اولین کنفرانس مهندسی مکاترونیک ایران
دانشگاه آزاد اسلامی قزوین
۷ و ۸ خرداد ۱۳۸۲
ICME2003

رفع نویز در تصاویر با استفاده از اتوماتای یادگیر سلولی پیاده‌سازی شده بر روی یک سیستم قابل بازپیکربندی

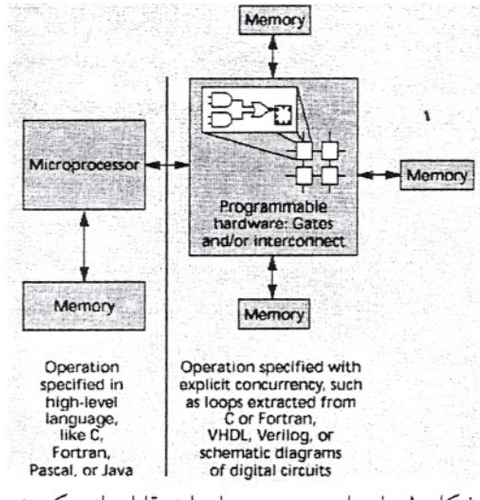
مرتضی صاحب‌الزمانی فرهاد مهدی پور محمد رضا میبدی
آزمایشگاه طراحی مدارهای VLSI
دانشکده مهندسی کامپیوتر و فناوری اطلاعات- دانشگاه صنعتی امیرکبیر
{szamani, mehdipur, meybodi}@ce.aku.ac.ir

چکیده : *سیستم‌های محاسبات قابل بازپیکره‌بندی امکان پیاده‌سازی کاربردها و افزایش کارآیی آنها را فراهم می‌کنند. پیاده‌سازی این نوع سیستم‌ها عمدتاً بر روی افزاره‌های برنامه‌پذیر مانند FPGA انجام می‌شود. اتوماتای سلولی یادگیر به عنوان ترکیبی از اتوماتای سلولی و اتوماتای یادگیر معرفی شده است. ممکن است برای بعضی از انواع کاربردها، پارامترهای مربوط به اتوماتای سلولی و اتوماتای یادگیر در زمان اجرای فرآیند نیاز به تغییر داشته باشند. بنابراین پیکره‌بندی زمان اجرا برای چنین کاربردهایی ضرورت پیدا می‌کند. در این مقاله طراحی و پیاده‌سازی اتوماتای یادگیر سلولی که برای رفع نویز در تصاویر سیاه و سفید مورد استفاده قرار گرفته است، بر روی یک سیستم قابل بازپیکربندی و نتایج حاصل از آن ارائه می‌شود. نتایج آزمایش‌ها حاکی از کارآیی قابل توجه پیاده‌سازی سخت‌افزاری نسبت به پیاده‌سازی نرم‌افزاری است. پیاده‌سازی سخت‌افزاری اتوماتای یادگیر سلولی با در نظر گرفتن قابلیت‌های بازپیکربندی انجام شده است. یکی از مزایای این روش این است که زمان حاصل از اجرای سخت‌افزاری مستقل از ابعد اتوماتای یادگیر سلولی است در حالی که در اجرای نرم‌افزاری به دلیل سریال بودن آن، زمان اجرا به نسبت ابعاد افزایش می‌یابد.*

کلمات کلیدی : سیستم‌های محاسبات قابل باز پیکربندی، اتوماتای یادگیر، اتوماتای سلولی، اتوماتای یادگیر سلولی، پردازش تصویر، افزاره برنامه‌پذیر.

۱-مقدمه : سیستم‌هایی که از اجزاء منطقی با قابلیت برنامه‌ریزی مجدد(مانند FPGA) برای پیاده‌سازی واحدهای محاسباتی و مجموعه دستورالعمل‌های خاص در جهت بهبود کارآیی یک کاربرد خاص بهره می‌گیرند، سیستم‌های محاسبات قابل پیکربندی نامیده می‌شوند. این سیستم‌ها می‌توانند پیکربندی مجدد شده و بر اساس ملزومات الگوریتم، وابستگی داده‌ها و تعاملات موجود با اجزاء خارجی، ازهمان اجزای سخت‌افزاری برای یک پیاده‌سازی یک طرح دیگر استفاده کنند[۳][۲].

قابلیت پیکربندی مجدد سخت‌افزار زمانی که در حال کار می‌باشد، امکان پیاده‌سازی سیستم‌های سخت‌افزار پویای قابل پیکربندی را فراهم می‌کند. سیستم‌هایی که دارای قابلیت بازپیکربندی هستند عموماً شامل یک پردازنده و یک FPGA به عنوان کمک‌پردازنده می‌باشند که اغلب با نام RFU[۱] نامیده می‌شوند. مدل کلی این سیستم‌ها در شکل ۱ آمده است. RFU را می‌توان در زمان اجرای برنامه بر روی پردازنده برای انجام سریع برخی محاسبات برنامه‌ریزی کرد. در زمان اجرا بخشی از کد در حال اجرا بر روی RFU نگاشت می‌شود. اگر در همین زمان نیاز به پیکربندی مجدد و انتقال بخش دیگری از برنامه به RFU باشد و برای این انتقال محلی وجود نداشته باشد، بایستی بخش از RFU را برای بارگذاری بخش جدیدتر خالی کرد. مدیریت این عملیات نیاز به ابزارهای طراحی فیزیکی قدرتمندی می‌باشد. مدیریت RFU می‌تواند بصورت Offline و Online انجام شود. در حالت Offline جریان برنامه از قبل مشخص است. بنابراین بهینه‌سازی RFUها و زمانبندی آنها می‌تواند از قبل صورت گیرد. در نسخه Online، تصمیم‌گیری در رابطه با اینکه کدامیک از عملیات بایستی پیکربندی شود از قبل معلوم نیست. و چون جریان برنامه نامشخص است(مانند سیستم‌های Multithreading)، پیکربندی RFU بصورت On the انجام می‌گیرد[۳][۲].



