بخشی از بهره[1] کل سیستم در نظر گرفت. ضرب K_i ها را به صورت زیر تعریف می کنیم:

$$K = \prod_{i=0}^{n} K_i \qquad (8\text{-}3)$$

اگر تعداد تکرارها به سمت بینهایت میل کند، مقدار K تقریباً برابر 0/6073 خواهد شد. بنابراین می توان این عدد ثابت را فقط یک بار در مرحله آخر در بهره سیستم ضرب کرد.

الگوریتم CORDIC در دو حالت کار می کند: در حالت اول که مد دوران[2] نام دارد، بردار ورودی توسط یک زاویه مشخص دوران می کند. در حالت دوم که مد برداری[3] نام دارد، بردار ورودی به سمت محور x دوران می کند و زاویه مورد نیاز برای این دوران ذخیره می شود.

3-9-1- مد دوران

برای مد دوران روابط الگوریتم CORDIC به صورت زیر در می آید:

$$\begin{aligned} x_{i+1} &= x_i - y_i d_i 2^{-i} \\ y_{i+1} &= y_i + x_i d_i 2^{-i} \\ z_{i+1} &= z_i - d_i \tan^{-1}(2^{-i}) \end{aligned} \quad , \quad \begin{cases} x_0 = x_{in} \\ y_0 = y_{in} \\ z_0 = z_{in} \end{cases} \qquad (9\text{-}3)$$

که در آن

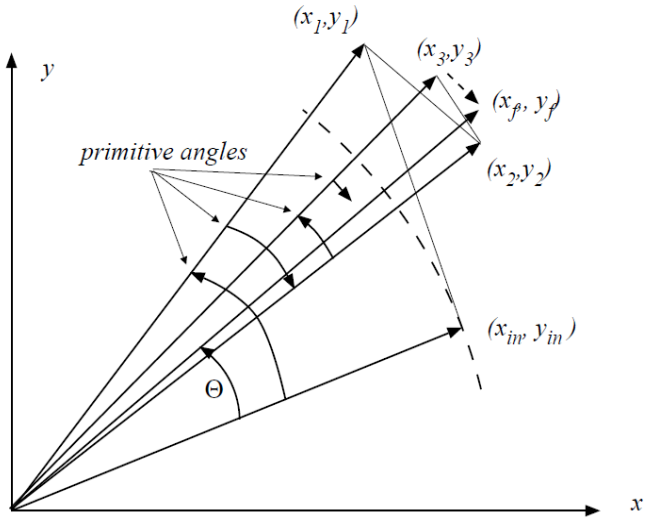
$$d_i = \begin{cases} -1 & z \text{ هر گاه } 0 \\ +1 & \text{سایر} \end{cases} \qquad (10\text{-}3)$$

اگر روابط (3-9)، n بار تکرار شود، مقادیر نهایی زیر را نتیجه می دهد:

[1] Gain
[2] Rotation mode
[3] Vectoring mode

$$
\begin{aligned}
x_f &= A_n[x_0\cos(z_0) - y_0\sin(z_0)] \\
y_f &= A_n[y_0\cos(z_0) + x_0\sin(z_0)] \\
z_f &= 0 \\
A_n &= \prod_n \sqrt{1+2^{-2i}}
\end{aligned}
\qquad (11\text{-}3)
$$

این روابط، شکل 15-3 را نتیجه می دهد که الگوریتم CORDIC در مد دورانی می باشد.



شکل 15-3: چرخش بردار (x_{in},y_{in}) با استفاده از دورانهای کوچک و رسیدن به بردار (x_f,y_f)

3-9-2- مد برداری

در مد برداری، معادلات الگوریتم CORDIC به همان صورت معادله (3-9) است، اما شرایط تغییر علامت تابع تصمیم d_i به صورت زیر تغییر می کند:

$$
d_i = \begin{cases} +1 & y \text{ بردار } i \text{ گام } 0 \\ -1 & \text{سایر بردار} \end{cases}
\qquad (12\text{-}3)
$$

بنابراین مقادیر نهایی به صورت زیر به دست می آید:

$$
\begin{aligned}
x_f &= A_n\sqrt{x_0^2 + y_0^2} \\
y_f &= 0 \\
z_f &= z_0 + \tan^{-1}(y_0 / x_0) \\
A_n &= \prod_n \sqrt{1+2^{-2i}}
\end{aligned}
\qquad (13\text{-}3)
$$

3–10– محاسبه سی‌نوس و کسی‌نوس

با استفاده از مد دورانی الگوریتم CORDIC به راحتی می توان مقادیر سینوس و کسینوس یک زاویه را محاسبه کرد. اگر در روابط مد دورانی مولفه y بردار ورودی صفر باشد، خواهیم داشت:

$$\begin{aligned} x_f &= A_n.x_0 \cos(z_0) \\ y_f &= A_n.x_0 \sin(z_0) \end{aligned} \qquad (14\text{-}3)$$

در این حالت اگر x_0 مساوی با $1/A_n$ قرار داده شود، مقادیر نهایی x و y به ترتیب برابر با کسینوس و سینوس زاویه ورودی می شود.

3–11– محاسبه تانژانت معکوس

اگر در مد برداری الگوریتم CORDIC ، مقدار ابتدایی x و z به ترتیب برابر 1 و 0 در نظر گرفته شود (یعنی $x_0=1$ و $z_0=0$) ، آنگاه مقدار نهایی z مستقیماً برابر با تانژانت معکوس مقدار ابتدایی y خواهد بود. یعنی $z_f=\tan^{-1}(y_0)$.

3–12– کاربردهای مختلف الگوریتم CORDIC

با تکنیکهایی شبیه دو بخش قبل، الگوریتم CORDIC می تواند به طور مستقیم توابع sin، cos، sinh، cosh، $\tan^{-1}$ ، $\tanh^{-1}$، $\tan^{-1}(y/x)$، y+x.z، $(x^2+y^2)^{1/2}$، $(x^2-y^2)^{1/2}$ و همچنین عملیات ضرب و تقسیم را محاسبه کند.

فصل 4– مروری بر مطالعات و تحقیقات گذشته

4–1– مقدمه

تحقیقات چندان زیادی در زمینه اجرای الگوریتم های ماشین بردار پشتیبان برای کاربردهای بلادرنگ بر روی FPGA انجام نگرفته است. به دلیل پیچیدگی روابط ماشین بردار پشتیبان در فاز آموزش، اخیراً محققان به پیاده سازی سخت افزاری فاز تست آن روی آورده اند.

دیوید آنگوئیتا [18] یک ساختار سخت افزاری کاملاً دیجیتال برای اجرای فاز آموزش و تست ماشین بردار پشتیبان با استفاده از توابع کرنل خطی و RBF[1] ارائه کرده است. ساختار ارائه شده علاوه بر مصرف زیاد فضای سخت افزار، سرعت عملکرد قابل توجهی نداشت. ماکزیمم فرکانس به دست آمده توسط آنگوئیتا 35/3MHz بود. فیصل خان [19] از محاسبات ممیز ثابت اجتناب کرده و از سیستم عددی لگاریتمی برای پیاده سازی فاز تست ماشین بردار پشتیبان خطی استفاده کرده است. هر چند که ساختار ارائه شده برای سخت افزار مناسب است، اما همواره ممکن است مشکلاتی در تبدیل اعداد واقعی به معادل لگاریتمی آنها وجود داشته باشد.

در این فصل به بررسی تعدادی از تحقیقاتی که در سالهای اخیر در زمینه پیاده سازی فاز تست ماشین بردار پشتیبان بر روی FPGAها، برای کاربردهای مختلف، صورت گرفته است می پردازیم.

4–2– پی‌اده سازی ماشی‌ن بردار پشتی‌بان کرنل خطی بر روی FPGA

پینارامیرز[20] کلاسه بند ماشین بردار پشتیبان خطی را بر اساس رابطه زیر بر روی Vitex-II XC2V1000-4 پیاده سازی کرد.

$$d(\mathbf{x}) = \sum_{i=1}^{SV} (\alpha_i y_i \mathbf{x}_i^T \mathbf{x}) + b \qquad (1\text{-}4)$$

ساختار سخت افزاری ارائه شده به پنج مرحله اصلی زیر تقسیم بندی می شود:

1- محاسبه ضرب برداری $P = \mathbf{X}_i^T \mathbf{X}$

2- محاسبه ضرب عددی $S = \alpha_i y_i$

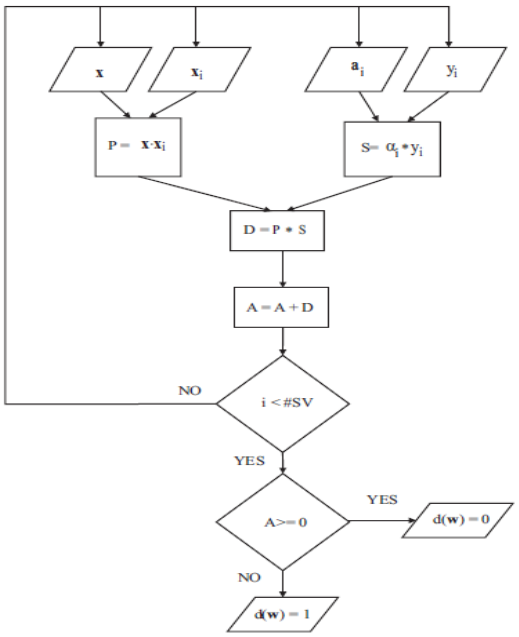
[1] Radial Basis Function

3- محاسبه ضرب عددی $D=P\times S$

4- انباره $A=A+D$

5- تصمیم گیری

شکل 4-1 الگوریتم پیاده سازی مراحل فوق را نشان می دهد.



شکل 4-1: الگوریتم پیاده سازی کلاسه بند ماشین بردار پشتیبان خطی [20]

به منظور ارزیابی عملکرد ساختار ارائه شده، دو کلاسه بند ماشین بردار پشتیبان خطی طراحی شده است. اولی (SVM1) برای جداسازی داده های گوسی مصنوعی 6 بعدی با میانگین های متفاوت و واریانس های مساوی، و دومی (SVM2) برای کلاسه بندی تصاویر MRI سه بعدی استفاده شده است. هر دو کلاسه بند برای جداسازی دو کلاس مورد استفاده قرار گرفته است. در هر دو مورد، از مدل Libsvm [21] برای اجرای فاز آموزش ماشین بردار پشتیبان در محیط نرم افزاری MATLAB به صورت خارج خط برای 1000 داده آموزشی استفاده شده است. بعد از آموزش، ماشین اول 6 بردار پشتیبان و ماشین دوم 78 بردار پشتیبان تولید کرد.

100 داده برای تست کلاسه بندهای مورد نظر در محیط نرم افزاری MATLAB و بر روی FPGA مورد استفاده قرار گرفته است. در محیط نرم افزاری MATLAB با محاسبات ممیز شناور هر دو کلاسه بند دقت عملکرد 100% را ارائه کرد. برای ارزیابی محاسبات ممیز ثابت، هر دو کلاسه بند ماشین بردار پشتیبان خطی بر روی FPGA مدل Virtex-II- XC2V1000-4

پیاده سازی شد. جدول 4-1 و جدول 4-2 ماتریس مقایسه عملکرد دقت سیستم برای محاسبات ممیز شناور و ثابت را به ترتیب برای کلاسه بندهای SVM1 و SVM2 نشان می دهد.

جدول 4-1: ماتریس مقایسه عملکرد کلاسه بند SVM1 در محیط نرم افزاری MATLAB و بر روی VirtexII

VirtexII		MATLAB		
C_{-1}	C_1	C_{-1}	C_1	
2	49	0	50	C_1
48	1	50	0	C_{-1}

جدول 4-2: ماتریس مقایسه عملکرد کلاسه بند SVM2 در محیط نرم افزاری MATLAB و بر روی VirtexII

VirtexII		MATLAB		
C_{-1}	C_1	C_{-1}	C_1	
4	49	0	50	C_1
46	1	50	0	C_{-1}

که در آن C_1 نشان دهنده کلاس 1 و C_{-1} نشان دهنده کلاس 2 است.

همانطور که ملاحظه می شود کلاسه بند SVM1 با مجموع 3 خطا عملکرد 97% ، و کلاسه بند SVM2 با مجموع 5 خطا عملکرد 95% را به دست آورده است.

به منظور مقایسه زمان پردازش، کلاسه بند SVM2 در محیط C بر روی یک پردازشگر AMD,Athlon550MHz اجرا شد. زمان اجرای آن 59/5 ثانیه طول کشید، که 1/84 برابر سریعتر از FPGA بود، در حالیکه ماکزیمم فرکانس کاری FPGA برابر 50 مگاهرتز یعنی یک یازدهم فرکانس AMD بود.

جدول 4-3 نیز منابع سخت افزاری که برای اجرای کلاسه بند SVM2 بر روی VirtexII-XC2V1000-4 مصرف شده است را نشان می دهد.

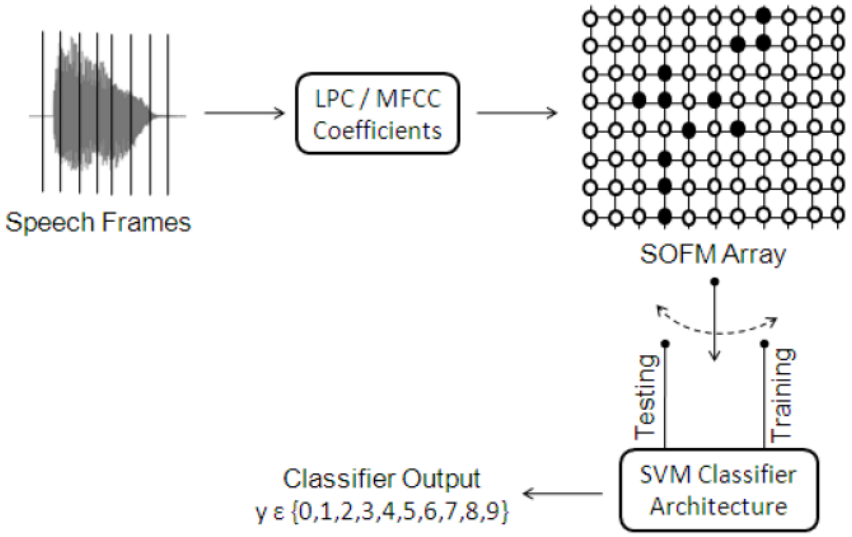
جدول 4-3: منابع سخت افزاری استفاده شده برای پیاده سازی SVM2 بر روی VirtexII

منبع سخت افزاری	تعداد موجود	درصد استفاده
Slice Flip Flops	908	8%
Occupied Slices	751	14%
Shift registers	3	-
Bonded IOBs	9	2%

-	2	IOB Flip Flops
30%	12	Block RAMs
15%	6	MULT18X18s
6%	1	GCLKs

4-3- پی‌اده سازی ماشی‌ن بردار پشتی‌بان بر روی FPGA بر اساس سی‌ستم تشخی‌ص ارقام مجزا

مانیکاندان[22] یک ماشین بردار پشتیبان خطی چند کلاسه برای تشخیص صوتی رقمهای 0 تا 9 ، از مجموعه پایگاه داده TI46 [23]، بر روی FPGA از خانواده Altera Cyclone II مدل EP2C35F672C6 پیاده سازی کرده است. ویژگیهای MFCC[1] و LPC[2] از هر کدام از نمونه های صوتی استخراج شد. به دلیل بزرگ بودن اندازه ابعاد بردار ویژگی در سیستمهای صوتی تشخیص ارقام، و همچنین برای یکسان کردن ابعاد این ویژگیها برای ورود به کلاسه بند ماشین بردار پشتیبان، پس از استخراج ویژگی های LPC و MFCC ، از روش SOFM[3] نیز استفاده شده است. بلوک دیاگرام کلی سیستم مورد نظر به صورت شکل 4-2 است.



شکل 4-2: بلوک دیاگرام کلی سیستم تشخیص صوتی ارقام 0 تا 9

براي ارزيابي عملكرد نرم افزاري سيستم، ابتدا ماشين بردار پشتيبان مورد نظر در محيط نرم افزاری MATLAB اجرا شده است. درستي كلاسه بند موردنظر با اندازه هاي مختلف SOFM مورد بررسي قرار گرفته است، كه نتايج آن در جدول 4-4 قابل مشاهده است.

[1] Mel Frequency Cepstral Coefficients
[2] Linear Predictive Coefficients
[3] Self Organized Feature Mapping

جدول 4-4: عملکرد کلاسه بند تشخیص ارقام مجزا در MATLAB با تابع کرنل خطی و اندازه های مختلف SOFM

ویژگی ←	LPC		MFCC	
اندازه SOFM ↓	صحت کلاسه بند	تعداد بردارهای پشتیبان	صحت کلاسه بند	تعداد بردارهای پشتیبان
12×12	85%	627	100%	602
18×18	97%	692	100%	691
24×24	94%	739	99%	747

جدول 4-4 نشان می دهد که با انتخاب ویژگي MFCC و اندازه 12×12 براي SOFM، هم صحت کلاسه بند 100درصد مي شود و هم تعداد بردارهاي پشتیبان کاهش مي یابد که خود باعث کاهش تعداد عملیات ضرب و جمع در FPGA خواهد شد.

در جدول 4-5 با انتخاب اندازه 12×12 براي SOFM، عملکرد سیستم با توابع کرنل مختلف و همچنین تابع Wavelet مورد بررسي قرار گرفته است. طبق جدول با انتخاب ویژگي MFCC در همه موارد عملکرد سیستم درستي 100درصد را نشان مي دهد، اما انتخاب تابع کرنل خطي براي پیاده سازي بر روي FPGA مناسب تر به نظر مي رسد، چرا که هم ساختار آن ساده است و هم تعداد بردارهاي پشتیباني که از مرحله آموزش تولید کرده است از سایر روشها کمتر است که این خود باعث کاهش حجم محاسبات در FPGA خواهد شد.

جدول 4-5: عملکرد کلاسه بند تشخیص ارقام مجزا در MATLAB با توابع کرنل مختلف و اندازه های 12×12 برای SOFM

ویژگی ←	**LPC**		**MFCC**	
تابع کرنل ↓	صحت کلاسه بند	تعداد بردارهای پشتیبان	صحت کلاسه بند	تعداد بردارهای پشتیبان
خطی	85%	627	100%	602
چندجمله ای	87%	759	100%	747
RBF	90%	677	100%	611
Sigmoid	90%	653	100%	669
Wavelet	90%	703	100%	682

بنابراین با پیش آنالیز سیستم مورد نظر با استفاده از MATLAB مي توان به این نتیجه رسید که با انتخاب MFCC براي استخراج ویژگي، و اندازه 12×12 براي SOFM، و استفاده از تابع کرنل خطي مي توان ماشین بردار پشتیبان را بر روي FPGA پیاده سازي کرد.

برای پیاده سازی کلاسه بند خطی تشخیص ارقام صوتی بر روی FPGA، ابتدا فقط از عناصر منطقی با فرمت ممیز شناور 1,6,11 (1بیت برای علامت، 6 بیت برای توان، 5 بیت برای اعشار) استفاده شده است. صحت کلاسه بند مورد نظر در این حالت 100% گزارش شده است، اما تعداد عناصر منطقی مورد استفاده در FPGA زیاد می باشد. بنابراین از فرمت ممیز ثابت نیز استفاده شده است. تعداد عناصر منطقی در حالت ممیز ثابت با ضریب 3/29 کمتر شده است.

با افزایش طول کلمات، تعداد عناصر منطقی مورد نیاز برای پیاده سازی FPGA بسیار زیاد خواهد شد و نیاز به راه حل بهتری جهت محدود کردن تعداد آنها می باشد. با استفاده از پردازنده Nios II Soft-Core با محاسبات ممیز شناور، تعداد عناصر منطقی لازم برای کلاسه بندی با ضریب 25 کاهش یافته است.

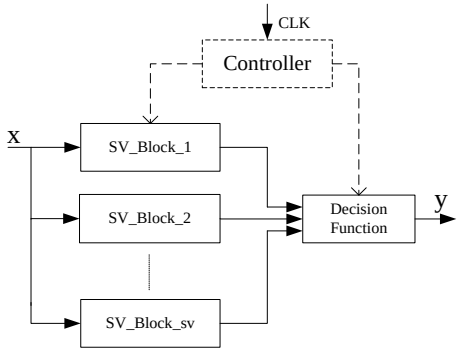
4-4- پی‌اده سازی ماشی‌ن بردار پشتی‌بان بر روی FPGA برای شناسای‌ی اشی‌اء 3 بعدی

مارتا روئیز[24] یک ساختار موازی برای پیاده سازی فاز کلاسه بند ماشین بردار پشتیبان غیرخطی براي کلاسه بندی دو کلاس در ابعاد بسیار زیاد برای تشخیص اشیاء مختلف از پایگاه داده COIL [25] بر روی FPGA طراحی کرد. از یک نوع تابع کرنل خاص به نام "سخت افزار پسند"[1] که رابطه آن به صورت زیر می باشد، استفاده شده است:

$$K\left(\mathbf{x}_i,\mathbf{x}\right)=2^{-\gamma\left\|\mathbf{x}_i-\mathbf{x}\right\|_1} \qquad (2\text{-}4)$$

بلوک دیاگرام کلی ساختار سخت افزاری طراحی شده به صورت شکل 4-3 است.

[1] Hardware friendly



شکل 4-3: بلوک دیاگرام کلی ساختار سخت افزاری کلاسه بند ماشین بردار پشتیبان

محاسبه قسمت $\alpha_i y_i K(\mathbf{x}_i,\mathbf{x})$ از رابطه تابع تصمیم ماشین بردار پشتیبان (معادله (2-27))، در بلوکهای SV_Block به صورت موازی انجام شده، که این خود افزایش سرعت کار سیستم را درپی داشته است.

برای پیاده سازی تابع کرنل سخت افزار پسند، مارتا روئیز از الگوریتم CORDIC استفاده کرده است. با بهره مندی از الگوریتم CORDIC، تابع کرنل بدون بکارگیری ضرب کننده ها و فقط با استفاده از شیفت دهنده ها و جمع کننده ها پیاده سازی می شود. قبل از شروع پارامترهای آموزش ماشین بردار پشتیبان بین 0 و $1-2^{-k}$ ، که k تعداد بیتهای لازم برای نمایش پارامترهای ماشین بردار پشتیبان است، مقیاس بندی شده است.

برای محاسبه تابع کرنل از روابط تکرارکننده زیر استفاده شده است:

$$B_{j+1} = B_j\left(1+d_j 2^{-j}\right) \qquad (3\text{-}4)$$

$$E_{j+1} = E_j - log_2\left(1+d_j 2^{-j}\right) \qquad (4\text{-}4)$$

که در آن d_j={-1,0} طوری انتخاب می شود که $|E_j+1| \le |E_j|$.
و مقادیر شروع الگوریتم به صورت زیر است:

$$B_1 = \beta_i \qquad (5\text{-}4)$$

$$E_1 = -\gamma\left\|\mathbf{x_i} - \mathbf{x}\right\|_1 \qquad (6\text{-}4)$$

که در آن

$$\beta_i = \alpha_i \frac{1-2^k}{C} \qquad (7\text{-}4)$$

بعد از k مرحله، پاسخ نهایی سیستم به صورت زیر به دست می آید:

$$B_{k+1} = B_1 2^{E_1} = \alpha_i 2^{-\gamma\|\mathbf{x}_i - \mathbf{x}\|_1} = \alpha_i K(\mathbf{x}_i, \mathbf{x}) \qquad (8\text{-}4)$$

ماشین بردار پشتیبان غیرخطی طراحی شده ابتدا برای جداسازی دو کلاس از پایگاه داده Iris بر روي FPGA مدل Altera Cyclone II-EP2C20 پیاده سازي شده است. پایگاه داده Iris شامل 120 داده دو بعدي است که به صورت غیرخطي جداپذیر هستند. فاز آموزش ماشین بردار پشتیبان با استفاده از جعبه ابزار bioinformatics نرم افزار Matlab انجام شده است. ساختار سخت افزاري فاز تست ماشین بردار پشتیبان با استفاده از نرم افزار Atera Quartus II با استفاده از طراحي شماتیك و زبان VHDL پیاده سازي شده است. براي نمایش پارامترهاي ماشین بردار پشتیبان از طول کلمه 8 بیت استفاده شده است. خروجي کلاسه بند مورد نظر بعد از 10 سیکل کاري به دست آمده که 8 سیکل توسط الگوریتم CORDIC صرف شده است. ماکزیمم فرکانس کاری سیستم در این حالت 33MHz به دست آمده است، یعنی کلاسه بندی هر داده جدید در عرض $0/3\mu s$ انجام می شود. با انتخاب C=10 و γ=2 و تعداد 10 براي بردارهاي پشتیبان، خطاي 10% در مقایسه با خطاي 5% نرم افزاري به دست آمده است، که این اختلاف خطا به دلیل محدود بودن طول کلمه است. در این ساختار افزایش تعداد بردارهاي پشتیبان تاثیري در سرعت سیستم ندارد، چون بردارهاي پشتیبان به صورت موازي در سیستم قرار گرفته اند.

در ساختار ارائه شده هر بلوک SV_Block حدود 180 المان منطقی را مصرف می کند که با تعداد 10 بردار پشتیبان حدود 11% از کل المانهای منطقی در قطعه مورد نظر را اشغال کرده است.

پایگاه داده COIL نیز، که شامل 72 تصویر با زوایاي متفاوت از اجسام مختلف است، براي ارزیابي عملکرد سیستم مورد نظر، مورد استفاده قرار گرفته است. هر یک از تصاویر بعد از یك پیش پردازش ساده به یك ماتریس 32×32 تبدیل شده، و سپس از این ماتریس یك بردار 1024 بعدی براي ورودي ماشین بردار پشتیبان استخراج شده است.

اولین آزمایش برای جداکردن دو کلاس *obj72* و *obj74* از یکدیگر انجام شده است. 12 تصویر از هر کلاس متناظر با 12 زاویه مختلف از یک شیء برای آموزش، و بقیه 60 تصویر باقیمانده برای تست ماشین بردار پشیتبان استفاده شده است. در این حالت خطای کلاسه بندی هم در اجرای نرم افزاری و هم پیاده سازی سخت افزاری به صفر درصد رسیده است.

بقیه تصاویر نیز به صورت دو به دو با هم مقایسه شده است. نتایج نشان می دهد که اگر تعداد بردارهاي پشتیبان بین 20 تا 40 انتخاب شود، خطاي کلاسه بندي صفر است، در حالي که اگر این تعداد به 12 برسد، خطاي کلاسه بندي بین 0 تا 5% افزایش مي یابد.

