به نام خداوند بخشنده مهربان

معماری کامپیوتر

دانشکده مهندسی کامپیوتر

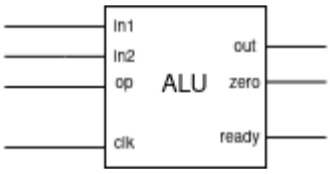
دکتر اسدی
بهار ۱۴۰۳

مهدی علی نژاد، ۴۰۱۱۰۶۲۶۶ ، امیرحسین صوری، ۴۰۱۱۰۶۱۲۸

تمرین سوم، بخش عملی

تمارین عملی

۱. در ادامه تمرینات عملی قصد داریم به طراحی پردازنده MIPS بپردازیم. در این تمرین می‌خواهیم قسمت ALU این پردازنده را طراحی کنیم. این واحد پردازشی دو عدد ۸ بیتی (in1, in2) و یک کد عملیات ۳ را می‌گیرد. اما خروجی بر اساس نوع عملیات متفاوت است. خروجی یک عملیات ممکن است یک عدد ۸ بیتی به اسم out باشد یا یک سیگنال یک بیتی به اسم zero باشد که برای مقایسه‌ها استفاده می‌شود. علاوه براین یک سیگنال به اسم ready نیز داریم، این سیگنال هنگامی که جواب آماده شد فعال می‌شود. در شکل زیر می‌توانید ساختار کلی ALU را مشاهده کنید.



حال در ادامه عملیات‌هایی که برروی دو عدد ۸ بیتی انجام می‌شود را بررسی می‌کنیم.

آ) AND :

$$op = 0000$$
$$out = in1 \text{ AND } in2$$

ب) OR :

$$op = 0001$$
$$out = in1 \text{ OR } in2$$

ج) XOR :

$$op = 0010$$
$$out = in1 \text{ XOR } in2$$

د) Add : Carry Select Adder آنها را جمع می‌کند و خروجی ۸ بیتی را به ما می‌دهد.

$$op = 0011$$
$$out = in1 + in2$$

ه) عملیات تفریق:

$$op = 0100$$
$$out = in1 - in2$$

و) عملیات ضرب: از الگوریتم booth برای ضرب با علامت استفاده می‌کند.

$$op = 0101$$
$$out = in1 \times in2$$

ز) Equal :