شکل ۱: اجزای سیستم محاسبات قابل باز پیکربندی

در این مقاله به ارائه نتایج حاصل از پیاده‌سازی سخت‌افزاری اتوماتای یادگیر سلولی بر روی یک افزاره قابل باز پیکربندی می‌پردازیم. همچنین از اتوماتای یادگیر سلولی سخت‌افزاری برای رفع نویز در تصاویر سیاه و سفید استفاده کرده و مقایسه‌ای از نتایج حاصل از پیاده‌سازی نرم‌افزاری و سخت‌افزاری را ارائه خواهیم داد. اتوماتای یادگیر سلولی ترکیبی از اتومای سلولی و اتومای یادگیر است که اخیراً ارائه شده است و برای مدل‌سازی سیستم‌هایی با اجزا ساده و رفتار پیچیده قابل استفاده است. اتومای سلولی[۲] در اواخر دهه ۱۹۴۰ به عنوان مدلی برای بررسی رفتار سیستم‌های پیچیده پیشنهاد شد. یک اتوماتای سلولی

۱- Reconfigurable Functional Unit
۲- Cellular Automata

متشکل از یک شبکه منظم از سلول‌ها است که هر سلول می‌تواند چند مقدار مختلف داشته باشد. اتوماتای یادگیر[1] نیز در اوایل دهه ۱۹۶۰ مطرح شد. یک اتوماتای یادگیر در محیطی تصادفی عمل نموده و قادر است بر اساس ورودیهای دریافت شده از محیط، احتمال انجام عملیات خود را بروز در آورد تا بدین طریق کارآیی خود را بهبود بخشد. هر اتوماتای یادگیر بر اساس یک الگوریتم یادگیری عمل نموده و در طی زمان یاد می‌گیرد که چگونه از طریق فعل و انفعال با محیط، بهترین عمل را انتخاب نماید. مدل اتوماتای یادگیر سلولی[2] (CLA) ترکیبی از دو مدل فوق است که در آن هر سلول در اتوماتای سلولی مجهز به یک اتوماتای یادگیر است که تعیین کننده وضعیت سلول است. اتوماتای سلولی یادگیر دارای کاربردهای متعددی از جمله مدلسازی شبکه های تجارت و پردازش تصاویر می‌باشد.

در بخشهای ۳،۲ و٤ به ترتیب به معرفی اتوماتای سلولی اتوماتای سلولی یادگیر و اتوماتای یادگیر سلولی خواهیم پرداخت. در بخش ٥ نیز نحوه پیاده سازی CLA بر روی سیستم قابل بازپیکربندی را ارائه داده و سپس در بخش ٦ روش رفع نویز در تصاویر را مورد بررسی قرار می‌دهیم. در بخش آخر گزارش به ارائه نتایج حاصل از پیاده‌سازی‌های انجام شده می‌پردازیم.

۲-اتوماتای سلولی

اتوماتای سلولی یک مدل ریاضی برای سیستم‌هایی است که در آنها چندین مؤلفه ساده برای تولید الگوهای پیچیده‌تر با هم همکاری می‌کنند. در اتوماتای سلولی یک مجموعه منظم از سلولها وجود دارد که هر کدام می‌توانند با چند مقدار متفاوت مقداردهی شوند. این سلولها بطور همگام و در زمانهای گسسته بر طبق یک قانون محلی[3] به هنگام رسانی می‌شوند. در تعیین مقدار جدید برای هر سلول، مقادیر سلولهای همسایه نیز تأثیرگذار است. شبکه مربوط به سلولها می‌تواند بصورت یک بعدی، دو بعدی و غیره باشد. با توجه به تعداد مقادیری که سلولها می‌توانند اختیار کنند، اتوماتای سلولی به دو نوع دودویی و چند مقداره تقسیم می‌شود. شکل همسایگی در اتوماتای سلولی نیز ممکن است پریودیک و یا غیرپریدیک باشد. انتخاب قوانین مختلف برای بهنگام‌رسانی، انواع متفاوتی از اتوماتای سلولی را بوجود می‌آورد. این قانون ممکن است بصورت قطعی و یا احتمالاتی باشد. قانون مورد استفاده در اتوماتای سلولی را می‌توان به شکل‌های مختلفی نمایش داد. یک شیوه معمول برای نمایش قوانین استفاده از یک شماره برای هر قانون است. در این روش، شماره قانون برابر با معادل دهدهی مقدار جدید سلول مرکزی به ازای تمامی ترکیبات مختلف سلولهای همسایه است. به عنوان مثال در اتوماتای سلولی یک بعدی، شماره قانون ٥٤ بصورت "۰۰۱۱۰۱۱۰" بیان شده و مطابق با شکل ۲ باعث بروزرسانی مقادیر هر سلول مرکزی $a_i(t)$ نسبت به دو سلول همسایه خود یعنی $a_{i+1}(t)$ و $a_{i-1}(t)$ می‌شود. شکل کلی این قوانین که قوانین عمومی[4] نامیده می‌شوند، بصورت زیر است:

$$a_i(t+1) = \Phi[a_{i-1}(t), a_i(t), a_{i+1}(t)] \qquad \text{رابطه (۱)}$$

$(a_{i-1}(t), a_i(t), a_{i+1}(t))$	(۱،۱،۱)	(۱،۱،۰)	(۱،۰،۱)	(۱،۰،۰)	(۰،۱،۱)	(۰،۱،۰)	(۰،۰،۱)	(۰،۰،۰)
$a_i(t+1)$	۰	۰	۱	۱	۰	۱	۱	۰

شکل ۲: نحوه اعمال قانون در اتوماتای سلولی

[1]- Learning Automata
[2]- Cellular Learning Automata
[3]- Local Rule
[4]- General

متشکل از یک شبکه منظم از سلول‌ها است که هر سلول می‌تواند چند مقدار مختلف داشته باشد. اتوماتای یادگیر[1] نیز در اوایل دهه ۱۹۶۰ مطرح شد. یک اتوماتای یادگیر در محیطی تصادفی عمل نموده و قادر است بر اساس ورودیهای دریافت شده از محیط، احتمال انجام عملیات خود را بروز در آورد تا بدین طریق کارآیی خود را بهبود بخشد. هر اتوماتای یادگیر بر اساس یک الگوریتم یادگیری عمل نموده و در طی زمان یاد می‌گیرد که چگونه از طریق فعل و انفعال با محیط، بهترین عمل را انتخاب نماید. مدل اتوماتای یادگیر سلولی[2] (CLA) ترکیبی از دو مدل فوق است که در آن هر سلول در اتوماتای سلولی مجهز به یک اتوماتای یادگیر است که تعیین کننده وضعیت سلول است. اتوماتای سلولی یادگیر دارای کاربردهای متعددی از جمله مدلسازی شبکه های تجارت و پردازش تصاویر می‌باشد.

