

# معماری کامپیوتر

دانشکده مهندسی کامپیوتر

دکتر اسدی  
بهار ۱۴۰۳

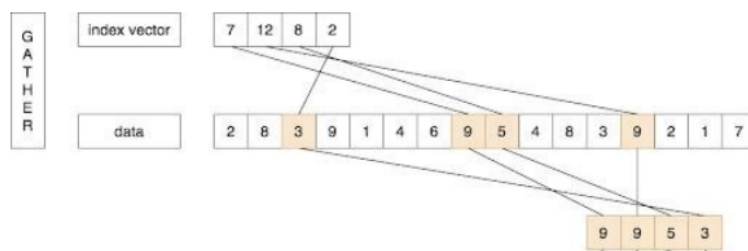
مهدی علی نژاد، ۴۰۱۱۰۶۲۶۶، امیرحسین صوری، ۴۰۱۱۰۶۱۲۸



## تمرین دوم، بخش عملی

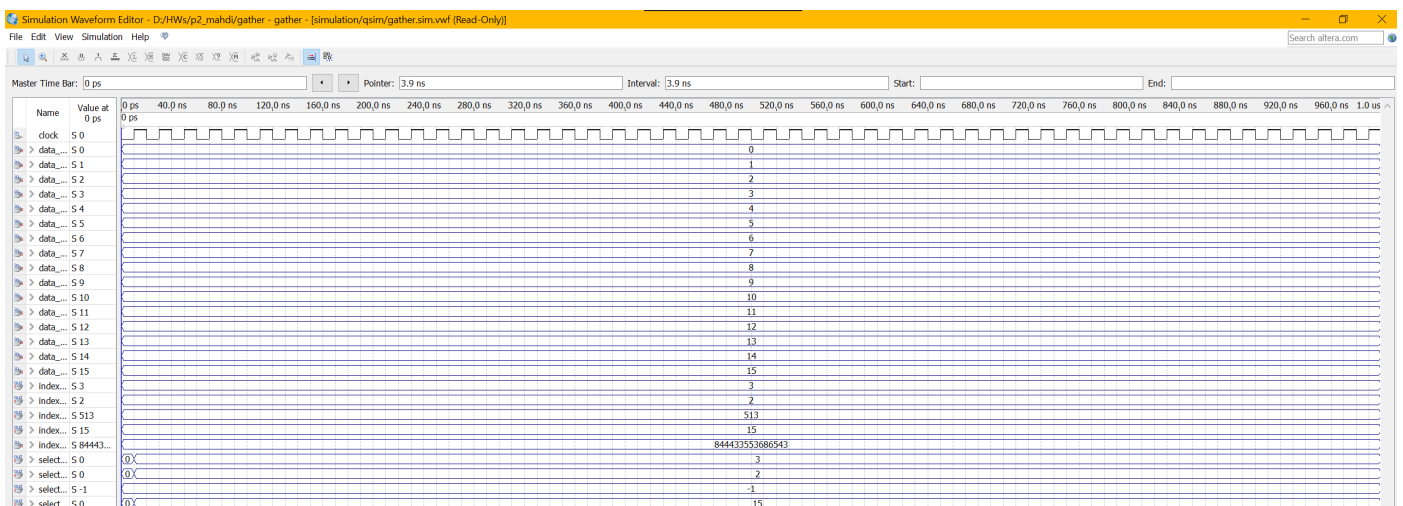
سوال ۱

۲. در این تمرین می‌خواهیم که با عملیات جمع‌آوری<sup>۴</sup> که در برخی از پردازنده‌هایی با ثبات‌های بزرگ انجام می‌شود، آشنا شویم. فرض کنید که پردازنده‌ای در اختیار دارید که ۴ ثبات ۲ بیتی و یک ثبات ۸ بیتی در اختیار دارد. فرض کنید که می‌خواهیم هر دو بایت از ثبات ۸ بیتی را با مقادیر یکی از ثبات‌های دو بیتی پر کنیم. یکی از کارهایی که در اینجا می‌توان انجام داد این است که از شیفت و ORهای متوالی در ثبات ۸ بیتی کمک بگیریم که آن را پر کنیم. اما می‌توان از عملیات جمع‌آوری نیز بهره برد. کاری که عملیات جمع‌آوری انجام می‌دهد، این است که ابتدا یک آرایه به طول ۴ به نام بردار اندیس<sup>۵</sup> که در آن اندیس ثبات‌ها نوشته شده است و باید به ترتیب در ثبات بزرگ ۸ بیتی قرار گیرند را به عنوان ورودی دریافت می‌کند. به عنوان مثال به شکل زیر توجه کنید:



در این شکل هر خانه در ردیف داده نشان‌دهنده یک ثبات ۲ بیتی است و ۴ خانه نارنجی پایینی نشان‌دهنده یک ثبات ۸ بیتی است.

