این فاز چهار پروژه مجزا و مستقل تعریف شده است که هر گروه از دانشجویان (هر گروه دو نفر) یک پروژه را پیاده‌سازی می‌کنند. شرح هر پروژه در ادامه ذکر گردیده است. این فاز در سه مرحله انجام خواهد شد.

مرحله اول: انتخاب اولویت‌بندی تا تاریخ ۱۳۹۵/۳/۵

در این مرحله هر گروه از دانشجویان با ارسال یک ایمیل (به آدرس pmahmoody@gmail.com) اسامی اعضای گروه را اعلام می‌کنند و همچنین اولویت انجام هر چهار پروژه را مشخص می‌کنند (افرادی که سریع‌تر اعلام کنند در اولویت هستند).

مرحله دوم: پیاده‌سازی بخش سخت‌افزاری تا تاریخ ۱۳۹۵/۳/۱۸

فایل پروژه شامل کدهای بخش سخت‌افزاری، Test bench و شبیه‌سازی را ارسال می‌کنند (نیازی به گزارش کتبی در این فاز نیست).

مرحله سوم: پیاده‌سازی کامل تا تاریخ ۱۳۹۵/۴/۷

فایل پروژه شامل کدهای بخش سخت‌افزاری، نرم‌افزاری، Test bench، شبیه‌سازی و گزارش کتبی را ارسال می‌کنند. لازم به ذکر است زمان ارائه حضوری متعاقبا اعلام خواهد شد.

در صورت وجود هر گونه ابهام در صورت پروژه ها و یا عملکرد این سیستم‌ها، می‌توانید با ارسال ایمیل یا مراجعه حضوری به آزمایشگاه دکتر صاحب الزمانی توضیحات تکمیلی را دریافت کنید.

پروژه A:

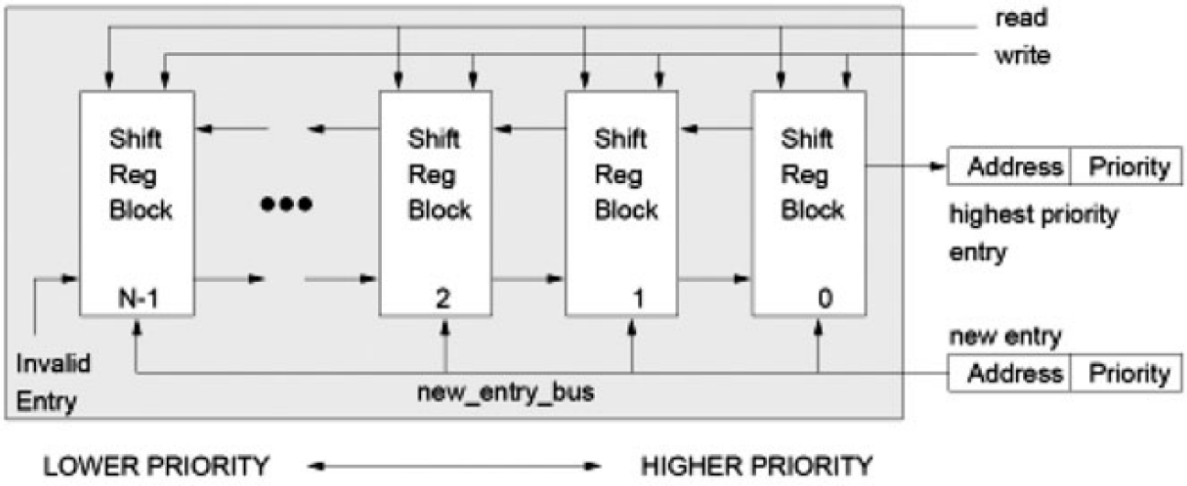
در طراحی زمانبند سخت‌افزاری در بیشتر موارد با استفاده از یک صف که براساس اولویت وظایف مرتب شده است، عمل زمانبندی را انجام می‌دهند به گونه‌ای که در هر زمان که یک منبع بی‌کار می‌شود، وظیفه‌ای که در ابتدای صف است، توسط زمانبند برای اجرا انتخاب می‌شود.