در بخشهای ۳،۲ و٤ به ترتیب به معرفی اتوماتای سلولی اتوماتای سلولی یادگیر و اتوماتای یادگیر سلولی خواهیم پرداخت. در بخش ٥ نیز نحوه پیاده سازی CLA بر روی سیستم قابل بازپیکربندی را ارائه داده و سپس در بخش ٦ روش رفع نویز در تصاویر را مورد بررسی قرار می‌دهیم. در بخش آخر گزارش به ارائه نتایج حاصل از پیاده‌سازی‌های انجام شده می‌پردازیم.

۲-اتوماتای سلولی

اتوماتای سلولی یک مدل ریاضی برای سیستم‌هایی است که در آنها چندین مؤلفه ساده برای تولید الگوهای پیچیده‌تر با هم همکاری می‌کنند. در اتوماتای سلولی یک مجموعه منظم از سلولها وجود دارد که هر کدام می‌توانند با چند مقدار متفاوت مقداردهی شوند. این سلولها بطور همگام و در زمانهای گسسته بر طبق یک قانون محلی[3] به هنگام رسانی می‌شوند. در تعیین مقدار جدید برای هر سلول، مقادیر سلولهای همسایه نیز تأثیرگذار است. شبکه مربوط به سلولها می‌تواند بصورت یک بعدی، دو بعدی و غیره باشد. با توجه به تعداد مقادیری که سلولها می‌توانند اختیار کنند، اتوماتای سلولی به دو نوع دودویی و چند مقداره تقسیم می‌شود. شکل همسایگی در اتوماتای سلولی نیز ممکن است پریودیک و یا غیرپریدیک باشد. انتخاب قوانین مختلف برای بهنگام‌رسانی، انواع متفاوتی از اتوماتای سلولی را بوجود می‌آورد. این قانون ممکن است بصورت قطعی و یا احتمالاتی باشد. قانون مورد استفاده در اتوماتای سلولی را می‌توان به شکل‌های مختلفی نمایش داد. یک شیوه معمول برای نمایش قوانین استفاده از یک شماره برای هر قانون است. در این روش، شماره قانون برابر با معادل دهدهی مقدار جدید سلول مرکزی به ازای تمامی ترکیبات مختلف سلولهای همسایه است. به عنوان مثال در اتوماتای سلولی یک بعدی، شماره قانون ٥٤ بصورت "۰۰۱۱۰۱۱۰" بیان شده و مطابق با شکل ۲ باعث بروزرسانی مقادیر هر سلول مرکزی $a_i(t)$ نسبت به دو سلول همسایه خود یعنی $a_{i+1}(t)$ و $a_{i-1}(t)$ می‌شود. شکل کلی این قوانین که قوانین عمومی[4] نامیده می‌شوند، بصورت زیر است:

رابطه (۱) $$a_i(t+1) = \Phi[a_{i-1}(t), a_i(t), a_{i+1}(t)]$$

$(a_{i-1}(t), a_i(t), a_{i+1}(t))$	(۰،۰،۰)	(۰،۰،۱)	(۰،۱،۰)	(۰،۱،۱)	(۱،۰،۰)	(۱،۰،۱)	(۱،۱،۰)	(۱،۱،۱)
$a_i(t+1)$	۰	۱	۱	۰	۱	۱	۰	۰

شکل ۲: نحوه اعمال قانون در اتوماتای سلولی

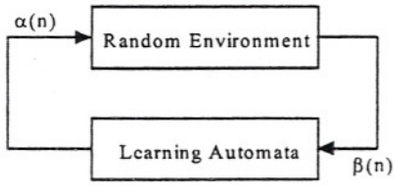
۱- Learning Automata
۲- Cellular Learning Automata
۳- Local Rule
٤- General

۳-اتوماتای یادگیر

اتوماتای یادگیر یک مدل انتزاعی است که تعداد معدودی عمل را می تواند انجام دهد. هر عمل انتخاب شده توسط محیطی احتمالی ارزیابی شده و پاسخی به اتوماتای یادگیر داده می شود. اتوماتای یادگیر از این پاسخ استفاده نموده وعمل خود را برای مرحله بعد انتخاب می کند. شکل ۳ ارتباط بین اتوماتای یادگیر و محیط را نشان می دهد.

محیط[۱] : محیط را می توان توسط سه تایی $E\equiv\{\alpha,\beta,c\}$ نشان داد که در آن $\alpha\equiv\{\alpha_1,\alpha_2,...\alpha_r\}$ مجموعه ورودیها، $\beta\equiv\{\beta_1,\beta_2,...\beta_m\}$ مجموعه خروجیها و $c\equiv\{c_1,c_2,...c_r\}$ مجموعه احتمالهای جریمه می باشد. هر گاه β مجموعه دو عضوی باشد ، محیط از نوع P می باشد. در چنین محیطی $\beta_1=1$ به عنوان جریمه و $\beta_2=0$ به عنوان پاداش در نظر گرفته می شود. در محیط از نوع Q ، $\beta(n)$ می تواند به طور گسسته یک مقدار از مقادیر محدود در فاصله [۰،۱] و در محیط از نوع S، $\beta(n)$ متغیر تصادفی در فاصله [۰،۱] است.

c_i احتمال اینکه عمل α_i نتیجه نامطلوب[۲] داشته باشد می باشد. در محیط ایستا[۳] مقادیر c_i بدون تغییر می مانند ، حال آنکه در محیط غیر ایستا[۴] این مقادیر در طی زمان تغییر می کنند. اتوماتای یادگیربه دو گروه با ساختار ثابت و با ساختار متغیر تقسیم می گردند. در ا دامه به شرح مختصری از اتوماتای یادگیر با ساختار متغیر که در این مقاله استفاده شده است می‌پردازیم