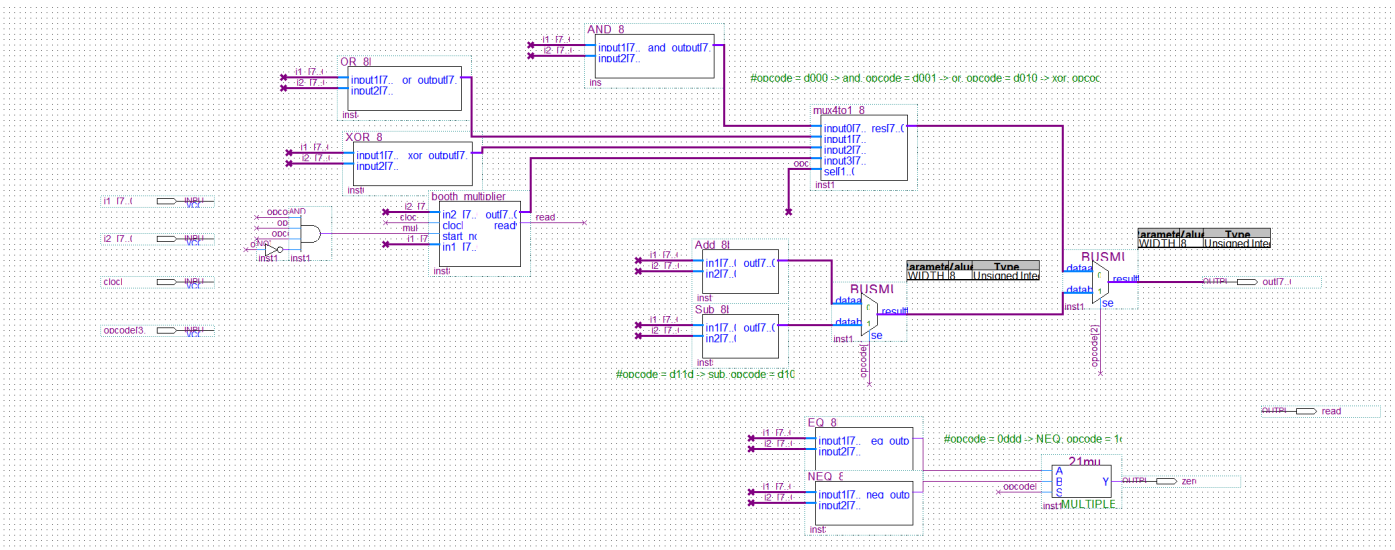
$$op = 0110$$
$$zero = (in1 == in2)$$

ح) N-Equal :

$$op = 0111$$
$$zero = (in1 \neq in2)$$

حتما طراحی خود را به صورت کامل و دقیق انجام دهید چرا که تمارین بعدی به این قسمت وابستگی دارند. همچنین در گزارش خود نیز برای هر حالت حداقل ۳ مثال بزنید و جواب آن‌ها را در گزارش خود بیاورید.
توجه کنید که اعداد علامت‌دار هستند.

من اول از همه توضیحی در مورد مدار و کلیت های آن بدم.