در اینجا یک زمانبند سخت‌افزاری قابل‌بازپیکربندی آورده شده است که در این زمانبند از یک ثبات انتقال[1] برای نگهداری وظایف آماده به صورت مرتب شده براساس اولویت آن‌ها استفاده می‌شود. در هنگام ورود هر وظیفه‌ی جدید به صف، تمام وظایفی که از قبل داخل صف می‌باشند با استفاده از یک مقایسه کننده‌ی محلی اولویت خود را با وظیفه‌ی جدید که می‌خواهد وارد شود مقایسه می‌کند و در صورتی که اولویت آن کمتر باشد به سمت انتهای صف یک بار جابه‌جا می‌شود و اگر اولویت آن بیشتر باشد بدون جابه‌جایی باقی می‌ماند. در انتها، یک خانه‌ی خالی در صف ایجاد می‌شود و وظیفه‌ای که قصد ورود به صف را داشت وارد آن خانه‌ی خالی می‌شود.

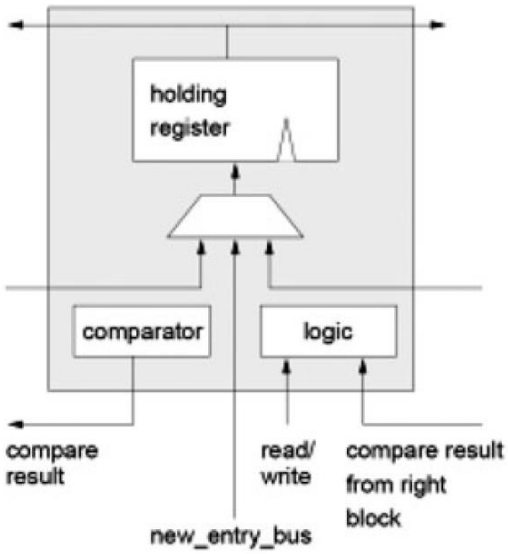
سازوکار فوق باعث می شود که وظایف موجود در صف همواره براساس اولویت آن‌ها مرتب بوده و وظیفه‌ی با اولویت بیشتر همیشه در ابتدای صف باشد. در هر زمانی که سیستم‌عامل از زمانبند تقاضای انتخاب وظیفه‌ی بعدی را کند، زمانبند وظیفه‌ای که در ابتدای صف است را بدون نیاز به انجام کار اضافی به سیستم عامل معرفی می کند. در این کار عمل ورود وظیفه به صف و همچنین انتخاب یک وظیفه برای اجرا با تعداد کمی پالس ساعت انجام می‌شود.

در شکل ۱ به صورت کلی یک زمان‌بند سخت‌افزاری مطابق توضیحات بالا را نشان می‌دهد و در شکل ۲ ساختار داخلی یک Shift Reg Block را نشان می‌دهد.

[1] Shift register


read
write
Shift Reg Block
N-1
Shift Reg Block
2
Shift Reg Block
1
Shift Reg Block
0
Address
Priority
highest priority entry
new entry
Address
Priority
Invalid Entry
new_entry_bus
LOWER PRIORITY
HIGHER PRIORITY

شکل ۱


holding register
comparator
logic
compare result
read/ write
compare result from right block
new_entry_bus

شکل ۲

تاریخ تحویل: ۹۵/۳/۱۸ و ۹۵/۴/۷

در این پروژه شما باید صف مذکور را در بخش سخت‌افزاری پیاده‌سازی کرده و در بخش نرم‌افزاری یک مدیریت ساده با دستورهای ارسال وظیفه به صف و گرفتن وظیفه با اولویت بالاتر از صف را انجام دهید. نرم‌افزار باید طبق جدول زیر وظایف را ارسال و دریافت کند. مراحل زیر با فواصل زمانی کافی انجام می‌گیرند.

۱. وظیفه‌ی A به صف ارسال شود.
۲. وظیفه‌ی بعد از صف گرفته شود.
۳. وظیفه‌ی B به صف ارسال شود.
۴. وظیفه‌ی C به صف ارسال شود.
۵. وظیفه‌ی بعد از صف گرفته شود.
۶. وظیفه‌ی D به صف ارسال شود.
۷. وظیفه‌ی بعد از صف گرفته شود.
۸. وظیفه‌ی بعد از صف گرفته شود.

وظیفه	اولویت
A	5
B	3
C	5
D	1

پروژه B:

همان طور که پیش‌تر ذکر شد در طراحی زمانبند سخت‌افزاری در بیشتر موارد با استفاده از یک صف که براساس اولویت وظایف مرتب شده است، عمل زمانبندی را انجام می‌دهند به گونه‌ای که در هر زمان که یک منبع بی‌کار می‌شود، وظایفی که در ابتدای صف می‌باشد توسط زمانبند برای اجرا انتخاب می‌شود. در ادامه ما یک مدل دیگر از این زمان‌بندها را معرفی می‌کنیم.