شکل ۳: ارتباط بین اتوماتای یادگیر و محیط

اتوماتای یاد گیر با ساختار متغیر[۵]: اتوماتای یادگیر با ساختار متغیر توسط ٤ تائی $\{\alpha,\beta,p,T\}$ نشان داده می شود که در آن $\alpha\equiv\{\alpha_1,\alpha_2,...,\alpha_r\}$ مجموعه عمل های اتوماتا، $\beta\equiv\{\beta_1,\beta_2,...,\beta_m\}$ مجموعه ورودیهای اتوماتا، $p\equiv\{p_1,p_2,...p_r\}$ بردار احتمال انتخاب هر یک از عملها و $p(n+1)=T[\alpha(n),\beta(n),p(n)]$ الگوریتم یادگیری می باشد. در این نوع از اتوماتاها، اگر عمل α_i در مرحله n ام انتخاب شود و پاسخ مطلوب از محیط دریافت نماید، احتمال $p_i(n)$ افزایش یافته و سایر احتمالها کاهش می یابند. و برای پاسخ نامطلوب احتمال $p_i(n)$ کاهش یافته و سایر احتمالها افزایش می یابند. در هر حال، تغییرات به گونه‌ای صورت می‌گیرد تا حاصل جمع $p_i(n)$ ها همواره ثابت و مساوی یک باقی بماند. الگوریتم زیر یک نمونه از الگوریتمهای یادگیری خطی در اتوماتای با ساختار ثابت است.

الف- پاسخ مطلوب

$$p_i(n+1)=p_i(n)+a[1-p_i(n)]$$ رابطه (۲)

$$p_j(n+1)=(1-a)p_j(n)\qquad \forall j\quad j\neq i$$ رابطه (۳)

ب- پاسخ نامطلوب

$$p_i(n+1)=(1-b)p_i(n)$$ رابطه (٤)

۱- Environment
۲- Unfavorable
۳- Stationary
٤- Non-Stationary
۵- Variable Structure

رابطه (۵) $p_j(n+1)=\frac{b}{r-1}+(1-b)p_j(n) \quad \forall j \quad j \neq i$

در روابط فوق، a پارامتر پاداش و b پارامتر جریمه می باشد. با توجه به مقادیر a و b سه حالت زیر را می توان در نظر گرفت. زمانیکه a و b با هم برابر باشند، الگوریتم را L_{RP} [۱] می نامیم. زمانیکه b از a خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$ [۲] می نامیم و زمانیکه b مساوی صفر باشد، الگوریتم را L_{RI} [۳] می نامیم.

۱. اتوماتای یادگیر سلولی

اتوماتای یادگیر سلولی که اخیراً پیشنهاد شده است، مدلی برای سیستم‌هایی است که از اجزاء ساده‌ای تشکیل شده اند و رفتار هر جزء بر اساس رفتار همسایگانش و نیز تجربیات گذشته اش تعیین و اصلاح می شود. اجزاء ساده تشکیل دهنده این مدل، از طریق کنش و واکنش با یکدیگر می توانند رفتار پیچیده ای از خود نشان دهند. هر اتوماتای یادگیر سلولی، از یک اتوماتای سلولی تشکیل شده است که هر سلول آن به یک یا چند اتوماتای یادگیر مجهز می باشد که وضعیت این سلول را مشخص می سازد. مانند اتوماتای سلولی، یک قانون محلی در محیط حاکم است که تعیین می کند که آیا عمل انتخاب شده توسط یک اتوماتان در یک سلول بایستی پاداش داده شود ویا جریمه شود. عمل دادن پاداش و یا جریمه منجر به بروز در آوردن ساختار اتوماتای یادگیر سلولی بمنظور نیل به یک هدف مشخص می گردد.

یک اتوماتای یادگیر سلولی به صورت پنج تایی $<L,V,Q,\Omega,\Phi>$ نشان داده می شود. $L=\{l_1,l_2,...,l_n\}$ مجموعه سلولهای موجود در اتوماتای یادگیر سلولی می باشد که در یک شبکه کارتزین قرار گرفته اند $V=\{v_i,i\in L\}$ مجموعه سلولهای همسایه یک سلول در اتوماتای یادگیر سلولی است. $Q=\{q_1,q_2,...,q_k\}$ مجموعه اعمال مجاز یک اتوماتای ساکن در یک سلول و $\Omega=\{x:L\to Q\}=Q^N$ فضای حالت و Φ قانون حاکم بر اتوماتای یادگیر سلولی می باشد.

- **همسایگی**

مجموعه $V=\{v_i,i\in L\}$ در صورتی سلولهای همسایه یک سلول در اتوماتای یادگیر سلولی است که دارای دو خصوصیت زیر باشد:

رابطه (٦) ۱) $i\notin v_i \quad \forall i\in L$

رابطه (۷) ۲) $i\in v_i \quad$ iff $\quad j\in v_i \quad \forall i,j\in L$

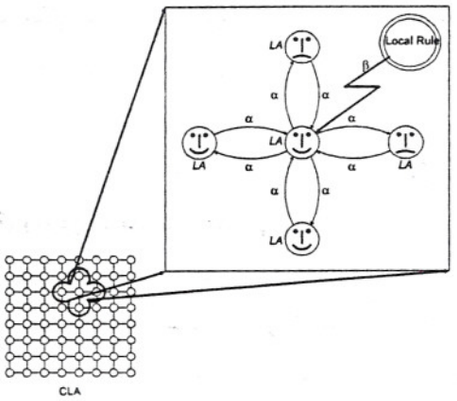
در آن صورت v_i را همسایگی i می نامند. در اتوماتای یادگیر سلولی می توان از ساختارهای مختلفی برای همسایگی استفاده نمود.در حالت کلی هر مجموعه مرتب از سلولها را می توان به عنوان همسایه در نظر گرفت اما معمول‌ترین آنها همسایگی ون نیومن و همسایگی مور می باشند که به نزدیکترین همسایگان مشهور می باشند. این همسایگی ها در شکل ٤ نشان داده شده اند.