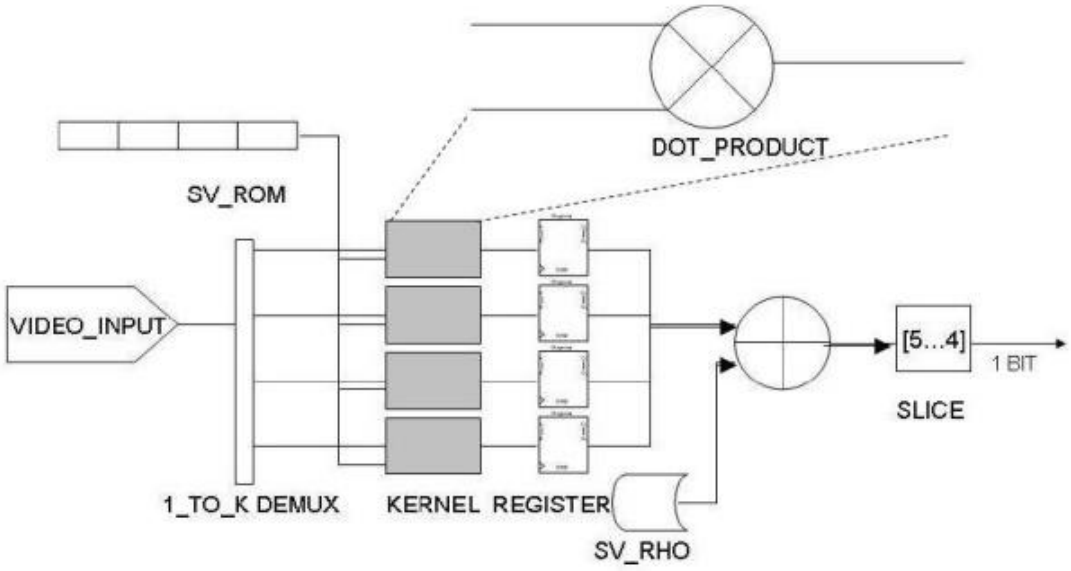
4-5- پیاده سازی ماشین بردار پشتیبان بر روی FPGA برای آشکارسازی مرز نما در تصاویر ویدئویی

چونسو[26] یک کلاسه بند ماشین بردار پشتیبان خطی برای آشکار سازی مرز نما[1] در تصاویر ویدئویی بر روی FPGA نوع Virtex4-XC4VSX35 پیاده سازی کرده است. بخش آموزش ماشین بردار پشتیبان که زمان بر است، برای یکبار با استفاده از مدل Libsvm به صورت نرم افزاری در MATLAB انجام شده و از نتایج آن به صورت آفلاین برای پیاده سازی فاز کلاسه بندی استفاده شده است. به منظور کاهش دادن تعداد ضرب کننده ها، بخشی از فرمول تابع تصمیم کلاسه بند ماشین بردار پشتیبان قبل از پیاده سازی سخت افزاری، در MATLAB محاسبه شده است.

ماشین بردار پشتیبان خطی موردنظر برای کلاسه بندی دو کلاس طراحی شده است: کلاس نما و کلاس غیرنما. نسبت داده های نما نسبت به غیرنما حدود 90 به 1 است. این اختلاف در حالت عادی باعث عدم تعادل در نتایج کلاسه بندی می شود. برای حل این مشکل از روش "نمونه برداری شبه منفی تصادفی"[2] استفاده شده است. با این روش صحت کلاسه بندی 93/5% به دست آمده است. بلوک دیاگرام کلی ساختار سخت افزاری کلاسه بند مورد نظر در شکل 4-4 مشاهده می شود.

با انتخاب طول کلمه 5 بیت دسیمال، در حالی که ابعاد بردار ویژگی 10 است، ماکزیمم فرکانس کاری سیستم 251/62MHz به دست آمده است.

[1] Shot boundary
[2] Random pseudo-negative sampling method



شکل 4-4: بلوک دیاگرام کلی ساختار کلاسه بند ماشین بردار پشتیبان خطی برای آشکارسازی مرز نما در تصاویر ویدئویی[26]

فصل 5- الگوری‌تم پی‌شنهادی جهت پی‌اده سازی ماشی‌ن بردار پشتی‌بان

همانطور که در چکیده نیز اشاره شد، جهت پیاده سازی ساختار کلاسه بند ماشین بردار پشتیبان بر روی سخت افزار ابتدا باید فاز آموزش ماشین بردار پشتیبان را که دارای الگوریتم پیچیده ای است با استفاده از نرم افزار به صورت خارج خط اجرا کنیم، و سپس از پارامترهای استخراج شده جهت پیاده سازی سخت افزاری فاز تست یا همان کلاسه بند ماشین بردار پشتیبان استفاده نماییم. در این تحقیق برای اجرای فاز آموزش ماشین بردار پشتیبان از مدل libsvm [21] در محیط نرم افزاری MATLAB استفاده می شود و فاز تست آن به کمک شبیه ساز سخت افزاری گرافیکی SYSTEM GENERATOR طراحی و اجرا خواهد شد.

همچنین به منظور واقعی کردن طرح مورد نظر و نتایج حاصل، از داده های سه کلاس مختلف مربوط به تصاویر ارقام دستنوشته فارسی [27] جهت نمونه های ورودی آموزش و تست ماشین بردار پشتیبان در پیاده سازی سیستم تشخیص ارقام فارسی استفاده می شود.

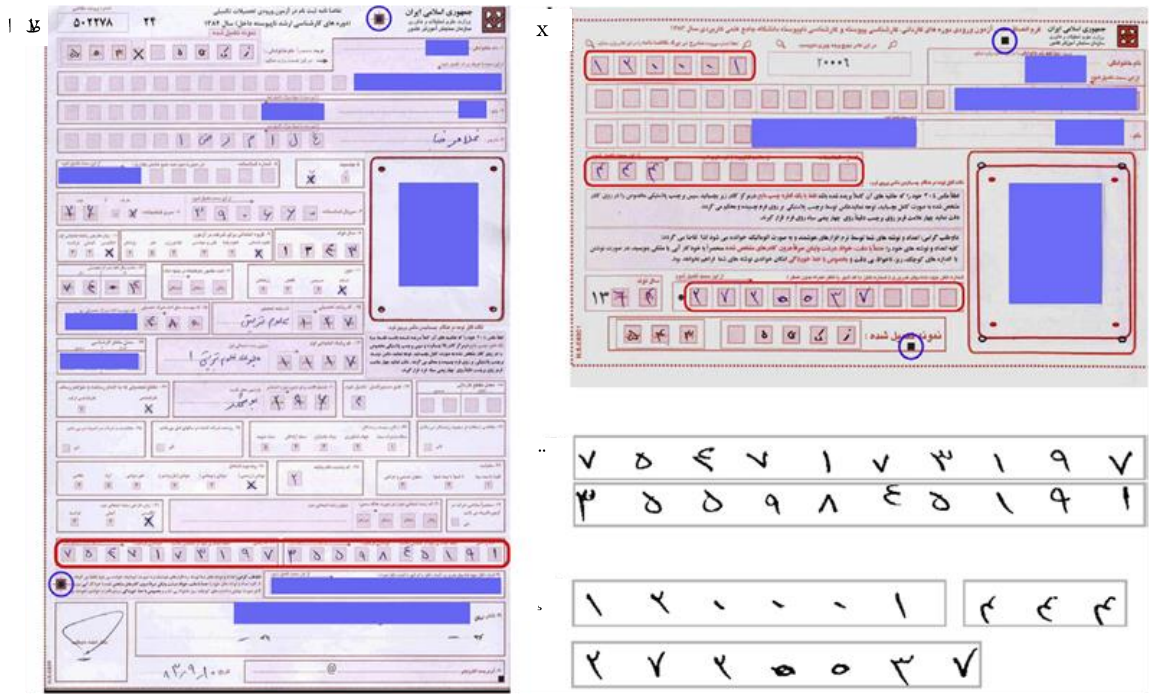
در این فصل اجرای نرم افزاری ماشین بردار پشتیبان تشخیص ارقام فارسی به همراه نتایج آن، نحوه پیاده سازی سخت افزاری توابع مختلف، کلاسه بندی سخت افزاری چند کلاس به طور همزمان و با ابعاد بردار ویژگی متفاوت به طور مفصل شرح داده خواهد شد. ساختار سخت افزاری مورد نظر طوری طراحی شده است که می توان آن را با کمی تغییر برای هر تعداد ابعاد ویژگی به سادگی پیاده سازی کرد، تنها محدودیت موجود در این خصوص فضای فیزیکی سخت افزار مورد نظر است.

از نتایج به دست آمده از فاز آموزش در این فصل، برای محاسبه نتایج شبیه سازی و آنالیز سخت افزاری در فصل 6 استفاده خواهد شد.

5-1- معرفی سی‌ستم تشخی‌ص ارقام فارسی دستنوشته

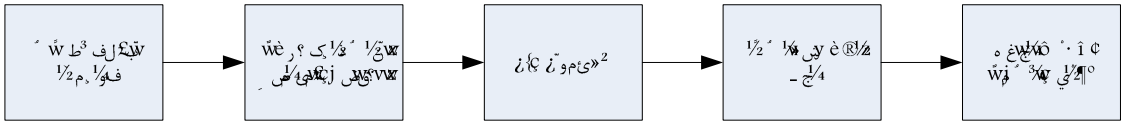
حسین خسروی[27] یک مجموعه داده بسیار بزرگ از تصاویر ارقام فارسی دستنوشته و روشی برای برای یافتن تفاوت در نگارش ارقام دستنوشته ارائه کرده است. 11942 نمونه فرم ثبت نام آزمون کارشناسی ارشد سراسری (نوع1) و آزمون ورودی دانشگاه سراسری (نوع2) برای تهیه این مجموعه مورد استفاده قرار گرفته است. بخش کدپستی و کدملی 10

رقمی از نوع 1، و بخش شماره پرونده، شماره شناسنامه و شماره تلفن، که رقمهای آن متغیر و حداکثر تا 26 رقم است، از نوع 2 استفاده شده است. فرمهای معتبر و محل ارقام مربوط به بخش های ذکر شده در هر فرم، با پیدا کردن مربع های سیاه رنگ در بالا و پایین فرمها، و از طریق یافتن مختصات محل بخش ها مشخص می شود. شکل 5-1 الف و ب نمونه ای از این فرمها را نشان می دهد.



شکل 5-1: الف)نمونه فرمهای ثبت نام نوع 1، ب)نمونه فرمهای نوع 2، ج و د) نمونه تصاویر باینری استخراج شده از ارقام دستنوشته مورد نظر [27]

پس از یافتن محل بخش های مربوط به ارقام دستنوشته مورد نظر، مراحل انجام عملیات پیش پردازش طبق شکل 5-2 اعمال می شود، تا در نهایت از هر رقم یک شکل باینری حاصل شود.



شکل 5-2: مراحل استخراج ارقام دستنوشته، پیش پردازش و تشخیص آنها

در این تحقیق کلاسه بند ماشین بردار پشتیبان را برای تشخیص اعداد 1، 4 و 8 از یکدیگر طراحی خواهیم کرد. برای هر کدام از کلاسهای مورد نظر 800 داده، (در مجموع 2400 نمونه) در اختیار داریم، که 600 داده از هر کلاس را برای اجرای فاز آموزش و 200 داده باقیمانده را برای اجرای فاز تست ماشین بردار پشتیبان به کار خواهیم برد. داده های مورد نظر دارای ابعاد بردار ویژگی 7 و 24 هستند، که با استفاده از ویژگی گشتاورهای هندسی[1] [28] استخراج شده اند.

با توجه به اینکه داده های مورد نظر دارای ابعاد بردار ویژگی 7 و 24 هستند، بنابراین کلاسه بند ماشین بردار پشتیبان را یک بار برای داده های 7بعدی و یک بار برای داده های 24بعدی پیاده سازی می کنیم.

5-2- اجرای نرم افزاری ماشین بردار پشتیبان برای تشخیص ارقام فارسی

قبل از پیاده سازی سخت افزاری ماشین بردار پشتیبان، لازم است که هم مرحله آموزش و هم مرحله تست آن به صورت نرم افزاری بر روی کامپیوتر اجرا شود، تا با تغییر پارامترهای مختلف، کمترین نرخ خطای کلاسه بندی به دست آید، و علاوه بر این پارامترهای لازم در حالت کمترین نرخ خطا برای کلاسه بندی خارج خط بر روی سخت افزار نیز استخراج شود.

جهت اجرای نرم افزاری ماشین بردار پشتیبان می توان یکی از ابزارها یا مدلهای معتبر در زمینه اجرای ماشین بردار پشتیبان که به عنوان مرجع شناخته می شوند، مورد استفاده قرار داد. به غیر از جعبه ابزار SVM در نسخه 2009 نرم افزار MATLAB، کدهای برنامه نویسی زیادی توسط محققان مختلف در سراسر دنیا جهت اجرای ماشین بردار پشتیبان نوشته شده است. در این میان مدل libsvm به عنوان یکی از بهترین و مورد اعتمادترین ابزارها جهت پیاده سازی ماشین بردار پشتیبان معرفی شده است که در بسیاری از مقالات و تحقیق های دانشگاهی از آن استفاده می شود.

در این تحقیق نیز بخش آموزش ماشین بردار پشتیبان توسط مدل libsvm در محیط نرم افزاری MATLAB اجرا می شود. برای این منظور کافیست پوشه حاوی بسته libsvm را به مسیرهای MATLAB اضافه و سپس دستور زیر را اجرا کنیم:

```
model = svmtrain(training_label_vector,
training_instance_matrix,['libsvm_options']);
```

[1] Geometric moments

که در آن training_label_vector همان بردار شاخص $\mathbf{y}_i$ شامل مقادیر 1+ و 1- است، که برای کلاس 1 مقدار 1- و برای کلاس 2 مقدار 1+ را در نظر گرفته ایم. training_instance_matrix ماتریس آموزش با اندازه n×d، شامل n نمونه آموزشی مربوط به کلاس 1 و 2 است که دارای ابعاد بردار ویژگی d می باشد. و libsvm_options شامل یک سری از کدهای خاص رشته ای با فرمت libsvm می باشد که مقادیر پارامترهای C، نوع تابع کرنل و پارامترهای متناظر، تعداد کلاسها و .... را مشخص می کند.

پس از اجرای دستور فوق، مقادیر پارامترهایی نظیر بردار حاصلضرب $\mathbf{y}_i$ در $\boldsymbol{\alpha}_i$ و همچنین SVها (بردارهای پشتیبان) در فضای کاری MATLAB ظاهر می شود. با توجه به رابطه (2-27) فقط همین پارامترها برای به دست آوردن مقدار تابع تصمیم و کلاسه بندی الگوی مورد نظر کافی است. بنابراین با داشتن این پارامترها می توان به راحتی فاز تست یا کلاسه بند ماشین بردار پشتیبان را به صورت خارج خط اجرا کرد.

در اینجا قصد داریم ماشین بردار پشتیبان را برای تشخیص سه رقم از ارقام فارسی، که در بخش 5-1 توضیح داده شد، اجرا کنیم. سیستم مورد نظر قرار است اعداد 1، 4 و 8 را از هم تمیز دهد، یعنی در واقع سه کلاس مختلف داریم که باید از هم جدا شوند. به صورت قراردادی برای عدد 1 کلاس 1، برای عدد 4 کلاس 2 و برای عدد 8 کلاس 3 را در نظر می گیریم.

در بخش 2-4- روش کلاسه بندهای دوتایی به عنوان یکی از روشهای کلاسه بندی در سیستمهای ماشین بردار پشتیبان چند کلاسه معرفی شد. در این روش اگر n کلاس برای کلاسه بندی داشته باشیم، $n(n-1)/2$ ماشین بردار پشتیبان دوتایی تشکیل می شود و در نهایت کلاسی که رای بیشتری داشته باشد، به عنوان کلاس مورد نظر انتخاب خواهد شد. در اینجا چون 3 کلاس در اختیار داریم، پس باید 3 کلاسه بند دوتایی طراحی شود.

برای اجرای سیستم تشخیص ارقام فارسی، 600 داده از هر کلاس یک بار با بردارهای ویژگی 7 بعدی و یک بار با بردارهای ویژگی 24 بعدی که پس از مرحله پیش پردازش[1] و استخراج ویژگی[2] به دست آمده است، به صورت تصادفی انتخاب و به کلاسه بندهای دوتایی، جهت آموزش، وارد می شود. در این صورت برای آموزش هر کلاسه بند دوتایی، 1200 داده ورودی خواهیم داشت.

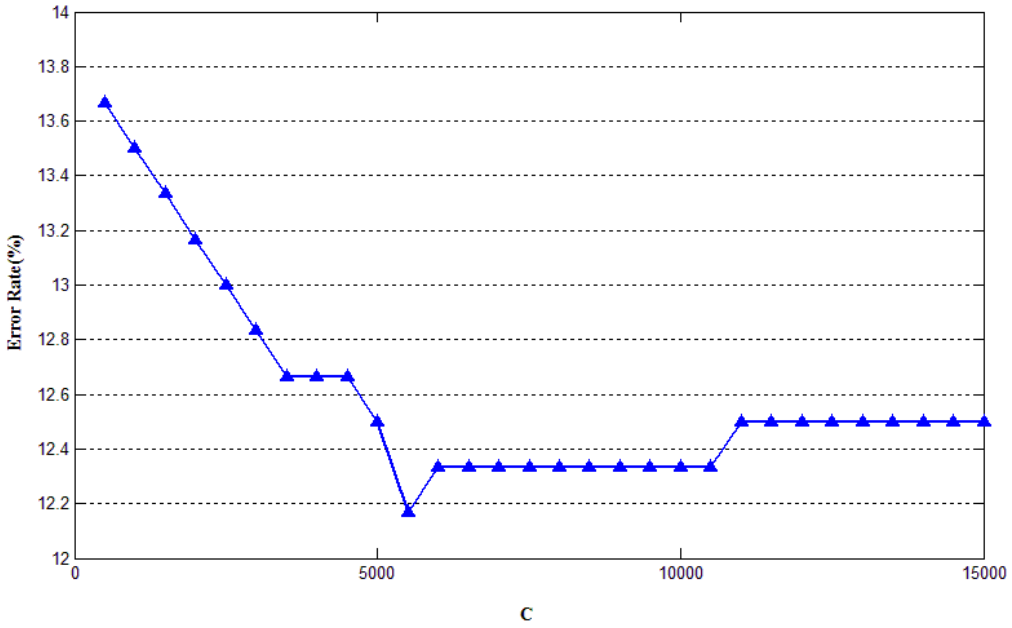
[1] Pre-processing
[2] Feature extraction

به طور کلی در اجرای ماشین بردار پشتیبان، قبل از شروع باید نوع تابع کرنل را مشخص کرد. در این تحقیق برای اجرای ماشین بردار پشتیبان در تشخیص ارقام فارسی، از دو نوع تابع کرنل خطی و گوسی استفاده می شود.

5-2-1- اجرای ماشین بردار پشتیبان با تابع کرنل خطی

تابع کرنل خطی ساده ترین نوع از توابع کرنلی است که در جدول 2-1 به آن اشاره شد. یکی از ویژگیهای بارز این تابع، سادگی روابط ریاضی و سرعت انجام محاسبات آن است. بنابراین تابع کرنل خطی گزینه بسیار مناسبی برای پیاده سازی بر روی سخت افزار می باشد. در این بخش اجرای نرم افزاری ماشین بردار پشتیبان خطی بررسی می شود و در بخش 5-3 کلاسه بند ماشین بردار پشتیبان خطی را برای پیاده سازی سخت افزاری طراحی خواهیم کرد.

در استفاده از ماشین بردار پشتیبان و توابع کرنل، پارامترهای عددی ویژه ای وجود دارد که توسط کاربر مشخص می شود و مقدار آنها نقش بسیار مهمی در نرخ خطای کلاسه بندی تعیین می کند. هر چند که تابع کرنل خطی پارامتر خاصی ندارد که بتوان با تغییر دادن آن نرخ خطا را بهینه کرد، اما در ساختار ماشین بردار پشتیبان خطی پارامتری به نام C وجود دارد که می توان با تغییر آن نرخ خطای کلاسه بندی را بهینه کرد. در این تحقیق آزمایش های انجام شده نشان می دهد که در کلاسه بندی ارقام فارسی با استفاده از ماشین بردار پشتیبان خطی، تغییر پارامتر C باعث ایجاد تغییر در نرخ خطای کلاسه بندی می شود، در حالیکه این تغییرات در کلاسه بندی ارقام فارسی با استفاده از ماشین بردار پشتیبان غیرخطی که در بخش بعدی توضیح داده خواهد شد، بسیار ناچیز است و بنابراین اثر این تغییرات در کلاسه بندی غیرخطی بررسی نخواهد شد. این وضعیت سبب می شود تا بتوان با تغییر دادن مقدار پارامتر C به بهترین نرخ خطای کلاسه بندی خطی دست یافت. به منظور رسیدن به این هدف، نموداری از تغییر نرخ خطای کلاسه بندی ماشین بردار پشتیبان خطی در مقابل تغییرات مقدار پارامتر C در یک بازه محدود را به دست آورده و مورد بررسی قرار می دهیم. شکل 5-3 نمودار حاصل از تشخیص ارقام فارسی با بردارهای ویژگی 7 بعدی، با استفاده از ماشین بردار پشتیبان خطی بازای تغییرات مقدار پارامتر C را نشان می دهد.



شکل 5-3: نمودار حاصل از تغییرات نرخ خطا در مقابل تغییر C در کلاسه بندی خطی ارقام فارسی 7 بعدی

نمودار شکل 5-3 نشان می دهد که با انتخاب C=5500 بهترین نرخ خطا در کلاسه بندی خطی ارقام فارسی 7 بعدی به دست می آید، که برابر با 12/17% می باشد. انتخاب این مقدار برای C اگر چه خطای کلاسه بندی را تا حدی کاهش می دهد، اما اگر بخواهیم همین مقدار C را برای پیاده سازی روی سخت افزار استفاده کنیم، باید تعداد بیتهای خیلی زیادی برای انجام عملیات محاسباتی به آن اختصاص دهیم. همین امر باعث مصرف شدن فضای سخت افزاری بسیار زیاد و کاهش شدید سرعت سیستم می شود.

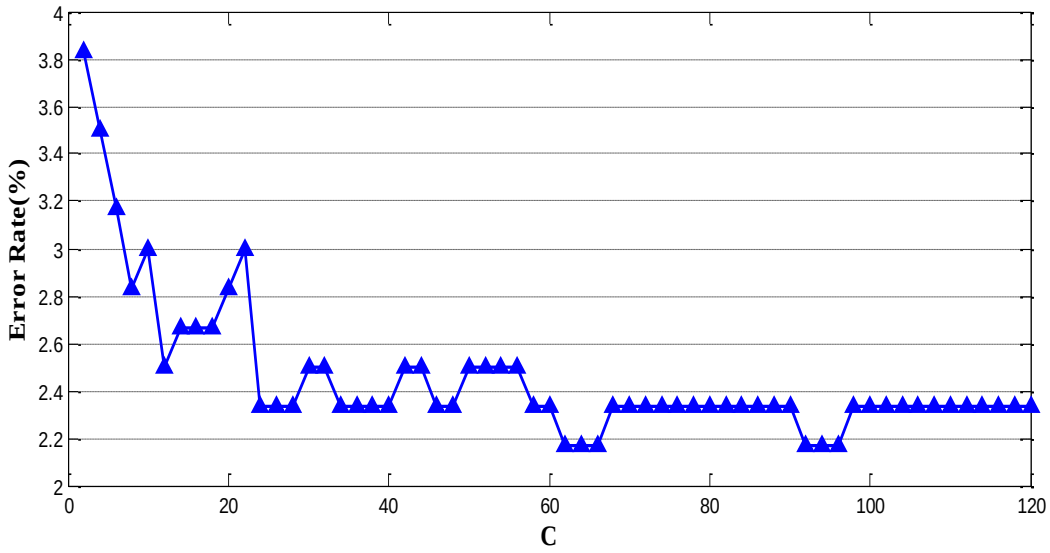
اگر مقدار پارامتر C را برابر با 100 انتخاب کنیم، نرخ خطا به 13/83% افزایش می یابد. با این مقدار C به دلیل کم شدن تعداد بیتها، به راحتی می توان کلاسه بند مورد نظر را بر روی سخت افزار پیاده سازی کرد، در حالی که افزایش نرخ خطای کلاسه بندی خیلی محسوس نیست و محدویت های فضای سخت افزاری و کاهش سرعت محاسباتی نیز تا حدودی از بین خواهد رفت.

با قراردادن مقدار 100 برای پارامتر C، مجدداً ماشین بردار پشتیبان خطی را برای کلاسه بندی داده های ارقام فارسی 7 بعدی مورد نظر اجرا می کنیم. جدول 5-1 نتایج حاصل از این کلاسه بندی را با روش کلاسه بندهای دوتایی نشان می دهد، که در آن SV مشخص کننده تعداد بردارهای پشتیبان است و b نشان دهنده بایاس می باشد.

جدول 5-1: نتایج حاصل از کلاسه بندی خطی ارقام فارسی 7 بعدی با مقدار C=100

کلاس 2 و 3		کلاس 1 و 3		کلاس 1 و 2		
کلاس3	کلاس2	کلاس3	کلاس1	کلاس2	کلاس1	
243	243	7	6	35	35	SV
-10/4076		3/7228		-0/7895		b
20/75%		10/5%		10/25%		Error Rtae

برای حفظ عمومیت مسئله با تغییر ابعاد ویژگی، مجدداً ماشین بردار پشتیبان خطی را این بار بر روی داده های ارقام فارسی 24 بعدی اجرا می کنیم. در این حالت نیز برای رسیدن به بهترین نتیجه، باید نموداری از تغییرات نرخ خطا در مقابل تغییر پارامتر C به دست بیاوریم. شکل 4-5 نمودار مورد نظر را بازای تغییرات پارامتر C در یک بازه عددی محدود نشان می دهد.



شکل 4-5: نمودار حاصل از تغییرات نرخ خطا در مقابل تغییر C در کلاسه بندی خطی ارقام فارسی 24 بعدی

با توجه به نمودار شکل 4-5 مقدار بهینه خطا در C=62 به دست می آید که برابر با 2/167% می باشد. مقدار عددی پارامتر C در این حالت خیلی بزرگ نیست و البته به راحتی و با تعداد بیت های کم قابل پیاده سازی بر روی سخت افزار می باشد. در بخش5-3- در مورد پیاده سازی سخت افزاری این کلاسه بند بحث خواهد شد. جدول 5-2 نتایج حاصل از کلاسه بندی ارقام فارسی 24 بعدی با مقدار C=62 را نشان می دهد.