مشابه پروژه A در اینجا هم یک زمانبند سخت‌افزاری قابل‌بازپیکربندی آورده شده است که در این زمانبند از یک ثبات انتقال[2] برای نگهداری وظایف آماده به صورت مرتب شده براساس اولویت آن‌ها استفاده می‌شود. در این زمانبند بر پایه زمانبند قبلی مدلی برای صف اولویت ارائه شده است که در آن مقایسه کننده‌ها و ثبات‌های محلی که در پروژه‌ی A برای اتخاذ تصمیمات به صورت محلی استفاده می‌گردید حذف شده است و یک پرچم در هر خانه از صف برای این‌که نشان دهد آن خانه خالی است قرار داده شده است. همچنین به هر وظیفه داخل سیستم یک شناسه اختصاص داده شده است و وظایف داخل صف با آن شناسه شناسایی می‌شوند. اطلاعات مربوط به زمانبندی در حافظه‌ی جداگانه نگهداری می‌شود که فقط در تحت کنترل زمانبند است. یک کنترل‌کننده مرکزی وظیفه کنترل صف را برعهده گرفته است. به این صورت که در زمان ورود هر وظیفه، اطلاعات مربوط به زمانبندی آن وظیفه را از حافظه می‌گیرد و سپس به صورت خطی یا دودویی شروع به جستجو در صف می‌کند تا مکان مناسب برای وظیفه جدید را پیدا کند. بعد از یافتن مکان مناسب برای وظیفه‌ی مذکور، یک فرمان به وظایف داخل صف می‌دهد تا آن‌ها با جابه‌جایی مکان مورد نظر را خالی کنند. سپس وظیفه‌ی مذکور در مکان خود قرار می‌گیرد. وظیفه‌ای که در ابتدای صف قرار دارد همواره دارای اولویت بالاتر می‌باشد و به محض نیاز می‌تواند برای اجرا انتخاب شود. در این مدل امکان حذف وظایف از میان صف نیز وجود دارد، به این صورت که کنترل‌کننده مرکزی فرمان لازم برای حذف وظیفه‌ی مورد نظر و جابه‌جایی سایر وظایف را صادر می‌کند.

[2] Shift register

سازوکار فوق باعث می‌شود که وظایف موجود در صف همواره براساس اولویت آن‌ها مرتب بوده و وظیفه‌ی با اولویت بیشتر همیشه در ابتدای صف باشد. در هر زمانی که سیستم‌عامل از زمانبند تقاضای انتخاب وظیفه‌ی بعدی را کند، زمانبند وظیفه‌ای که در ابتدای صف است را بدون نیاز به انجام کار اضافی به سیستم عامل معرفی می کند. در این کار عمل ورود وظیفه به صف و همچنین انتخاب یک وظیفه برای اجرا با تعداد کمی پالس ساعت انجام می‌شود.

شکل ۳ به صورت کلی یک زمان‌بند سخت‌افزاری مطابق توضیحات بالا را نشان می‌دهد و در شکل ۴ ساختار داخلی یک Shift Reg Block را نشان می‌دهد.