۱- Linear Reward Pealty
۲- Linear Reward Epsilon Pealty
۳- Linear Reward Inaction



همسایگی Moor همسایگی Von Neumann

شکل ٤: دو نوع همسایگی در CA دوبعدی(سمت چپ) همسایگی ون نیومن (سمت راست) همسایگی مور



شکل ٥: اتوماتای یادگیر سلولی (CLA)

- **عملکرد اتوماتای یادگیر سلولی**

عملکرد اتوماتای یادگیر سلولی را می توان بصورت زیر شرح داد. در ابتدا هر اتوماتای یادگیر در اتوماتای یادگیر سلولی یکی از اعمال از مجموعه اعمال خود را انتخاب می کند. این عمل می‌تواند بر اساس مشاهدات قبلی ویا بصورت تصادفی انتخاب می شود. عمل انتخاب شده با توجه به اعمال انتخاب شده توسط سلول های همسایه و قانون حاکم بر اتوماتای یادگیر سلولی پاداش داده می‌شود و یا جریمه می‌گردد. با توجه به اینکه عمل انتخاب شده پاداش گرفته و یا جریمه شده است، بردار داخلی اتوماتا بروز می گردد. عمل بروز در آوردن تمام اتوماتاها در اتوماتای یادگیر سلولی بصورت همزمان انجام می شود. بعد از بروز درآوردن هر اتوماتا در اتوماتای یادگیر سلولی، دوباره یک عمل از مجموعه اعمال خود را انتخاب و انجام می دهد. نتیجه عمل منجر به دادن پاداش و یا جریمه به آن عمل می گردد. فرآیند انتخاب عمل و دادن پاداش یا جریمه تا زمانی‌که سیستم به یک حالت پایدار برسد و یا یک معیار از قبل تعریف شده ای بر قرار شود ادامه می یابد. عمل بروز در بردار ساختار اتوماتای یادگیر سلولی توسط الگوریتم یادگیری در هر سلول انجام میگیرد. شکل ٥ اتوماتای یادگیر سلولی را نشان می دهد که در آن از همسایگی ون نیومن استفاده شده است. در این شکل، اتوماتاهایی که خوشحال هستند در مرحله پاداش و اتوماتاهایی که ناراحت هستند در مرحله قبل جریمه شده اند.

- **قوانین**

قوانین در اتوماتای یادگیر سلولی به سه دسته *general* ، *totalistic* و *outer totalistic* تقسیم میگردند [Wolf٨٦][Mitc]. در قوانین *general* مقدار یک سلول در مرحله بعدی به مقادیر همسایه های آن سلول بستگی دارد.

در قوانین *totalistic* مقدار یک سلول تنها به مجموع همسایه های آن سلول بستگی داردو در قوانین *outer totalistic* مقدار یک سلول در مرحله بعدی هم به مقادیر همسایه های آن سلول و هم به خود سلول بستگی دارد.

قوانین *General*: برای بیان قوانین *general* از همسایگی ون نیومن استفاده کرده و همسایگان را بصورتی که در شکل ۷ نشان داده شده است نامگذاری می کنیم. برای قوانین *general* تعریف شده در زیر هر اتوماتان در CLA دارای دو عمل می باشد. اگر اتوماتان عمل α_1 را انتخاب کند سلول آن اتوماتا به صورت پر و اگر اتوماتا عمل α_2 را انتخاب کند سلول آن اتوماتا به صورت خالی نشان داده می شود.برای آشنائی با *general* چند نمونه از این قوانین در زیر آمده است.

قانون *And All*: در این قانون زمانی به عمل انتخاب شده توسط اتوماتا پاداش داده می شود که خود اتوماتا و تمام هشت همسایه اش عمل α_1 را انتخاب کرده باشند و در غیر اینصورت عمل انتخاب شده توسط اتوماتا جریمه می شود. قانون *And All* به صورت زیر بیان می شود.

AND(A۱,A۲,A۳,B۱,B۲,B۳,C۱,C۲,C۳) رابطه (۸)

با توجه به اینکه ارزش عبارت منطقی فوق ورودی اتوماتا می باشد، می‌توان قانون فوق را به صورت دیگری نیز بیان کرد. بر اساس قانون فوق اگر یک اتوماتا در CLA عمل α_2 را انتخاب کند آن عمل قطعاً جریمه خواهد شد. و اگر یک اتوماتا در CLA عمل α_1 را انتخاب کند و همچنین تمام هشت همسایه این اتوماتا عمل α_1 را انتخاب کرده باشند عمل انتخاب شده توسط این اتوماتا یعنی عمل α_1 پاداش خواهد گرفت و در غیر اینصورت جریمه می شود.

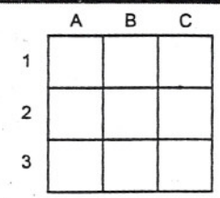
لازم به ذکر است که برای قوانین general مقدار True برای یک قانون بعنوان پاسخ مناسب محیط در نظر گرفته می شود.

قانون *Or All*: در این قانون زمانی به عمل انتخاب شده توسط اتوماتا پاداش داده می شود که خود اتوماتا یا یکی از هشت همسایه اش عمل α_1 را انتخاب کرده باشند و در غیر اینصورت عمل انتخاب شده توسط اتوماتا جریمه می شود. قانون Or All به صورت زیر بیان می شود.

OR(A۱,A۲,A۳,B۱,B۲,B۳,C۱,C۲,C۳) رابطه (۹)

به بیان دیگر بر اساس قانون فوق اگر یک اتوماتا در CLA عمل α_1 را انتخاب کند آن عمل قطعاً پاداش خواهد گرفت . و اگر یک اتوماتا در CLA عمل α_2 را انتخاب کند و حداقل یکی از هشت همسایه این اتوماتا عمل α_1 را انتخاب کرده باشند عمل انتخاب شده توسط این اتوماتا پاداش خواهد گرفت و در غیر اینصورت جریمه می شود.