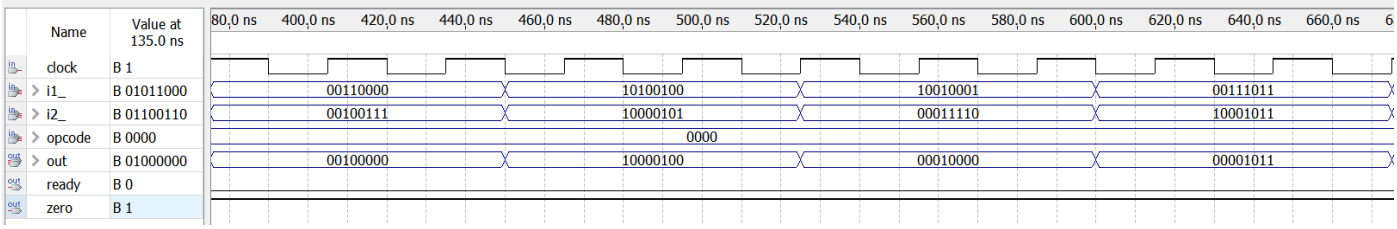


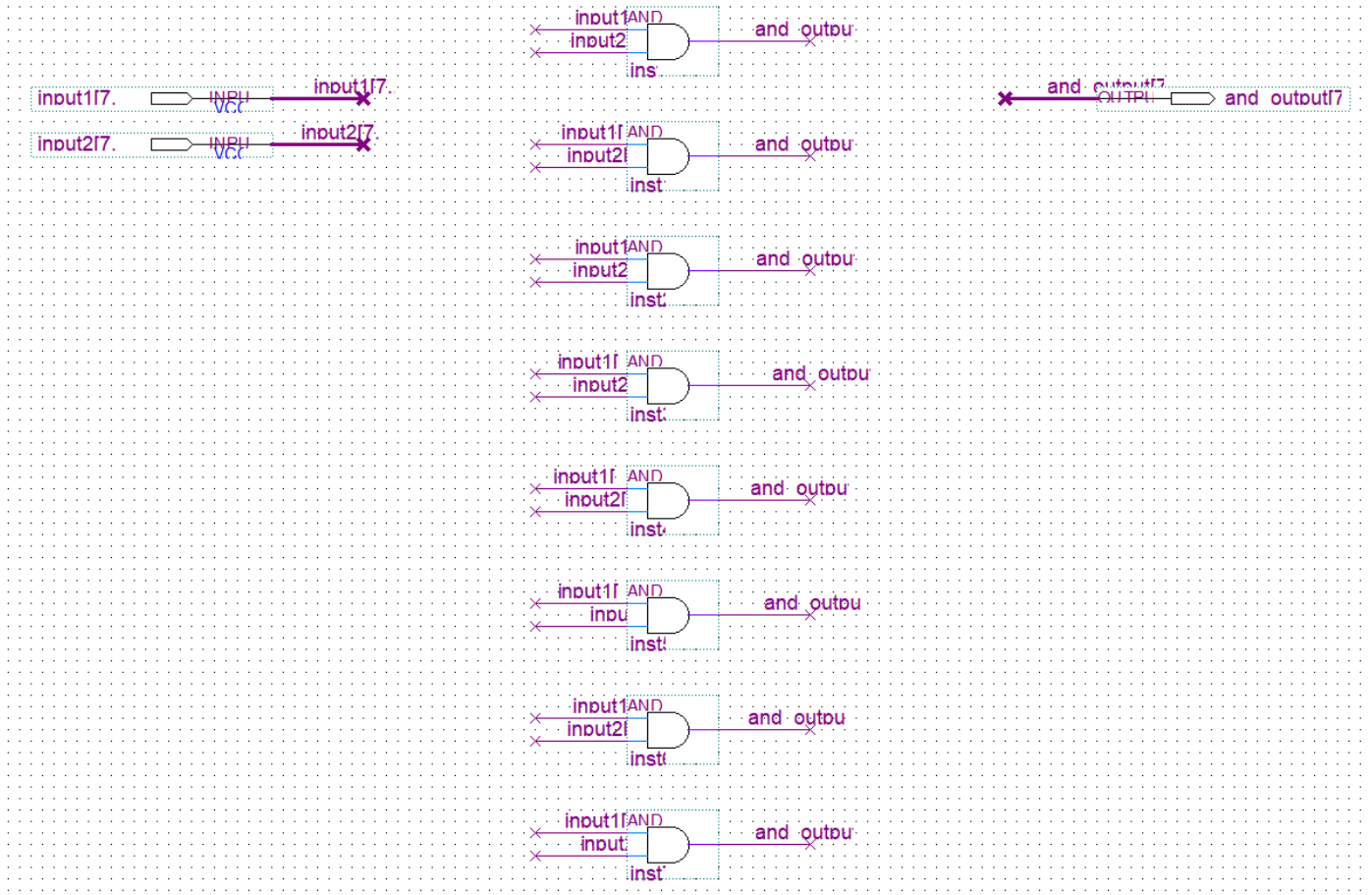
مدار بالا شماتیک کلیی از alu ماست. آن شامل ۸ بلاک ، که هر کدام یکی از محاسبات را انجام می دهند، و تعدادی MUX برای مشخص کردن خروجی. همچنین سلکت های ماکس برای سادگی طوری از روی opcode ساخته شده اند که تعدادی از بیت های opcode برای مدار به صورت don't care فرض شده است. برای مثال opcode اصلی equal، ۱۰۰۰ است ولی در ماکس فقط به یک بودن بیت اول توجه می کند.

همچنین برای راحتی، opcode دستورات را تغییر دادیم.

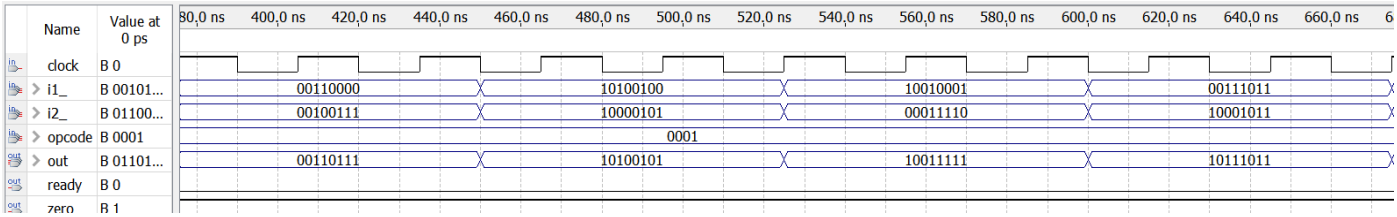
خب برویم مروری بر روی هر عملیات داشته باشیم.