جدول 5-2: نتایج حاصل از کلاسه بندی خطی ارقام فارسی 24 بعدی با مقدار C=62

کلاس 1 و 2		کلاس 1 و 3		کلاس 2 و 3		
کلاس1	کلاس2	کلاس1	کلاس3	کلاس2	کلاس3	
30	33	21	22	44	47	SV
-20/2396		-8/6731		0/1235		b
3%		0/5%		3%		Error Rtae

5-2-2- اجرای ماشین بردار پشتیبان با تابع کرنل گوسی

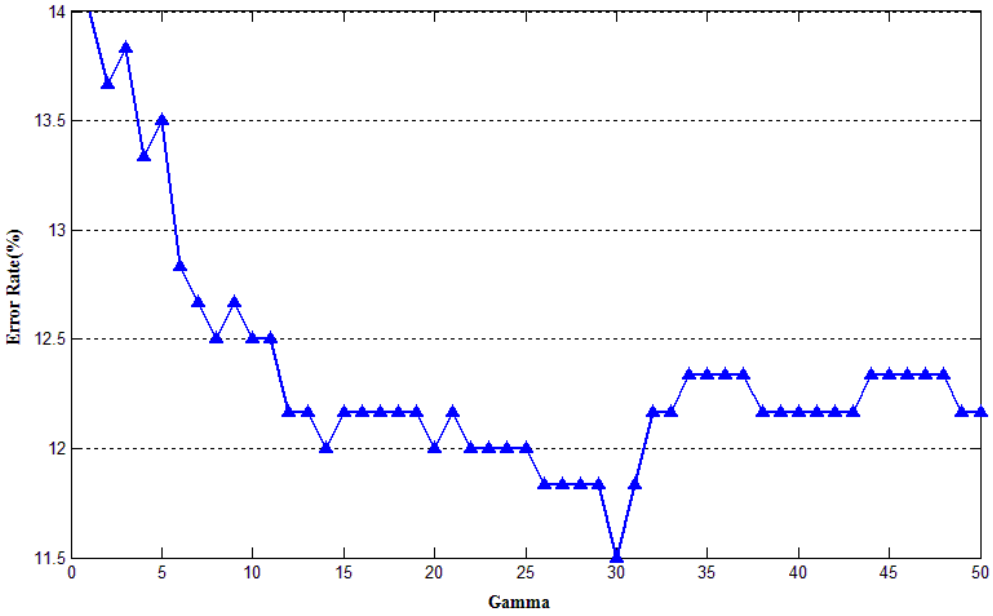
یکی از توابع کرنلی که در ماشین بردار پشتیبان زیاد استفاده می شود و نتایج تقریباً بهتری نسبت به سایر توابع کرنل از خود نشان داده است، تابع کرنل گوسی است [29]. در این تحقیق نیز از تابع کرنل گوسی جهت آموزش و تست ماشین بردار پشتیبان غیر خطی استفاده می شود.

نکته ای که در اینجا باید به آن دقت کرد این است که یکی از فاکتورهای مهمی که در استفاده از توابع کرنل غیرخطي مد نظر قرار می گیرد، مقدار پارامتر ثابت آنهاست. در تابع کرنل گوسی این پارامتر همان مقدار گاما است که در جدول 2-1 با $1/2\sigma^2$ نشان داده شده است و برای سادگی آن را برابر با γ (گاما) قرار می دهیم:

$$\gamma = 1/2\sigma^2 \qquad (1\text{-}5)$$

مقدار گاما نقش بسیار مهمی در به دست آمدن نرخ خطای بهینه دارد و باید دقت خاصی نسبت به آن داشت، چرا که انتخاب نامناسب گاما منجر به نتایج ضعیفی در کلاسه بندی خواهد شد. رابطه دقیقی برای محاسبه مقدار گاما وجود ندارد و غالبا با روش سعی و خطا می توان به مقدار گامای مناسب دست پیدا کرد.

در این بخش می خواهیم نتایج حاصل از کلاسه بندی ارقام فارسی، با شرایطی که در بخش 5-1 توضیح داده شد، توسط ماشین بردار پشتیبان غیرخطی بررسی کنیم. با توجه به اینکه تابع کرنل گوسی برای این هدف انتخاب شده است، برای رسیدن به مقدار بهینه ی پارامتر مهم گاما، نموداری از تغییرات نرخ خطا در مقابل تغییرات گاما در یک بازه عددی محدود مورد بررسی قرار می دهیم. این نمودار به ما کمک خواهد کرد تا بتوان مناسب ترین مقدار گاما برای رسیدن به بهترین نتیجه کلاسه بندی ممکن را به دست آورد. شکل 5-5 نمودار مورد نظر را در کلاسه بندی ارقام فارسی 7 بعدی نشان می دهد.



شکل 5-5: نمودار تغییرات نرخ خطا در مقابل تغییر گاما در یک بازه عددی محدود در کلاسه بندی غیرخطی ارقام فارسی 7 بعدی

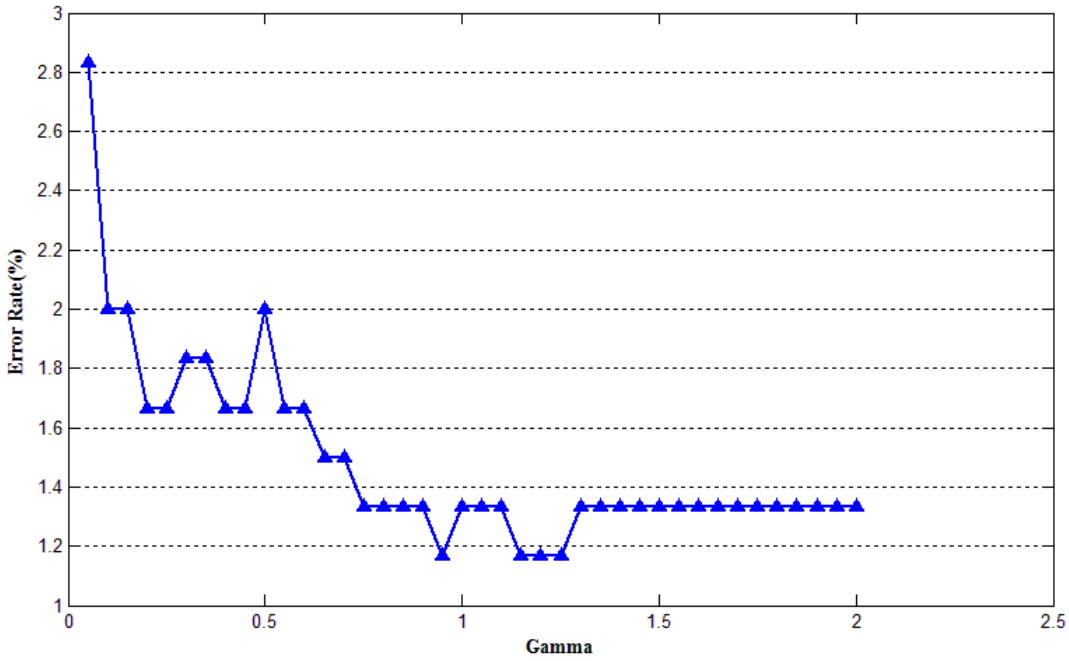
همانطور که از نمودار شکل 5-5 نیز کاملاً مشخص است، با انتخاب مقادیر کم برای گاما نرخ خطای بالایی به دست می آید، که مناسب نمی باشد. ضمن اینکه در مقادیر بالای گاما نیز خطای مناسبی حاصل نمی شود. مقدار 30 برای گاما کمترین نرخ خطای کلاسه بندی را ارائه می دهد که برابر با 11/5% است. بنابراین در مرحله بعد مقدار گاما را برابر با 30 قرار داده و مراحل آموزش و تست ماشین بردار پشتیبان را مجدداً اجرا می کنیم. جدول 5-3 نتایج حاصل از این اجرا را در حالت کلاسه بندی دوتایی نشان می دهد.

جدول 5-3: نتایج حاصل از کلاسه بندی ارقام فارسی 7 بعدی با تابع کرنل گوسی و γ= 30

کلاس 1 و 2		کلاس 1 و 3		کلاس 2 و 3		
کلاس1	کلاس2	کلاس1	کلاس3	کلاس2	کلاس3	
19	20	9	16	199	200	SV
0/0658		0/0628		-2/6997		b
8/25%		9%		17/25%		Error Rtae

در مرحله بعد ماشین بردار پشتیبان غیرخطی با تابع کرنل گوسی را برای تشخیص ارقام فارسی با بردارهای ویژگی 24 بعدی استفاده می کنیم تا عمومیت کلاسه بندی با تغییر ابعاد ویژگی نیز مورد بررسی قرار بگیرد. همانند قبل ابتدا باید مقدار بهینه را برای پارامتر تابع کرنل

گوسی، یعنی گاما، بیابیم. شکل 5-6 نمودار حاصل از این کلاسه بندی را به ازای تغییرات مقدار گاما در یک بازه عددی محدود نشان می دهد.



شکل 5-6: نمودار تغییرات نرخ خطا در مقابل تغییر گاما در یک بازه عددی محدود در کلاسه بندی غیرخطی ارقام فارسی 24 بعدی

همانطوری که نمودار شکل 5-6 نشان می دهد بهترین نرخ خطا برای تشخیص ارقام فارسی با ابعاد ویژگی 24 تایی، در گامای 0/95 به دست می آید. نرخ خطای مینیمم در این مقدار گاما برابر با 1/167% می باشد، که در مقایسه با کلاسه بند 7 بعدی بسیار بهتر است. به همین دلیل در این تحقیق در بخش 5-3-6- همین کلاسه بند را برای پیاده سازی سخت افزاری بر روی FPGA طراحی خواهیم کرد.

با بکارگیری مقدار گاما برابر با 0/95 و اجرای مجدد ماشین بردار پشتیبان برای تشخیص ارقام فارسی 24 بعدی، نتایج کلاسه بندی مطابق جدول 5-4 به دست می آید.

جدول 5-4: نتایج حاصل از کلاسه بندی ارقام فارسی 24 بعدی با تابع کرنل گوسی و γ= 0/95

کلاس 1 و 2		کلاس 1 و 3		کلاس 2 و 3		
کلاس1	کلاس2	کلاس1	کلاس3	کلاس2	کلاس3	
19	22	18	18	33	35	SV
-3/8358		-3/2601		-3/1587		b
1/5%		0/5%		1/5%		Error Rtae

اجازه دهید در پایان این بخش با ارائه جدول 5-5 نتایج کلاسه بندی ماشین بردار پشتیبان خطی و غیرخطی برای تشخیص ارقام فارسی 7 و 24 بعدی را با هم مقایسه کنیم. نتایج این جدول در فصل 6- با نتایج پیاده سازی سخت افزاری مقایسه خواهد شد.

جدول 5-5: مقایسه روشهای کلاسه بندی خطی و غیرخطی در تشخیص ارقام فارسی 7 و 24 بعدی

نوع تابع کرنل	پارامتر	ابعاد بردار ویژگی	نرخ خطا
خطی	C=100	7	13/83%
خطی	C=62	24	2/167%
گوسی	γ=30	7	11/5%
گوسی	γ=0/95	24	1/167%

جدول 5-5 نشان می دهد که نرخ خطای کلاسه بندی به روش ماشین بردار پشتیبان با تابع کرنل خطی و گوسی، با ابعاد بردار ویژگی یکسان، تفاوت چندانی ندارد. از آنجا که کلاسه بندی به روش ماشین بردار پشتیبان خطی دارای ساختار بسیار ساده تری است، بنابراین می توان با پذیرش درصد خطای ناچیز، به جای استفاده از ماشین بردار پشتیبان غیرخطی از ماشین بردار پشتیبان خطی برای پیاده سازی بر روی سخت افزار استفاده کرد، این کار باعث می شود که هم فضای کمتری از منابع سخت افزاری برای طراحی استفاده شود و هم سرعت عملکرد سیستم بالا رود.

در ادامه ساختار سخت افزاری کلاسه بند ماشین بردار پشتیبان خطی، به منظور پیاده سازی بر روی FPGA طراحی می گردد. همچنین یک روش ساده و مفید برای پیاده سازی سخت افزاری تابع کرنل گوسی ارائه می گردد که توسط آن کلاسه بند ماشین بردار پشتیبان غیرخطی نیز طراحی خواهد شد. روند انجام طراحی در بخش بعد توضیح داده می شود.

5-3- پیاده سازی سخت افزاری ماشین بردار پشتیبان

همانطور که قبلاً نیز توضیح داده شد، اجرای فاز آموزش ماشین بردار پشتیبان بر روی سخت افزار، به دلیل پیچیدگی روابط آن کاری بسیار مشکل و نیازمند فضای سخت افزاری زیادی می باشد که البته سرعت انجام محاسبات نیز کاهش می یابد. بنابراین برای پیاده سازی ماشین بردار پشتیبان بر روی سخت افزار، روش مناسب و کم هزینه تر این است که فاز آموزش ماشین بردار پشتیبان را به صورت

خارج خط با نرم افزار اجرا کنیم و از پارامترهای خروجی آن برای پیاده سازی فاز تست یا همان کلاسه بند ماشین بردار پشتیبان بر روی سخت افزار استفاده نماییم.

برای درک بهتر این موضوع بهتر است رابطه تابع تصمیم کلاسه بند خطی ماشین بردار پشتیبان را مجدداً مرور کنیم:

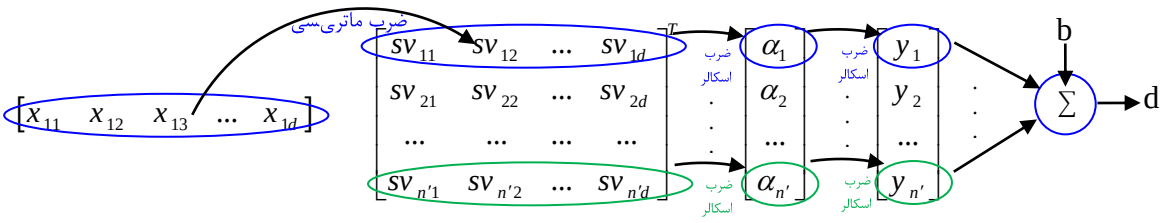
$$d(\mathbf{x}) = \mathbf{w_0}^T \mathbf{x} + b_0 = \sum_{i=1}^{SV} \alpha_i y_i \mathbf{x}_i \mathbf{x} + b \qquad (2\text{-}5)$$

این رابطه نشان می دهد که برای کلاسه بندی یک الگوی ورودی یا همان داده تست، باید هر یک از پارامترهای $\mathbf{x}_i$، y_i، α_i و b را داشته باشیم. با توجه به اینکه این پارامترها برای هر کلاسه بند ثابت می باشند، کافی است یکبار فاز آموزش ماشین بردار پشتیبان را با نرم افزار اجرا کنیم و سپس از پارامترهای به دست آمده برای اجرای فاز تست استفاده نماییم. در مدل libsvm تمامی این پارامترها را می توان پس از اجرای مرحله آموزش ماشین بردار پشتیبان استخراج نمود و سپس از آنها برای پیاده سازی بر روی FPGA استفاده کرد.

5-3-1- اعمال تکنیک تعادل قبل از پیاده سازی سخت افزاری

در بخش 5-2- فاز آموزش و تست ماشین بردار پشتیبان را برای تشخیص ارقام فارسی 1، 4 و 8 با ابعاد بردار ویژگی 7 و 24 به صورت نرم افزاری اجرا کردیم و توضیح دادیم که به دلیل سادگی و جلوگیری از محاسبات پیچیده که منجر به اشغال فضای سخت افزاری زیادی می شود، بهتر است در صورت عدم افزایش محسوس نرخ خطای کلاسه بندی، از ماشین بردار پشتیبان خطی به منظور پیاده سازی سخت افزاری استفاده کنیم.

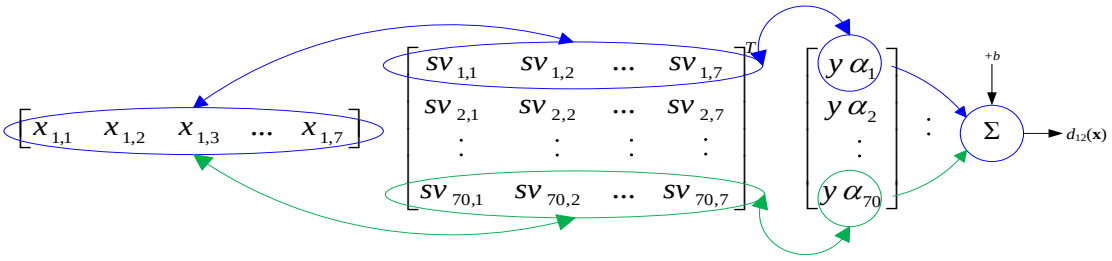
از مرحله آموزش ماشین بردار پشتیبان خطی مورد نظر که به صورت نرم افزاری اجرا شد، پارامترهای $\mathbf{x}_i$، y_i، α_i و b استخراج می شود و برای پیاده سازی در رابطه (2-5) قرار می گیرد. اجازه دهید تابع تصمیم کلاسه بند ماشین بردار پشتیبان خطی را که قرار است بر روی FPGA پیاده سازی کنیم، به فرم ماتریسی و قابل پیاده سازی بر روی FPGA به صورت شکل 5-7 در بیاوریم.



شکل 5-7: فرم ماتریسی کلی تابع تصمیم ماشین بردار پشتیبان خطی

فرم ماتریسی فوق نشان می دهد که برای محاسبه تابع تصمیم هر الگوی ورودی باید آن را در هر کدام از بردارهای پشتیبان، به صورت ماتریسی، و سپس در ضرایب آلفا و درایه های بردار شاخص متناظر، به صورت عددی، ضرب کرد و حاصل جمع این مقادیر را با مقدار b جمع نمود تا تابع تصمیم به صورت یک عدد به دست آید.

در جدول 5-1 تعداد بردارهای پشتیبان برای کلاسه بندی داده های ارقام فارسی 7 بعدی که از مرحله آموزش و به صورت نرم افزاری به دست آمد، مشخص شده است. می توانیم فرم ماتریسی شکل 5-7 را با توجه به نتایج به دست آمده از جدول 5-1 بازنویسی کنیم. همچنین به منظور ساده تر کردن عملیات محاسباتی بهتر است عملیات ضرب اسکالر بردار **α** در بردار شاخص **y** را در هم ادغام کنیم و بردار جدیدی به نام **yα** بسازیم. شکل 5-8 فرم ماتریسی تابع تصمیم ماشین بردار پشتیبان خطی را برای جداسازی کلاسهای 1 و 2 در سیستم تشخیص ارقام فارسی 7 بعدی نشان می دهد.



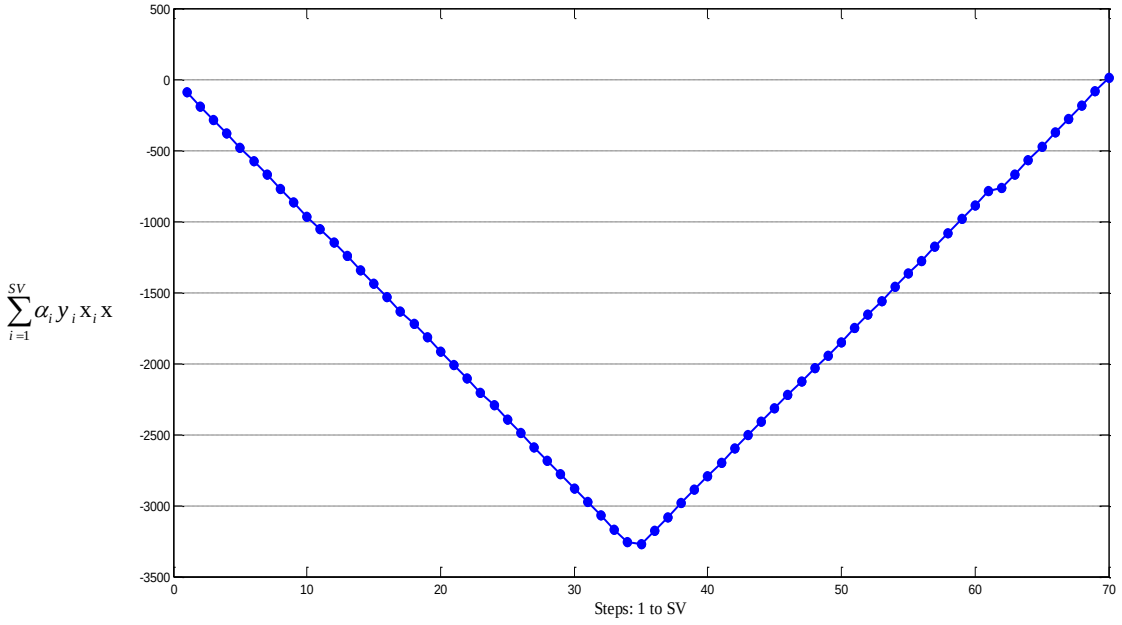
شکل 5-8: فرم ماتریسی تابع تصمیم ماشین بردار پشتیبان خطی در کلاسه بندی کلاسهای 1 و 2 ارقام فارسی 7 بعدی

با توجه به ماهیت ساختار FPGA، فرم ماتریسی فوق دید طراح را برای پیاده سازی بر روی FPGA باز می کند. تابع فوق فقط شامل عملیات ضرب و جمع است که می تواند در FPGA هم به صورت موازی و هم به صورت سری پیاده سازی شود. با توجه به اینکه تعداد بردارهای پشتیبان (SV) در هر کلاسه بند دوتایی متفاوت است، برای عمومیت دادن به ساختار طراحی بهتر است عملیات ضرب ماتریسی بردار تست در هر کدام از بردارهای پشتیبان و ضرب اسکالر نتیجه آن در بردار تلفیق شده **yα** به صورت موازی و همزمان، و عملیات جمع نتیجه هر کدام از این ضربها به صورت سری انجام شود.

در فرم ماتریسی شکل 5-8 بردارهای پشتیبان و همچنین درایه های بردار **yα** از 1 تا 35 مربوط به کلاس 1 و از 36 تا 70 مربوط به کلاس 2 است. با توجه به مقدار عددی

داریه های بردار y (1- برای کلاس 1 و 1+ برای کلاس 2) علامت نتیجه حاصلضرب ماتریسی بردار تست ورودی در هر کدام از بردارهای پشتیبان و سپس در درایه های ماتریس **yα** ، برای کلاس 1 و 2 متفاوت خواهد بود. در این صورت اگر بردارهای پشتیبان را به همین ترتیب کلاس 1 و2 در ماتریس بردار پشتیبان قرار دهیم، با توجه به اینکه قرار است عملیات جمع در رابطه $\sum_{i=1}^{SV}\alpha_i y_i \mathbf{x}_i \mathbf{x}$ به صورت سری انجام شود، در مرحله 35 ام (آخرین بردار پشتیبان کلاس 1) با یک عدد بزرگی روبرو خواهیم شد که در هنگام پیاده سازی بر روی FPGA برای محاسبه آن باید تعداد بیتهای زیادی را اختصاص دهیم که این خود باعث افزایش فضای سخت افزاری و کاهش سرعت سیستم می شود. هر چقدر تعداد بردارهای پشتیبان بیشتر باشد این عدد به همان نسبت افزایش می یابد و مشکل افزایش فضا و کاهش سرعت سیستم نیز بیشتر خواهد شد.

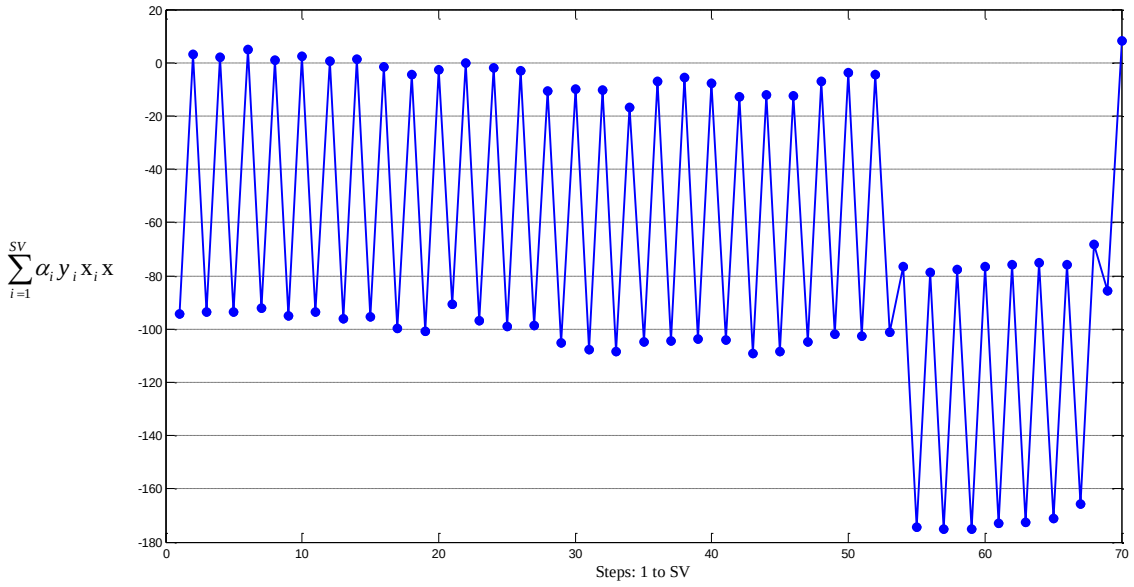
شکل 5-9 نموداری از مراحل افزایش مقدار عددی جمع مورد نظر در هنگام کلاسه بندی کلاسهای 1 و 2 داده های ارقام فارسی 7 بعدی را نشان می دهد. در مرحله 35 نمودار حاصل، به عدد 3/2746e3- می رسیم که برای فقط بخش صحیح آن به تعداد بیتی معادل 13 بیت نیاز می باشد. این تعداد بیت برای کلاسه بندی کلاسهای 2و 3 که تعداد بردارهای پشتیبان زیادتری دارد (243 بردار پشتیبان در هر کلاس)، بسیار بیشتر خواهد شد.



شکل 5-9: نمودار تغییرات مقدار $\sum_{i=1}^{SV}\alpha_i y_i \mathrm{x}_i \mathrm{x}$ از مرحله 1 تا SV ، در کلاسه بندی خطی کلاسهای 1 و 2 ارقام فارسی 7 بعدی، در حالتی که

بردارهای پشتیبان کلاسهای 1 و 2 پشت سر هم در ماتریس SV قرار گرفته اند.

به منظور جلوگیری از این افزایش تعداد بیت، روش پیشنهادی ما به نام "تکنیک تعادل" این است که ترتیب قرار گرفتن بردارهای پشتیبان کلاسهای 1 و 2 در ماتریس بردار پشتیبان **SV** و درایه های $y_i\alpha_i$ متناظر با آنها در بردار **yα** به صورت یک در میان باشد، یعنی ابتدا یکی از بردارهای پشتیبان کلاس 1 و سپس یکی از بردارهای پشتیبان کلاس2 در ماتریس SV قرار بگیرد و این ترتیب تا قرار گرفتن آخرین بردار پشتیبان هر کلاس ادامه پیدا کند. این وضعیت سبب می شود تا هر عملیات ضربی که صورت می پذیرد، نتیجه عملیات ضرب بعدی علامتی مخالف با علامت ضرب فعلی داشته باشد، بنابراین در جمع سری این نتایج، یعنی رابطه $\sum_{i=1}^{SV}\alpha_i y_i \mathbf{x}_i \mathbf{x}$، عدد خیلی بزرگی از نظر قدر مطلق حاصل نمی شود و در نتیجه تعداد بیت کمتری برای محاسبات آن در FPGA مورد استفاده قرار می گیرد. شکل 5-10 نموداری از تغییرات مقدار عددی این حاصل جمع را از مرحله 1 تا 70 در کلاسه بندی کلاسهای 1 و 2 نشان می دهد. همانطور که انتظار می رفت این مقدار مرتباً به صورت نوسانی افزایش و کاهش دارد و قدر مطلق آن هیچگاه عدد خیلی بزرگی نمی شود. در نمودار به دست آمده، حداکثر قدرمطلق مقدار عددی در حاصل جمع، 175/1940 است که نسبت به مقدار آن قبل از اعمال این روش (یعنی عدد 3/2746e3) عدد بسیار کوچکتری است. بخش صحیح این عدد را می توان با 9 بیت نمایش داد که در مقایسه با عدد قبلی 4 بیت صرفه جویی شده است.