قوانین *Totalistic*: این دسته از قوانین همانند نوع مشابه خود در اتوماتای سلولی می باشند با این تفاوت که از نتیجه قانون برای پاداش یا جریمه استفاده می شود. این قوانین به صورت N-M نشان داده می شوند که $M=\{M_1,M_2,...\}$ و $N=\{N_1,N_2,...\}$ میباشد و به صورت زیر تفسیر می‌گردند. اگر اتوماتای سلول مرکزی در CLA عمل α_1 را انتخاب کند و تعداد اتوماتاهای همسایه که عمل α_1 را انتخاب کرده باشند متعلق به مجموعه M باشد اتوماتای مرکزی جریمه شده در غیر اینصورت پاداش داده خواهد شد. اگر اتوماتای سلول مرکزی در CLA عمل α_2 را انتخاب کند و تعداد اتوماتاهای همسایه که عمل α_1 را انتخاب کرده باشند متعلق به مجموعه N باشد اتوماتای مرکزی جریمه در غیر اینصورت پاداش داده خواهد شد.



شکل ٦: نحوه نامگذاری همسایگی

۲. پیاده سازی CLA بر روی سیستم قابل بازپیکربندی

CLA دارای چند خاصیت ذاتی است که موجب می‌شود، پیاده‌سازی آن به وسیله یک سیستم محاسبات قابل بازپیکربندی به نحو مناسبی امکان‌پذیر شود. موازات ذاتی موجود در سلول‌ها، نیاز به حافظه کم، وجود ارتباطات محلی بین سلول‌های مجاور و وجود الگوهای تصمیم‌گیری نسبتاً ساده از جمله عواملی هستند که بیانگر مناسب بودن این مسأله برای پیاده‌سازی بر روی یک سیستم قابل بازپیکربندی می‌باشند. CLA برای حل مسائل مختلف مورد استفاده قرار گرفته است. در حل برخی مسائل لازم است که قوانین یادگیری در ضمن آموزش CLA تغییر یابند. گاهاً نیزقوانین یادگیری ثابت بوده و پارامترهای مربوط به قوانین و یا مسأله دچار تغییر می‌شوند. انجام هر نوع تغییر در قوانین یادگیری و یا پارامترهای آن باعث ایجاد تغییرات جزئی و یا کلی در سخت‌افزار پیاده‌سازی شده برای سلول‌های CLA می‌شود. بنابراین، در طول تکرارهای مختلف آموزش CLA، لازم است که پیکربندی آن دچار تغییر شود. پیاده سازی این سیستم، به شکل نرم‌افزاری و سخت‌افزاری صورت گرفته است. در ادامه به بررسی جزئیات این پیاده‌سازیها و ارائه نتایج حاصل می‌پردازیم.

- **پیاده‌سازی نرم‌افزاری**

پیاده‌سازی نرم‌افزاری CLA در محیط C++Builder ۵ انجام شده است. کامپایل و بهینه‌سازی لازم بر روی کد CLA صورت گرفته است. در شکل ۷، بخش اصلی این برنامه را مشاهده می‌کنید. در پیاده‌سازی انجام شده، از اتوماتای یادگیر با ساختار ثابت استفاده شده است که دارای دو عمل می‌باشد. پیاده‌سازی اتوماتای سلولی نیز بر اساس قانون Majority می‌باشد. بر اساس این قانون اگر مجموع تعداد عمل(اول) انجام شده(از دو عمل موجود)، توسط هر سلول و همسایه‌های آن از یک حد آستانه باشد، در این صورت سلول مورد نظر همان عمل اول را برای انجام دادن انتخاب خواهد نمود. مقدار حد آستانه قابل تغییر می‌باشد. در پیاده‌سازی سخت‌افزاری، این مسأله می‌تواند باعث ایجاد تغییر در سخت‌افزار سلول می‌شود. با اندازه‌گیری زمان اجرای کد فوق بر روی یک ماشین PC با پردازنده پنتیوم ٤ با ۵۱۲مگابایت حافظه RAM و ۲۱۰۰مگاهرتز ، نتایج جدول ۱ حاصل شده است. آزمایش‌های انجام شده برای CLA با ابعاد مختلف و با دورهای متفاوت انجام گرفته است. دو نتیجه مهم حاصل از آزمایش‌ها به صورت زیر است:

زمان اجرای برنامه کاملاً وابسته به ابعاد CLA تغییر می‌کند.

زمان اجرای برنامه به صورت خطی به تعداد تکرار الگوریتم وابسته است.

پیاده‌سازی سخت‌افزاری

پیاده‌سازی انجام شده، ماجول‌هایی برای اتوماتای یادگیر و اتوماتای سلولی پیاده‌سازی شده است. در شکل ۷ دیاگرام کلاری مربوط به مدار سنتز شده یک سلول اتوماتای یادگیر سلولی مجهز به اتوماتای یادگیر $L_{2N,2}$ که از قانون M استفاده می‌کند دیده می‌شود. برای پیاده‌سازی سخت‌افزاری، کد مربوط به هر یک از اجزای CLA با استفاده از

VHDL نوشته شده و بعد از شبیه‌سازی عملکرد آن با استفاده از نرم‌افزار Modelsim ۵،۵ سنتز آن به وسیله نرم‌افزار Leonardo Expectrum ۲۰۰۱ بر روی افزاره برنامه‌پذیر انجام شده است.

توصیف ارائه شده برای CLA برروی انواع FPGA پیاده‌سازی شده است. نتایج حاصل نشان می‌دهد که امکان پیاده‌سازی CLA بر روی انواع مختلف FPGA با فرکانس حداقل ۲۰مگاهرتز وجود دارد. مشاهده می‌شود که امکان پیاده‌سازی سخت-افزاری CLA بر روی افزاره‌های برنامه‌پذیر با فرکانس ۲۰۰مگاهرتز نیز وجود دارد. با توجه به موازی بودن ساختار CLA، می‌توان به کارآیی بالایی نسبت به پیاده‌سازی نرم‌افزاری که ماهیت سریال دارد دست یافت.

```
CLA( ){
_ init_cla( );
        for (k= ۰; k< MaxIter  ; k++){
         for (t= ۱; t<= INPUT_PATTERNS; t++){
                  _init_input_patterns ( ) ;
                  ۲D_CA();
                  for (i= ۰; i< MaxRow; i++)
                    for (j= ۰; j< MaxCol; j++)
                                        la[i][j]= L۲N-۲();
        }
      }
```