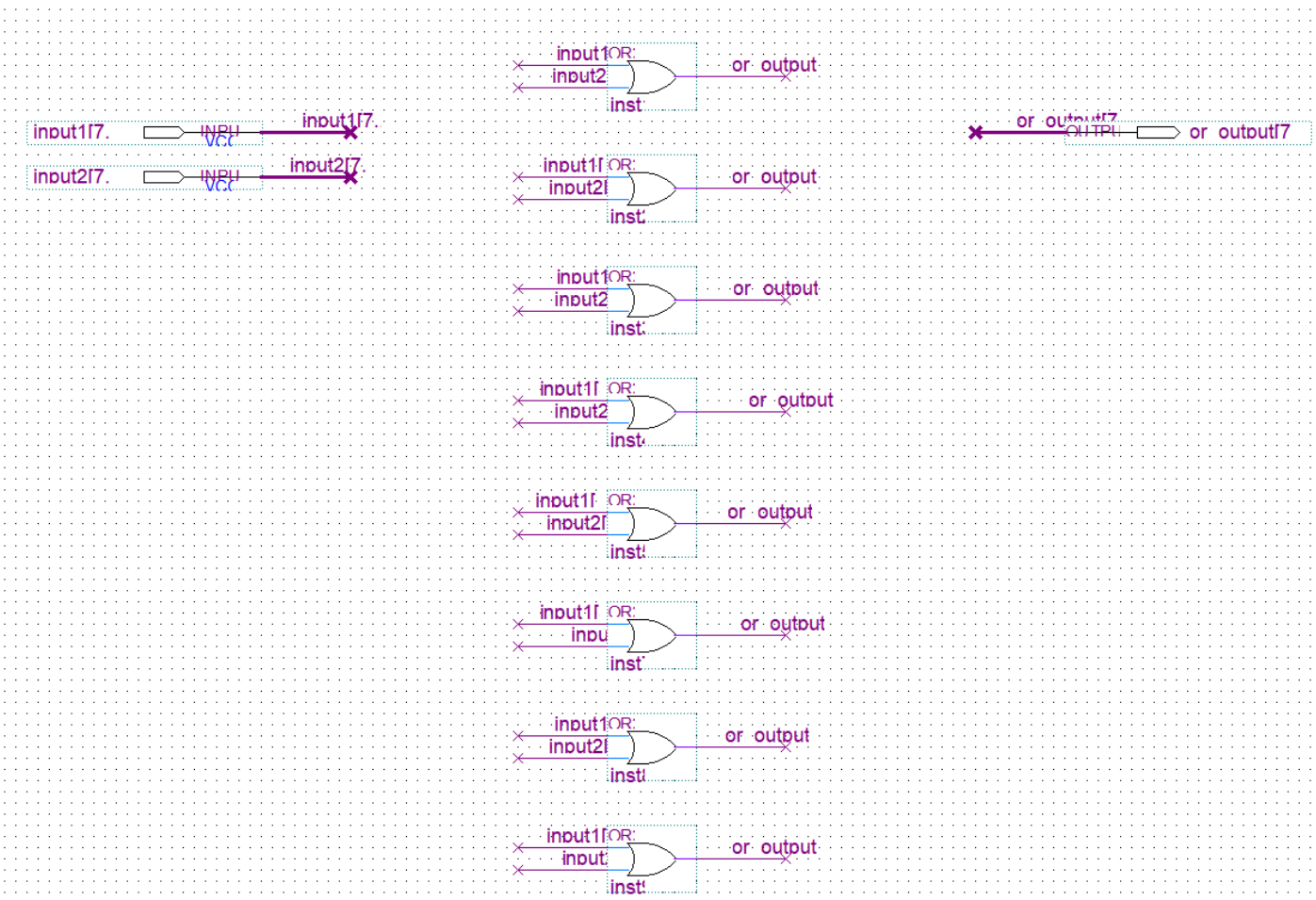
(آ) AND با $opcode = 0000$



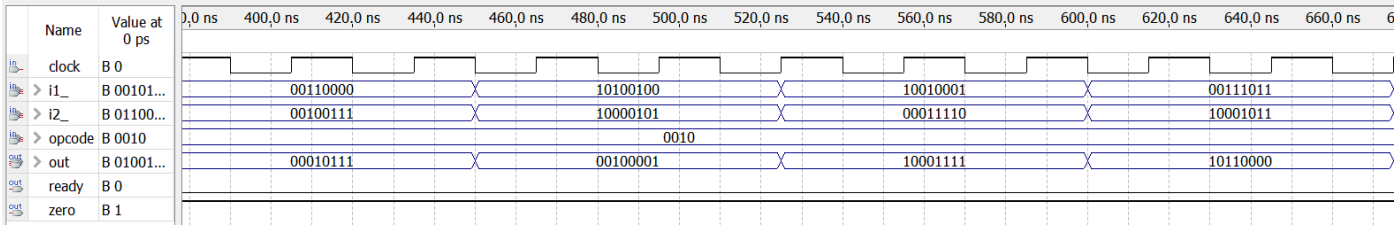


