مجله علمی پژوهشی «رادار»
سال ششم، شماره ۱، بهار و تابستان ۱۳۹۷؛ ص ۲۶- ۱۷

طراحی و پیاده‌سازی کارآمد STAP به‌روش چندبرداری برای آشکارسازی هدف در رادارهای هوایی

نرجس حسنی‌خواه[۱]، سیاوش امین‌نژاد[۲*]، غفار درویش[۳]، محمدرضا منیری[۴]

۱- دانشجوی دکتری، دانشگاه آزاد اسلامی، واحد علوم و تحقیقات، گروه مهندسی برق- الکترونیک، تهران، ایران ۲- استادیار، دانشگاه گیلان،
۳- دانشیار، دانشگاه آزاد اسلامی، واحد علوم و تحقیقات، گروه مهندسی برق- الکترونیک، تهران، ایران
۴- استادیار، دانشگاه آزاد اسلامی، واحد یادگار امام خمینی (ره) شهر ری، دانشکده برق، تهران، ایران
(دریافت: ۹۶/۰۸/۳؛ پذیرش: ۹۷/۰۶/۲۱)

چکیده

پردازش وفقی زمانی- مکانی (STAP) یک روش اساسی در بهبود عملکرد رادارهایی است که در حضور اختلالات شدید دینامیکی مانند کلاتر و جمینگ عمل می‌کنند. عملکرد STAP بر اساس نمونه‌برداری با نرخ بسیار بالا از سیگنال‌هایی است که به‌طور هم‌زمان از چندین آرایهٔ آنتنی و تعدادی پالس دریافت شده‌اند. لذا، حجم محاسبات بسیار بالا بوده و پیاده‌سازی آن نیز دشوار است. در این مقاله، روش چندبرداری برای کاهش حجم محاسبات STAP ارائه گردیده است. نتایج پیاده‌سازی نشان می‌دهد که روش پیشنهادی علاوه بر کاهش حجم محاسبات منجر به کاهش منابع سخت‌افزاری، کاهش توان مصرفی و افزایش حداکثر فرکانس عملیاتی نیز می‌شود. به‌طور مثال، حجم محاسبات با ۶ آرایهٔ آنتنی، ۱۰ پالس دریافتی و ۲۰۰ نمونه فاصله با اندازهٔ بردار ۱۷، ۲۸٫۱ GFLOPs با حداکثر فرکانس ۲۷۸ MHz و زمان اجرای ۲۶۲ µs روی تراشهٔ Arria 10 به‌دست آمده است. بنابراین، روش چندبرداری می‌تواند نیازمندی‌های زمان- آنی وزن‌های STAP را برآورده سازد. الگوی وفقی زاویه- داپلر و تست استاتیکی برای آشکارسازی هدف نشان می‌دهند که استفاده از روش مذکور برای محاسبهٔ وزن‌ها بسیار کاربردی است.

واژگان کلیدی

پردازش وفقی زمانی- مکانی، جمینگ، رادار هوایی، کلاتر

۱- مقدمه

پردازش وفقی زمانی- مکانی (STAP)[۱] یک پردازش سیگنال پیشرفته است که برای تشخیص وجودِ یک شئ در سطح زمین و تعیینِ بُرد آن به‌کار می‌رود. STAP که از ترکیبِ دو پردازش در حوزه‌های زمانی و مکانی حاصل می‌شود، سیگنال‌های دریافتی از چندین آنتن در یک آرایهٔ مکانی و از چندین پالس دریافتی در حوزهٔ زمان را به‌شکل وفقی با هم ترکیب می‌نماید [۳-۱]. همچنین به شیوه‌های متفاوتی می‌تواند عملکردِ رادارهای هوایی را بهبود بخشد. از جمله می‌توان به تشخیصِ اهدافی با سرعت کم به‌وسیله حذف بهترِ لُبِ[۲] اصلی کلاتر، تشخیص اهدافی با سطح مقطع کوچک که ممکن است توسط لُبِ جانبی کلاتر پوشانده شوند و تشخیص اهداف در محیط‌های حاوی کلاتر و جمینگ، اشاره نمود [۵-۴].

هرچند که اصول STAP در اوایل ۱۹۷۰ بنا نهاده شد [۶]، ولی با توجه به حجم بسیار سنگین محاسبات و لزوم اجرای آن در بازهٔ زمانی بسیار کوتاه، تلاش برای تحقق و پیاده‌سازی زمان-آنی[۳] آن به‌طور کامل موفقیت‌آمیز نبوده است. پیچیدگی محاسباتی STAP قابلیت‌های محاسباتی پردازنده‌ها را محدود ساخته است. یک پردازندهٔ STAP مجموعه‌ای از وزن‌های وفقی را از حل یک مدل زمان- آنی که شامل معادلات خطی از مرتبهٔ (NM)[۳] است، به‌دست می‌آورد. در یک شیوهٔ کاملاً وفقی، هنگامی که رادار حجمی از فضا را اسکن نموده و در محدودهٔ سرعت معمولِ هدف به‌دنبال آن می‌گردد، فاصلهٔ زمانی پردازش پیوسته معمولاً کمتر از یک ثانیه خواهد بود. همچنین، یک پردازندهٔ STAP باید بتواند مسأله محاسبهٔ وزن‌های وفقی را چندین بار شکل داده و در این فاصلهٔ زمانی کوتاه حل نماید. حاصل چنین شرایطی می‌تواند بازدهٔ محاسباتی از مرتبهٔ صدها

*رایانامه نویسنده پاسخگو: amin@liv.ac.uk

1- Space time adaptive processing (STAP)
2- Lobe
3 -Real-time

میلیارد عملیات ممیز شناور در ثانیه باشد. بنابراین، پیاده‌سازی کارآمد STAP با وجود جریان دادهٔ بزرگ یکی از چالش‌های عمده در این حوزه بوده و پردازش در واحد اندازه، توان و وزن (SWaP)[1] نیز اهمیتِ خاصی دارد [۷-۹]. در سکوی رادارهای هوابُرد برای حداقل نمودن SWaP به الگوریتم‌های پردازش سیگنال با پیچیدگی محاسباتی کم، نیاز است. از آنجا که الگوریتمِ پیشرفتهٔ پردازشِ سیگنالِ STAP به‌شدت محاسباتی است؛ نیازهای آموزشی و بار محاسباتی آن با قید SWaP پایین متناقض می‌باشد. بنابراین، STAP چالشی را برای طراحان پردازش سیگنال مطرح نموده؛ به‌طوریکه می‌توان گفت هدفِ اصلی بهبود سامانه‌های STAP به افزایشِ سرعت مورد نیاز آنها برای عملکرد کارآمدتر در شرایط پویا و مختلط وابسته است. همچنین اکثر پردازنده‌های دیجیتال از طول کلمهٔ ۱۶ بیتی و فرمت ممیز ثابت بهره می‌گیرند که برای STAP مناسب نمی‌باشد. گزینهٔ دیگر استفاده از پردازنده‌های ممیز شناور است که محدودهٔ دینامیکی وسیعی را فراهم می‌نمایند. بنابراین، پیاده‌سازی ممیز شناور STAP از مهمترین حوزه‌های تحقیقاتی روز به‌شمار می‌رود.