شکل ۷: شبه-کد نرم‌افزاری CLA

جدول ۱ : زمان اجرای نرم‌افزاری CLA

ابعاد CLA	تعداد تکرار	زمان سپری شده(بر حسب میلی ثانیه)
۱۰×۱۰	۱۰۰۰۰	۲۶۰
۱۰×۱۰	۵۰۰۰۰	۱۳۲۰
۲۰ ×۲۰	۱۰۰۰۰	۹۹۰
۲۰ ×۲۰	۵۰۰۰۰	۵۰۰۰
۳۰×۳۰	۱۰۰۰۰	۲۲۳۰
۳۰×۳۰	۵۰۰۰۰	۱۱۲۵۰

مشکلی که در پیاده سازی سخت‌افزاری CLA بر روی FPGA به چشم می‌آید. افزایش I/O مورد نیاز در صورت افزایش ابعاد CLA می‌باشد. برای رفع این مشکل سلول CLA مطابق شکل ۸ طوری طراحی شده است که امکان جابجایی نتیجه حاصل از سلول به سلول بعدی وجود داشته باشد. خروجی هر سلول به ورودی سلول سمت راست آن قابل انتقال است. بنابراین کافی است با فعال کردن ورودی observe از هر سلول و اعمال سیگنال ساعت، خروجی آن را به سلول بعدی و در نهایت با تکرار این فرآیند به آخرین سلول سمت راست انتقال دهیم. برای یک CLA با ابعاد NxN به تعداد N سیگنال ساعت لازم است تا وضعیت تمامی سلول‌ها کاملاً قابل مشاهده باشد.

مشکل دیگر مسأله محدود بودن ظرفیت FPGA برای پیاده‌سازی CLA است. در صورت افزایش ابعاد CLA ممکن است ناچار به استفاده از سیستم مولتی-FPGA باشیم که مطمئناً این مسأله در ضمن افزایش هزینه‌های پیاده‌سازی باعث کاهش کارآیی نیز خواهد شد.

۳. استفاده از اتوماتای یادگیر سلولی برای رفع نویز

یکی از کاربردهای اتوماتای یادگیر سلولی در پردازش تصویر می‌باشد. در این بخش به بیان جزئیات و نتایج حاصل از پیاده‌سازی سخت‌افزاری CLA که از آن برای رفع نویز در تصاویر سیاه وسفید استفاده شده است می‌پردازیم. فرآیند یادگیری CLA را در شکل ۹ مشاهده می‌کنید. در این فرآیند از اتوماتای یادگیر $L_{2N,2}$ شامل دو عمل استفاده شده است. یک عمل متناظر با رنگ سیاه و عمل دیگر متناظر با رنگ سفید انتخاب شده است. همچنین از قانون Majority برای پاداش و یا جریمه دادن به سلول‌ها استفاده می‌شود. بر اساس این قانون اگر تعداد سلول‌های همسایه‌ای که یک عمل را انتخاب کرده‌اند بهمراه خود سلول از یک حد آستانه(مثلاً تعداد نصف تعداد سلول‌های موجود در همسایگی) بیشتر شود به سلول مورد نظر پاداش و در غیر اینصورت جریمه داده می‌شود. در الگوریتم اولیه از همسایگی نوع مور استفاده می‌شود. در یک الگوریتم بهبود یافته که از آن برای پیاده‌سازی سیستم قابل باز پیکربندی استفاده کرده‌ایم از دو نوع همسایگی مور و ون نیومن استفاده می‌شود. به این ترتیب که اگر تعداد مجموع سلول‌هایی که یک عمل را انتخاب کرده‌اند با مقدار آستانه برابر باشد در اینصورت همسایگی از نوع مور به ون نیومن تغییر می‌یابد. این تغییر همسایگی در تغییر ساختار سخت‌افزار نیز تاثیرگذار است. الگوریتم اخیر که از همسایگی ترکیبی بهره می‌گیرد، بر اساس شبیه‌سازی نرم‌افزاری انجام شده دارای نتایج بهتری نسبت به روش اولیه است.

٤. نتایج حاصل

آزمایش‌های انجام شده حاکی از آن است که استفاده از روش پیاده‌سازی سخت‌افزاری باعث بهبود قابل توجه در زمان جرای فرآیند می‌شود. پیاده‌سازی روش رفع نویز مبتنی بر CLA بر روی تراشه VIRTEX از شرکت Xilinx انجام شده ست. زمان لازم برای اجرای نسخه‌های نرم‌افزاری و سخت‌افزاری اندازه‌گیری شده است. در آزمایش‌های انجام شده، به صاویر ۳۰٪ نویز اضافه و سپس از روش‌های نرم‌افزاری و سخت‌افزاری برای رفع نویز در تصاویر استفاده شده است. CLA ورد استفاده متناسب با اندازه تصاویر دارای ابعاد ۱۲۸X۱۲۸ بوده است. با توجه به زمانهای اجرای حاصل که در جدول ۳ عکس شده است، CLA قابل پیکربندی کارآیی روش رفع نویز را حداقل به اندازه ۳۰۰۰ برابر نسبت به روش نرم‌افزاری بود بخشیده است. از مزایای مهم دیگر این روش به غیر از کارآیی قابل توجه آن، عدم وابستگی زمان اجرای روش سخت-زاری به ابعاد CLA می‌باشد که این به دلیل ساختار موازی ذاتی موجود در آن است. در حالی‌که در روش نرم‌افزاری به یل ماهیت سریال آن، زمان اجرا به همان نسبتی که تعداد سلول‌ها زیاد می‌شود افزایش پیدا می‌کند. در شکل ۱۰ نمونه‌ای از صاویر رفع نویز شده را مشاهده می‌کنید.

مراجع

[۱] Adamatzky, A., "Identification of Cellular Automata", Taylor & Francis Ltd., ۱۹۹٤.

[۲] K. Compton, S. Hauck, "Reconfigurable Computing: A Survey of Systems and Software", ACM Computing Surveys(CSUR), vol. ۳٤, Issue ۲, June ۲۰۰۲.

[۳] S. Hauck, A. Agrawal, "Software Technologies for Reconfigurable Systems", Northwestern University Department Of Electrical and Computer Engineering, Technical report, ۱۹۹٦.