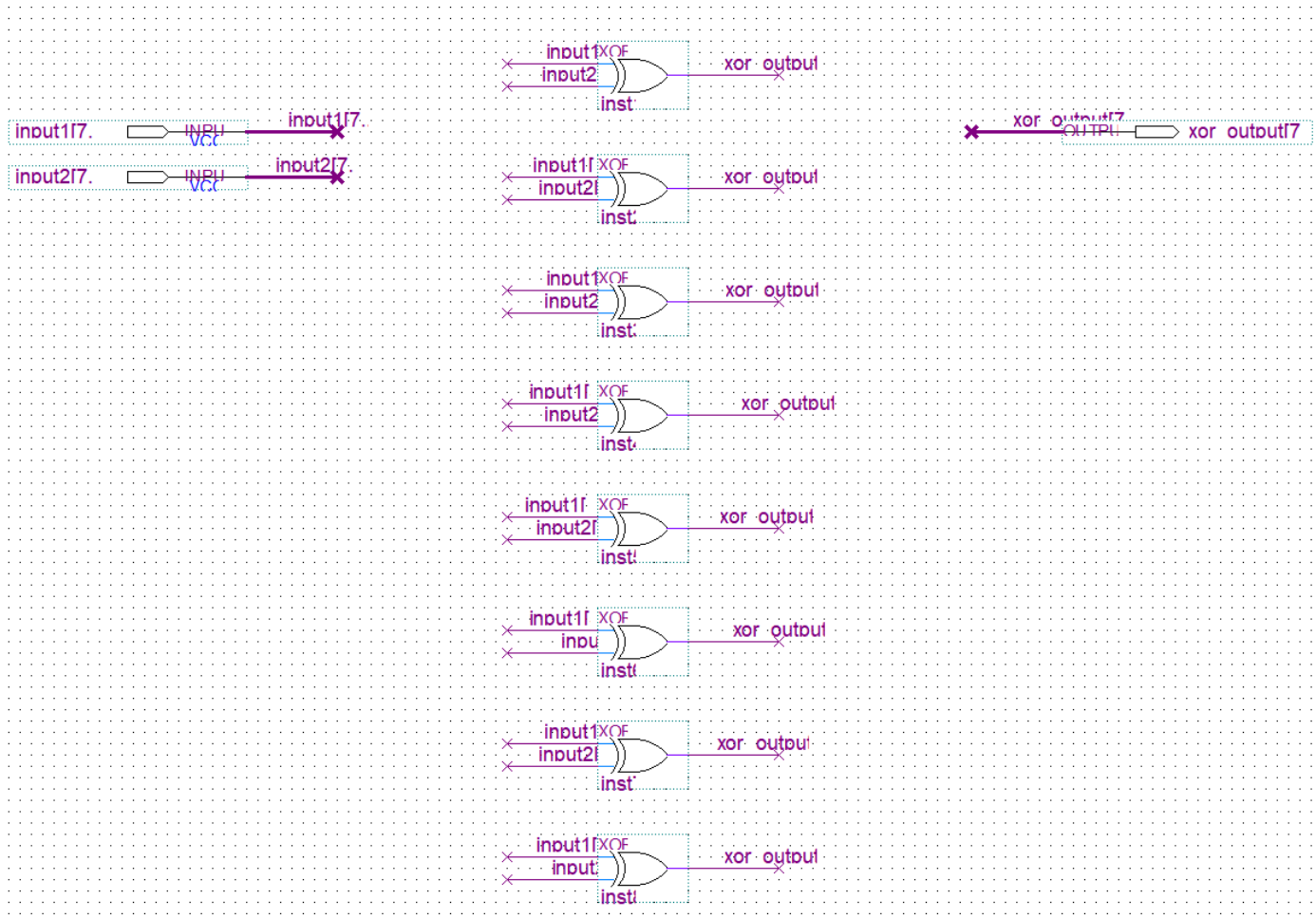
(ب) OR با $opcode = 0001$