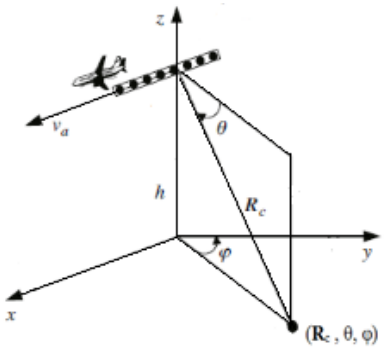
در گذشته، ابتدا از کامپیوترهای با کارآیی بالا (HPC)[2]ها برای پیاده‌سازی STAP استفاده می‌شد [۱۰-۱۱]. با عرضهٔ تراشه‌های DSP[3]، از آنها برای پیاده‌سازی زمان- آنی STAP استفاده گردید [۱۲-۱۳]. با توسعه و گسترش کارآیی STAP در حوزه‌های مختلف، انگیزه‌های بیشتری برای پیاده‌سازی آن ایجاد شد. در مرجع [۱۴]، چهار شیوهٔ پیاده‌سازی تجزیهٔ QR (QRD) برای کاربرد STAP شامل الگوریتم گرام- اشمیت کلاسیک (CGS)[4]، الگوریتم گرام- اشمیت اصلاح‌شده (MGS)[5]، دوران گیونز (GR)[6] و تبدیل هاوس هولدر (HT)[7] بررسی شدند و نتایج حاصل از پیاده‌سازی بر روی HPC و DSP مقایسه گردید. نتایج نشان داد که الگوریتم MGS از سرعت بالاتری نسبت به سایر روش‌ها برخوردار است. همچنین سرعت پیاده‌سازی DSPها بالاتر از HPCها است. در مرجع [۱۵] نیز، طراحی و پیاده‌سازی زمان- آنی بخش محاسبات QRD با استفاده از الگوریتم MGS روی FPGA انجام گرفت. هر چند که طراحی ارائه‌شده از مصرف منابع سخت‌افزاری مناسبی برخوردار بود، ولی استفادهٔ صرف از یک اندازهٔ ماتریسی کوچک و عدم تست انعطاف‌پذیری و کارآمدی آن برای اندازه های ماتریسی بزرگ از جملهٔ معایب تحقیق مذکور به‌شمار می‌رود. در مطالعهٔ دیگری، طراحی مبتنی بر FPGA الگوریتم D^3 و یک بستهٔ نرم‌افزاری جدید برای STAP ارائه شد [۱۶]. نتایج آنها با تست چند اندازهٔ ماتریسی کوچک نشان داد که حداکثر فرکانس عملیاتی (F_{max}) بسیار پایین و مصرف منابع سخت‌افزاری بسیار بالا است. مرجع [۱۷] به پیاده‌سازی STAP روی FPGA برای آشکارسازی هدف در رادارهای دوپایه و غیرفعال اختصاص یافته است. این تحقیق بر طراحی و پیاده‌سازی فیلترینگ تطبیق یافته تمرکز داشت و به حجم محاسبات موردنیاز در پیاده‌سازی STAP اشاره‌ای نشد. در مرجع [۱۸] نیز به پیاده‌سازی STAP روی GPU[8]های مختلف پرداخته شد؛ ولی در نهایت توان مصرفی به ۷۵ W و زمان محاسبات به ۳ ms کاهش یافت. بنابراین، پیاده‌سازی STAP بر روی پردازشگرهای قابل برنامه‌ریزی هنوز به‌عنوان یک مسأله چالش برانگیز باقی مانده است و در این بخش نیاز به فعالیت و پژوهش بیشتری محسوس است. لذا در این مقاله، به طراحی و پیاده‌سازی مدلی کارآمد برای STAP پرداخته‌ایم. ابتدا، بلوک دیاگرامی برای محاسبهٔ بردار وزن بدون نیاز به محاسبهٔ صریح ماتریس کواریانس ارائه گردید. سپس با انجام اصلاحاتی در دستورات QRD-MGS توانستیم ضمن خارج نمودن مسیر کامل محاسبهٔ Q، سرعت الگوریتم را به‌طور نسبی افزایش داده و طراحی QRD را ساده‌تر نمائیم. ما برای برقراری مصالحه بین زمانِ اجرای محاسبات و سایر پارامترها مانند توان مصرفی، مصرف منابع سخت‌افزاری و بار محاسباتی، بلوک QRD را به‌صورت انعطاف‌پذیر با در نظر گرفتن پارامتری به‌نام "اندازهٔ بردار" طراحی نمودیم. از آنجا که پردازش هر ماتریس به‌صورت ستونی (برداری) انجام می‌شود؛ با انتخاب اندازهٔ بردار دلخواه، هر ستون به قسمت‌های کوچکتری تقسیم شده و پردازش هر یک از بخش‌ها به‌ترتیب انجام می‌شود. ما روش پردازش چندبرداری را برای پیاده‌سازی کارآمد ماتریس‌های با ابعاد بزرگ در نظر گرفتیم که در حوزهٔ STAP تاکنون چنین روشی استفاده نشده است. برای سنجش صحت بردار وزن نیز از الگوی وفقی زاویه- داپلر و تست استاتیکی برای تشخیص هدف استفاده می‌شود.

ادامهٔ مقاله به‌صورت زیر سازمان یافته است. در بخش بعدی مدل سامانه شرح داده می‌شود. سپس روش ارائه‌شده برای طراحی و پیاده‌سازی STAP و ساختارهای پیشنهادی آن معرفی خواهند شد. نتایج شبیه‌سازی و پیاده‌سازی به‌ترتیب در بخش‌های چهارم و پنجم ارائه می‌شوند. بخش ششم نیز به نتیجه‌گیری اختصاص دارد.

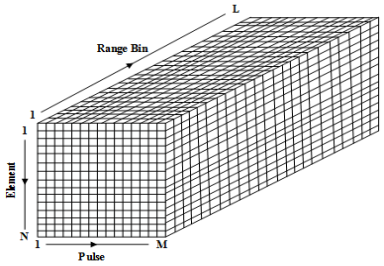
1- Size, weight and power (SWaP)
2- High performance computer (HPC)
3- Digital signal processing (DSP)
4- Classical Gram-schmidt (CGS)
5- Modified gram-schmidt (MGS)
6- Givens rotation (GR)
7- Householder transformation (HT)
8- Graphics processor unit (GPU)

۲- مدل سامانه

مدل سامانه مطابق شکل (۱) در نظر گرفته شده است. در این شکل، یک رادار پهلونگر[1] پالسی و تک پایه با آنتنی با چند مرکز فاز مانند آنتن آرایهٔ فازی درنظرگرفته شده که بر روی یک هوابُرد قرار دارد. هوابُرد در امتداد محورِ آنتن که متشکل از یک آرایهٔ خطی یکنواخت است؛ در ارتفاع h از سطح زمین و با سرعت v_a در حال حرکت است. مختصات هدف با (R_c, θ, φ) نشان داده شده است. R_c بُرد هدف و (θ, φ) نیز موقعیت مکانی هدف را مشخص می‌کنند. سیگنال دریافتی مطابق شکل (۲) به‌صورت یک مکعب داده است که N تعداد عناصر آرایهٔ آنتنی و M تعداد پالس‌های دریافتی است. در فاصلهٔ هر پالس، سیگنال‌های دریافتی با نرخ بالایی نمونه‌برداری می‌شوند که نمونه‌های فاصله نامیده شده و با L نشان داده شده‌اند. سیگنال دریافتی خالص نبوده و با سیگنال‌های ناخواسته‌ای همچون نویز و تداخل‌هایی مانند کلاتر و جمینگ همراه است. STAP عمل فیلترینگ برداری را بر روی ترکیب داده‌های عناصر آرایهٔ آنتنی و پالس‌های مجاور برای یک نمونه فاصلهٔ خاص انجام می‌دهد. این روند برای تمامی نمونه‌های فاصله صورت می‌گیرد.



شکل (۱): مدل سامانه



شکل (۲): مکعب دادهٔ STAP

با محاسبهٔ وزن‌های وفقی می‌توان سیگنال مطلوب را با کیفیت مناسب از مجموع سیگنال‌های دریافتی تفکیک نمود. در STAP وزن‌های بهینه از رابطهٔ (۱) محاسبه می‌شوند. t بردار فرمان[2] مکان-زمان، S_I ماتریس کواریانس تداخل و k ضریب نرمالیزه است.

$$w = kS_I^{-1}t^* \qquad (۱)$$

t مطابق رابطهٔ (۲) از ضرب کرونیکر[3] بردارهای فرمان مکانی و زمانی حاصل می‌شود که به‌ترتیب در روابط (۳) و (۴) نشان داده شده است.

$$t = \alpha(\theta,\varphi) \otimes \alpha(f_s) \qquad (۲)$$

$$\alpha(\theta,\varphi) = \left[1,.....,e^{j\,2\pi(N-1)\frac{d}{\lambda}\mathrm{Sin}\theta\mathrm{Cos}\varphi}\right]^T \qquad (۳)$$

$$\alpha(f_s) = \left[1,.....,e^{j\,2\pi(M-1)f_s/f_r}\right]^T \qquad (۴)$$

λ طول موج، d فاصلهٔ بین آرایه‌های آنتنی مجاور، f_s فرکانس داپلر و f_r فرکانس تکرار پالس است.