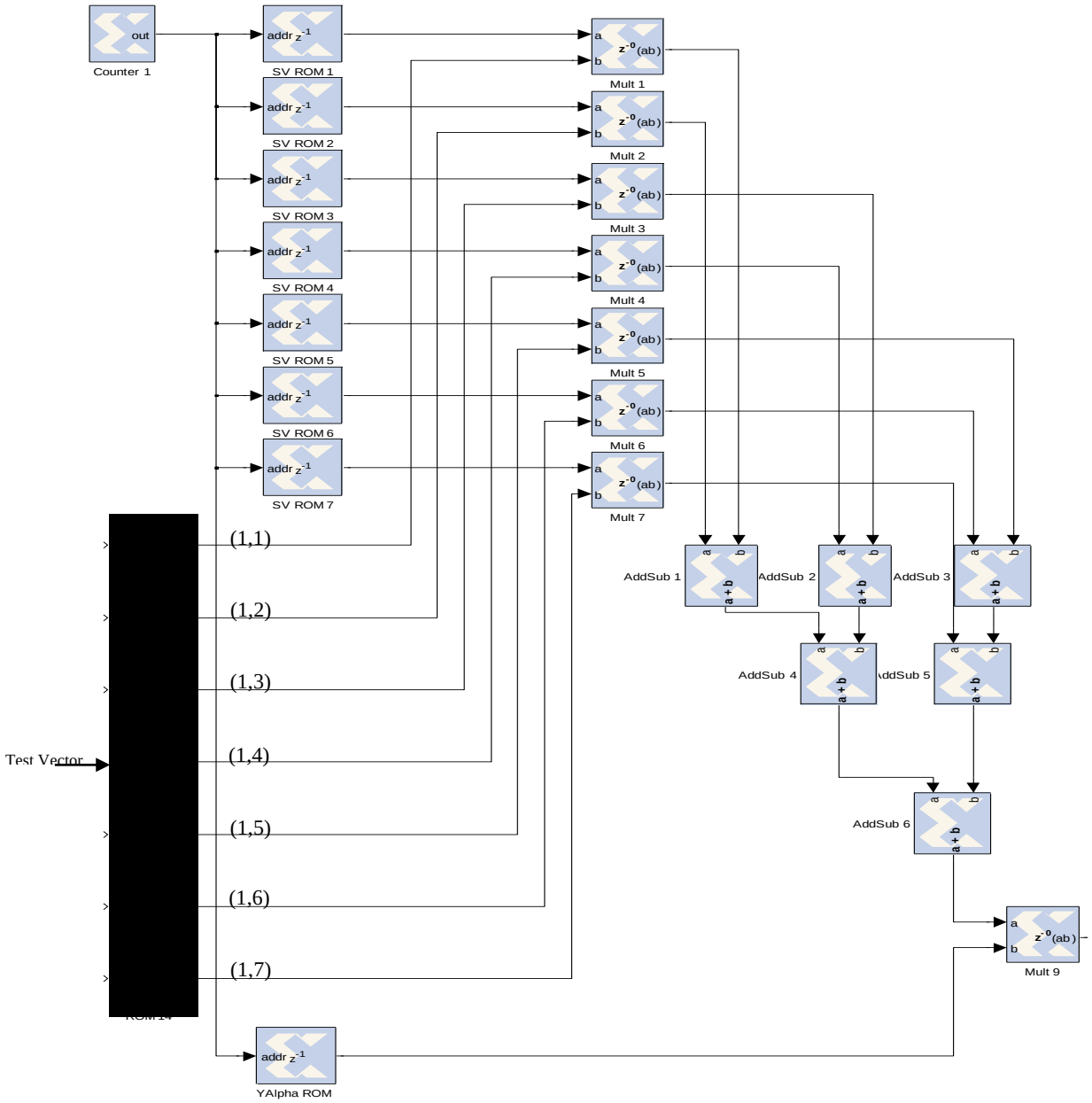


شکل 5-10: نمودار تغییرات مقدار $\sum_{i=1}^{SV}\alpha_i y_i \mathrm{x}_i \mathrm{x}$ از مرحله 1 تا SV ، در کلاسه بندی خطی کلاسهای 1 و 2 ارقام فارسی 7 بعدی، در حالتی که بردارهای پشتیبان کلاسهای 1 و 2 به صورت یک در میان در ماتریس SV قرار گرفته اند.

5-3-2- استفاده از تکنیک Pipelining برای افزایش فرکانس

تا اینجای کار با آماده کردن ماتریس **SV** و بردار **yα** به ترتیبی که توضیح داده شد، بستر مناسب برای طراحی سخت افزاری ماشین بردار پشتیبان فراهم شده است. برای شروع روند طراحی، یک بار دیگر به رابطه ماتریسی شکل 5-8 بر می گردیم. هدف این است که ساختاری طراحی کنیم که بتواند محاسبات مورد نظر را بر روی FPGA پیاده سازی کند. همانطور که قبلاً ذکر کردیم، در این تحقیق از شبیه ساز گرافیکی و قدرتمند System Generator برای طراحی ساختار سخت افزاری مورد نظر استفاده شده است.

در اولین مرحله، به طراحی قسمت ضرب ماتریسی بردار تست ورودی در بردارهای ماتریس **SV** می پردازیم. شکل 5-11 طراحی سخت افزاری مورد نظر را در محیط شبیه ساز System Generator نشان می دهد. فرض بر این است که بردار تست ورودی از هر درگاهی می تواند وارد سیستم شود، بنابراین این بردار به صورت بک بلوک سیاه رنگ نشان داده شده است. در ادامه برای آنالیز و شبیه سازی، بردار تست ورودی را در ROM های داخلی FPGA قرار داده و سیستم را آنالیز می کنیم.



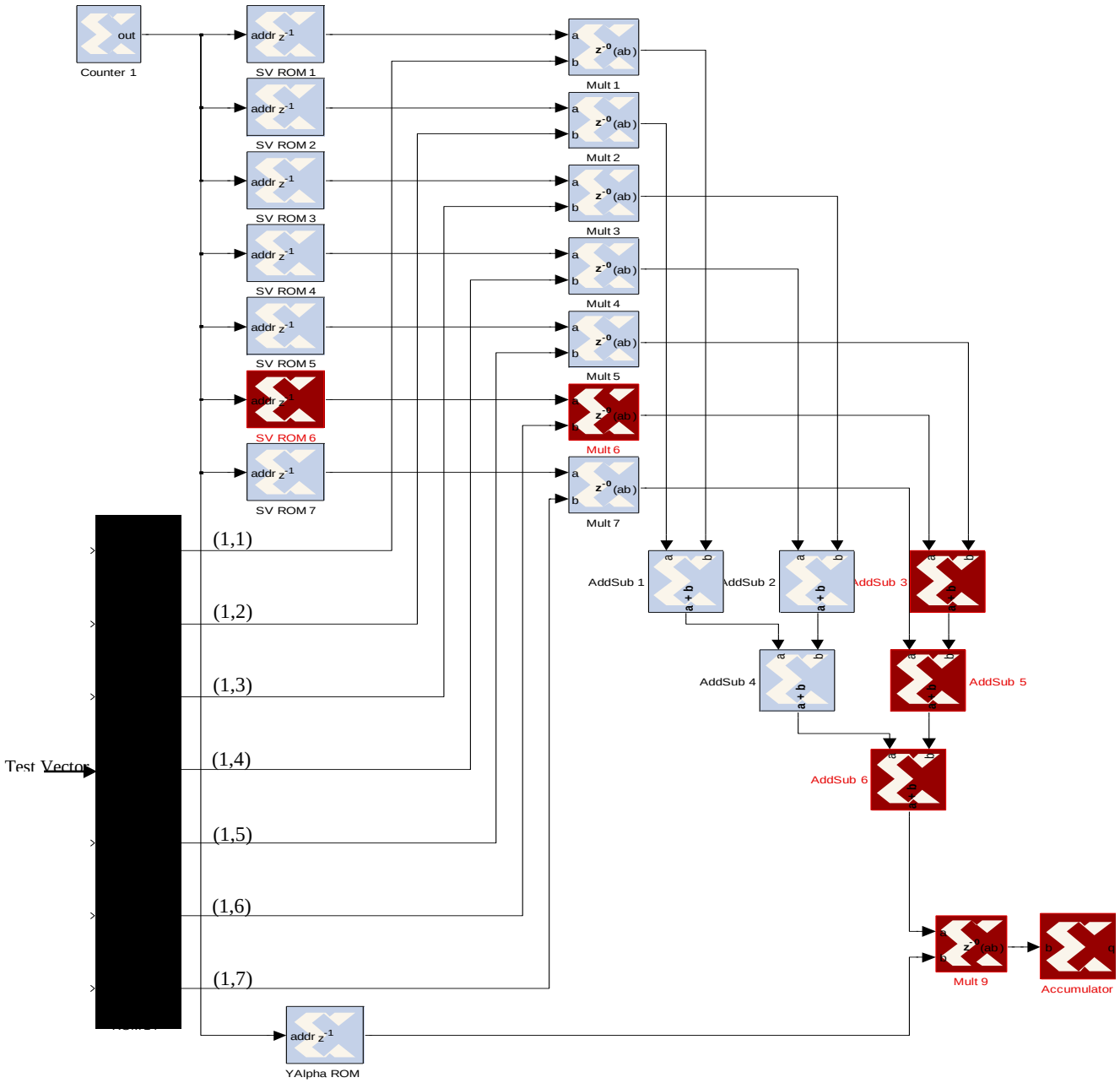
شکل 5-11: طرح سخت افزاری ضرب بردار تست ورودی در ماتریس SV و بردار yα (رابطه $\alpha_i y_i \mathbf{x}_i \mathbf{x}$)

در شکل 5-11 هر کدام از ستونهای ماتریس **SV** در حافظه های SV ROM1 تا SV ROM7 قرار دارد، این ROM ها ذاتاً به اندازه یک کلاک تاخیر داخلی دارند که قابل کاهش نیست. این ROM ها همچنین پورت آدرسی به نام addr دارند که عدد ورودی آن درایه ای از ستون مورد نظر که قرار است در کلاک بعد در خروجی قرار بگیرد را نشان می دهد. اگر پورت آدرس را به شمارنده ای وصل می کنیم که از مقدار صفر تا تعداد بردارهای پشتیبان منهای 1 را بشمارد، در هر کلاک در خروجی مجموعه ROM های 7 تایی، درایه های بردار پشتیبان متناظر با عدد شمارنده را خواهیم داشت.

بلوک های Addsub1 تا Addsub6 به عنوان جمع کننده به کار رفته اند. نتیجه حاصل از ضرب بردار تست در هر یک از بردارهای پشتیبان در خروجی Addsub6 ظاهر خواهد شد. این نتیجه باید در درایه بردار **yα** متناظر نیز ضرب عددی شود، که این ضرب در بلوک Mult9 انجام می شود، در حالیکه بردار **yα** در بلوک YAlpha ROM قرار گرفته و شمارنده ای که پورت addr آن را آدرس دهی می کند با شمارنده ROM های SV مشترک است.

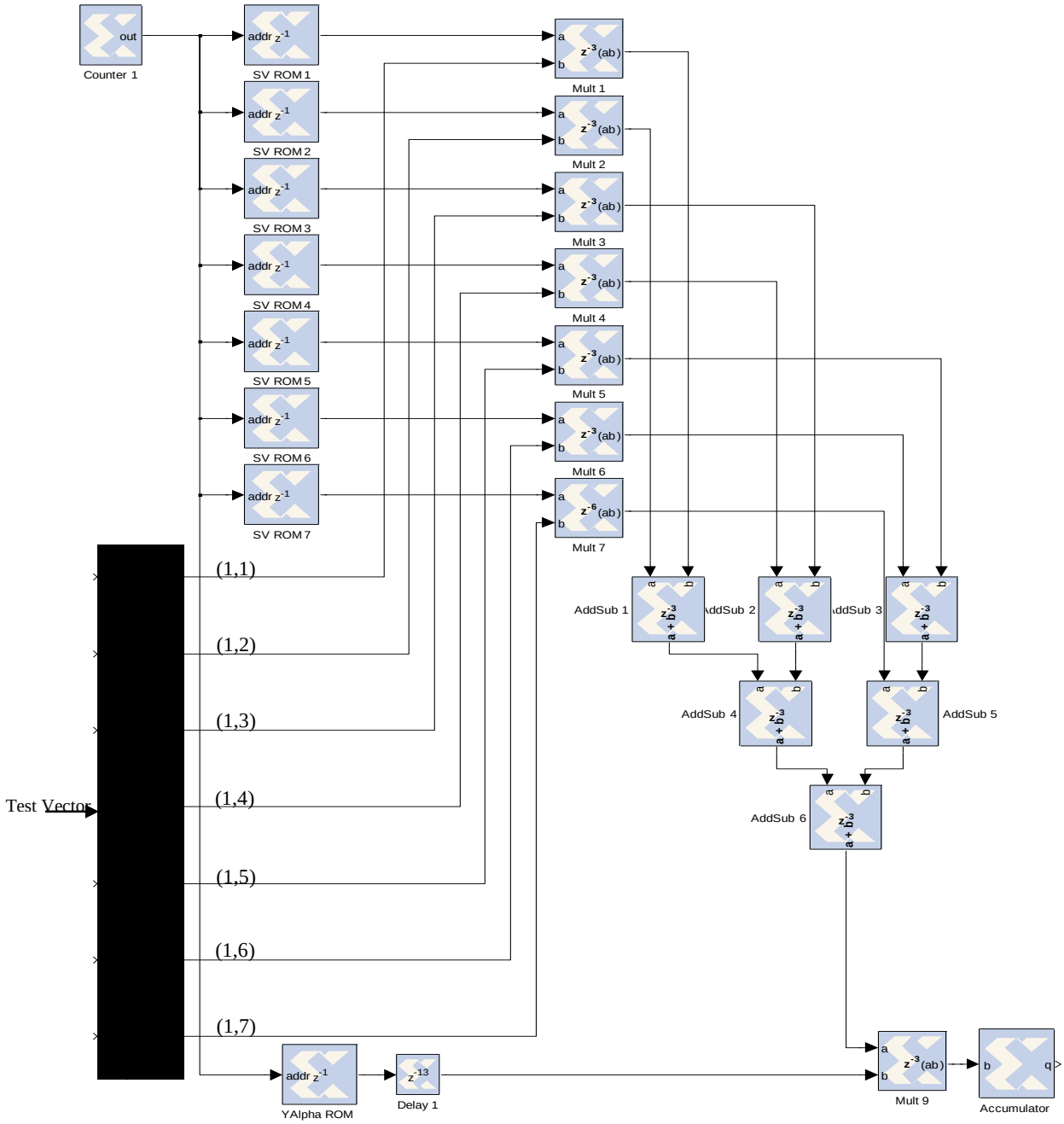
هر چند که طراحی شکل 5-11 از نظر ساختاری یک طرح صحیح و قابل اجراست اما باید در نظر داشت که تمام عملیات ضرب و جمع از خروجی SV ROM ها تا بلوک Mult9 همگی باید در یک کلاک انجام شود، بنابراین باید یک کلاک با دوره تناوب طولانی برای چنین طرحی فراهم شود تا در هر مرحله نتایج عملیات ضرب و جمع به صورت صحیح انجام گردد. این وضعیت سبب می شود تا ماکزیمم فرکانس کاری سیستم به شدت افت کند. در System Generator ابزاری به نام تجزیه و تحلیل زمان[1] وجود دارد که هم به صورت گزارش متنی و هم به صورت گرافیکی کوتاهترین مسیر در سیستم طراحی شده و مسیرهایی که تاخیر آنها بیش از حد مجاز است را مشخص می کند. شکل 5-12 نتیجه حاصل از این آنالیز را نشان می دهد که در آن طولانی ترین مسیر با رنگ تیره تر مشخص شده است. بلوک Accumulator به عنوان جمع کننده سری نتایج حاصلضرب به کار رفته است. این نکته را باید در نظر داشت که تاخیر در طولانی ترین مسیر، شامل تاخیر داخلی بلوکها و تاخیر سیم کشی می باشد. در یک آزمایش آنالیز زمانی با تعداد بیتهای مشخص و FPGA خاص، مشخص شد که این مسیر طولانی ماکزیمم فرکانس کاری سیستم را روی مقدار 40/243MHz محدود می کند. در آنالیز انجام شده فقط 29/5% از تاخیر مسیر، مربوط به سیم کشی گزارش شده است. این مقدار نشان دهنده این است که بلوکهای محاسباتی تاخیر درونی خیلی زیادی دارند. پس باید تکنیکهای خاصی برای کاهش این تاخیر اعمال کرد.

[1] Timing analysis



شکل 5-12: مشخص شدن طولانی ترین مسیر سیستم طراحی شده پس از تجزیه و آنالیز زمان

در بخش 3-7- در مورد تکنیکهای بهینه سازی طراحی در FPGA به طور مفصل شرح داده شد. یکی از روشهایی که برای افزایش فرکانس کاری طرح سخت افزاری در مورد آن توضیح داده شد، روش pipelining بود که در آن با افزودن رجیسترها و بلوکهای تاخیری بین لاجیکهای دیجیتال باعث توزیع ابر پراکندگی بلوکها و افزایش سرعت سیستم می شود. در اینجا نیز برای افزایش فرکانس کاری سیستم با بهره گیری از تکنیک pipelining ساختار قبل را به صورت شکل 5-13 طراحی می کنیم.



شکل 5-13: استفاده از تکنیک pipelining برای افزایش فرکانس کاری سیستم

همانطور که در شکل 5-13مشاهده می شود، تاخیر درونی بلوکهای مربوط به مسیر طولانی شکل 5-12 تغییر کرده است. در این ساختار آزمایشهای متعدد نشان داد که با قرار دادن تاخیر درونی این بلوکها به اندازه 3 کلاک، با استفاده از روش pipelining، بهترین نتیجه از لحاظ فرکانس کاری حاصل می شود. در اینجا این نکته را باید در نظر داشت که با تغییر دادن تاخیر در هر کدام از مسیرهای طرح سخت افزاری، همزمانی آن با سایر مسیرها نیز تغییر می کند. بنابراین باید تاخیر همه مسیرهای موازی با مسیر موردنظر، دقیقاً به همان نسبت تغییر یابد. پس با

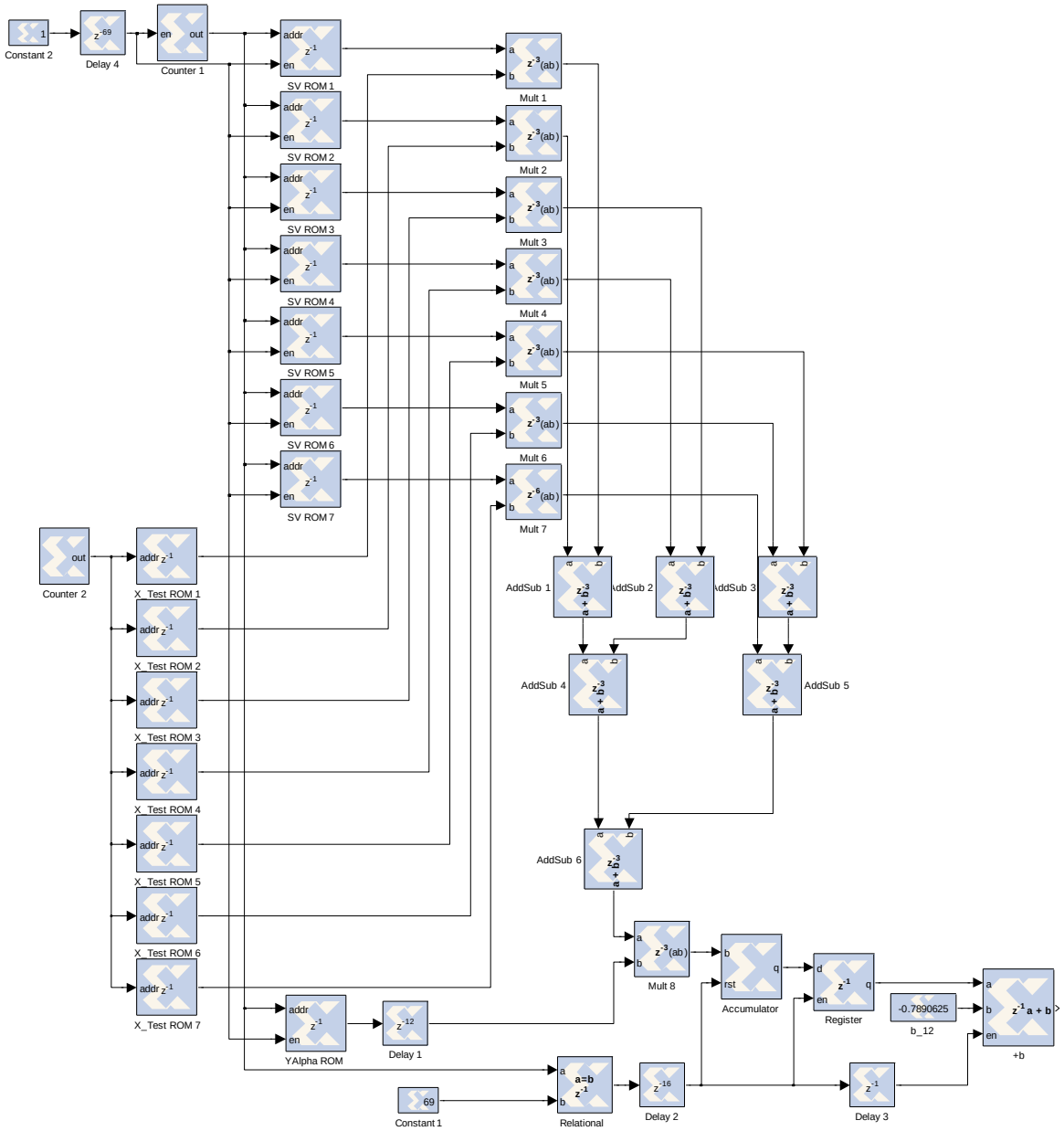
این حساب باید تاخیر همه بلوکهای Mult1 تا Mult7 و Addsub1 تا Addsub6 نیز به اندازه 3 کلاک افزایش پیدا کند. البته تاخیر درونی بلوک Mult7 را باید با دقت بیشتری انتخاب کرد، چرا که خروجی Mult7 باید با خروجی Addsub3 جمع شود، بنابراین باید خروجی آن همزمان با خروجی Addsub3 ظاهر شود. پس باید تاخیر بلوک Mult7 را به اندازه مجموع تاخیرهای بلوکهای Mult6 و بلوک Addsub3 قرار داد که برابر با 6 کلاک می شود.

همچنین به منظور برابر شدن تاخیر در هر یک از شاخه های موازی، بلوک تاخیری Delay1 به اندازه 12 کلاک قرار داده شده است تا عملیات ضرب در Mult9 به صورت همزمان انجام شود. با این تغییر کوچک در ساختار طراحی، ماکزیمم فرکانسی کاری سیستم به طور نسبتاً زیادی افزایش می یابد، به طوری که مقدار آن از 40/243MHz در حالت قبل از اعمال این روش، به 223/164MHz در این مرحله رسید که بیش از 5 برابر افزایش داشته است.

5-3-3- طراحی تابع تصمیم کلاسه بند خطی برای دو کلاس

یکبار دیگر می خواهیم طرح را کامل تر کنیم. در نظر داشته باشید که قرار است تعداد زیادی داده تست را یکجا کلاسه بندی کنیم. می توانیم داده های تست را همانند بردارهای پشتیبان در داخل بلوکهای ROM قرار دهیم و با یک شمارنده ای آنها را فراخوانی کنیم. با توجه به اینکه به ازای هر داده تست، باید ابتدا عملیات ضرب برداری آن داده در بردارهای پشتیبان و سپس ضرب اسکالر در درایه های بردار $\mathbf{y}\alpha$، جمع با ثابت b و در نهایت کلاسه بندی آن داده تست انجام شود تا نوبت به داده تست بعدی برسد و این فرآیند تکرار شود، بنابراین باید دوره تناوب این شمارنده به اندازه طول بزرگترین ماتریس بردار پشتیبان انتخاب شود.

شکل 5-14 طرح مورد نظر را برای کلاسه بندی بین کلاسهای 1 و 2 ارقام فارسی 7 بعدی نشان می دهد. داده های تست در بلوکهای X_Test ROM1 تا X_Test ROM7 قرار گرفته اندکه توسط شمارنده Counter2 با دوره تناوب 70 (به اندازه تعداد بردارهای پشتیبان کلاسهای 1 و 2) آدرس دهی می شوند.



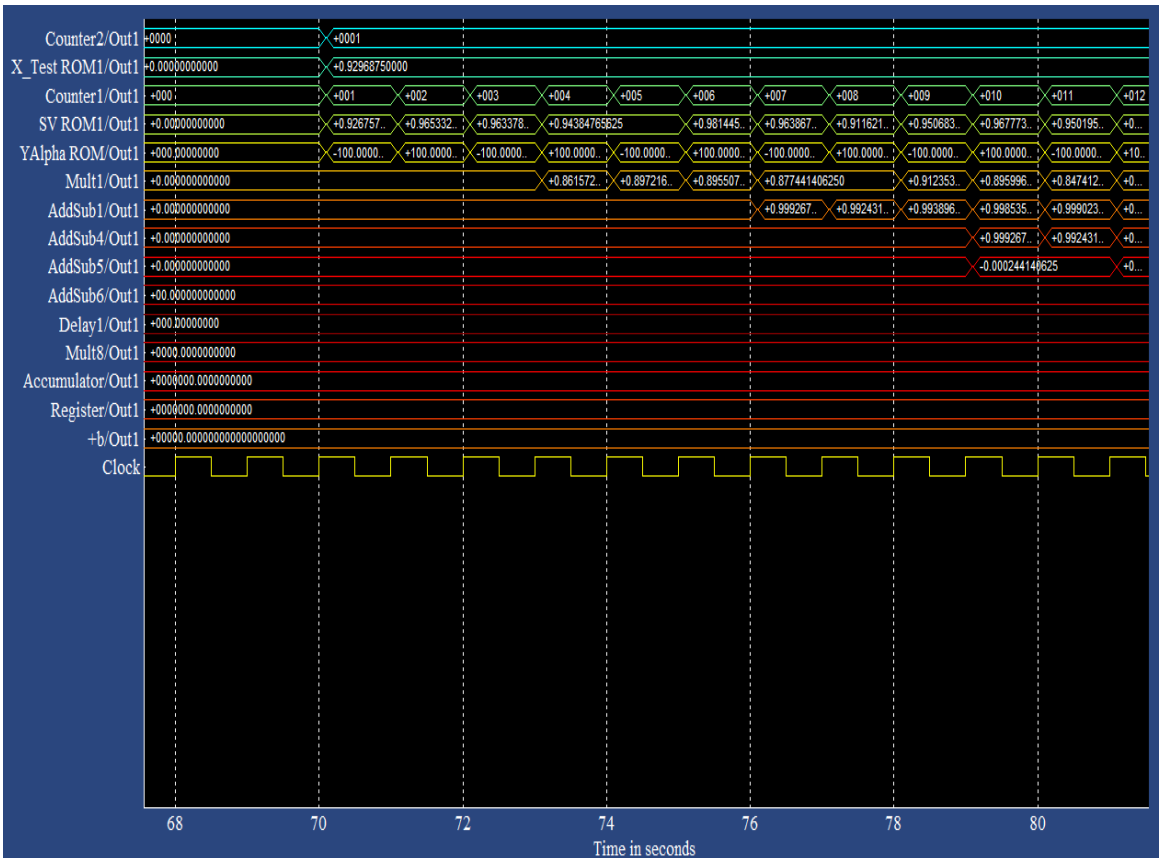
شکل 5-14: طراحی سخت افزاری تابع تصمیم کلاسه بند خطی برای تشخیص کلاسهای 1 و 2 ارقام فارسی

در طراحی شکل 5-14در اولین کلاک سیستم، خروجی هر کدام از ROM ها صفر است و سپس با تغییر خروجی شمارنده ها، خروجی ROMها نیز ظاهر می شود. بنابراین ROM های SV و YAlpha باید تاخیری به اندازه یک دوره تناوب X_Test ROM ها داشته باشند تا همزمان با اولین خروجی در ROM های X_Test، ROM های SV و YAlpha نیز خروجی خود را ظاهر کنند. بلوک Delay4 وظیفه ایجاد این تاخیر را بر عهده دارد.