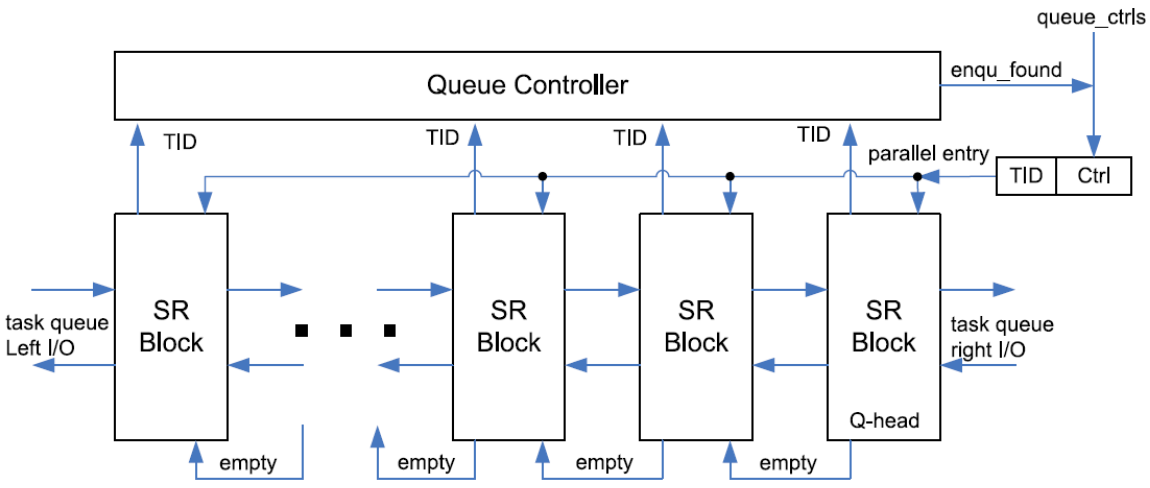


Fig. 4. Basic SR task queue.

شکل ۳

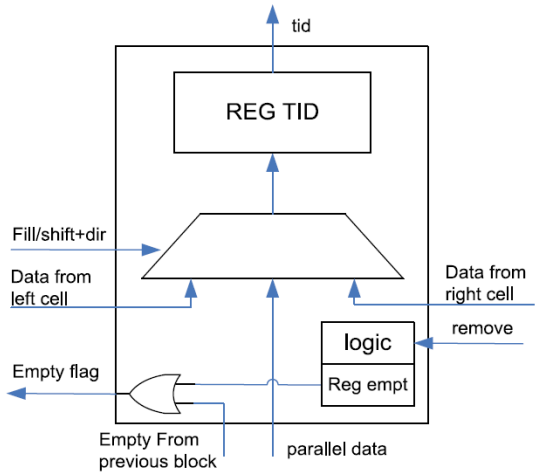


Fig. 5. BSR task queue block.

شکل ۴

در این پروژه شما باید صف مذکور را در بخش سخت‌افزاری پیاده‌سازی کرده و در بخش نرم‌افزاری یک مدیریت ساده با دستورهای ارسال وظیفه به صف و گرفتن وظیفه با اولویت بالاتر از صف را انجام دهید. نرم‌افزار باید طبق جدول زیر وظایف را ارسال و دریافت کند. مراحل زیر با فواصل زمانی کافی انجام می‌گیرند.

۱. وظیفه‌ی A به صف ارسال شود.
۲. وظیفه‌ی بعد از صف گرفته شود.
۳. وظیفه‌ی B به صف ارسال شود.
۴. وظیفه‌ی C به صف ارسال شود.
۵. وظیفه‌ی C از صف حذف شود.
۶. وظیفه‌ی D به صف ارسال شود.
۷. وظیفه‌ی بعد از صف گرفته شود.
۸. وظیفه‌ی بعد از صف گرفته شود.

وظیفه	اولویت
A	5
B	3
C	5
D	1

پروژه C:

همان طور که پیش‌تر ذکر شد در طراحی زمانبند سخت‌افزاری در بیشتر موارد با استفاده از یک صف که براساس اولویت وظایف مرتب شده است، عمل زمانبندی را انجام می‌دهند به گونه‌ای که در هر زمان که یک منبع بی‌کار می‌شود، وظایفی که در ابتدای صف می‌باشد توسط زمانبند برای اجرا انتخاب می‌شود. در ادامه ما یک مدل دیگر از این زمان‌بندها را معرفی می‌کنیم.

مشابه پروژه‌های A و B در اینجا هم یک زمانبند سخت‌افزاری قابل‌بازپیکربندی آورده شده است که در این زمانبند از یک ثبات انتقال[3] برای نگهداری وظایف آماده به صورت مرتب شده براساس اولویت آن‌ها استفاده می‌شود. در این زمانبند بر پایه زمانبند B، مدلی برای صف اولویت ارائه شده است. با این تفاوت که در این زمانبند از صف فقط برای وظایف آماده استفاده نمی‌شود بلکه برای وظایف متوقف شده (آن‌هایی که بدلیل آماده نبودن یک منبع یا داده موقتا قادر به ادامه کار نیستند) هم استفاده می‌شود. برای این منظور به هر خانه از صف یک پرچم برای این که مشخص شود کدام وظیفه آماده است یا خیر اضافه شده است.

سازوکار فوق باعث می‌شود که وظایف موجود در صف همواره براساس اولویت آن‌ها مرتب بوده و وظیفه‌ی با اولویت بیشتر همیشه در ابتدای صف باشد ولی به دلیل آن‌که بعضی از وظایف داخل صف آماده‌ی اجرا نیستند، در هر زمانی که سیستم‌عامل از زمانبند تقاضای انتخاب وظیفه‌ی بعدی را کند، زمانبند با استفاده از یک Priority Encoder وظیفه با اولویت بالاتر که بیت Ready آن یک است را به سیستم عامل معرفی می کند. در این کار عمل ورود وظیفه به صف و همچنین انتخاب یک وظیفه برای اجرا با تعداد کمی پالس ساعت انجام می‌شود.