برای داشتن بهرهٔ واحد در جهت سیگنال مطلوب، می‌توان از قید $w't=1$ استفاده نمود که به آن شکل‌دهندهٔ پرتو بدون اعوجاج اطلاق می‌شود. با اعمال این قید به رابطهٔ (۲)، ضریب k مطابق رابطهٔ (۵) تعیین می‌شود [۱۹].

$$k = \frac{1}{t^H\left(S_I^{-1}t^*\right)^*} \qquad (۵)$$

با قراردادن رابطهٔ (۵) در رابطهٔ (۱)، رابطهٔ (۶) برای بردار وزن به‌دست می‌آید.

$$w = kS_I^{-1}t^* = \frac{S_I^{-1}t^*}{t^H\left(S_I^{-1}t^*\right)^*} = \frac{u}{t^Hu^*} \qquad (۶)$$

در رابطهٔ فوق، $u \equiv S_I^{-1}t^*$ است. محاسبهٔ مستقیم w با به‌دست آوردن معکوس $\boldsymbol{S_I}$ برای مسائلی همچون شکل‌دهنده‌های پرتو وفقی با تعداد متوسط آنتن‌ها امکان‌پذیر است. اما در STAP به دلیل ابعاد بالای ماتریس $\boldsymbol{S_I}$، محاسبهٔ معکوس آن از لحاظ عددی حساس و محاسباتش بسیار پیچیده می‌باشد. لذا، به‌ندرت این‌کار صورت می‌گیرد. از آنجا که در رابطهٔ (۶) نتیجهٔ نهایی w است، $\boldsymbol{S_I}$ یک فاکتور میانجی محسوب می‌شود. از طرف دیگر، حل معادلهٔ (۱) معادل با یافتن پاسخ برای مجموعه معادلات خطی $wS_I = kt^*$ است. بنابراین، با توجه به حجم بسیار سنگین محاسبات ناشی از محاسبهٔ معکوس $\boldsymbol{S_I}$ در رابطهٔ (۶)، می‌توان رابطهٔ $u \equiv S_I^{-1}t^*$ را به $S_Iu \equiv t^*$ تبدیل نمود؛ سپس از روش‌های حل معادلات خطی برای تعیین u استفاده کرد. ولی معمولاً

1- Side looking

2- Steering

3- Kronecker product

ماتریس کواریانس از پیش مشخص نیست و با توجه به داده‌های دریافتی تعیین می‌شود. یکی از روش‌های بسیار متداول برای محاسبهٔ $\boldsymbol{S_I}$، الگوریتم شکل‌دهندهٔ پرتو SMI[1] است [۱۹]. در این الگوریتم، ماتریس کواریانس با انتخاب تعدادی از نمونه‌های فاصله که سلول‌های آموزشی (L_s) نامیده می‌شوند، به‌دست می‌آید. L_s نباید شامل سلول‌هایی باشد که اکوهای هدف را در بر می‌گیرد. برای جلوگیری از آلودگی ماتریس کواریانس با اکوهای هدف، تعدادی سلول محافظ قبل و بعد از سلول هدف درنظر گرفته می‌شوند. به این ترتیب تخمینی از $\boldsymbol{S_I}$ به‌دست می‌آید که در رابطهٔ (۷) نشان داده شده است.

$$\hat{S}_I = E\left\{YY^H\right\} = \frac{1}{L_s}\sum_{i=1}^{L_s} y(i)y^H(i) \qquad (۷)$$

در رابطهٔ فوق، ابعاد ماتریس Y، $NM\times L_s$ است که $NM<L_s<L$ می‌باشد و از L_s بردار $y(i)$ تشکیل شده است. $y(i)$ها تصاویر لحظه‌ای[2] از مکعب داده هستند. برای ایجاد $y(i)$، ابتدا یک برش دوبُعدی از مکعب داده در نمونهٔ فاصلهٔ موردنظر (l_0) که یک تصویر لحظه‌ای دو بعدی در ابعاد N×M است، صورت می‌گیرد. سپس، تصویر لحظه‌ای دو بعدی مطابق رابطهٔ (۸) به یک بردار ستونی y_{l_0} در ابعاد NM×1 تبدیل می‌شود. ماتریس کواریانس برای هریک از L_s نمونهٔ فاصله به‌طور مجزا محاسبه شده و سپس با متوسط‌گیری، ماتریس $\hat{\boldsymbol{S}}_{\boldsymbol{I}}$ به‌دست می‌آید. پس از تعیین $\hat{\boldsymbol{S}}_{\boldsymbol{I}}$ از رابطهٔ (۶) برای محاسبهٔ w استفاده می‌شود.

$$y_{l_0} = \begin{bmatrix} y(l_0,0,0) \\ y(l_0,0,1) \\ \vdots \\ y(l_0,0,N-1) \\ y(l_0,1,0) \\ \vdots \\ y(l_0,1,N-1) \\ \vdots \\ y(l_0,M-1,0) \\ \vdots \\ y(l_0,M-1,N-1) \end{bmatrix} \qquad (۸)$$

برای پرهیز از محاسبهٔ معکوس $\hat{\boldsymbol{S}}_{\boldsymbol{I}}$ می‌توان از روش‌های حل معادلات خطی بهره گرفت. بخش وسیعی از کاربردهای پردازش سیگنال و مخابرات به مسائل حداقل مربعات (LS)[3] مرتبط بوده و به حل آنها نیازمندند. مسأله LS با حل معادلات خطی **Ax=b** شناخته می‌شود. روش‌های متفاوتی برای حل این معادلات وجود دارد که مهمترین آنها شیوهٔ حل مستقیم است [۲۰, ۲۱]. روش‌های مستقیم به طرق مختلفی مانند حذف گوسی، فاکتورگیری LU[4]، تجزیهٔ چالسکی[5] و QRD متداول هستند که همهٔ آنها بعد از تعدادی عملیات مشخص، راه حل دقیقی را برای معادلات ارائه می‌دهند. ایدهٔ اصلی روش‌های ذکر شده، کاهش تعداد معادلات کلی به یک فرم بالامثلثی و یا پایین‌مثلثی است. سپس با استفاده از روش‌های جایگزینی پسرو و پیشرو، نتیجهٔ نهایی با دقت مطلوبی به‌دست می‌آید. در این میان، روش QRD کاربرد گسترده‌تری دارد. QRD به دو شیوه می‌تواند برای یافتن بردار وزن در الگوریتم SMI استفاده شود که عبارت است از: حوزهٔ توان و حوزهٔ ولتاژ [۱۹]. روش حوزهٔ توان با تشکیل ماتریس کواریانس $\hat{\boldsymbol{S}}_{\boldsymbol{I}}$ آغاز می‌شود. در این روش QRD مستقیماً به ماتریس $\hat{\boldsymbol{S}}_{\boldsymbol{I}}$ اعمال می‌شود. از آنجا که رابطهٔ (۷) شامل ضرب مقادیر دامنهٔ ولتاژ است، مقادیر ماتریس کواریانس در دامنهٔ توان می‌باشند. از معایب روش حوزهٔ توان، اثر مجذورنمودن مقادیر ناشی از ضرب خارجی است. در بسیاری از سامانه‌های واقعی، نرخ واقعی محاسبات لازم برای پردازش وفقی بسیار بالا است و سخت‌افزارهای مورد استفاده معمولاً ممیز ثابت می‌باشند. مجذورکردن یک سیگنال، دامنهٔ دینامیکی آن را برحسب دسیبل دو برابر نموده و بنابراین تعداد بیت‌های لازم برای نمایش بدون سرریز آن را نیز دو برابر می‌نماید. درنتیجه با افزایش نرخ داده در پردازشگر، ابعاد آن و همچنین، پیچیدگی طراحی افزایش می‌یابد. لذا، مطلوب است که سیگنال بی‌جهت مجذور نشود. در حوزهٔ ولتاژ، QRD مستقیماً به بردارهای دادهٔ مکان- زمان اعمال و از محاسبهٔ صریح ماتریس کواریانس $\hat{\boldsymbol{S}}_{\boldsymbol{I}}$ اجتناب می‌گردد. در این روش ابتدا یک ماتریس Y در ابعاد $NM\times L_s$ از به‌هم چسباندن بردارهای دادهٔ مکان-زمان مطابق رابطهٔ (۹) تشکیل می‌شود.

$$Y = [y_0, y_1, \ldots, y_{L_s-1}] \qquad (۹)$$

از ماتریس $\boldsymbol{Y}$ برای محاسبهٔ ماتریس $\hat{\boldsymbol{S}}_{\boldsymbol{I}}$ نیز می‌توان بهره گرفت که در رابطهٔ (۱۰) آمده است. ولی از آنجا که مجذور عناصر را شامل می‌شود، محاسبهٔ آن مطلوب نیست.

$$\hat{S}_I = Y^* Y' \qquad (۱۰)$$

سپس، QRD به $\boldsymbol{Y}'$ اعمال می‌شود. ابعاد ماتریس‌های $\boldsymbol{Q}$ و $\boldsymbol{R}$ که از تجزیهٔ $\boldsymbol{Y}'$ به‌دست می‌آیند، به‌ترتیب $L_s\times L_s$ و $L_s\times NM$ است. $\boldsymbol{Q}$ ماتریس متعامد و $\boldsymbol{R}$ با رابطهٔ (۱۱) بیان می‌شود. $\hat{\boldsymbol{R}}$، یک ماتریس بالامثلثی در ابعاد NM×NM و $\boldsymbol{0}$، ماتریسی با درایه‌های صفر در ابعاد $(L_s\text{-}NM)\times NM$ است.

$$R = \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix} \qquad (۱۱)$$

با قراردادن ماتریس‌های $\boldsymbol{Q}$ و $\boldsymbol{R}$ در رابطهٔ (۱۰)، رابطهٔ (۱۲)

1- Sample Matrix inversion (SMI)
2- Snapshots
3- Least squares (LS)
4- Lower upper (LU) factorization
5- Cholesky decomposition

به‌دست می‌آید.