بلوک Accumulator که به عنوان جمع کننده سری مورد استفاده قرار گرفته است باید بعد از هر بار که یکی از داده های تست در همه بردارهای پشتیبان ضرب برداری و در YAlpha ضرب عددی می شود، مقادیر را یکی یکی جمع کرده و فقط مقدار نهایی را نگه دارد و ریست شود تا دوباره

برای داده تست بعدی همین فرآیند تکرار شود. همانطور که در شکل 5-14 مشاهده می شود، پورت ریست (rst) بلوک Accumulator به همین منظور در نظر گرفته شده است که توسط خروجی بلوک Relational فعال می شود. یعنی در صورتی که خروجی شمارنده Counter1 به شماره آخرین بردار پشتیبان رسید، پس از انجام عملیات ضرب برداری در بردار تست ورودی و ضرب عددی در YAlpha، Accumulator ریست می شود، در حالی که آخرین مقدار آن باید نگه داشته شود. بلوک Register وظیفه نگه داشتن آخرین مقدار را بر عهده دارد که بعد از ریست شدن Accumulator و با یک تاخیر، مقدار مورد نظر را به بلوک b+ می دهد. وظیفه بلوک b+ اضافه کردن مقدار ثابت b به آخرین مقدار Accumulator قبل از ریست شدن است، یعنی باید خروجی Register را با مقدار ثابت b جمع کند. بنابراین این بلوک باید همزمان با خروجی Register، که یک کلاک تاخیر درونی دارد، فعال شود. بلوک تاخیری Delay3 به همین منظور استفاده شده است.

اجازه دهید یک بار دیگر فرآیند فوق را از نظر کلاک زمانی مرور کنیم و بررسی نماییم که در هر کلاک چه اتفاقی رخ می دهد. شکل 5-15 نمودار سیگنالهای خروجی بخشهای مختلف طرح در فاصله زمانی کلاک 68 تا 80 را نشان می دهد.



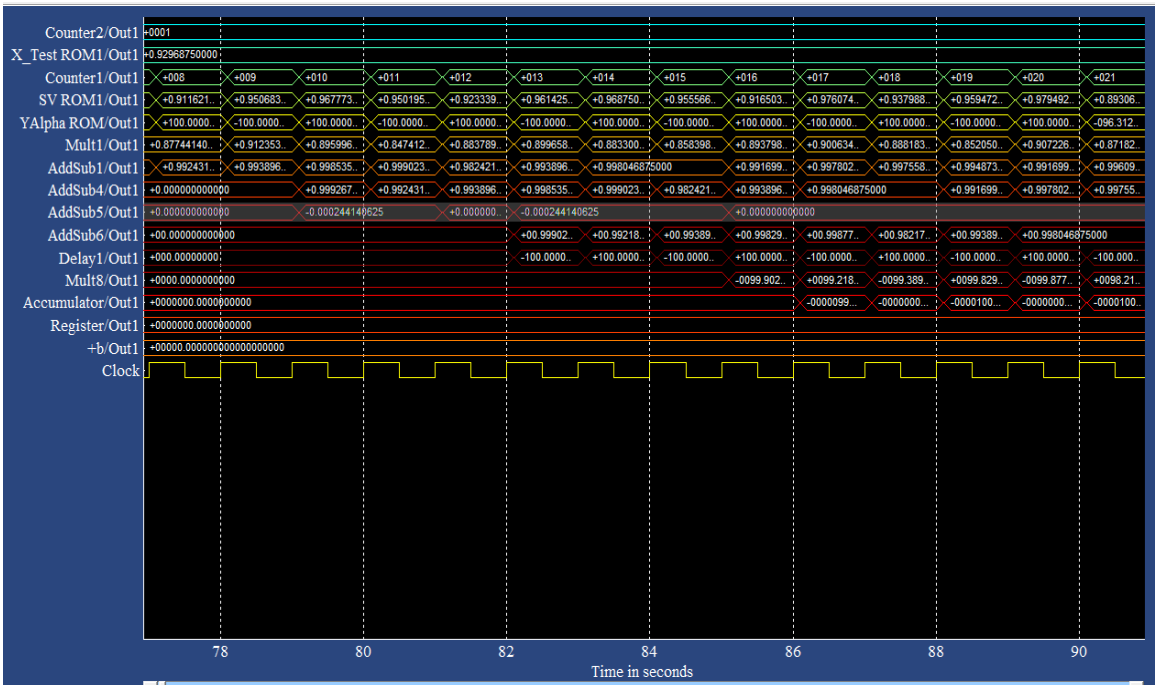
شکل 5-15: سیگنالهای زمانی خروجی بلوکهای مختلف مربوط به ساختار شکل 5-14 از کلاک 68 تا 80

برای درک بهتر، ساختار شکل 5-14 و سیگنالهای زمانی شکل 5-15 را همزمان بررسی می کنیم. همانطور که در شکل 5-15 ملاحظه می شود، تا قبل از کلاک 70 (اندازه تعداد بردارهای پشتیبان) خروجی همه بلوکها صفر است و هیچ اتفاقی نیفتاده است. همانطور که گفتیم این تاخیر به دلیل همزمان شدن خروجی ROM های SV و ROM های X_Test اجتناب ناپذیر است. از کلاک 70 شمارنده Conuter2 شروع به شمردن می کند. بلافاصله داده های ROM های X_Test نیز در خروجی آنها ظاهر می شوند (در اینجا سیگنال خروجی X_Test ROM1 به عنوان نمونه در شکل آورده شده است). دوره تناوب شمارنده Conuter2 به اندازه تعداد بردارهای پشتیبان یعنی 70 است. همزمان با شمارنده Counter2، شمارنده Counter1 نیز شمارش خود را آغاز می کند و داده های ROMهای SV و YAlpha را در خروجی آنها قرار می دهد (در اینجا نیز سیگنال خروجی SV ROM1 به عنوان نمونه در شکل قرار داده شده است). در این حالت خروجی های SV ROM1 و X_Test ROM1 همزمان وارد بلوک Mult1 می شوند. همانطور که انتظار می رفت سیگنال خروجی Mult1 پس از 3 کلاک (به اندازه تاخیر درونی آن که با روش pipelining اعمال شد) ظاهر می شود. همین روند تا خروجی Mult2 نیز انجام می شود. سیگنالهای خروجی Mult1 و Mult2 به طور همزمان وارد

Addsub1 می شوند و در این بلوک نیز پس از انجام عمل جمع و تاخیری برابر با 3 کلاک، نتیجه خروجی وارد بلوک Addsub4 می شود.

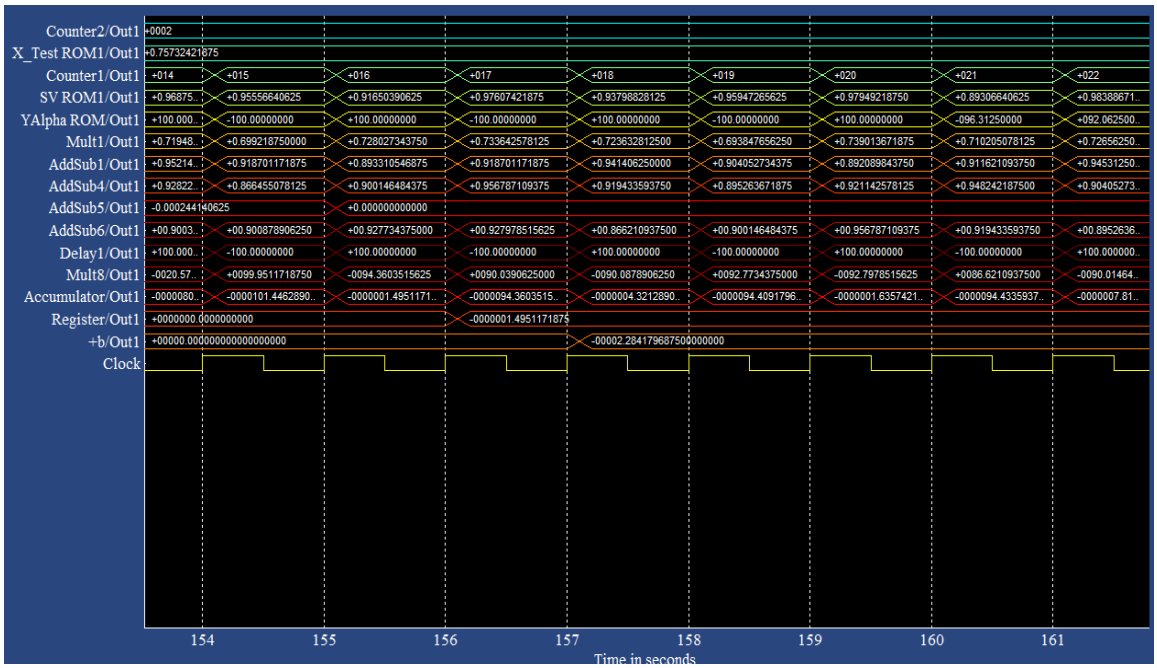
تا اینجای کار پس از شروع شمارش شمارنده ها، 6 تاخیر در هر یک از بلوکهای موازی پشت سر گذاشته شده است. بنابراین انتخاب تاخیر 6 کلاک برای بلوک Mult7 موجب شده است که خروجی بلوکهای موازی Addsub4 و Addsub5 ، پس از 3 تاخیر دیگر، به طور همزمان وارد بلوک Addsub6 شود. پس تا قبل از بلوک Addsub6 ، 9 کلاک تاخیر پشت سر گذاشته شده است. در بلوک Addsub6 نیز 3 تاخیر درونی وجود دارد، بنابراین عملیات جمع در این بلوک پس از 3 تاخیر دیگر انجام می شود، پس جمع تاخیرهای ایجاد شده تا خروجی Addsub6 به عدد 12 می رسد. این در حالی است که خروجی Addsub6 باید در مقدار عددی خروجی YAlpha ROM به صورت موازی و همزمان ضرب شود، بنابراین باید مقدار خروجی YAlpha ROM با همین میزان تاخیر وارد بلوک Mult8 شود، یعنی انتخاب مقدار 12 برای بلوک تاخیری Delay1 انتخاب مناسبی است.

در ادامه سیگنالهای زمانی خروجی بلوکها نشان داده شده است. همانطور که توضیح داده شد و انتظار می رفت خروجی های Delay1 و Addsub6 همزمان با هم در کلاک 82 ظاهر شده اند و وارد Mult8 می شوند. در بلوک Mult8 نیز پس از 3 تاخیر، عملیات ضرب انجام شده و خروجی آن به دست می آید.



شکل 5-16: سیگنالهای زمانی خروجی بلوکهای مختلف مربوط به ساختار شکل 5-14 از کلاک 78 تا 90

از این مرحله به بعد بلوک Accumulator در هر کلاک، خروجی Mult8 را با مقدار قبلی آن جمع می کند و این کار را ادامه می دهد تا نوبت به آخرین بردار پشتیبان برسد. با توجه به اینکه Accumulator در کلاک 86 شروع به عملیات جمع کرده و تعداد بردارهای پشتیبان 70 است، بنابراین طبق پیش بینی بلوک Register باید در کلاک 156 فعال شود تا آخرین مقدار Accumulator را نگه دارد. همانطور که گفته شد بلوکهای Relational و Delay2 این وظیفه را بر عهده دارند. برای درک بهتر، سیگنالهای زمانی طرح سخت افزاری را در فاصله زمانی کلاک 154 تا 161 نشان می دهد.



شکل 5-17: سیگنالهای زمانی خروجی بلوکهای مختلف مربوط به ساختار شکل 5-14 از کلاک 154 تا 161

در کلاک 155 آخرین مقدار Accumulator مربوط به اولین داده تست ظاهر می شود و در کلاک بعدی، این مقدار توسط Register نگه داشته می شود. در همین زمان بلوک Accumulator ریست می شود، تا شروع به جمع کردن مقادیر مربوط به داده تست بعدی کند. در کلاک بعدی مقداری که توسط بلوک Register نگه داشته شد، با مقدار ثابت b جمع می شود و نتیجه آن در کلاک 157 در خروجی بلوک b+ ظاهر می شود. خروجی بلوک b+ نتیجه تابع تصمیم کلاسه بند ماشین بردار پشتیبان خطی است.

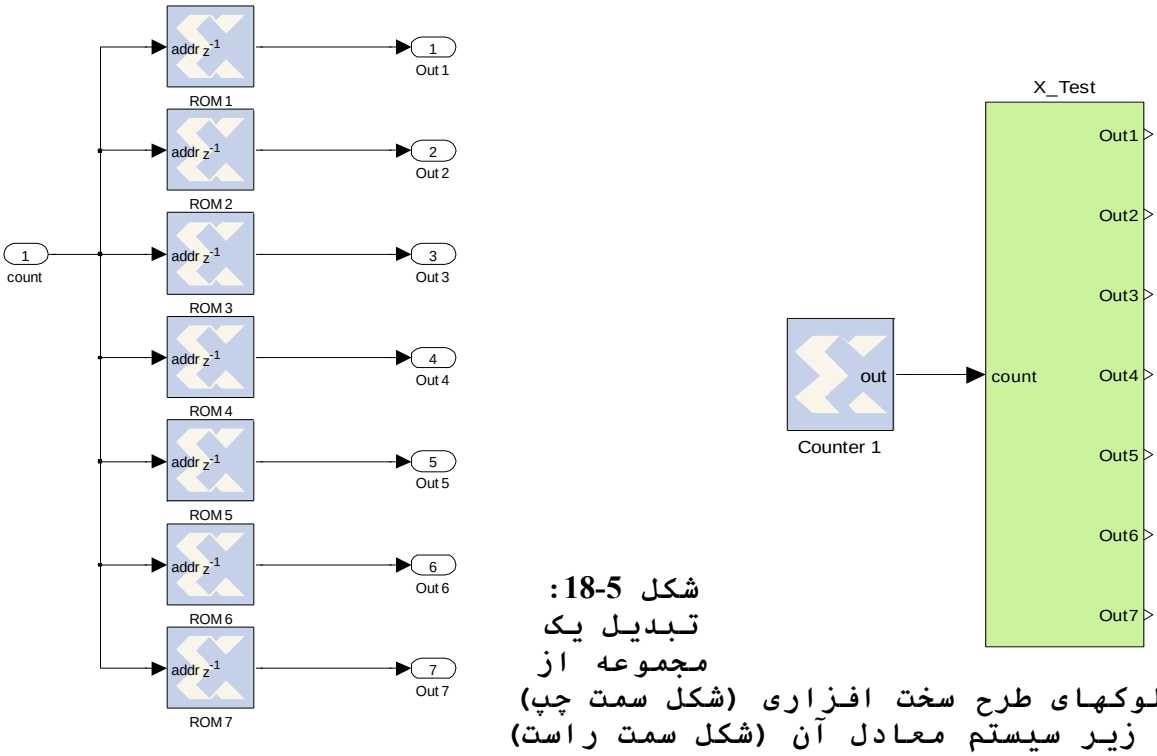
5-3-4- کلاسه بندی همزمان سه کلاس

می خواهیم یک بار دیگر و این بار با هدف کلاسه بندی همزمان چندین داده تست بین سه کلاس مختلف، طرح را کامل تر کنیم. به بخش 5-2- اشاره می کنیم که در آن داده های ارقام فارسی 1 ، 4 و 8 با بردار ویژگی 7 بعدی به ترتیب در کلاسهای 1 ، 2 و 3 قرار داده شدند. قرار است این داده ها با طراحی سخت افزاری ماشین بردار پشتیبان خطی کلاسه بندی شوند. نتایج آموزش سیستم مورد نظر توسط ماشین بردار پشتیبان خطی قبلاً در جدول 5-1 نشان داده شد.

برای پیاده سازی سخت افزاری کلاسه بند خطی مورد نظر، فرض کنیم از هر کلاس 200 داده، مستقل از داده های آموزشی، به صورت رندم جهت تست در اختیار داریم، یعنی مجموعا 600 داده تست برای سه کلاس. داده های تست را

همگی در داخل ماتریسی به نام X_Test قرار می دهیم و مانند بخش قبل از طریق بلوکهای X_Test ROM آنها را یکی یکی فراخوانی می کنیم. چون هدف طراحی کلاسه بندی دوتایی و به صورت همزمان است، باید سه کلاسه بند دوتایی را به صورت موازی طراحی کنیم. بهتر است جهت جلوگیری از شلوغی طرح از نظر گرافیکی، هر مجموعه ای از بلوکها را در داخل یک زیر سیستم[1] قرار دهیم. مثلا برای اینکه بلوکهای X_Test ROM را در داخل یک زیرسیستم قرار دهیم، کافیست طبق شکل 5-18 یک پورت ورودی به نام Count جهت اتصال به شمارنده و هفت پورت خروجی به نامهای Out1 تا Out7 جهت اتصال خروجی ROM ها به بیرون برای آن قرار دهیم، این زیر سیستم را X_Test نامگذاری می کنیم.

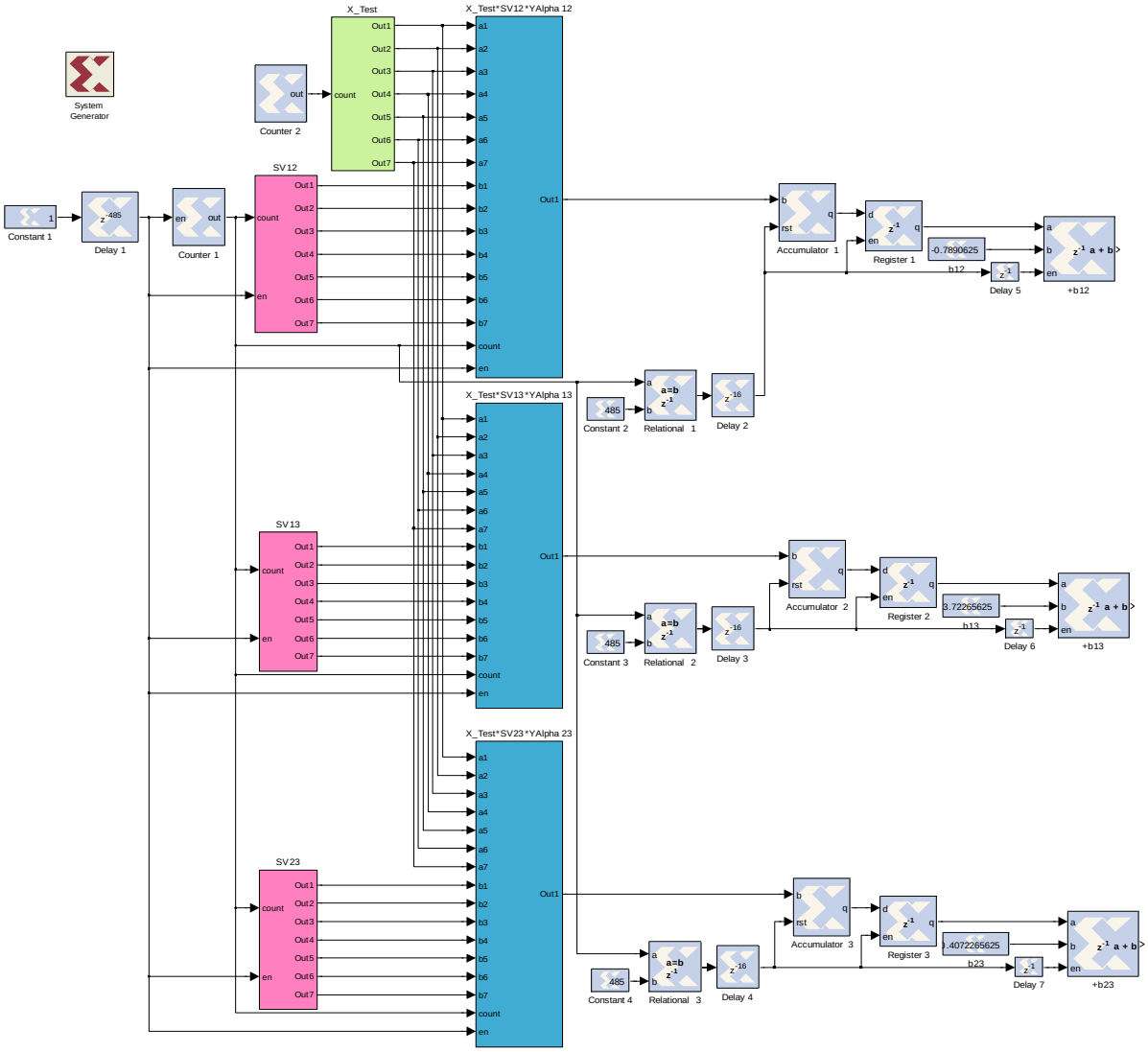
[1] Subsytem



شکل 5-18: تبدیل یک مجموعه از بلوکهای طرح سخت افزاری (شکل سمت چپ) به زیر سیستم معادل آن (شکل سمت راست) جهت فشرده کردن طرح گرافیکی

برای هر کدام از مجموعه ROM های ذخیره کننده بردارهای پشتیبان کلاسهای 1و2 ، 1و3 ، 2و3 نیز یک زیر سیستم، به همین ترتیبی که برای بردارهای تست انجام شد، قرار می دهیم و آنها را به ترتیب SV12، SV13، SV23 نامگذاری می کنیم. مجموعه بلوکهایی که عملیات ضرب برداری داده های تست در بردارهای پشتیبان و ضرب عددی در YAlpha را انجام می دهند، نیز در داخل زیر سیستمی به نام X_Test*SV*YAlpha قرار می دهیم.

بدین ترتیب طرح سخت افزاری مورد نظر برای کلاسه بندی همزمان بین سه کلاس به صورت شکل 5-19 در می آید.

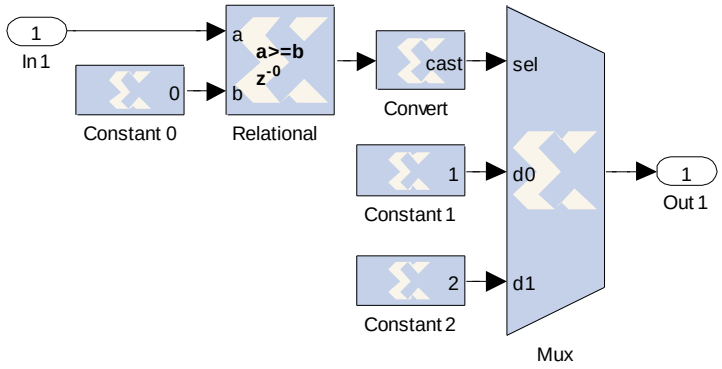


شکل 5-19: ساختار تابع تصمیم کلاسه بند ماشین بردار پشتیبان خطی برای کلاسه بندی دوتایی ارقام فارسی 7 بعدی

داده های تست ورودی به طور همزمان در سه مسیر موازی جهت کلاسه بندی دوتایی قرار می گیرند و مقدار تابع تصمیم هر مسیر در خروجی بلوکهای b+ همزمان با هم ظاهر می شود. اگر هدف فقط کلاسه بندی بین دو کلاس 1 و 2 بود، خیلی راحت از علامت عدد خروجی بلوک b12+ می توانستیم کلاس مورد نظر را تشخیص دهیم. اما اینجا هدف کلاسه بندی بین سه کلاس است، پس باید وقت بیشتری برای ادامه طراحی بگذاریم.

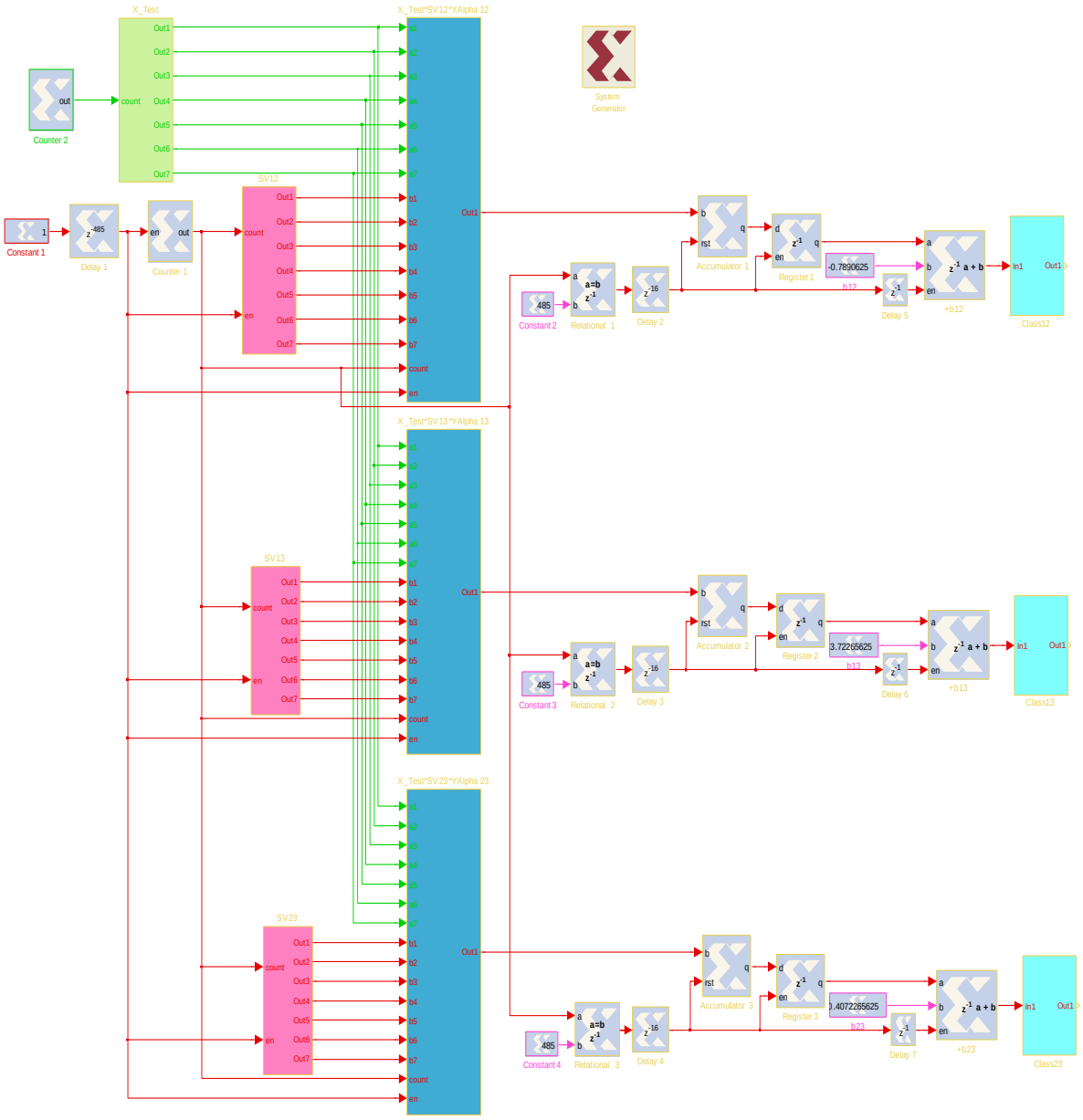
در بخش 2-4- گفته شد در روش کلاسه بندهای دوتایی کلاسی را انتخاب کنیم که در مجموع رای بیشتری داشته باشد. برای پیاده سازی سخت افزاری این پروسه ابتدا باید در هر مسیر بعد از بلوک های b+ با توجه به علامت خروجی آن، کلاس مورد نظر را مشخص کنیم. به عنوان مثال در مسیر اول، که برای کلاسه بندی بین کلاسهای 1 و 2 است، اگر علامت خروجی بلوک b12+ منفی باشد، داده مربوط به کلاس 1 و

اگر مثبت باشد، داده مربوط به کلاس 2 است. برای پیاده سازی این وضعیت، بعد از بلوکهای b+ ساختار شکل 5-20 را طراحی می کنیم، که در آن ورودی In1 از خروجی بلوک b12+ می آید. مقایسه کننده Relational مقدار ورودی را با مقدار ثابت صفر که در بلوک Constant0 قرار گرفته است مقایسه می کند؛ اگر بزرگتر یا مساوی صفر باشد، بلوک Mux (مالتی پلکسر) مقدار ثابت 1 را از بلوک Constant1 می گیرد و در خروجی قرار می دهد، در غیر این صورت مقدار ثابت 2 از بلوک Constant2 به خروجی مالتی پلکسر فرستاده می شود.