در شکل ۵ به صورت کلی یک زمان‌بند سخت‌افزاری مطابق تو ضیحات بالا را نشان می‌دهد و در شکل ۶ ساختار داخلی یک Shift Reg Block را نشان می‌دهد.

[3] Shift register

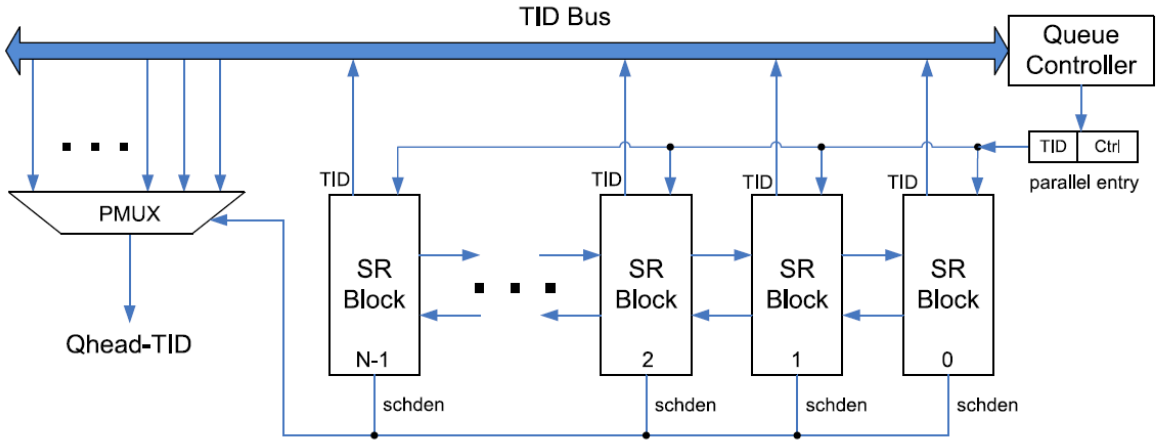


Fig. 7. Advanced SR task queue model.

شکل ۵

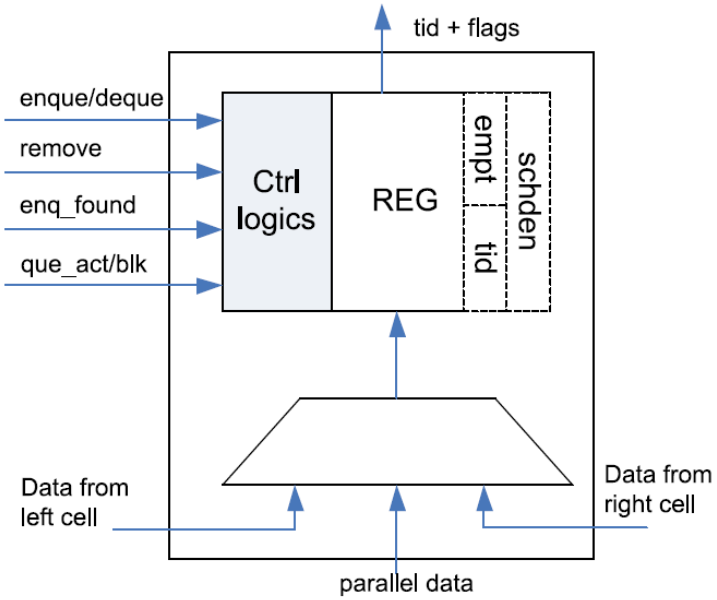


Fig. 8. Advanced SR task queue block.

شکل ۶

در این پروژه شما باید صف مذکور را در بخش سخت‌افزاری پیاده‌سازی کرده و در بخش نرم‌افزاری یک مدیریت ساده با دستورهای ارسال وظیفه به صف و گرفتن وظیفه با اولویت بالاتر از صف را انجام دهید. نرم‌افزار باید طبق جدول زیر وظایف را ارسال و دریافت کند. مراحل زیر با فواصل زمانی کافی انجام می‌گیرند.