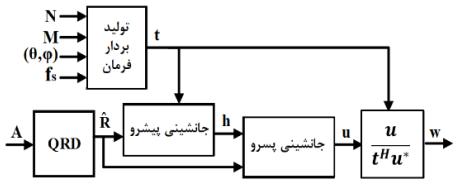
$$\hat{S_1} = Y^{*}Y' = R^H Q^H QR = R^H R = \hat{R}^H \hat{R} \qquad (۱۲)$$

اگر رابطهٔ فوق را در $\hat{S_1}u = t^*$ جایگذاری نماییم، رابطهٔ (۱۳) حاصل می‌شود که با فرض $h \equiv \hat{R}u$ به رابطهٔ (۱۴) می‌رسیم.

$$\hat{R}^H \hat{R}u = t^* \qquad (۱۳)$$

$$\hat{R}^H h = t^* \qquad (۱۴)$$

در رابطهٔ (۱۴)، $\hat{R}^H$ یک ماتریس پایین مثلثی است. بنابراین، بردار h به طول NM با استفاده از جانشینی پیشرو به‌دست می‌آید. سپس از رابطهٔ $h \equiv \hat{R}u$ بردار u با استفاده از جانشینی پسرو محاسبه می‌شود. نهایتاً بردار وزن w از رابطهٔ (۶) به‌دست خواهد آمد. در شکل (۳)، بلوک دیاگرام روش فوق‌الذکر برای پیاده‌سازی STAP به‌منظور تعیین بردار وزن نشان داده شده است. $\boldsymbol{A}$ همان ماتریس $\boldsymbol{Y}'$ است.



شکل (۳): بلوک دیاگرام مدل STAP برای محاسبهٔ بردار وزن

۳- طراحی مدل STAP

اولین مرحلهٔ بسیار مهم پس از محاسبهٔ ماتریس دادهٔ $\boldsymbol{Y}$، تجزیهٔ QR است. تحقیقات و پژوهش‌ها نشان می‌دهند؛ از میان روش‌های ذکر شده برای QRD در حوزهٔ STAP، GR و HT از کارایی کمتری برخوردارند [۱۴]. MGS تغییریافتهٔ CGS است که از لحاظ هزینهٔ پیاده‌سازی تقریباً مشابه آن می‌باشد، ولی پایداری عددی بهتری را نشان می‌دهد [۲۱]. دستورات استاندارد الگوریتم MGS همراه با تعداد عملیات موردنیاز در جدول (۱) آمده است. درایه‌های ماتریس $\hat{\boldsymbol{R}}$، سطر به سطر محاسبه می‌شوند و در هر سطر ابتدا عنصر قطری و سپس سایر عناصر محاسبه خواهند شد. محاسبهٔ عناصر ماتریس $\boldsymbol{Q}$ نیز به‌صورت ستونی انجام می‌شود. به‌عبارت دیگر، پس از تشکیل سطر اول ماتریس $\hat{\boldsymbol{R}}$، ستون اول ماتریس $\boldsymbol{Q}$ تشکیل می‌شود. سپس ستون‌های باقیماندهٔ ماتریس $\boldsymbol{A}$ که در مرحلهٔ بعد مورد نیازند، محاسبه خواهند شد. بنابراین، در هر چرخه ابتدا یک سطر از $\hat{\boldsymbol{R}}$، سپس یک ستون از $\boldsymbol{Q}$ و نهایتاً ستون‌های باقیمانده‌ای از $\boldsymbol{A}$ که در چرخه‌های بعد موردنیاز هستند، محاسبه می‌شوند. این روند به تعداد ستون‌های $\boldsymbol{A}$ ادامه می‌یابد تا در نهایت ماتریس‌های خروجی $\hat{\boldsymbol{R}}$ و $\boldsymbol{Q}$ تشکیل شوند. همان‌طور که در دستورات استاندارد MGS آمده است، برای محاسبهٔ عناصر قطری $\hat{\boldsymbol{R}}$ از تابع نُرم استفاده می‌شود. از آنجا که مقصود اصلی در این مقاله، سازگاری طراحی از لحاظ پیاده‌سازی سخت‌افزاری می‌باشد؛ سطر دوم را تغییر می‌دهیم تا با اجزای سخت افزاری موجود، آسان‌تر پیاده‌سازی شود. همچنین به‌محاسبهٔ $\hat{\boldsymbol{R}}(k,k)$ قبل از $\hat{\boldsymbol{R}}(k,j)$ ،و محاسبهٔ $\hat{\boldsymbol{R}}(k,j)$ و $\boldsymbol{Q}(1{:}\ L_s,k)$ پیش از $\boldsymbol{A}(1{:}\ L_s, j)$ نیاز است. این روند تعدادی زیادی سیکل غیرفعال ایجاد می‌نماید که مانع بهره‌برداری مناسب از منابع سخت‌افزاری می‌شود و پیاده‌سازی ممیز شناور آن را نیز دشوار می‌سازد. بنابراین، تغییراتی مطابق جدول (۲) در دستورات استاندارد MGS ایجاد می‌نماییم.

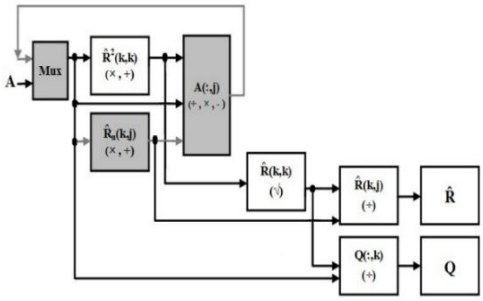
دستورات جدول (۲) از دو حلقهٔ اصلی تشکیل شده است. در حلقهٔ اول، ملزومات محاسباتیِ اولیه برای عناصر قطری ماتریس $\hat{\boldsymbol{R}}$ و ستون‌های باقیمانده از ماتریس $\boldsymbol{A}$ که در چرخهٔ بعد موردنیاز هستند، انجام می‌شوند. بنابراین مطابق بلوک دیاگرام شکل (۴) که بر اساس جدول (۲) رسم شده است، عملگر جذر از مسیر بحرانی محاسبهٔ $\boldsymbol{A}$ خارج شده است. در حلقهٔ دوم نیز، محاسبهٔ عناصر قطری و سایر عناصر غیرصفر برای هر سطر از ماتریس $\hat{\boldsymbol{R}}$ به‌پایان رسیده و ستون $\boldsymbol{Q}(1{:}\ L_s,k)$ محاسبه می‌شود. ترتیب مراحل مذکور از مزایایی نیز برخوردار است. از جمله این‌که؛ محاسبهٔ ستون‌های باقیماندهٔ ماتریس $\boldsymbol{A}$ به عملیات کمتری نیاز دارد. با توجه به این‌که ملزومات محاسباتی برای عناصر هر سطر از ماتریس $\hat{\boldsymbol{R}}$ در حلقهٔ اول لحاظ شده است؛ حلقهٔ دوم می‌تواند تا در دسترس قرار گرفتن داده‌های موردنیاز، زمان‌بندی شود. همچنین در شکل (۳) مشاهده می‌شود که نیازی به محاسبهٔ $\boldsymbol{Q}$ برای تعیین بردار وزن وجود ندارد. بنابراین، همان‌طور که در شکل (۴) نشان داده شده، محاسبهٔ آن به‌طور کامل از مسیر بحرانی خارج شده است. به این ترتیب، می‌توانیم به طراحی QRD ساده‌تری دست یابیم. از جدول‌های (۱ و ۲) مشهود است که مزایای ذکر شده منجر به افزایش تعداد عملگرها نشده است.