شکل 5-20: طرح سخت افزاری برای مشخص کردن کلاس 1 و 2

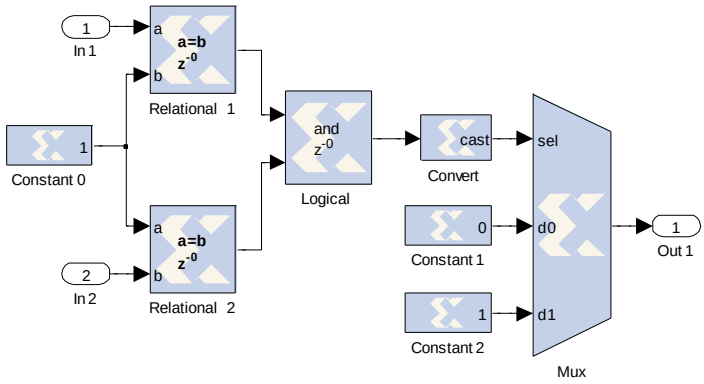
این مجموعه را در درون زیر سیستمی به نام Class12 قرار می دهیم. برای دو مسیر موازی دیگر نیز به همین ترتیب طراحی انجام می شود، با این تفاوت که مقادیر بلوکهای Connstant1 و Constant2 تغییر می کند. مثلاً در مسیر کلاسه بندی کلاسهای 1 و 3، برای بلوک Constant1 مقدار 1 و برای بلوک Constant2 مقدار 3 باید قرار داده شود، و در مسیر کلاسه بندی کلاسهای 2 و 3 برای بلوک Constant1 مقدار 2 و برای بلوک Constant2 مقدار 3 مطلوب است. خروجی هر کدام از این زیر سیستمها عدد مربوط به یکی از کلاسهاست. با این تغییرات، ساختار طرح به صورت شکل 5-21 در می آید.



شکل 5-21: ساختار کلاسه بند ماشین بردار پشتیبان خطی بعد از اضافه کردن زیر سیستم های Class xx

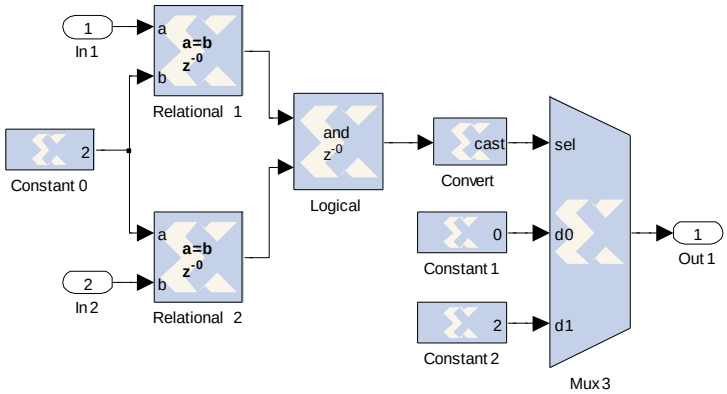
در مرحله بعد هدف این است که سیستمی طراحی کنیم که تشخیص دهد کدام کلاس بیشترین رای را داشته است. چون کلاً سه کلاس داریم، کلاسی که دو رای داشته باشد قطعاً بیشترین رای را داشته و به عنوان کلاس منتخب تشخیص داده می شود. شکل 5-22 طراحی مربوط به این مرحله را نشان می دهد. ورودی های In1 و In2 از خروجی بلوکهای Class12 و Class13 در ساختار شکل 5-21 گرفته شده است. سناریو به این صورت است که در صورتی که خروجی هر دو بلوک Class12 و Class13 برابر 1 باشد، بلوکهای Relational1 و Relational2 مقدار Constant0 یعنی عدد 1 را در خروجی خود ظاهر می کنند. بلوک Logical این دو خروجی را در هم ضرب دیجیتال یعنی and می کند و به ورودی Mux می دهد. تنها در صورتی که هر دو مقدار خروجی

بلوکهای Class12 و Class13 برابر 1 باشد، خروجی Logical یا ورودی Mux یک خواهد شد. بلوک Mux هم با توجه به مقدار ورودی آن، یکی از دو مقدار Constant1 (عدد 0) و یا Constant2 (عدد 1) را در خروجی خود ظاهر می کند. در صورتی که خروجی این مرحله برابر 1 باشد به این معناست که در کلاسه بندی بین کلاسهای 1 و 2 و همچنین بین کلاسهای 1 و 3، کلاس 1 انتخاب شده است. و این به معنای بیشترین رای به کلاس 1 در کل سیستم است، پس می توان گفت که داده مورد نظر در کلاس 1 قرار دارد. این مرحله از طراحی را زیر سیستم Class1 نامگذاری می کنیم.



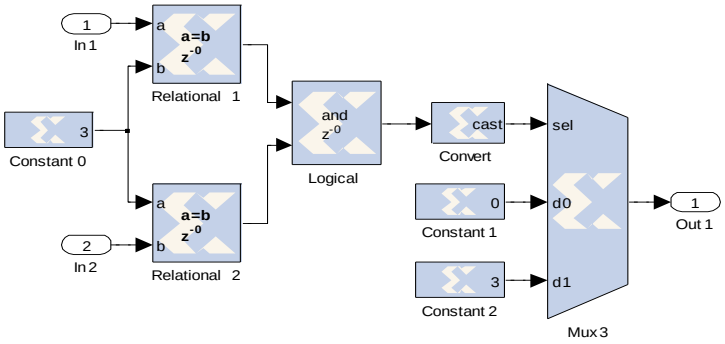
شکل 5-22: زیر سیستم Class1

همین فرم طراحی را به صورت موازی برای خروجی بلوکهای Class12 و Class23 نیز انجام می دهیم تا مشخص شود که آیا کلاس 2 بیشترین رای را داشته است یا خیر. شکل 5-23 طراحی زیر سیستم Class2 را نشان می دهد. در اینجا مقدار عددی بلوکهای Constant0 و Constant2 تغییر کرده و به عدد 2 تبدیل شده است، در حالی که سناریوی قبلی حفظ شده است. بلوکهای این قسمت از طراحی را در یک زیر سیستم جدید به نام Class2 قرار می دهیم.



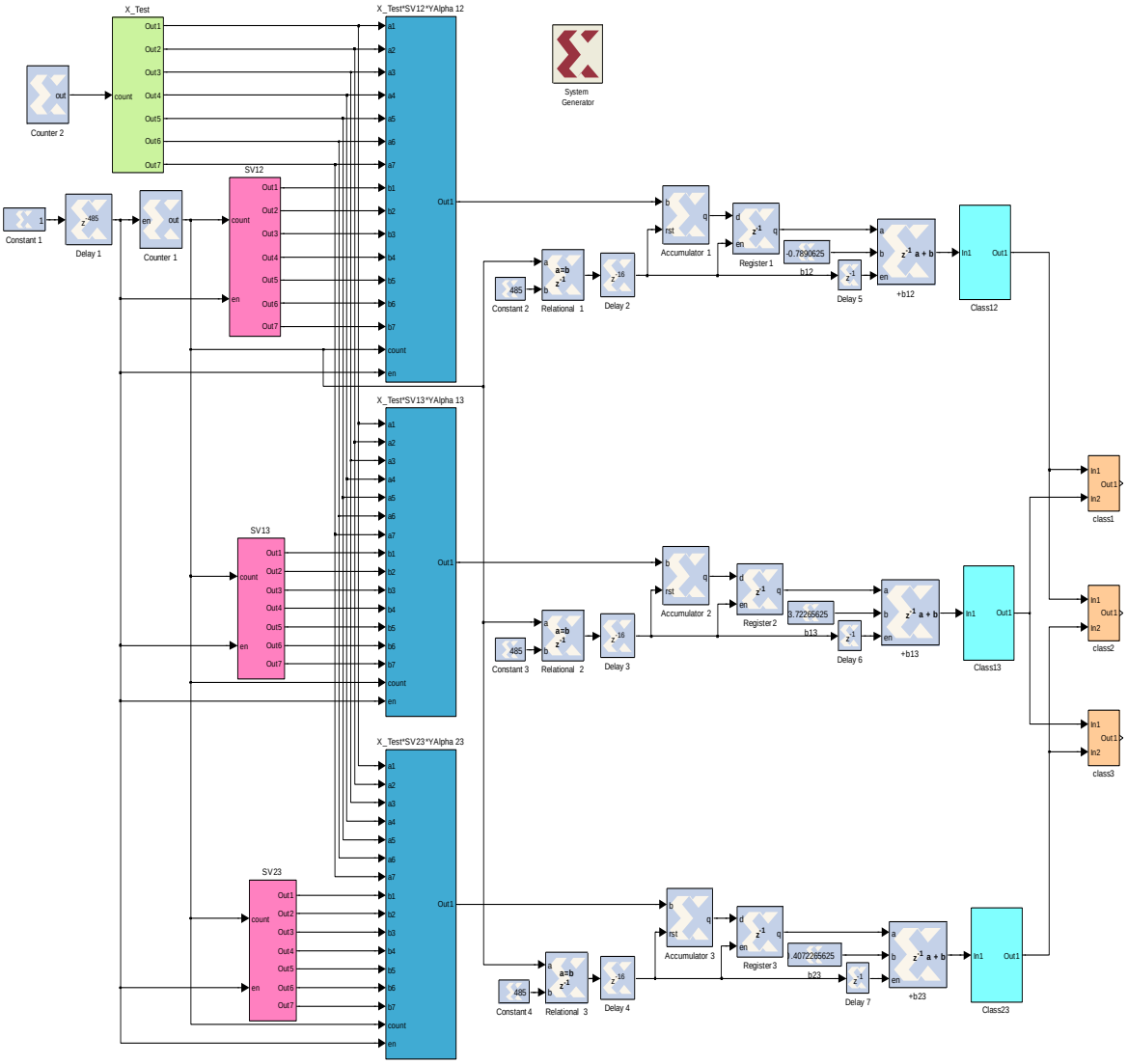
شکل 5-23: زیر سیستم Class2

یک بار دیگر فرم طراحی فوق را به صورت موازی برای خروجی بلوکهای Class13 و Class23 نیز انجام می دهیم. در حالی که سناریوی قبلی باز هم حفظ شده است. شکل 5-24 طراحی زیر سیستم Class3 برای تشخیص کلاس 3 را نشان می دهد. در اینجا نیز مقدار عددی بلوکهای Constant0 و Constant2 به عدد 3 تغییر داده شده است. این مجموعه را نیز در قالب زیر سیستم Class3 نامگذاری می کنیم.

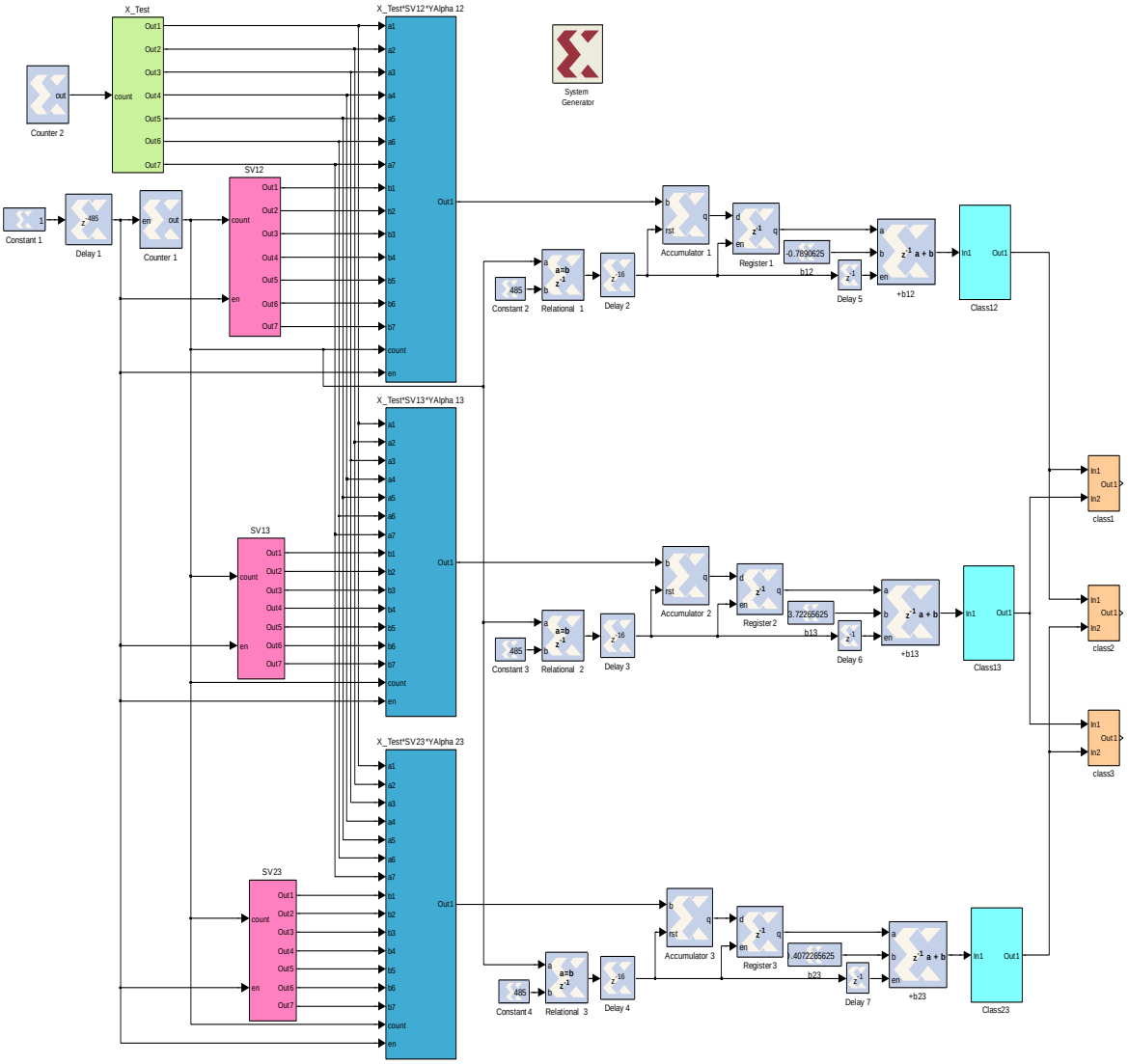


شکل 5-24: زیر سیستم Class3

با اضافه شدن زیر سیستم های Class1، Class2 و Class3 ساختار کلی طراحی سیستم به صورت شکل 5-25

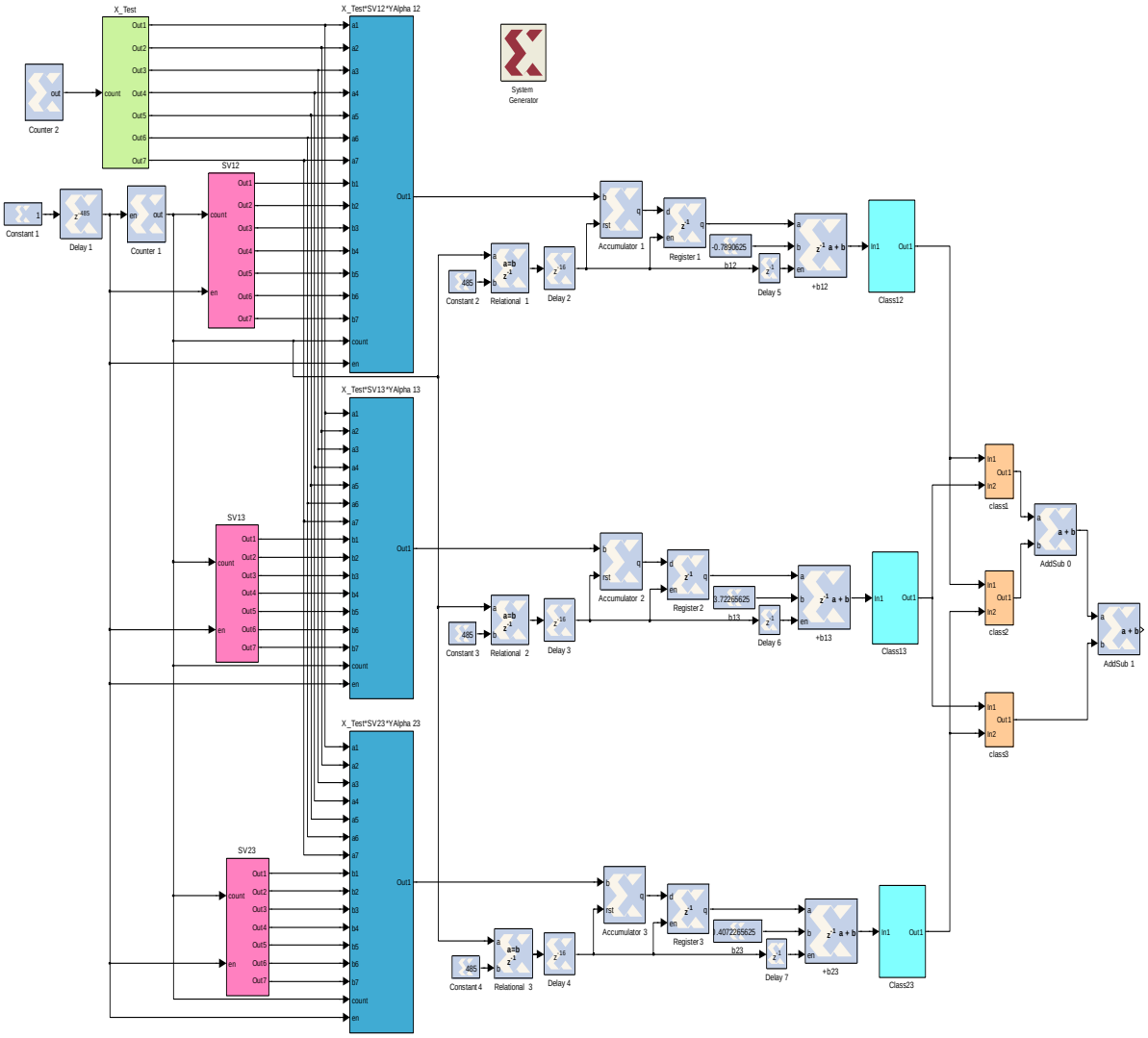

X_Test
Counter 2
SV12
Constant 1
Delay 1
Counter 1
X_Test*SV12*YAlpha 12
System Generator
Accumulator 1
Register 1
Delay 5
+b12
Class12
Constant 2
Relational 1
Delay 2
X_Test*SV13*YAlpha 13
SV13
Accumulator 2
Register 2
Delay 6
+b13
Class13
Constant 3
Relational 2
Delay 3
X_Test*SV23*YAlpha 23
SV23
Accumulator 3
Register 3
Delay 7
+b23
Class23
Constant 4
Relational 3
Delay 4
class1
class2
class3

در می آید.



شکل 5-25: ساختار کلی طرح کلاسه بندی سه کلاس، بعد از اضافه شدن زیر سیستمهای Class x

با توجه به اینکه در هر لحظه از سه خروجی زیر سیستمهای Class1، Class2 و Class3 فقط یکی از آنها کلاس مورد نظر را در خروجی خود ظاهر می کند و بقیه خروجیها صفر خواهند بود، بنابراین برای اینکه مسیر خروجی فقط شامل یک مسیر باشد که همان مسیر نیز کلاس منتخب در بین سه کلاس را مشخص کند، کافیست این سه خروجی را در هر لحظه با هم جمع کنیم. این قسمت در شکل 5-26 با اضافه کردن بلوکهای Addsub0 و Addsub1 انجام می شود.

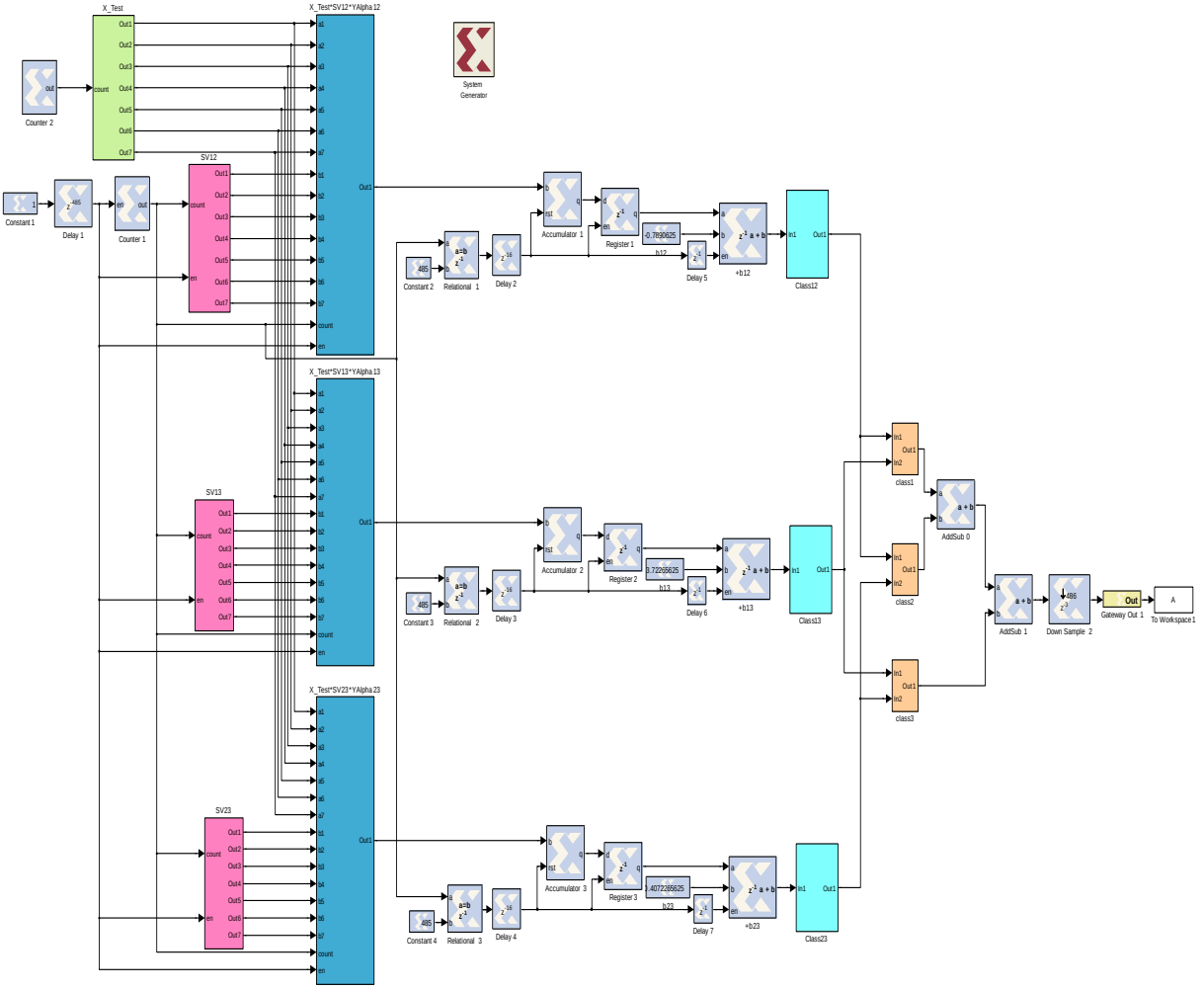


شکل 5-26: اضافه شدن بلوکهای Addsub0 و Addsub1 برای یکسو کردن مسیرهای کلاسه بندیهای دوتایی

در نهایت در خروجی این مرحله، در هر کلاک از سیستم یکی از اعداد 0، 1، 2، 3 ظاهر می شود. اعداد 1، 2 و 3 کلاس مورد نظر را نشان می دهند و عدد 0 نشان دهنده این است که یا هنوز ورودی سیستم وارد نشده است، یا ورودی سیستم اعمال شده اما در قسمت تاخیر اولیه هستیم و یا اینکه داده مورد نظر ورودی به هیچ یک از کلاسهای فوق تعلق ندارد. حالت آخری قبلاً در شکل 2-10 نشان داده شده است.