[٤] Kharazmi, M. R. and Meybodi, M. R., "Application of Cellular Learning Automata to Image Segmentation", Proceeding of Tenth Conference on Electrical Engineering (۱۰th ICEE), University of Tabriz, Vol. ۱, pp. ۲۹۸-۳۰٦, ۲۰۰۱.

[٥] Lakshmivarahan, S., "Learning Algorithms: Theory and Applications", New York, Springer Verlag, ۱۹۸۱.

[٦] Mars, P., Chen, J.R. and Nambir, R., "Learning Algorithms: Theory and Applications in Signal Processing, Control and Communications", CRC Press, Inc., pp. ٥-۲٤, ۱۹۹٦.

[۷] Meybodi, M.R. and Lakshmivarahan, S., "ε – Optimality of a General Class of Absorbing Barrier Learning Algorithms", Information Sciences, Vol. ۲۸, pp. ۱-۲۰, ۱۹۸۲.

[۸] Meybodi, M. R. and Beygi, H.Taherkhani, M.," Cellular Learning Automata", Proceedings of ٦th Annual CSI Computer Conference, Computer Engineering Department, University of Isfahan, pp. ۱٥۳-۱٥٦, ۲۰-۲۲ Feb ۲۰۰۱.

[۹] Meybodi, M. R. and Khojasteh, M. R., "Cellular Learning Automata as a Model for Commerce Networks", Proceedings of ٦th Annual CSI Computer Conference, CE Department, University of Isfahan, pp. ۲۸٤ –۲۹٥, ۲۰-۲۲ Feb. ۲۰۰۱.

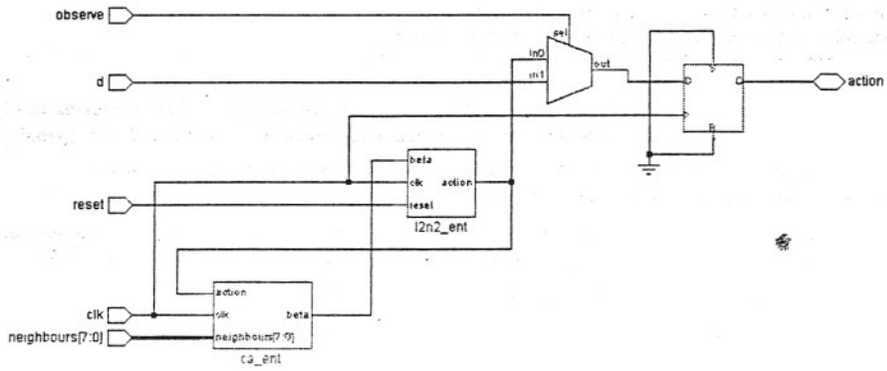
[۱۰] Mitchell, M., "Computation in Cellular Automata: A Selected Review", Technical Report, Santa Fe Institute, Santa Fe, U.S.A., ۱۹۹٦.

[۱۱] Narendra, K.S. and Thathachar, M.A.L., "Learning Automata: An Introduction", Prentice Hall, Inc., ۱۹۸۹.

[۱۲] Taherkhani, M., "Proposing and Studying of Cellular Learning Automata as a Tool for Modeling Systems", M.Sc. Thesis, Computer Eng. Dept., Amirkabir University of Technology, Tehran, Iran, ۲۰۰۰.

[۱۳] Toffoli, T. and Margolus, N., "Cellular Automata Machines", the MIT Press, ۱۹۸۷.

[۱٤] Wolfram, S., "Statistical Mechanics of Cellular Automata", Review of Modern Physics, ٥٥, pp. ٦٠١-٦٤٤, ۱۹۸۳.



شکل ۸: سلول سخت‌افزاری

جدول ۲ : نتایج حاصل از پیاده‌سازی CLA بر روی انواع FPGA

نوع FPGA مورد استفاده	حداکثر تأخیر(بر حسب نانو ثانیه)
ACEX	۰٤،٥۰
APEX۲۰K	۲۰،۲۲
FLEX۱۰K	۲۲،٤۰
MAX٥۰۰۰	۳۹،۰۰
MAX۷۰۰۰	۱۹،۰۰
SPARTAN XL	۱۹،۱۸
VIRTEX E	۱۱،۱٥
XC۹٥۰۰XV	۲٦،٥۰

جدول ۳ : مقایسه زمان‌های اجرای نرم‌افزاری و سخت‌افزاری

Image No.	Software running time(sec)	Hardware running time(μsec)	speedup

۱	۰،۸	۲۱۰	۳۸۰۹
۲	۰،۷۵	۱۹٦	۳۸۲٦
۳	۰،۸۷	۲۱۰	٤۱٤۲
٤	۰،۸	۱۸۲	٤۳۹٥
٥	۰،۰٦	۱٤	٤۲۸٥
٦	۰،۷٤	۱٥٤	٤۸۰٥
۷	۰،۱۳	٤۲	۳۰۹٥

Assign a cell of CLA to each pixel of image
Select one of the two actions for each cell of CLA randomly
While stability criteria is not satisfied do
*** Majority algorithm for emitting reward or penalty**
For each cell in CLA do
If the number of neighborhood cells and cell itself which performs an identical action is greater th a threshold value (e.g. ٥) then *Beta*= ٠(reward) otherwise *Beta*= ١(penalty)
*** Learning Automaton receives Beta and determines an appropriate action as follows**
If *Beta*= ٠(cell has received reward) then
If current state of learning automata (*state_no*) is less than the number of states for each action (*N*) then *state_n* *state_no* +١ otherwise *state_no* = *state_no* −١
If *Beta*= ١ (cell has received penalty) *then*
If *state_no*= *N* then *state_no* = ٢*N* else if *state_no*= ٢*N* then *state_no* = *N* Otherwise
state_no= *state_no* +١;
*** Determining appropriate action for cell**
If *state_no*<= *N* and *state_no*>= ١ then *action*=١ (select black color for respective pixel) Otherwise actio

شکل ۹: فرآیند یادگیری CLA برای رفع نویز در تصاویر سیاه و سفید



(ج) (ب) (الف)

شکل ۱۰: حذف نویز در تصاویر سیاه و سفید(الف) تصویر اولیه(ب) تصویر نویزی(ج) تصویر رفع نویز شده