جدول (۱): دستورات استاندارد الگوریتم QRD-MGS

شماره	دستور	جذر	تقسیم	ضرب
۱	**for** k=1:NM			
۲	$\hat{\mathbf{R}}(k,k) = \text{norm}(\mathbf{A}(1{:}\ L_s, k))$;	k		$k \times L_s$
۳	**for** j = k+1:NM			
۴	$\hat{\mathbf{R}}(k,j) = \text{dot}(\mathbf{A}(1{:}\ L_s, k), \mathbf{A}(1{:}\ L_s, j))/\hat{\mathbf{R}}(k,k)$;		$k^2/2$	$L_s \times k^2/2$
۵	**end**			
۶	$\mathbf{Q}(1{:}\ L_s, k) = \mathbf{A}(1{:}\ L_s, k)/\hat{\mathbf{R}}(k,k)$;		k	
۷	**for** j = k+1:NM			
۸	$\mathbf{A}(1{:}\ L_s, j) = \mathbf{A}(1{:}\ L_s, j) - \hat{\mathbf{R}}(k,j) \times \mathbf{Q}(1{:}\ L_s,k)$;			$L_s \times k^2/2$
۹	**end**			
۱۰	**end**			
مجموع		k	$k+k^2/2$	$(k+k^2)L_s$

جدول (۲): دستورات اصلاح شدهٔ الگوریتم QRD-MGS

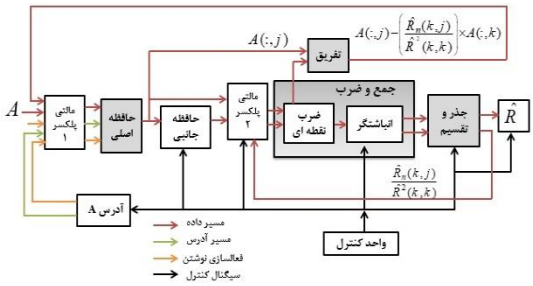
شماره	دستور	جذر	تقسیم	ضرب
۱	**for** k=1:NM			
۲	$\hat{\mathbf{R}}^2$(k, k) = dot (**A**(1: L_s, k), **A**(1: L_s, k));			k×L_s
۳	$\hat{\mathbf{R}}$(k, k) = sqrt($\hat{\mathbf{R}}^2$(k, k));	k		
۴	**for** j = k+1:NM			
۵	$\hat{\mathbf{R}}_n$(k, j) = dot (**A**(1: L_s, k), **A**(1: L_s, j));			L_s×k^2/2
۶	**A**(1: L_s, j) = **A**(1: L_s, j) – ($\hat{\mathbf{R}}_n$(k,j)/ $\hat{\mathbf{R}}^2$(k, k)) × **A**(1: L_s,k);	k^2/2		L_s×k^2/2
۷	$\hat{\mathbf{R}}$(k, j) = $\hat{\mathbf{R}}_n$(k,j)/ $\hat{\mathbf{R}}$(k,k);	k^2/2		
۸	**end**			
۹	**Q**(1: L_s, k) = **A**(1: L_s, k) / $\hat{\mathbf{R}}$(k, k);	k		
۱۰	**end**			
مجموع		k	$k+k^2$	$(k+k^2)L_s$



شکل (۴): بلوک دیاگرام دستورات اصلاح شدهٔ الگوریتم QRD-MGS

پیاده‌سازی سامانه‌ها به‌صورت دیجیتال با روش‌های مختلفی قابل اجرا است. از روش‌های جدیدی که با هدف پیاده‌سازی روی FPGA استفاده می‌شود، ابزار DSP Builder است [۲۲]. این بستهٔ نرم‌افزاری به برنامهٔ متلب الحاق گردیده و کتابخانه‌هایی را به قسمت سیمولینک اضافه می‌نماید. با برخورداری از این ابزار می‌توان سامانه‌های بزرگ و پیچیده را طوری طراحی نمود که پس از تست نتایج اولیه در متلب با لینک‌شدن به نرم‌افزار کوارتس[1]، مستقیماً پروژهٔ HDL ایجاد شده را مورد ارزیابی قرار داد. همچنین با لینک شدن به نرم‌افزار مُدل‌سیم[2]، پروژهٔ HDL مورد شبیه‌سازی سخت‌افزاری قرار می‌گیرد. این روند، فرصت‌های جدیدی را برای طراحی در اختیار طراحان قرار می‌دهد و آن را از پیاده‌سازی‌های مبتنی بر قوائد ملالت‌آور رها می‌سازد. این‌گونه طراحی‌ها، به سازگاری یک طراحی بحرانی ناشی از آزمایشی بودن بُرد نیز کمک می‌نماید. در این مقاله، از ابزارهای مذکور برای طراحی و پیاده‌سازی استفاده شده است.

طراحی بلوک QRD در شکل (۵) نشان داده شده است. طراحی QRD از پنج بخش اصلی مشتمل بر واحد کنترل، حافظهٔ اصلی، حافظهٔ جانبی، واحد جمع و ضرب، و واحد جذر و تقسیم تشکیل شده است. واحد کنترل متشکل از یک ماشین حالت است که از چهار حالت تشکیل شده است. حالت‌ها عبارتند از: محاسبهٔ اندازهٔ مقادیر، محاسبهٔ ضرب‌های نقطه‌ای، موجود بودن اندازه‌ها، محاسبهٔ مقادیر جدید ماتریس ***A***، مالتی پلکسر ۱، منبع داده‌ای را که به طبقهٔ بعد هدایت می‌شود را انتخاب می‌کند. داده‌ها می‌توانند از خارج از محیط طراحی به‌عنوان ورودی اولیه و یا از داخل آن باشند. در این بلوک بردارهایی به‌تعداد ستون‌های ماتریس ورودی، برای ذخیرهٔ داده‌ها ایجاد شده است. بنابراین، تمامی عملیات کاملاً برداری صورت می‌گیرد. در این طراحی تعداد ضرب‌کنندها با پارامتری به‌نام اندازهٔ بردار مشخص شده‌اند که بیانگر تعداد ضرب‌کننده‌های مختلط و ممیز شناوری است که می‌توانند به‌طور موازی عمل نمایند. حافظهٔ اصلی از یک حافظهٔ دومنظوره و یک مولد آدرس تشکیل شده است. در ابتدا حافظه دومنظوره با ماتریس ورودی A مقداردهی می‌شود. سپس در طی اجرای الگوریتم با داده‌های میانی که در گذرهای آتی استفاده می‌شوند، مقداردهی خواهد شد.



شکل (۵): طراحی الگوریتم اصلاح شدهٔ QRD-MGS

یک پیاده‌سازی کارآمد به استفادهٔ مؤثر از حافظه بستگی دارد. در QRD نیاز است به‌طور مستمر از یک حافظهٔ بزرگ خوانده شده و در آن نوشته شود. به‌طوریکه بتواند کل ماتریس را ذخیره نماید. بنابراین، از آنجا که مقادیر جدید ماتریس ***A*** و مقادیر $\hat{\boldsymbol{R}}_{\boldsymbol{n}}$ به بردار ***A***(1:m,k) و سایر بردارهای ماتریس ***A*** نیاز دارند، مطلوب است که دو پورت مجزای خواندن برای محاسبهٔ آنها درنظرگرفته شود. به جای استفاده از پورت مجزای خواندن برای حافظهٔ بزرگ ***A***، یک حافظهٔ جانبی که تنها اولین بردار را ذخیره می‌کند، به‌کار گرفته شده است. حافظهٔ جانبی از یک حافظهٔ دومنظوره و شمارنده‌هایی برای تأمین پورت آدرس تشکیل شده است. این حافظه، هنگام محاسبهٔ $\hat{\boldsymbol{R}}$(k,k) فراخوانده می‌شود. سپس ***A***(1:m,k) برای استفاده مجدد در سایر بخش‌ها، ذخیره خواهد شد. مالتی پلکسر ۲، مسیر داده را به ضرب‌کننده هدایت می‌کند. واحد ضرب و جمع از ضرب نقطه‌ای و انباشتگر[3] تشکیل شده است. محاسبهٔ $\hat{\boldsymbol{R}}^2$(k,k) و به‌هنگام نمودن‌های مکرر ستون‌های باقی‌ماندهٔ ماتریس ***A*** در این بخش انجام می‌شود. از آنجا که این واحد روی بردارها عمل می‌کند، پیاده‌سازی تمام موازی آن نیز امکان‌پذیر است. همان‌طور که مطرح شد، در این