اگر طرح سخت افزاری را تا همین جای کار رها کنیم و خروجی را از بلوک Addsub1 بگیریم، این خروجی به صورت یک برداری در می آید که در هر کلاک سیستم یک مقدار خواهد داشت و در نتیجه طول بسیار زیادی خواهد شد. اما اصول کار این است که به ازای هر بردار تست ورودی یکی از اعداد 1، 2 یا 3 برای مشخص شدن کلاسی که آن بردار تست متعلق به آن است، در خروجی داشته باشیم. از آنجا که

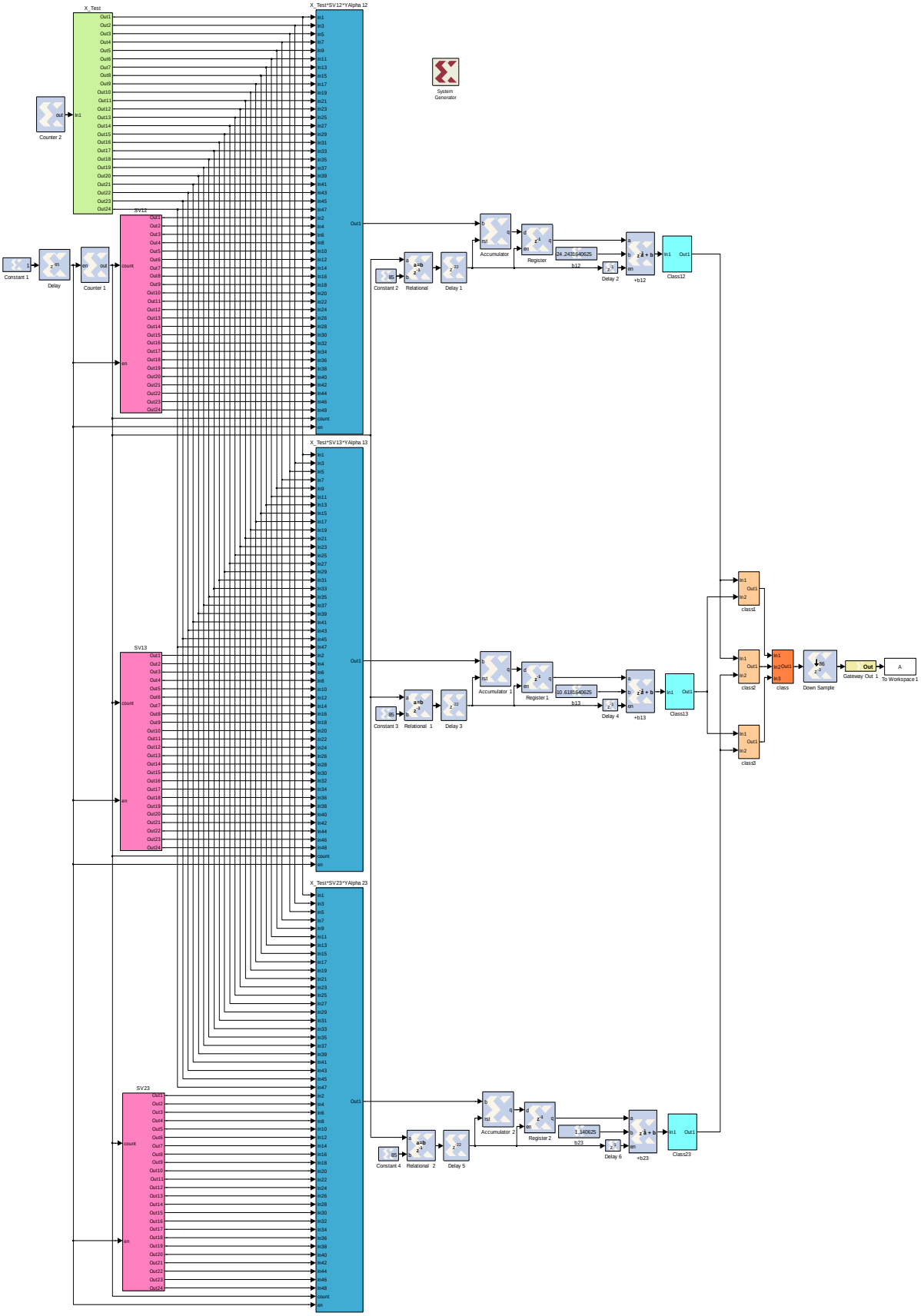
بردارهای تست ورودی توسط شمارنده Counter2 فراخوانی می شوند و دوره تناوب این شمارنده به اندازه بیشترین تعداد بردارهای پشتیبان در هر یک از کلاسه بندهای دوتایی است، بنابراین باید در خروجی کلی سیستم از یک بلوک Down Sample استفاده کنیم که نرخ نمونه برداری آن به اندازه دوره تناوب Counter2 باشد. در این صورت بازای هر بردار تست ورودی فقط یکی از درایه های بردار خروجی یکی از اعداد 1، 2 یا 3 را در بر خواهد گرفت. شکل 5-27 ساختار نهایی کلاسه بند ماشین بردار پشتیبان خطی برای کلاسه بندی موازی ارقام فارسی 1 ، 4 و 8 ، با بردار ویژگی 7 بعدی، از یکدیگر نشان می دهد. نتیجه نهایی طرح توسط بلوک GatewayOut به WorkSpace محیط MATLAB انتقال داده می شود و در بردار A ذخیره می شود.



شکل 5-27: ساختار نهایی کلاسه بند ماشین بردار پشتیبان خطی برای کلاسه بندی موازی ارقام فارسی 1 ، 4 و 8 با بردارهای ویژگی 7 بعدی

5-3-5- تغییر ابعاد بردار ویژگی

در بخش 5-2-1- ماشین بردار پشتیبان خطی برای کلاسه بندی ارقام فارسی 7 و 24 بعدی در MATLAB اجرا شد و نتایج آن در جدول 5-1 و جدول 5-2 نشان داده شد. از ابتدای بخش 5-3- تا اینجا به دلیل سادگی، مراحل طراحی سخت افزاری کلاسه بند خطی برای کلاسه بندی ارقام فارسی 7 بعدی توضیح داده شد. اما همانطور که قبلاً به آن اشاره شد، ساختار سخت افزاری کلاسه بند مورد نظر طوری طراحی شده است که به راحتی امکان تغییر ابعاد بردار ویژگی در آن نیز وجود دارد، بدون اینکه کلیات طراحی دچار تغییر شود. در این تحقیق ساختار کلاسه بند ماشین بردار پشتیبان خطی برای کلاسه بندی داده هایی با بردارهای ویژگی 24 بعدی نیز طراحی شده است. کافیست که در زیر سیستمهای X_Test و SV xx تعداد ROM ها را از 7 به 24 افزایش دهیم. در این صورت به دلیل افزایش تعداد بلوکهای Addsub و Mult، عملیات ضرب برداری در زیر سیستمهای X_Test*SVxx*YAlpha xx با تاخیر بیشتری مواجه می شود. بقیه ساختار طرح بدون تغییر باقی می ماند. شکل 5-28 ساختار نهایی کلاسه بند ماشین بردار پشتیبان خطی برای کلاسه بندی ارقام فارسی 24 بعدی نشان می دهد. برای سادگی بلوکهای Addsub0 و Addsub1 در انتهای طراح، در زیر سیستم Class قرار داده شده است.



شکل 5-28: ساختار کلاسه بند ماشین بردار پشتیبان خطی برای کلاسه بندی ارقام فارسی 24 بعدی

5-3-6- ماشین بردار پشتیبان غیرخطی

در بخش 5-2-2- ماشین بردار پشتیبان غیرخطی با استفاده از تابع کرنل گوسی اجرا شد و نتایج آن در جدول 5-3 و جدول 5-4 آورده شد. می خواهیم ساختار سخت افزاری تابع تصمیم کلاسه بند ماشین بردار پشتیبان غیرخطی با تابع کرنل گوسی را طراحی کنیم. اجازه دهید یک بار دیگر رابطه تابع تصمیم کلاسه بند ماشین بردار پشتیبان غیرخطی با تابع کرنل گوسی را بررسی کنیم:

$$d(\mathbf{x}) = \mathbf{w_0}^T \mathbf{x} + b_0 = \sum_{i=1}^{SV} \alpha_i y_i \, exp(-\gamma \|\mathrm{x} - \mathrm{x}_i\|^2) + b \qquad (3\text{-}5)$$

که در آن γ طبق رابطه (5-1) تعریف می شود.

به دلیل محدویت های ساختاری در FPGA، بهتر است قبل از پیاده سازی رابطه فوق بر روی FPGA، آن را کمی ساده تر کنیم. هچنین تابع نرم[1] در شبیه ساز System Generator وجود ندارد، پس باید به نحوی آن را به صورت عملیات جمع و ضرب قابل پیاده سازی بر روی FPGA تبدیل کرد. برای این منظور فرض کنیم A یک بردار به صورت زیر باشد:

$$A = [a_1 \;\; a_2 \;\; ...] \qquad (4\text{-}5)$$

تابع نرم A به صورت زیر تعریف می شود:

$$\|A\| = \left(|a_1|^2 + |a_2|^2 + ...\right)^{1/2} \qquad (5\text{-}5)$$

بنابراین مجذور نرم A به صورت زیر در می آید:

$$\|A\|^2 = \left(|a_1|^2 + |a_2|^2 + ...\right) \qquad (6\text{-}5)$$

یا به عبارتی:

$$\|A\|^2 = \left(a_1^2 + a_2^2 + ...\right) \qquad (7\text{-}5)$$

با استفاده از رابطه (5-7) به راحتی می توان تابع $\|\mathrm{x} - \mathrm{x}_i\|^2$ را فقط با استفاده از توابع جمع و ضرب پیاده سازی کرد.

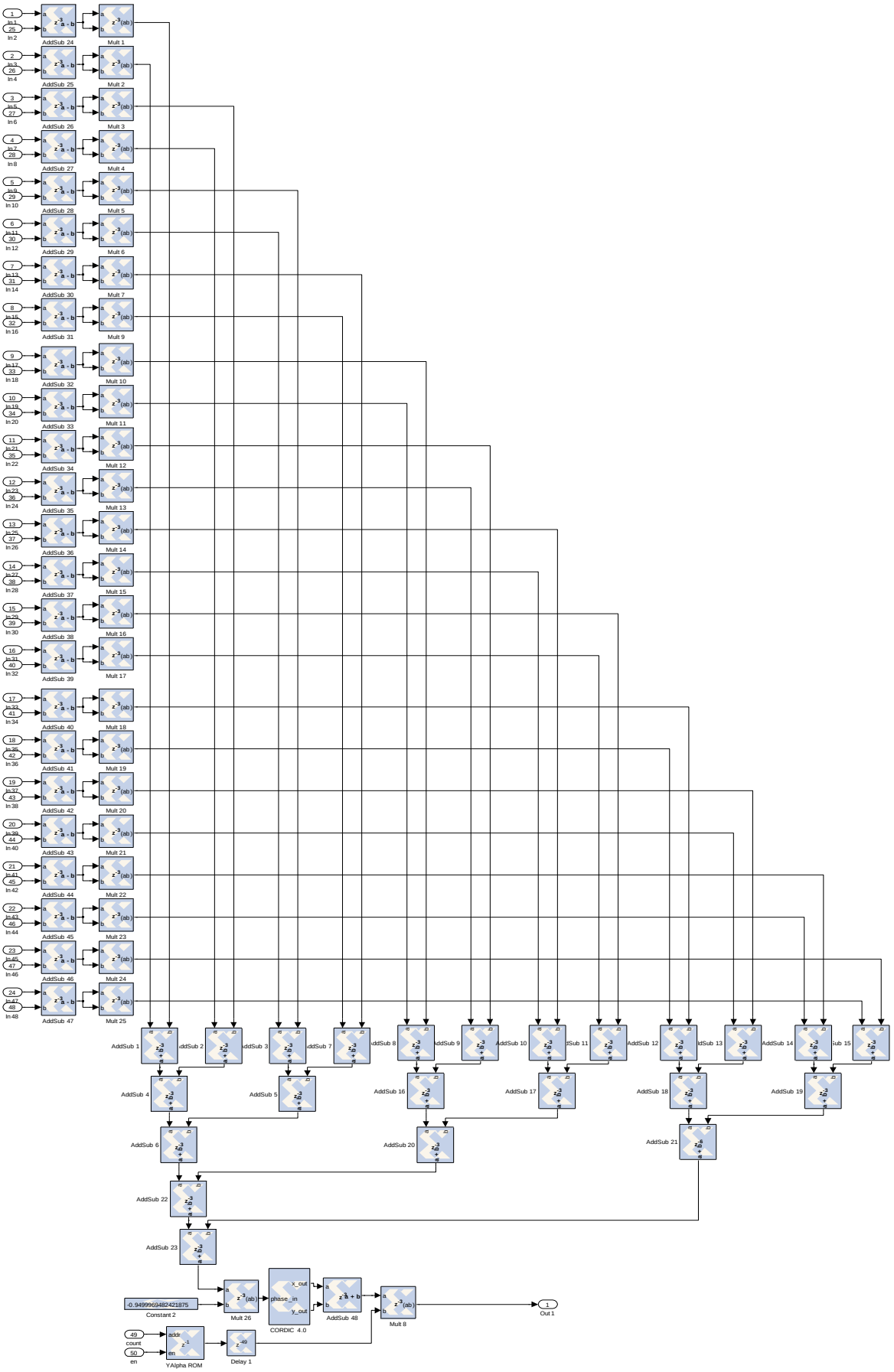
در System Generator تابعی برای محاسبه مستقیم exp وجود ندارد. اما بلوکی به نام CORDIC4.0 وجود دارد که خروجی های Sinh و Cosh را طبق الگوریتم CORDIC، که در بخش 3-9- شرح داده شد، تولید می کند. توابع مثلثاتی ذکر شده با تابع exp طبق معادله زیر با هم رابطه دارند:

[1] Norm

$$exp(x) = Sinh(x) + Cosh(x) \qquad (8\text{-}5)$$

با بهره گیری از رابطه (5-8) می توان تابع exp را با استفاده از بلوک CORDIC4.0 به راحتی پیاده سازی کرد.

بنابراین با استفاده از روابط (5-7) و (5-8) و با توجه به طراحی انجام شده در شکل 5-28، برای پیاده سازی کلاسه بند ماشین بردار پشتیبان غیرخطی با بردار ویژگی 24 بعدی فقط کافیست در ساختار ماشین بردار پشتیبان خطی شکل 5-28، زیرسیستم های X_Test*SVxx*YAlpha xx به صورت شکل 5-29 پیاده سازی شود، در حالیکه سایر قسمتهای طراحی بدون تغییر می ماند. تاخیر در نظر گرفته شده در بلوک Delay1 این ساختار، حاصل از مجموع تاخیرهای بلوکهای موازی Addsub و Mult و بلوک CORDIC می باشد. این تاخیر مشخص می کند که تاخیر درونی بلوک CORDIC برابر با 22 کلاک می باشد.



شکل 5-29: زیر سیستم X_Test*SVxx*YAlpha xx برای پیاده سازی ماشین بردار پشتیبان غیر خطی

فصل 6- نتایج شبیه سازی و آنالیز

در این فصل نتایج شبیه سازی و آنالیز کلاسه بند ماشین بردار پشتیبان خطی و غیرخطی برای تشخیص ارقام فارسی 1، 4 و 8 با بردارهای ویژگی 7 و 24 بعدی را، که در فصل 5- طراحی شد، با دو FPGA متفاوت و با تعداد بیتهای مختلف به دست می آوریم.

6-1- انتخاب نوع FPGA

از آنجا که به دلیل تفاوت در سرعت کلاک و حجم فضای سخت افزاری، انتخاب نوع FPGA در نتایج شبیه سازی، آنالیز و سنتز موثر است، در این تحقیق سعی کردیم طرح سخت افزاری خود را بر روی دو FPGA از دو خانواده متفاوت Xilinx اجرا کنیم. بدین منظور خانواده های Spartan3 و Virtex4 انتخاب شدند. از خانواده Virtex4 مدل xc4vsx35 و از خانواده Spartan3 مدل xc3s1500I جهت هدف پیاده سازی به کار گرفته می شود.

هر طراحی سخت افزاری که در System Generator ایجاد می شود، برای اجرای شبیه سازی باید حداقل یک بلوک علامت[1] در آن طرح داشته باشد. این بلوک علامت به هیچ چیز دیگری اتصال ندارد اما برای اجرای پروسه شبیه سازی و آنالیز، حتماً باید وجود داشته باشد. در این بلوک می توان قطعه FPGA مورد نظر را تعریف کرد. برای شروع پروسه شبیه سازی، قطعه xc4vsx35 از خانواده Virtex4 را انتخاب کرده و در ادامه به بررسی نتایج شبیه سازی بر روی قطعه xc3s1500I از خانواده Spartan3 می پردازیم. نتایج شبیه سازی بر روی هر دو FPGA با تعداد بیتهای مختلف نیز مقایسه خواهد شد.

6-2- انتخاب نوع کوانتی‌زاسیون[2] و تعداد بیت

در بخش 3-5- توضیح دادیم که بهتر است در پیاده سازی سخت افزاری بر روی FPGA از محاسبات ممیز ثابت استفاده شود، چرا که علاوه بر کاهش در حجم فضای سیلیکونی و مصرف توان، افزایش سرعت سیستم را نیز در بر خواهد داشت. هر چند که پیاده سازی ممیز شناور بر روی FPGA نیز امکان پذیر است اما در این صورت باید از داشتن این ویژگیها چشم پوشی کرد.

[1] Token
[2] Quantization

در این تحقیق ما از محاسبات ممیز ثابت برای رسیدن به هدف کلاسه بندی استفاده می کنیم. در همان بخش 3-5- توضیح داده شد که در محاسبات ممیز ثابت طول کلمه با انجام محاسبات پشت سر هم افزایش پیدا می کند و این باعث افزایش بی رویه طول کلمه، طولانی تر شدن زمان محاسبات مسیرهای بلوک های دیجیتال و کاهش سرعت سیستم می شود. راه حلی که برای رفع این مشکل پیشنهاد شد، استفاده از روش قطع کردن و یا روندکردن بود.

در این تحقیق از تکنیک قطع کردن برای محدود کردن طول کلمه استفاده شده است، چرا که فضای کمتری از سخت افزار را اشغال می کند.

6-3- نتایج شبیه سازی کلاسه بندی خطی 7 بعدی بر روی Virtex4

اجازه دهید یک بار دیگر به شکل 5-11 در ابتدای طرح سخت افزاری مراجعه کنیم. بلوکهای SV ROM1 تا SV ROM7 ، X_Test ROM1 تا X_Test ROM7 و YAlpha ROM مقدار ورودی خود را از work space محیط MATLAB می گیرند. برای قرار گرفتن در محیط سخت افزاری و انجام محاسبات ممیز ثابت، خروجی این بلوکها باید از نظر نوع محاسبات و تعداد بیت ها مشخص شود. بلافاصله بعد از بلوکهای SV ROM و X_Test ROM بلوکهای Mult، برای عمل ضرب، و سپس بلوکهای Addsub، برای عمل جمع، وجود دارند که در مجموع عملیات ضرب برداری را انجام می دهند. اصولاً در هر یک از این مراحل عملیات ضرب و جمع، طول کلمه افزایش پیدا خواهد کرد. همانطور که گفته شد در هر یک از این مراحل با استفاده از تکنیک قطع کردن، طول کلمه را به مقدار دلخواه و مناسب محدود می کنیم. این کار تا پایان طرح سخت افزاری ادامه پیدا خواهد کرد. در نهایت بزرگترین طول کلمه استفاده شده درطرح را به عنوان دقت ممیز ثابت سیستم خواهیم شناخت.

6-3-1- نتایج با دقت Q64.40

انتخاب دقت Q64.40 [1] به عنوان اولین دقت ممیز ثابت در نتایج شبیه سازی صرفاً به دلیل حصول اطمینان از عملکرد کلاسه بند مورد نظر است، هر چند که می دانیم این انتخاب باعث صرف منابع سخت افزاری زیاد و پایین آمدن سرعت سیستم خواهد شد. به همین منظور نتایج شبیه سازی سخت افزاری ممیز ثابت در System Generator با نتایج ممیز شناور در نرم افزار MATLAB مقایسه می شود.

[1] 40 بیت برای بخش اعشاری و 24 بیت برای بخش صحیح

جدول 6-1 نتایج حاصل از شبیه سازی ساختار کلاسه بند ماشین بردار پشتیبان خطی برای تشخیص ارقام فارسی با بردارهای ویژگی 7 بعدی که طراحی آن در شکل 5-27 انجام شد را در محیط نرم افزاری MATLAB و همچنین بر روی Virtex4 مدل xc4vsx35 با دقت ممیز ثابت Q64.40 نشان می دهد. فاز آموزش این کلاسه بند قبلاً در بخش 5-2-1- به صورت خارج خط توسط نرم افزار MATLAB انجام و نتایج آن آورده شد. تعداد کل داده های تست برای کلاسه بندی 600 نمونه 7 بعدی است. نتایج نشان می دهد که ساختار سخت افزاری طراحی شده برای کلاسه بند ماشین بردار پشتیبان خطی صحیح است، چرا که در مقایسه با کلاسه بندی در MATLAB، نرخ خطا هیچ تغییری نکرده است.

جدول 6-1: نتایج شبیه سازی کلاسه بندی خطی 7 بعدی – مقایسه با MATLAB

MATLAB (ممیز شناور)			Sys. Gen. (ممیز ثابت)			
کلاس1	کلاس2	کلاس3	کلاس1	کلاس2	کلاس3	
200	0	0	200	0	0	کلاس 1
6	159	35	6	159	35	کلاس 2
0	42	158	0	42	158	کلاس 3
13/83%			13/83%			نرخ خطا
2/1GHz Intel, Core2 Dou			83/872MHz Virtex4			فرکانس کاری سیستم
0/45s			3/5ms			زمان انجام محاسبات

نکته جالب در نتایج جدول 6-1، زمان انجام محاسبات برای کلاسه بندی 600 داده تست است. در حالی که فرکانس پردازشگر کامپیوتر 2/1GHz، در مقابل فرکانس FPGA ، 25 برابر است؛ اما FPGA محاسبات را 128 برابر سریعتر از کامپیوتر انجام داده است و این به دلیل خصوصیت پردازش موازی در FPGA است. زمان انجام محاسبات از رابطه زیر به دست می آید:

زمان محاسبات FPGA=(تعداد بردارهای تست×ماکزیمم SVها / فرکانس کاری) + تاخیر اولیه

در جدول 6-2 منابع سخت افزاری استفاده شده در Virtex4 برای پیاده سازی ساختار کلاسه بند خطی 7 بعدی با دقت بیت Q64.40 نشان داده شده است.

جدول 6-2: منابع سخت افزاری استفاده شده در Virtex4 برای کلاسه بندی خطی 7 بعدی با دقت Q64.40

لاجیک های موجود	تعداد استفاده شده	تعداد موجود	درصد استفاده
Slice Flip Flop	3726	30720	12%
4 input LUT	1709	30720	5%
Occupied Slice	2266	15360	14%
Bonded IOB	387	448	86%
BUFG/BUFGCTRL	1	32	3%
FIFO16/RAMB16	38	192	19%
DSP48	102	192	53%

6-3-2- نتایج با دقت Q32.20

با توجه به اینکه انتخاب دقت 64 بیت برای شبیه سازی و آنالیز طرح مورد نظر صرفاً جهت اطمینان از کارکرد صحیح سیستم بود و نتایج خوبی از نظر فضای سخت افزاری و فرکانس کاری حاصل نشد، این بار سیستم را با دقت 32 بیت، که کمی معقولانه تر اما نه کاملاً بهینه است، آنالیز می کنیم. بدون شک انتظار می رود که با دقت 32 بیت نسبت به دقت 64 بیت، هم فرکانس کاری سیستم زیاد شود و هم منابع سخت افزاری کمتری استفاده شود، در حالی که احتمال افزایش نرخ خطا در کلاسه بندی نیز وجود خواهد داشت.

جدول 6-3 نتایج حاصل از شبیه سازی ساختار سخت افزاری کلاسه بند ماشین بردار پشتیبان خطی 7 بعدی برای تشخیص ارقام فارسی که طراحی آن در شکل 5-27 انجام شد را بر روی Virtex4 با دقت ممیز ثابت Q32.20 نشان می دهد. همانطور که در جدول نتایج مشاهده می شود، از بین کل داده های تست که مجموع آنها 600 داده است، 5 مورد نسبت به کلاسه بند نرم افزاری MATLAB متفاوت کلاسه بندی شده است و این صرفاً به دلیل محدود شدن تعداد بیتها است، هر چند که این وضعیت باعث کاهش نرخ خطای کلاسه بندی شده است.

جدول 6-3: نتایج شبیه سازی کلاسه بندی خطی 7 بعدی با محاسبات ممیز ثابت Q32.20

	کلاس1	کلاس2	کلاس3
کلاس 1	200	0	0
کلاس 2	6	162	32
کلاس 3	0	42	158
نرخ خطا	13/33%		
تفاوت کلاسه بندی نسبت به MATLAB	5		
فرکانس کاری سیستم	131/683MHz Virtex4		
زمان انجام محاسبات	2/2ms		

جدول 6-4 منابع سخت افزاری استفاده شده در Virtex4 مدل xc4vsx35 برای پیاده سازی ساختار کلاسه بند ماشین بردار پشتیبان سه کلاسه برای تشخیص ارقام فارسی 7 بعدی، با دقت بیت Q32.20 را نشان می دهد.

جدول 6-4: منابع سخت افزاری استفاده شده در Virtex4 برای کلاسه بندی خطی 7 بعدی با دقت Q32.20

لاجیک های موجود	تعداد استفاده شده	تعداد موجود	درصد استفاده
Slice Flip Flop	2010	30720	6%
4 input LUT	992	30720	3%
Occupied Slice	1468	15360	9%
Bonded IOB	220	448	49%
BUFG	1	32	3%
RAMB16	38	192	19%
DSP48	96	192	50%

6-3-3- نتایج با دقت Q24.16

یک بار دیگر تعداد بیتها را کمتر می کنیم تا به فرکانس کاری بالاتر و منابع سخت افزاری کمتری دسترسی پیدا کنیم، البته باید مراقب باشیم که نرخ خطای کلاسه

بندی خیلی زیاد نشود. این بار پیش بینی می کنیم با انتخاب دقت ممیز ثابت Q24.16 نتایج بهتری از نظر سرعت و فضای سخت افزاری حاصل می شود.
جدول 6-5 نتایج حاصل از شبیه سازی ساختار سخت افزاری کلاسه بند ماشین بردار پشتیبان خطی 7 بعدی برای تشخیص ارقام فارسی را بر روی Virtex4 با دقت بیت Q24.16 نشان می دهد.

جدول 6-5: نتایج شبیه سازی کلاسه بندی خطی 7 بعدی با محاسبات ممیز ثابت Q24.16

	کلاس1	کلاس2	کلاس3
کلاس 1	200	0	0
کلاس 2	6	169	25
کلاس 3	0	41	159
نرخ خطا	12%		
تفاوت کلاسه بندی نسبت به MATLAB	13		
فرکانس کاری سیستم	202/840MHz		
زمان انجام محاسبات	1/4ms		

از محتوای جدول پیداست که باز هم به دلیل محدود تر شدن تعداد بیتها، تفاوت کلاسه بندی نسبت به پردازش ممیز شناور MATLAB افزایش پیدا کرده است، اما آنچه که ما اینجا به دست آورده ایم رسیدن به ماکزیمم فرکانس سیستم برابر با 202/840MHz است. بنابراین می توان کلاسه بند ماشین بردار پشتیبان برای تشخیص ارقام فارسی 7 بعدی را با دقت 24 بیت و فرکانس 202/840MHz روی Virtex4 پیاده سازی کرد.

جدول 6-6 نیز منابع سخت افزاری استفاده شده در Virtex4 برای پیاده سازی ساختار کلاسه بند ماشین بردار پشتیبان خطی برای تشخیص ارقام فارسی 7 بعدی، با دقت بیت Q24.16 را نشان می دهد. همانطور که در این جدول ملاحظه می شود، منابع سخت افزاری استفاده شده در این حالت، نسبت به دقت ممیز ثابت Q32.20 کاهش یافته است.