۱. وظیفه‌ی A به صف ارسال شود.
۲. وظیفه‌ی بعد از صف گرفته شود.
۳. وظیفه‌ی B به صف ارسال شود.
۴. وظیفه‌ی A معلق می‌شود و به صف باز می‌گردد.
۵. وظیفه‌ی C به صف ارسال شود.
۶. وظیفه‌ی C از صف حذف شود.
۷. وظیفه‌ی D به صف ارسال شود.
۸. وظیفه‌ی بعد از صف گرفته شود.

وظیفه	اولویت
A	5
B	3
C	5
D	1

در صورت نیاز به توضیح بیشتر در مورد پروژه A، B و C به مقاله زیر مراجعه کنید.

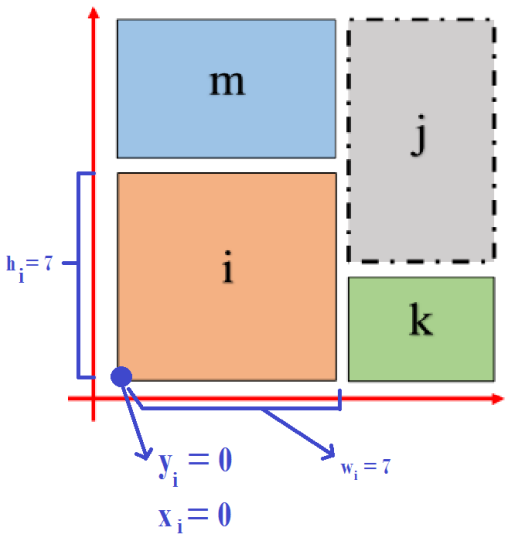
Y. Tang and N. W. Bergmann, “A hardware scheduler based on task queues for FPGA-based embedded real-time systems,” *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1254–1267, 2015.

پروژه D:

یکی از مهم‌ترین چالش‌های زمانبندی وظایف بر روی FPGA وابستگی عمل‌کرد سیستم‌های زمانبندی به الگوریتم‌های جایابی است. در زمانبندی برای اتخاذ تصمیمات زمانبندی نیازمندیم بدانیم که FPGA مورد نظر برای کدام یک از وظایفی که در هر لحظه در صف موجود است فضای کافی دارد.

برای این کار در این‌جا ما فرض می‌کنیم که لیستی از وظایفی که در FPGA موجود است در نرم‌افزار داریم و ابتدا یک سری نقاط که حدس می‌زنیم برای قرار دادن وظیفه‌ی جدید مناسب باشد را در نظر می‌گیریم (برای مثال نقاطی در لبه‌ی وظایف موجود در FPGA) سپس با کمک سخت‌افزار چک می‌کنیم که آیا آن نقاط با وظایف موجود در FPGA تداخل دارند یا خیر. در بخش سخت‌افزاری با پیاده‌سازی تعدادی مقایسه کننده مشخص می‌کنیم که نقطه مناسب است یا خیر.

برای مثال فرض کنید مطابق شکل زیر سه وظیفه‌ی m، k و i هم اکنون در FPGA موجود هستند اگر بخواهیم بدانیم که آیا مکان در نظر گرفته شده برای وظیفه‌ی j مناسب است یا خیر باید آن را تک تک با وظایف m، k و i مقایسه و مشخص کنیم که تداخلی وجود دارد یا خیر. معادله ۱ چک می کند که آیا وظیفه‌ی j با وظیفه‌ی k تداخل دارد یا خیر. در صورتی که حداقل یکی از شرط‌های معادله‌ی ۱ برقرار باشد این به این معنی است که تداخلی وجود ندارد. مثلا اگر شرط $y_k + h_k < y_j$ برقرار باشد یعنی وظیفه‌ی j کاملا بالای وظیفه‌ی k است.



$$\begin{cases} x_k+w_k<x_j \\ y_k+h_k<y_j \\ x_k-w_j>x_j \\ y_k-h_j>y_j \end{cases} (۱)$$

$$\begin{cases} x_m+w_m<x_j \\ y_m+h_m<y_j \\ x_m-w_j>x_j \\ y_m-h_j>y_j \end{cases} (۲)$$

$$\begin{cases} x_i+w_i<x_j \\ y_i+h_i<y_j \\ x_i-w_j>x_j \\ y_i-h_j>y_j \end{cases} (۳)$$

فرض کنید حداکثر هشت وظیفه می تواند در FPGA وجود داشته باشد و با توجه به شکل زیر سخت‌افزار لازم را پیاده‌سازی کنید.