1- Quartus
2- Modelsim
3- Accumulator

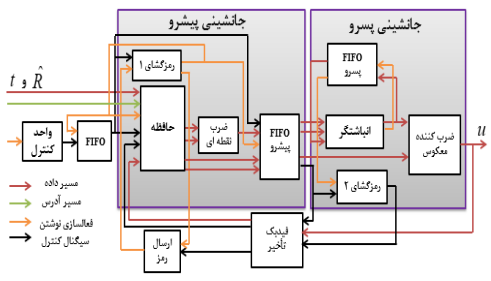
طراحی با در نظر گرفتن پارامتری به‌نام اندازهٔ بردار، پردازش در چندین مرحله انجام می‌شود. لذا، بردارها در بلوک انباشتگر جمع‌آوری می‌شوند. با تغییر اندازهٔ بردار و یا تعداد ستون‌های ماتریس ورودی، تعداد مراحل پردازشی نیز تغییر می‌کند. اگر بخواهیم طراحی را به‌طور تمام برداری با تنها یک بردار پردازش نماییم، می‌توان اندازهٔ بردار را برابر با تعداد ستون‌های ماتریس ورودی درنظرگرفت. در این صورت، پردازش کاملاً برداری و در یک مرحله انجام می‌شود و بلوک انباشتگر نیز حذف خواهد شد. بنابراین، QRD به‌طور کاملاً انعطاف‌پذیر طراحی شده و برای هر ماتریس با ابعاد دلخواه و اندازه های برداری مختلف قابل استفاده است.

بلوک t نیز مطابق روابط (۲) تا (۴) طراحی شده است. با به‌دست آوردن $\hat{\boldsymbol{R}}$ و t، در مراحل بعدی به جانشینی‌های پیشرو و پسرو نیاز است. مطابق روابط زیر، ابتدا از جانشینی پیشرو بردار h به‌دست می‌آید؛ سپس بردار u از جانشینی پسرو حاصل می‌شود. در تمامی روابط k=1,....,NM است. روابط (۱۵) و (۱۶) به ترتیب جانشینی های پیشرو و پسرو را نشان می‌دهند.

$$h_k = \frac{t^*_k - \sum_{j=1}^{k-1} \hat{R}^H_{k,j} h_j}{\hat{R}^H_{k,k}} \qquad (۱۵)$$

$$u_k = \frac{h_k - \sum_{j=1}^{k-1} \hat{R}_{k,j} u_j}{\hat{R}_{k,k}} \qquad (۱۶)$$

در شکل (۶) طراحی‌های جانشینی‌های پسرو و پیشرو نشان داده شده که از سه بخش اصلی واحد کنترل، واحد حافظه و مسیر داده تشکیل شده است. واحد حافظه، ورودی‌های $\hat{\boldsymbol{R}}$ و t را دریافت می‌کند و از چندین حافظهٔ دومنظوره تشکیل شده است. واحد کنترل شامل پنج حالت است. حالت اول به اولین درایه از هر سطر و حالت دوم به آخرین درایه از همان سطر اشاره دارد. حالت سوم به تکمیل سطر قبلی پیش از شروع سطر جدید می‌پردازد. حالت چهارم به هر محاسبهٔ جدید در بردار خروجی و حالت پنجم به مزدوج مختلط اختصاص دارد. از آنجا که داده‌های ورودی مختلط هستند، برای ترانهاده نمودن $\hat{\boldsymbol{R}}$ به حالت پنجم واحد کنترل نیاز است.



شکل (۶): طراحی جانشینی‌های پیشرو و پسرو

۴- نتایج شبیه‌سازی

همانطور که در قسمت دوم ذکر شد، با طراحی یک رادار پهلونگر پالسی تک‌پایه و استفاده از پردازش STAP برای تعیین موقعیت هدف و تضعیف تداخل‌ها، مکعب داده‌ای مطابق شکل (۲) به‌دست آمد. مشخصات کامل داده‌ها، در جدول (۳) آمده است. با خارج نمودن سلول هدف و سلول‌های محافظ اطراف آن، ۱۸۹ سلول بُرد باقی می‌ماند که از این تعداد، ۱۷۰ سلول برای آموزش در نظر گرفته شد. بنابراین، ماتریس A در ابعاد ۶۰×۱۷۰ به‌دست آمد.

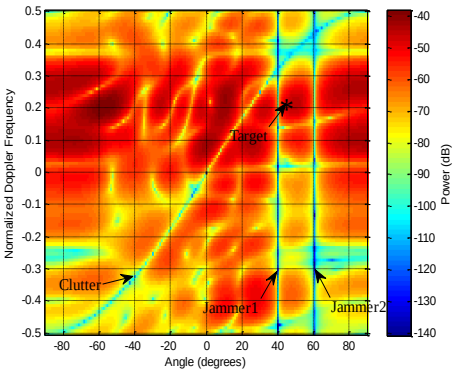
جدول(۳): مشخصات مکعب داده

تعداد آرایه‌های آنتنی	۶
تعداد پالس‌ها	۱۰
تعداد سلول‌های بُرد	۲۰۰
تعداد سلول‌های آموزشی	۱۷۰
تعداد سلول‌های محافظ	۱۰
مختصات مکانی هدف	(۴۵°, -۳۵°)
بُرد هدف	۱۷۳۲ m
فرکانس حامل	۱۰ GHz
طول موج	۳ cm
فرکانس تکرار پالس	۳۰ KHz
فرکانس داپلر	۶۳۴۳ Hz
سرعت هوابُرد	۲۲۵ m/s
ارتفاع هوابُرد	۱۰۰۰ m
نسبت کلاتر به نویز	۶۵ dB
نسبت سیگنال به نویز در گیرنده	۵ dB
زاویهٔ جمر ۱	۴۰°
زاویهٔ جمر ۲	۶۰°
توان جمرها	۱۰۰ W
حداکثر بُرد	۵۰۰۰ m

از آنجا که، یکی از مهمترین وظایف بخش پردازش یک گیرندهٔ راداری، تضعیف سیگنال‌های مزاحمی همچون کلاتر و جمر است [۲۳, ۲۴]؛ از الگوی وفقی زاویه- داپلر استفاده می‌نماییم. شکل (۷) الگوی وفقی زاویه- داپلر را نشان می‌دهد که بر اساس وزن‌های به‌دست آمده، ترسیم شده است. مطابق جدول (۳)، دو سیگنال جمینگ با توان یکسان در زوایای مختلف وجود دارد. در شکل (۷)، توانی حدود ۱۳۰ dB- در زوایای ۴۰° و ۶۰° مشهود است که بیانگر تضعیف جمرها است. خط انحنادار که در طول محور داپلر گسترده شده، نشان‌دهندهٔ کلاتر است که با قدرتی حدود ۸۰ dB- تا ۱۰۰ dB- تضعیف شده است. طبق جدول (۳)، زاویهٔ هدف ۴۵° آزیموت[1] و فرکانس داپلر نرمالیزه

1- Azimuth

حـــدود ۰/۲۱ اســـت. بنـــابراین، مختصـــات (۰/۲۱ و 45°) در شکل (۷)، بیانگر موقعیت هدف است که توانی حـدود ۴۰dB- در آن مشهود است.



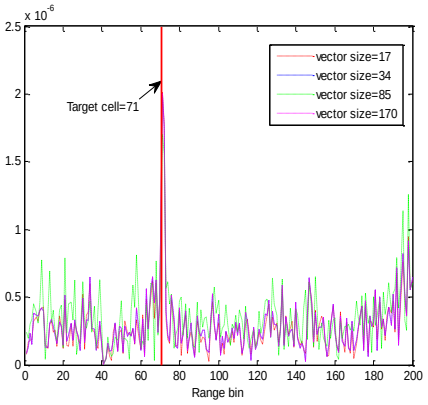
شکل (۷): الگوی وفقی زاویه-داپلر

همان‌طور که پیشتر مطرح شد، در این مقالـه بـرای کـاهش حجم محاسبات STAP روش چندبـرداری پیشـنهاد شـده اسـت. اندازه‌های برداری به‌گونه‌ای انتخاب شدند که مضربی از سلول‌های آموزشی باشند. بنابراین، چهار اندازهٔ برداری ۱۷، ۳۴، ۸۵ و ۱۷۰ در نظر گرفته شد که به‌ترتیب پـردازش در ۱۰، ۵، ۲ و ۱ مرحلـه انجام می‌شود. از اینجا به بعد، تـأثیر انـدازه‌هـای بـرداری نیـز در عملکرد مدل مورد بررسی قرار می‌گیرد.