جدول 6-6: منابع سخت افزاری استفاده شده در Virtex4 برای کلاسه بندی خطی 7 بعدی با دقت Q24.16

لاجیک های موجود	تعداد استفاده	تعداد موجود	درصد استفاده

		شده	
4%	30720	1422	Slice Flip Flop
2%	30720	748	4 input LUT
5%	15360	849	Occupied Slice
37%	448	167	Bonded IOB
3%	32	1	BUFG
16%	192	31	RAMB16
14%	192	27	DSP48

6-4- نتایج شبیه سازی کلاسه بندی خطی 7 بعدی بر روی Spartan3

همانطور که در ابتدای فصل گفته شد، به دلیل تفاوت در سرعت کلاک و حجم فضای سخت افزاری، انتخاب نوع FPGA در نتایج شبیه سازی موثر است. در بخش 6-3-3 بهترین حالت شبیه سازی کلاسه بندی خطی 7 بعدی برای تشخیص ارقام فارسی با دقت 24 بیت، در فرکانس 202/840MHz به دست آمد. بنابراین جهت مقایسه، با حفظ دقت ممیز ثابت بخش 6-3-3 (یعنی Q24.16)، FPGA دیگری از شرکت Xilinx از خانواده Spartan3، را به عنوان قطعه مقصد انتخاب و نتایج شبیه سازی را بر روی آن به دست آورده و با Virtex4 مقایسه می کنیم. بدون شک نتایج کلاسه بندی تغییری نخواهد کرد، چون پروسه طراحی در هر دو FPGA یکسان است و دقت کوانتیزاسیون نیز تغییری نکرده است. اما ماکزیمم فرکانس کاری و فضای سخت افزاری مورد استفاده دچار تغییر می شود. ماکزیمم فرکانس کاری به دست آمده با Spartan3 برابر با 81/268MHz است که نسبت به Virtex4 حدود 2/5 برابر کاهش یافته است. با این فرکانس زمان انجام محاسبات 3/6ms خواهد شد.

جدول 6-7 فضای سخت افزاری مورد استفاده در Spartan3 برای پیاده سازی ساختار کلاسه بند ماشین بردار پشتیبان خطی برای تشخیص ارقام فارسی 7 بعدی را نشان می دهد.

جدول 6-7: منابع سخت افزاری استفاده شده در Spartan3 برای کلاسه بندی خطی 7 بعدی با دقت Q24.16

درصد استفاده	استفاده شده	لاجیک
18%	7660	Slice Flip Flop
23%	9478	4 input LUT
27%	5615	Occupied Slice
30%	171	Bonded IOB
BUFGMUX	1	BUFG

12%		
77%	31	RAMB16
-	-	DSP48

با این حجم از منابع سخت افزاری استفاده شده در Spartan3 برای کلاسه بندی 7 بعدی، می توان ادعا کرد که این نوع FPGA برای پیاده سازی کلاسه بندهای 24 بعدی و همچنین برای افزایش تعداد کلاسه بندهای دوتایی مناسب نمی باشد، بنابراین بقیه نتایج شبیه سازی را روی قطعه Virtex4 متمرکز می کنیم.

6-5- نتایج شبیه سازی کلاسه بند خطی 24 بعدی با دقت Q24.16 بر روی Virtex4

همانطور که قبلاً توضیح داده شد، ساختار کلاسه بند ماشین بردار پشتیبان در این تحقیق به نحوی طراحی شده است که امکان افزایش ابعاد بردارهای ورودی به سادگی وجود دارد. هدف از این بخش این است که نتایج کلاسه بند سخت افزاری ماشین بردار پشتیبان خطی با بردارهای ویژگی 24 بعدی با دقت Q24.16، که قبلاً در شکل 5-9 طراحی شده است، را با نتایج به دست آمده از پیاده سازی نرم افزاری در MATLAB، که این مورد هم در بخش 5-2-1- انجام شد، مقایسه کنیم. جدول 6-8 نتایج حاصل از این کلاسه بندی را نشان می دهد.

جدول 6-8: نتایج شبیه سازی کلاسه بندی خطی 24 بعدی با محاسبات ممیز ثابت Q24.16 و مقایسه با MATLAB

Sys. Gen. (ممیز ثابت)			MATLAB (ممیز شناور)			
کلاس3	کلاس2	کلاس1	کلاس3	کلاس2	کلاس1	
0	1	199	0	1	199	کلاس 1
6	188	6	6	189	5	کلاس 2
199	1	0	199	1	0	کلاس 3
2/33%			2/17%			نرخ خطا
131/423MHz Virtex4			2/1GHz Intel, Core2 Dou			فرکانس کاری سیستم
0/43ms			0/18s			زمان انجام محاسبات

همانطور که از جدول 6-8 پیداست، از 600 داده تست تنها یک مورد در کلاسه بندی بر روی Vittex4 نسبت به کلاسه بندی نرم افزاری MATLAB ، خطا داشته است که آمار قابل قبولی است. علت کاهش زمان انجام محاسبات نسبت به کلاسه بندی خطی 7 بعدی، کمتر شدن ماکزیمم تعداد بردارهای پشتیبان

در کلاسه بندهای دوتایی است، که قبلاً در جدول 5-1 و جدول 5-2 این تعداد آورده شده است.

جدول 6-9 نیز منابع سخت افزاری مصرف شده در Virtex4 برای پیاده سازی ساختار کلاسه بند ماشین بردار پشتیبان برای تشخیص ارقام فارسی 24 بعدی را نشان می دهد.

جدول 6-9: منابع سخت افزاری استفاده شده در Virtex4 برای کلاسه بندی خطی 24 بعدی با دقت Q24.16

لاجیک های موجود	تعداد استفاده شده	تعداد موجود	درصد استفاده
Slice Flip Flop	3790	30720	12%
4 input LUT	2294	30720	7%
Occupied Slice	2446	15360	15%
Bonded IOB	207	448	46%
BUFG	1	32	3%
RAMB16	99	192	51%
DSP48	78	192	40%

6-6- نتایج شبیه سازی کلاسه بند 24 بعدی غیرخطی با دقت Q24.14

در بخش 5-3-6- ساختار کلاسه بند غیر خطی با بردارهای ویژگی 24 بعدی طراحی شد. در این بخش نتایج حاصل از شبیه سازی ساختار سخت افزاری طراحی شده به همراه نتایج نرم افزاری متناظر توسط MATLAB که قبلاً در بخش 5-2-2- انجام شد، مقایسه می شود. جدول 6-10 نتایج حاصل از این کلاسه بندی را نشان می دهد.

جدول 6-10: نتایج شبیه سازی کلاسه بندی غیرخطی 24 بعدی با محاسبات ممیز ثابت Q24.14 و مقایسه با MATLAB

	MATLAB (ممیز شناور)			Sys. Gen. (ممیز ثابت)		
	کلاس1	کلاس2	کلاس3	کلاس1	کلاس2	کلاس3
کلاس 1	199	1	0	199	1	0
کلاس 2	2	195	3	3	194	3
کلاس 3	0	1	199	0	1	199
نرخ خطا	1/17%			1/33%		
فرکانس کاری سیستم	2/1GHz Intel, Core2 Dou			151/286MHz Virtex4		
زمان انجام محاسبات	0/11s			0/27ms		

جدول 6-11 نیز منابع سخت افزاری مصرف شده در Virtex4 جهت کلاسه بندی ارقام فارسی 24 بعدی به روش ماشین بردار پشتیبان غیرخطی نشان می دهد.

جدول 6-11: منابع سخت افزاری استفاده شده در Virtex4 برای کلاسه بندی خطی 24 بعدی با دقت Q24.16

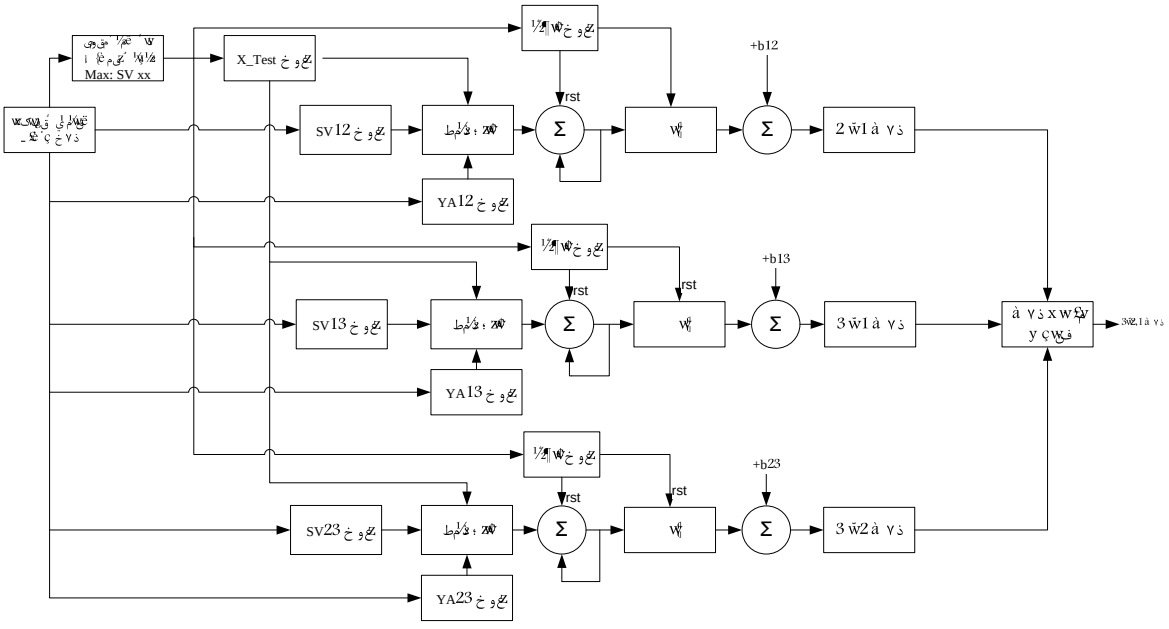
لاجیک های موجود	تعداد استفاده شده	تعداد موجود	درصد استفاده
Slice Flip Flop	11589	30720	37%
4 input LUT	9141	30720	29%
Occupied Slice	7261	15360	47%
Bonded IOB	241	448	53%
BUFG	1	32	3%
RAMB16	99	192	51%
DSP48	81	192	42%

فصل 7- نتی‌جه گی‌ری و پی‌شنهاد

در این تحقیق طراحی ساختار کلاسه بند ماشین بردار پشتیبان با توابع کرنل خطی و گوسی برای تشخیص ارقام دستنوشته فارسی 7 و 24 بعدی بر روی FPGA های Virtex4 و Spartan3 با دقت بیتهای مختلف انجام شد. با توجه به اینکه ارقام دستنوشته فارسی 1 ، 4 و 8 برای هدف کلاسه بندی انتخاب شدند، کلاسه بند مورد نظر برای کلاسه بندی 3 کلاس طراحی شد. بنابراین طبق روش کلاسه بندهای دوتایی، برای کلاسه بندی 3 کلاس، 3 کلاسه بند دوتایی طراحی شد و عملیات تابع تصمیم هر 3 کلاسه بند به صورت موازی و به طور همزمان انجام گردید.

طبق نتایج شبیه سازی در فصل 6 بهترین فرکانس کاری سیستم در کلاسه بندی خطی با بردارهای ویژگی 7 بعدی به دست آمد که برابر با 202/840MHz می باشد. همچنین بهترین حالت کلاسه بندی در کلاسه بندهای خطی و غیرخطی با بردارهای ویژگی 24 بعدی اتفاق افتاد که در هر دو حالت، از بین 600 داده تست، فقط یک مورد نسبت به کلاسه بندی در محیط نرم افزاری MATLAB انحراف داشته است.

اگر بخواهیم ساختار کلاسه بند طراحی شده را به صورت بلوک دیاگرام کلی رسم کنیم، شکل 7-1 را خواهیم داشت:



شکل 7-1: بلوک دیاگرام کلی ساختار کلاسه بند ماشین بردار پشتیبان برای کلاسه بندی بین 3 کلاس

هر چند که در این تحقیق کلاسه بند ماشین بردار پشتیبان را برای تشخیص 3 رقم از ارقام دستنوشته فارسی بر روی FPGA طراحی کردیم، اما باید در نظر داشت که هدف اصلی این تحقیق نحوه پیاده سازی کلاسه بندی چند کلاسه ماشین بردار پشتیبان خطی و غیرخطی بر روی FPGA است. ساختار کلاسه بند طراحی شده به گونه ای است که امکان افزایش تعداد کلاسه بندهای دوتایی به راحتی وجود دارد، و برای هر سیستم تشخیص مبتنی بر ماشین بردار پشتیبان خطی و غیرخطی (با تابع کرنل گوسی)، با هر تعداد بردار پشتیبان و بردار تست نیز قابل اجراست. به عنوان مثال می توان ساختار موردنظر را برای تشخیص شماره پلاک خودروهای ایرانی پیاده سازی کرد. همچنین می توان از ساختار طراحی شده در تشخیص حدود جاده برای کاربردهای خودرویی، یا جداسازی محصولات معیوب از سالم در کارخانجات تولیدی استفاده کرد، جایی که هم نیاز پرتابل و هم نیاز پردازش بلادرنگ بسیار ضروری است.

تنها عاملی که ممکن است پیاده سازی ساختار طراحی شده را با مشکل مواجه سازد، محدودیت فضای سخت افزاری است. به طور کلی عوامل اصلی که در مصرف فضای سخت افزاری ساختار طراحی شده موثر است، عبارتند از:

1. نوع کوانتیزاسیون و تعداد بیت
2. ابعاد بردار ویژگی
3. تعداد کلاسها
4. نوع FPGA

برای طراحی یک ساختار سخت افزاری بهینه باید هر چهار عامل فوق با هم مورد تجزیه و تحلیل قرار گیرد تا علاوه بر دستیابی به نرخ بازشناسی و ماکزیمم فرکانس کاری بالاتر، محدودیتهای فضای سخت افزاری نیز پوشش داده شود. در عمل برای هر کاربرد پردازشی، پس از مرحله پیش پردازش و استخراج ویژگی، ابعاد بردار ویژگی و تعداد کلاسها مشخص است. نوع کوانتیزاسیون و تعداد بیت هم که با سعی و خطا توسط کاربر با هدف رسیدن به بهترین نرخ بازشناسی و ماکزیمم فرکانس کاری بالاتر مشخص می شود. در این حالت اگر فضای سخت افزاری FPGA انتخابی جوابگوی ساختار طراحی شده نباشد، و همچنین امکان ارتقاء FPGA به مدل های بالاتر نیز وجود نداشته باشد، یکی از راهکارهای پیشنهادی استفاده از دو یا چند FPGA به صورت موازی به نحوی است که در هر FPGA بخشی از طرح سخت افزاری پیاده سازی و اجرا شود. مثلاً در ساختار کلاسه بند طراحی شده در این تحقیق می توان هر کدام از مسیرهای کلاسه بندی دوتایی، یا تعداد مشخصی از آنها، را بر روی یک FPGA پیاده سازی و اجرا کرد. در این تحقیق با توجه به نتایج فصل 6 به نظر می رسد ساختار طراحی شده برای

کلاسه بندی ماشین بردار پشتیبان خطی 7 بعدی با کوانتیزاسیون Q24.16 بر روی Virtex4، برای 10 کلاسه بند دوتایی موازی به راحتی جوابگو باشد. یعنی برای کلاسه بندی تا 5 کلاس مشکلی وجود نخواهد داشت. در صورت افزایش تعداد کلاسه بندهای دوتایی، می توان از راه حل های پیشنهادی فوق یا از حافظه های جانبی در کنار FPGA استفاده کرد.

به منظور ادامه کار این تحقیق، یافتن راهکاری برای پیاده سازی سایر توابع کرنل غیرخطی در جدول 2-1 نیز پیشنهاد می شود. با توجه به اینکه هر کدام از توابع کرنل برای کاربردهای خاصی پاسخ بهتری را نسبت به سایر توابع از خود نشان می دهد، در صورت یافتن راه حلی برای پیاده سازی همه این توابع دست طراح برای رسیدن به بهترین نتیجه در کلاسه بندی باز می شود.

پیشنهاد دیگر اجرای کل پروسه ماشین بردار پشتیبان شامل مرحله آموزش و تست آن بر روی FPGA است. فاز آموزش ماشین بردار پشتیبان شامل حل مسئله بهینه سازی (2-11) است، که برای هدف پیاده سازی سخت افزاری، پروسه ای بسیار وقت گیر و شامل محاسبات پیچیده است. اگر بتوان راه حلی که مناسب سخت افزار و قابل پیاده سازی بر روی FPGA باشد، برای ساده کردن حل مسئله بهینه سازی پیدا کرد، آنگاه می توان کل پروسه ماشین بردار پشتیبان را بر روی FPGA پیاده سازی نمود به طوریکه دیگر نیازی به اجرای مرحله آموزش در محیط نرم افزاری نباشد.

برای پیاده سازی عملی، با توجه به اینکه ساختار طراحی شده یک ساختار ترکیبی نرم افزار-سخت افزار می باشد، می توان از اسلات های کامپیوتری با هسته FPGA استفاده کرد. در این صورت بخشی از مشکلات احتمالی که در انتقال اطلاعات از کامپیوتر به FPGA از طریق کابل در فرکانسهای بالا بوجود می آید، از بین خواهد رفت.

فهرست مراجع

[1] V. N. Vapnik, “The Nature of Statistical Learning Theory,” Springer-Verlag, New York, 1995.

[2] N. Cristianini, and J. Shave-Taylor, “An Introduction to Support Vector Machines and other Kernel-Based Learning Methods,” Cambridge University Press, USA, 2000.

[3] Shigeo Abe, “Support Vector Machines for Pattern Classification,” Springer-Verlag, London, 2005.

[4] M. Martínez-Ramón, C. Christodoulou, “Support Vector Machines for Antenna Array Processing and Electromagnetics”, Morgan & Claypool Publishers, CO, USA, 2006

[5] A.J. Izenman, “Modern Multivariate Statistical Techniques: Regression, Classification and Manifold Learning,” Springer, New York, 2008.

[6] U. H.-G. Kreßel. Pairwise classification and support vector machines. In: B. Sch¨olkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in Kernel Methods: Support Vector Learning, pages 255–268. MIT Press, Cambridge, MA, 1999.

[7] R. Woods, J. McAllister, G. Lightbody, and Y. Yi, “FPGA-based Implementation of Signal Processing Systems,” John Wiley & Sons Ltd, United Kingdom, 2008.

[8] U. Meyer-Baese, “Digital Signal Processing with Field Programmable Gate Arrays,” 3rd ed., Springer, 2007.

[9] J. Serrano, CERN, Geneva, Switzerland, “Digital Signal Processing Using Fied Programmable Gate Arrays,” 13th Beam Instrumentation Workshop, Tahoe City, CA, USA, pp.29-38, 2008.

[10] A. Rushton, “VHDL for Logic Synthesis,” 2rd ed., John Wiley & Sons, 1998.

[11] S. Drimer, “Volatile FPGA design security: a survey,” Computer Laboratory, University of Cambridge, April 2008.

[12] John G. Proakis, Dimitris K. Manolakis, “Digital Signal Processing,” 4th ed. Prentice Hall, 2006.

[13] Wayne T. Padgett, David V. Anderson, “Fixed-Point Signal Processing,” Synthesis Lectures on Signal Processing, Morgan & Claypool, 2009.

[14] R. Andraka, “A survey of CORDIC algorithms for FPGA based computers,” Proc. ACM/SIGDA 6th International Symposium on Field Programmable Gate Array, Monterey, CA, pp. 191–200, 1998.

[15] Jack E. Volder, “The CORDIC Trigonometric Computing Technique”, IRE Transactions on Electronic Computers, pp330-334, September 1959.

[16] John S. Walther, “A Unified Algorithm for Elementary Functions”, Proc. of Spring Joint Computer Conference, pp379-385, May 1971.

[17] B. Lakshmi, and A. S. Dhar, “CORDIC Architectures: A Survey”, Hindawi Publishing Corporation, VLSI Design, Article ID 794891, Volume 2010.

[18] D. Anguita, A. Boni, S. Ridella, “A Digital Architecture for Support Vector Machines: Theory, Algorithm and FPGA Implementation”, IEEE Transactions on Neural Networks, pp. 993–1009, 2003.

[19] Faisal M.Khan, et.al. “Hardware-Based Support Vector Machine Classification in Logarithmic Number Systems”, IEEE Symposium on Circuits and Systems, ISCAS, 2005.

[20] O. Pina-Ramirez, R. Valdes-Cristerna and O. Yanez-Suarez , “An FPGA Implementation of Linear Kernel Support Vector Machines,” IEEE International Conference on Reconfigurable Computing and FPGA's, pp.1-6, 2006.

[21] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm

[22] J. Manikandan, B. Venkataramani, and V. Avanthi, “FPGA Implementation of Support Vector Machine Based Isolated Digit Recognition System”. Proc. 22nd Int. Conf. on VLSI Design, pp.347-352, 2009.

[23] http://www.ldc.upenn.edu/Catalog/readme_files/ti46.readme.html.

[24] M. Ruiz, and M. Y. Calvino, “FPGA Implementation of Support Vector Machines for 3D Object Identification”, Proceedings of the 19th International Conference on Artificial Neural Networks, Part I, Springer-Verlag, Berlin, pp. 467–474, 2009.

[25] http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php.

[26] Chun F.Hsu, Mong-Kai Ku, and Li-Yen Liu, “Support Vector Machine FPGA Implementation For Video Shot Boundary Detection Application”, IEEE International conf. on SOC 2009. SOCC 2009.

[27] H. Khosravi, E. Kabir, “Introducing a very large dataset of handwritten Farsi digits and a study on their varieties”, ELSEVIER, Pattern Recognition Letters 28, pp. 1133–1141, 2007.

[28] Theodoridis S., Koutroumbas K., “Pattern Recognition”, 3rd. edition, Elsevier 2006.

[29] H.-T. Lin; and C.-J. Lin, "A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods," Technical report, Department of Computer Science and Information Engineering, National Taiwan University, March 2003.

واژه نامه انگلیسی به فارسی

All-at-once	همه در یک بار
Buffering	بافرکردن
Clock	کلاک
Combinatorial path	مسیر طولانی
COrdinate Rotation DIgital Computer	محاسبه کننده دیجتال بر اساس دوران مختصات
Data mining	داده کاوی
Design entry	ورود طرح
Design flow	روند طراحی
Digital signal processing	پردازش سیگنال دیجیتال
Digital signal processor	پردازنده سیگنال دیجیتال
Discrete cosine transform	تبدیل کسینوسی گسسته
Dual Lagrange Function	تابع لاگرانژ دوگان
Error-correcting output code (ECOC)	کد خروجی تصحیح کننده خطا
Ethernet Medium Access Control (MAC) unit	واحد کنترل دسترسی میانی اترنت
Exponent	توان
Fast Fourier Transform	تبدیل فوریه سریع
Field Programmable Gate Array	آرایه منطقی قابل برنامه ریزی
Finite Impulse Response	پاسخ ضربه محدود
Fixed point	ممیز ثابت
Flash memory	حافظه فلش
Floating point	ممیز شناور
Frame rate	نرخ فریم
Gain	بهره
Gaussian Radial Basis Function	تابع گوسی اساس شعاعی
General purpose processors	پردازنده های همه منظوره
Geometric moments	گشتاورهای هندسی
Global positioning system	سیستم موقعیت یاب جهانی
Hardware Description Languages (HDL)	زبان توصیف سخت افزاری
Hardware friendly	سخت افزار پسند
Hyperplane	ابرصفحه
Infinite Impulse Response	پاسخ ضربه نامحدود
Input/Output	ورودی/خروجی
Intellectual Property (IP) core	هسته با مشخصه هوشمند
Kernel function	تابع کرنل
Kernel trick	حیله کرنل
Lagrange coefficient	ضرایب لاگرانژ
Laplacian	لاپلاس
Look-up table	جدول مراجعه
Mantissa or Fraction	اعشار
Multi-class SVM	ماشین بردار پشتیبان چند کلاسه
Multi-layer Neural Networks	شبکه های عصبی چند لایه

Netlist	نتلی‌ست
NonStationary	غی‌رای‌ستان
Objective function	تابع هدف
Offline	خارج خط
One against all	ی‌ک دربرابر همه
Online	برخط
Optimal separating hyperplane	ابرصفحه مجزاساز بهی‌نه
Pairwise	دوتای‌ی
Pass gates	دروازه های عبور سی‌گنال
Pattern recognition	تشخی‌ص الگو
Place & route	مکان ی‌ابی و سی‌م کشی
Primal Lagrange Function	تابع لاگرانژ اولی‌ه
Quadratic problem	مسئله درجه دو
Quantization	کوانتی‌زاسی‌ون
Real-time	بلادرنگ
Register Balancing	توازن رجی‌ستر
Rotation mode	مد دورانی
Rounding	روند کردن
Routing switches	سوئی‌چهای مسی‌ری‌اب
Serial Interface	واسط سری‌ال
Shot boundary	مرز نما
Single precision	تک دقتی

Slack variable	متغی‌ر کساد
Stationary	ای‌ستان
Stationary points	نقاط ای‌ستان
Subsytem	زی‌ر سی‌ستم
Support Vector Machine	ماشی‌ن بردار پشتی‌بان
Support Vectors	بردارهای پشتی‌بان
Synthesis	سنتز
Time multiplexing	زمان چندگانه
Timing analysis	آنالی‌ز زمانی
Token	علامت
Trade off	موازنه
Truncation	قطع کردن
Unclassified regions	نواحی کلاسه بندی نشده
Vectoring mode	مد برداری
VHSIC (Very-High-Speed Integrated Circuits) Hardware Description Languages	زبان توصی‌ف سخت افزاری مدارهای مجتمع خی‌لی سری‌ع
Random pseudo-negative sampling method	نمونه برداری شبه منفی تصادفی

Abstract: A simple hardware structure for implementation of pairwise Support Vector Machine (SVM) classifiers on FPGA is presented. Training phase of the SVM is performed offline, and the extracted necessary parameters used to implement the test phase on the hardware. In the designed structure, vector multiplication operation and the classification of binary classifiers is done in parallel and simultaneously. In order to realization, the dataset of Persian handwritten digits in three different classes is used for training and testing of SVM. Graphically Simulator, System Generator, has been used to simulate the desired hardware design. Implementation of linear and nonlinear SVM classifier using simple blocks and functions, possibility of increasing dimensions of feature vectors, generalized to allow multiple simultaneous pair wise classes, lack of complexity in hardware design, and simplicity of other blocks and functions used in the design are view of the obvious characteristics of this research. According to simulation results, reaching maximum frequency of 202.840MHz in linear classification, and classification accuracy of 98.67% in nonlinear one, shows outstanding performance of the hardware designed structure.

***Keywords**: hardware structure, Support Vector Machine, FPGA, Persian handwritten digits, System Generator.*

FPGA Implementation of Support Vector Machine Classifier

Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of Master of Science (M.Sc.)

in Electronic Engineering

Department of Electrical Engineering and Robotics

Shahrood University of Technology

By:

Davood Mahmoodi

Supervisor:

Dr. Ali Soleimani

Advisor:

Dr. Hossein Khosravi

Spring 2011