برای این کار کافی است که به صورت شماتیک مداری را در Quartus پیاده‌سازی کنید که عملیات جمع‌آوری را از ۴ ثبات دو بیتی به یک ثبات ۸ بیتی انجام می‌دهد. دقت کنید که بردار اندیس را می‌توانید ورودی مدار فرض کنید. همچنین برای اطلاعات بیشتر می‌توانید به اینجا مراجعه کنید.



در شکل گزارش شده data و index vector و کلاک به صورت ورودی به مدار داده می شوند. مدار بر اساس index های داده شده، دیتا ها را انتخاب می کند و آنها را در یک رجیستر ۸ بایتی می ریزد.

برای آسانی بررسی صحت مدار، index های تشخیص داده شده و اعداد انتخاب شده برای ریخته شدن در رجیستر ۸ بایتی را به صورت خروجی در نظر گرفتیم.

index_vector[0]	5 0
index_vector[1]	5 0
index_vector[2]	5 0
index_vector[3]	5 0
index_vector[4]	5 0
index_vector[5]	5 0
index_vector[6]	5 0
index_vector[7]	5 0
index_vector[8]	5 0
index_vector[9]	5 0
index_vector[10]	5 0
index_vector[11]	5 0
index_vector[12]	5 0
index_vector[13]	5 0
index_vector[14]	5 1
index_vector[15]	5 1

(عدد ۳)

index_vector[16]	5 0
index_vector[17]	5 0
index_vector[18]	5 0
index_vector[19]	5 0
index_vector[20]	5 0
index_vector[21]	5 0
index_vector[22]	5 0
index_vector[23]	5 0
index_vector[24]	5 0
index_vector[25]	5 0
index_vector[26]	5 0
index_vector[27]	5 0
index_vector[28]	5 0
index_vector[29]	5 0
index_vector[30]	5 1
index_vector[31]	5 0

(عدد ۲)

index_vector[32]	5 0
index_vector[33]	5 0
index_vector[34]	5 0
index_vector[35]	5 0
index_vector[36]	5 0
index_vector[37]	5 0
index_vector[38]	5 1
index_vector[39]	5 0
index_vector[40]	5 0
index_vector[41]	5 0
index_vector[42]	5 0
index_vector[43]	5 0
index_vector[44]	5 0
index_vector[45]	5 0
index_vector[46]	5 0
index_vector[47]	5 1

(عدد ۵۱۳)

index_vector[48]	5 0
index_vector[49]	5 0
index_vector[50]	5 0
index_vector[51]	5 0
index_vector[52]	5 0
index_vector[53]	5 0
index_vector[54]	5 0
index_vector[55]	5 0
index_vector[56]	5 0
index_vector[57]	5 0
index_vector[58]	5 0
index_vector[59]	5 0
index_vector[60]	5 1
index_vector[61]	5 1
index_vector[62]	5 1
index_vector[63]	5 1

(عدد ۱۵)

در بالا، محتویات index vector آمده است که هر ۲ بایت آن یک index را مشخص می کند. در صورتی که index مورد نظر از بازه ی صفر تا تعداد رجیستر های دیتا (در این مورد ۱۶) منهای یک، خارج باشد، عدد ۱- در خانه ی هدف ریخته می شود. index vector ما در بالا به این صورت است. ۳، ۲، [۵۱۳، ۱۵] که همانطور که می بینید خروجی ها همانطور که گفته شده آمده است.



این هم تصویر مدار است که از ۱۶ رجیستر به عنوان محل ذخیره سازی دیتا و ۴to۱۶ mux برای انتخاب خروجی و همچنین ۴to۱ mux برای تشخیص اینکه آیا مقدار index خارج از بازه است یا خیر.