پردازش وفقی زمانی- مکانی عمـل فیلترینـگ بـرداری را بـر روی ترکیب داده‌های مکانی و زمانی برای هر نمونه فاصلهٔ خـاص به‌طور مجزا انجام داده و از وزن‌های فیلتر وفقی برای آشکارسازی هدف استفاده می‌کند [۲۶-۲۵]. در اینجا از تست استاتیکی برای تعیین حضور یا غیاب هدف استفاده می‌نماییم که در رابطهٔ (۱۷) به‌صورت یک مسأله دو فرضیه‌ای آمده اسـت. فرضـیهٔ H_0 بیـانگر غیاب هدف و فرضیهٔ H_1 نشان‌دهندهٔ حضور هدف می‌باشد.

$$\begin{cases} H_0 : y = y_j + y_c + y_n \\ H_1 : y = \alpha y_t + y_j + y_c + y_n \end{cases} \qquad (۱۷)$$

در شکل (۸) سیگنال‌های جمع‌آوری شده توسط آرایهٔ آنتنی در هریـک از پـردازش‌هـای ۱۰، ۵، ۲ و ۱ بـرداری بعـد از انجـام پردازش STAP و اعمال فیلترینـگ آن، نشـان داده شـده اسـت. مطابق جدول (۳)، بُرد هدف ۱۷۳۲ m است که متناظر با سـلول شمارهٔ ۷۱ می‌باشد. در شکل (۸)، موقعیت سـلول هـدف بـا یـک خط عمودی قرمز نشان داده شده است. همان‌طـور کـه ملاحظـه می‌شود، تمامی خروجی‌ها حول خط مذکور دارای یـک پیـک بـا قدرت بالا هستند. بنـابراین، مـدل طراحـی شـده در هـر یـک از حالت‌ها به‌درستی موقعیت هدف را تشخیص داده است.



شکل (۸): تست استاتیکی برای تعیین موقعیت هدف

۵- نتایج پیاده‌سازی

در این قسمت به نتایج پیاده‌سـازی سـاختار طراحـی شـده روی تراشـهٔ Arria 10 FPGA پرداختـه مـی‌شـود. مصـرف منـابع سخت‌افزاری مدل STAP در فاکتورهای مختلف برای هـر یـک از اندازه‌های برداری در جدول (۴) نشـان داده شـده اسـت. مـاژول منطقی وفقی (ALM)[1]، پایه‌ترین بلوک سازندهٔ این نوع FPGAها است که از دو ALUT[2] تشکیل شده است. استفاده از یک FPGA با بلوک‌های DSP ممیز شناور مانند Arria 10 برای پیـاده‌سـازی مدل STAP، نشان داد که محدودیت در پیاده‌سازی یـک‌بـرداری ابعاد ماتریسی بزرگ، تعداد بلوک‌های DSP است. به این ترتیـب، تعداد ALMها حفظ می‌شـوند. مقـادیر نشـان مـی‌دهنـد کـه بـا افزایش اندازهٔ بردار، مصرف منابع نیز افزایش می‌یابـد. در جـدول (۴) برای روش یک برداری (۱۷۰)، حدود ۳۶٪ از ALMها و ۷۳٪ از DSPهای موجود، مصرف شده‌اند. همچنین در این حالت، ۹۶٪ از بلوک‌های DSP مصرف شده، ممیـز شـناور هسـتند. از سـویی دیگر، حدود ۷۰٪ صرفه‌جویی در مصرف منابع سخت‌افزاری برای کوچکترین اندازهٔ برداری (۱۷) نیز به‌دست آمده اسـت. بنـابراین، روش چندبرداری منجر به یک پیاده‌سازی کارآمد شده است.

از آنجا که داده‌های STAP مختلط هستند، پیاده‌سازی ارائـه شده در این مقاله نیز ممیز شناور تک دقتی[3] است. ولی بـه‌طـور معمـول، حجـم محاسـبات بـه‌صـورت ممیـز شـناور حقیقـی (RFLOP)[4] مطرح می‌شود. بر این اساس، RFLOPهای موردنیاز برای هر بخش از مـدل STAP بـر حسـب ابعـاد مکعـب داده در جدول (۵) آورده شده‌اند [۱۶]. مطلوب‌تر است که برای ارزیـابی پیاده‌سازی، RFLOP در هر ثانیه (RFLOPs)[5] محاسبه شود کـه با تقسیم RFLOPها بر زمان اجرای محاسبات به‌دست می‌آید. در

1- Adaptive Logic Module (Alm)
2- Adaptive Look-Up Tables (Alut)
3 -Single Precision
4- Real Floating Point
5- Rflop Per Second (Rflops)

جدول (۶) پارامترهای مختلف برای بررسی کارایی مدل، ارائه شده‌اند. حجم محاسبات ممیز شناور حقیقی در واحد زمان با GFLOPs[1] نشان داده شده است. همان‌طور که مشخص است، حجم محاسبات در اندازهٔ برداری ۱۷۰ تقریباً پنج برابر اندازهٔ برداری ۱۷ است. بنابراین، با کاهش اندازهٔ بردار، حجم محاسبات و همچنین توان مصرفی کاهش و F_{max} افزایش یافته است. هر چند که با بزرگ‌شدن اندازهٔ بردار، زمان اجرا نیز کاهش می‌یابد. ولی با افزایش مصرف منابع سخت‌افزاری، توان مصرفی و حجم محاسبات و همچنین کاهش F_{max} همراه است. این امر مانع بهره‌برداری مناسب از منابعِ تراشه‌های با امکانات محدود می‌گردد. بنابراین، نتایج نشان می‌دهد که روش پردازش چندبرداری به‌ویژه برای سامانه‌هایی مانند STAP که از حجم سنگین محاسبات برخوردارند و از ابعاد ماتریسی بزرگ استفاده می‌کنند، بسیار مؤثر و کارآمد است.

جدول (۴): مصرف منابع سخت‌افزاری مدل STAP برای چندین اندازهٔ برداری

اندازهٔ بردار	ALMها	ثبت‌ها	بیت‌های حافظهٔ MLAB	بلوک‌های DSP	بلوک‌های DSP ممیز شناور
۱۷	(٪۸) ۳۳۵۸۰	(٪۳) ۵۲۳۹۵	(٪۰/۵) ۷۱۰۱۷	(٪۱۲) ۱۸۸	(٪۷۴) ۱۳۹
۳۴	(٪۱۱) ۴۷۰۹۰	(٪۵) ۷۷۹۴۸	(٪۱) ۱۱۱۳۵۴	(٪۱۹) ۲۹۰	(٪۸۳) ۲۴۱
۸۵	(٪۲۱) ۹۰۷۰۰	(٪۹) ۱۵۵۷۴۳	(٪۲) ۲۲۷۶۹۱	(٪۳۹) ۵۹۸	(٪۹۲) ۵۴۹
۱۷۰	(٪۳۶) ۱۵۴۱۶۸	(٪۱۸) ۳۰۲۳۵۰	(٪۳) ۴۳۸۰۶۳	(٪۷۳) ۱۱۰۳	(٪۹۶) ۱۰۵۴

جدول(۵): RFLOPهای موردنیاز برای محاسبهٔ بردار وزن

تجزیهٔ QR	$8L_s(NM)^2 - 2/67(NM)^3$
بردار فرمان	$6(NM)$
جانشینی پیشرو	$4(NM)^2$
جانشینی پسرو	$4(NM)^2$
مجموع	$\{(NM)^2[8(L_s+1)-2/67(NM)]\}+6(NM)$

جدول(۶): پارامترهای مختلف برای بررسی کارایی مدل STAP

اندازهٔ بردار	زمان اجرا (µs)	توان (mW)	F_{max} (MHz)	GFLOPs	GFLOPs/ Watt
۱۷	۲۶۲	۶۳۶۹	۲۷۸	۲۸/۱	۴/۴۱
۳۴	۱۸۹	۷۳۳۰	۲۷۴	۵۱/۷	۷/۰۵
۸۵	۱۵۷	۱۰۴۱۶	۲۶۱	۸۶/۶	۸/۳۱
۱۷۰	۱۴۴	۱۵۴۴۱	۲۱۰	۱۴۰	۹/۰۶

1- Giga FLOPs (GFLOPs)

۶- نتیجه‌گیری

در این مقاله، به طراحی و پیاده‌سازی پردازش وفقی زمانی-مکانی در رادارهای هوایی پرداخته شد و روشی مؤثر برای پیاده‌سازی کارآمد آن پیشنهاد گردید. یک هوابُرد با رادار پهلونگر پالسی تک‌پایه که از پردازش وفقی زمانی- مکانی برای آشکارسازی هدف و تضعیف تداخل‌هایی همچون کلاتر و جمینگ استفاده می‌کند، در محیط متلب شبیه‌سازی شد و داده‌های موردنیاز برای تست و بررسی سامانه به‌دست آمدند. مدل ارائه‌شده برای محاسبهٔ بردار وزن STAP با چندین اندازهٔ برداری مورد شبیه‌سازی و پیاده‌سازی قرار گرفت. الگوی وفقی زاویه- داپلر و تست استاتیکی برای آشکارسازی هدف نشان داد که بردار وزن در هر یک از اندازه‌های برداری، به‌درستی محاسبه شده است. نتایج پیاده‌سازی روی تراشهٔ Arria 10 حاکی است که تقسیم پردازش یک مرحله‌ای به چندین مرحله پردازش متوالی، منجر به کاهش حجم محاسبات، کاهش مصرف منابع سخت‌افزاری، توان مصرفی پایین‌تر و فرکانس عملکرد بالاتر می‌گردد. هرچند اندازهٔ برداری بزرگتر به کمترین زمان اجرا منجر می‌شود. ولی حجم محاسبات نیز بسیار سنگین خواهد شد و این امر مانع بهره‌برداری مناسب از تراشه‌های با ظرفیت محدود می‌شود. لذا، با توجه به مزیت‌های فراوان روش چندبرداری نسبت به تک‌برداری، می‌توان از آن در پیاده‌سازی زمان- آنی سایر پردازش‌های چندبُعدی نیز بهره گرفت.

۷- مراجع

[1] W. L. Melvin, "A stap Overview: IEEE Aerospace and Electronic Systems Magazine," vol. 19, no. 1, pp. 19-35, 2004.

[2] R. Klemm, "Principles of Space-Time Adaptive Processing," IEE Radar, Sonar, Navigation and Avionics, London: IEE Press, 2002.

[3] J. Ward, "Space-Time Adaptive Processing for Airborne Radar," (Technical Report no. 1015) Lexington, MA: Lincoln Laboratory, MIT, 1994.

[4] W. L. Melvin, "Space-Time Adaptive Radar Performance in Heterogeneous Clutter," IEEE Transactions on Aerospace and Electronic Systems, vol. 36, no. 2, pp. 621-633, 2000.

[5] Z. Hao and M. Luo, "Adaptive Mainlobe Jamming Suppression Method for STAP Airborne Radar," IET International Radar conf., 2013.

[6] L. E. Brennan and L. S. Reed, "Theory of Adaptive Radar," IEEE Transactions on Aerospace and Electronic Systems, vol. 9, no. 2, pp. 237–252, 1973.

[7] A. S. Paulus, W. L. Melvin, and B. Himed, "Performance and Computational Trades for RD-STAP Algorithms in Challenging Detection Environments," IEEE Radar conf., 2016.

[8] N. A. Gawande, J. B. Manzano, A. Tumeo, N. R. Tallent, D. J. Kerbyson, and A. Hoisie, "Power and Performance trade-offs for Space Time Adaptive Processing," IEEE 26th International conf. on Application-Specific Systems, Architectures and Processors, 2015.

[9] L. B. Fertig, "Analytical Expressions for Space-time Adaptive Processing (STAP) Performance," IEEE Transactions on Aerospace and Electronic Systems, vol. 51, no. 1, pp. 42–53, 2015.

[10] J. M. Lebak and A.W. Bojanczyk, "Design and Performance Evaluation of A Portable Parallel Library for Space-Time Adaptive Processing," IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 3, pp. 287–298, 2000.

[11] W. K. Liao, A. Choudhary, D. Weiner, and P. Varshney, "Performance evaluation of a parallel pipeline computational model for space-time adaptive processing," The Journal of Supercomputing, vol. 31, no. 2, pp. 137-160, 2005.

[12] K. Rajan and L. M. Patnik, "Implementation of STAP Algorithms on IBM SP2 and on ADSP 21062 Dual Digital Signal Processor Systems," Microprocessors and Microsystems, vol. 27, no. 4, pp. 221-227, 2003.

[13] F. Xikun and W. Yongliang, "Real-time implementation of Airborne Radar Space-Time Adaptive Processing on Multi-DSP System," International conf. on Radar (CIE '06), 2006.

[14] S. Bin, L. Shaohong, R. Yi, and L. Jingsheng, "Realization and Comparison of QRD Algorithms for STAP," 2nd Asian-pacific conf. on Synthetic Aperture Radar (APSAR), Proc. of IEEE, pp. 306-309, 2009.

[15] Q. Li, J. S. Cao, and X. Pei, "Real-time and extensible calculation of STAP weights on FPGA," 12th International conf. on Signal Processing," pp. 425-428, 2014.

[16] A. Jarrah and M. Jamali, "Software tool for Efficient FPGA Design of Direct Data Domain Approach for Space-Time Adaptive Processing" Electronics Letters, vol. 49, no. 13, pp. 789–791, 2013.

[17] Z. U. Mahmood, M. Alam, K. Jamil, and Z. O. Al-Hekail, "FPGA Implementation of Space-Time Adaptive Processing (stap) Algorithm for Target Detection in Passive Radars," Progress in Electromagnetics Research C, vol. 35, pp. 35–48, 2013.

[18] T. M. Benson, R. K. Hersey, and E. Culpepper, "GPU-based Space-Time Adaptive Processing (STAP) for Radar," IEEE High Performance Extreme Computing conf. (HPEC), 2013.

[19] M. A. Richards, "Fundamentals of Radar Signal Processing," Tata McGraw-Hill Education, 2005.

[20] R. Woods, J. McAllister, G. Lightbody, and Y. Yi, "FPGA-based Implementation of Signal Processing Systems," John Wiley & Sons Ltd. U.K, 2008.

[21] G. H. Golub and C. F. Van Loan, "Matrix Computations," Johns Hopkins University Press, Baltimore, Third edition, 1996.

[22] M. Jervis, "Advances in DSP design tool flows for FPGAs," Military Communications conf., pp. 2041-2046, 2010.

[23] E. Rezagholizadeh, M. A. Sebt, "Clutter Mitigation in Airborne Radar Using DeterministicSubspace Clutter and Direct Data Domain", Journal of Radar, vol. 4, no. 2, pp. 1–10, 2016 (In Persian).

[24] S. Sadeghi, A. Sheikhi, "Jamming Noise Suppression Using Discrete Polynomial-Phase Transform in Pulsed Radars", Journal of Radar, vol. 5, no. 1, pp. 53–66, 2017 (In Persian).

[25] M. R. Taban, A. Sheikh Mozaffari, "Adaptive Radar Signal Detection in Gaussian Clutter with Autoregressive Model Using Kalman Fillter", Journal of Radar, vol. 2, no. 3, pp. 1–12, 2014 (In Persian).

Journal of "Radar"
Vol. 6, No. 1, 2018 (Serial No. 19)

Design & Efficient Implementation of STAP with the Multi-Vector Method for the Target Detection in Airborne Radars

N. Hasanikhah, S. Amin-Nejad*, Gh. Darvish, M. R. Moniri

University of Guilan, Rasht

(Received: 25/10/2017, Accepted: 12/09/2018)

Abstract

Space-Time Adaptive Processing (STAP) is a fundamental technique in improving the performance of radars which are acted in the presence of severe dynamic disturbances such as clatter and jamming. STAP operation is based on very high sampling rates from signals received simultaneously from several antenna arrays and a number of pulses. Therefore, the volume of computation is very high and its implementation is difficult. In this paper, multi-vector method is proposed to reduce the amount of STAP calculations. Implementation results show that the proposed method, in addition to reducing the computational volume, leads to reduced hardware resources, reduced power consumption and increased maximum operating frequency. For example, the calculation volume for 6 antenna arrays, 10 receiving pulses and 200 range samples with a vector size of 17 is obtained 28.1 GFLOPS with maximum frequency of 278 MHz, and the computation time of 262 μs on the Arria 10 chip. Therefore, the multi-vectoring method can meet the real-time requirements of STAP weights. Angle-doppler adaptive pattern and the static test for the target detection indicate that using the above method is very practical for calculating weights.

Keywords: Airborne Radar, Clutter, Jamming, Space-Time Adaptive Processing

* Corresponding author E-mail: amin@liv.ac.uk