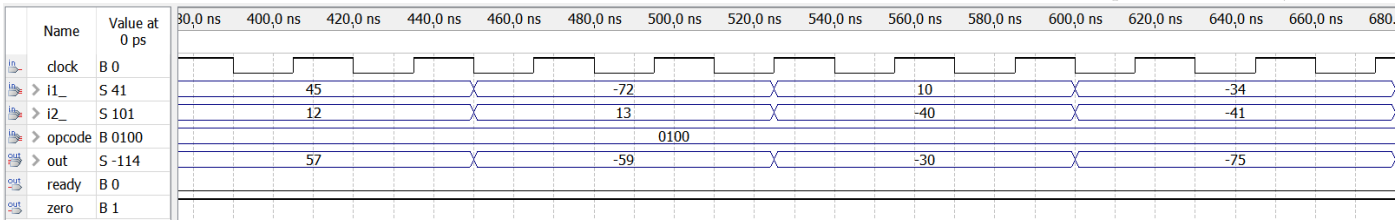


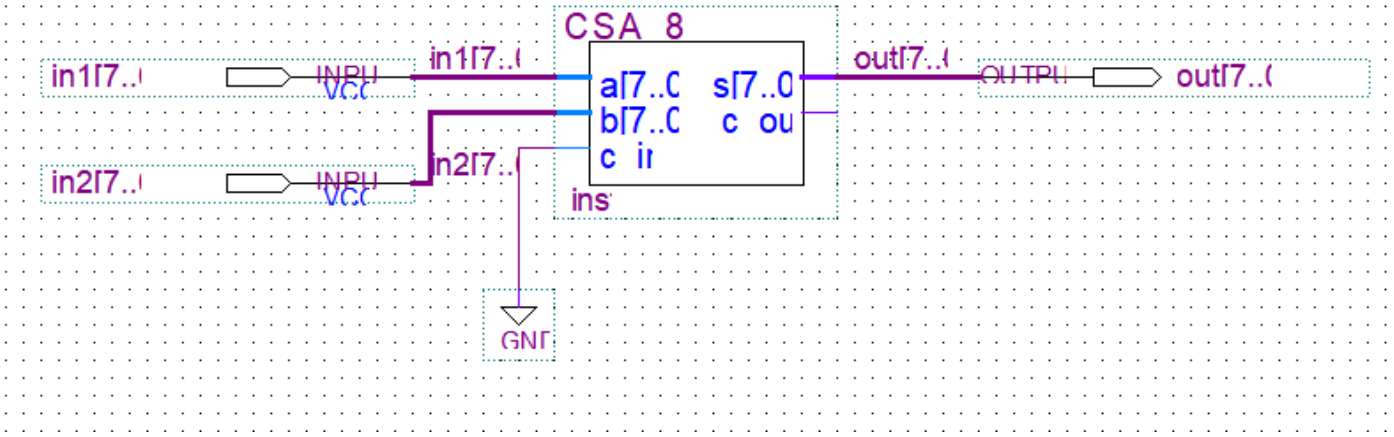
(ج) XOR با $opcode = ۰۰۱۰$



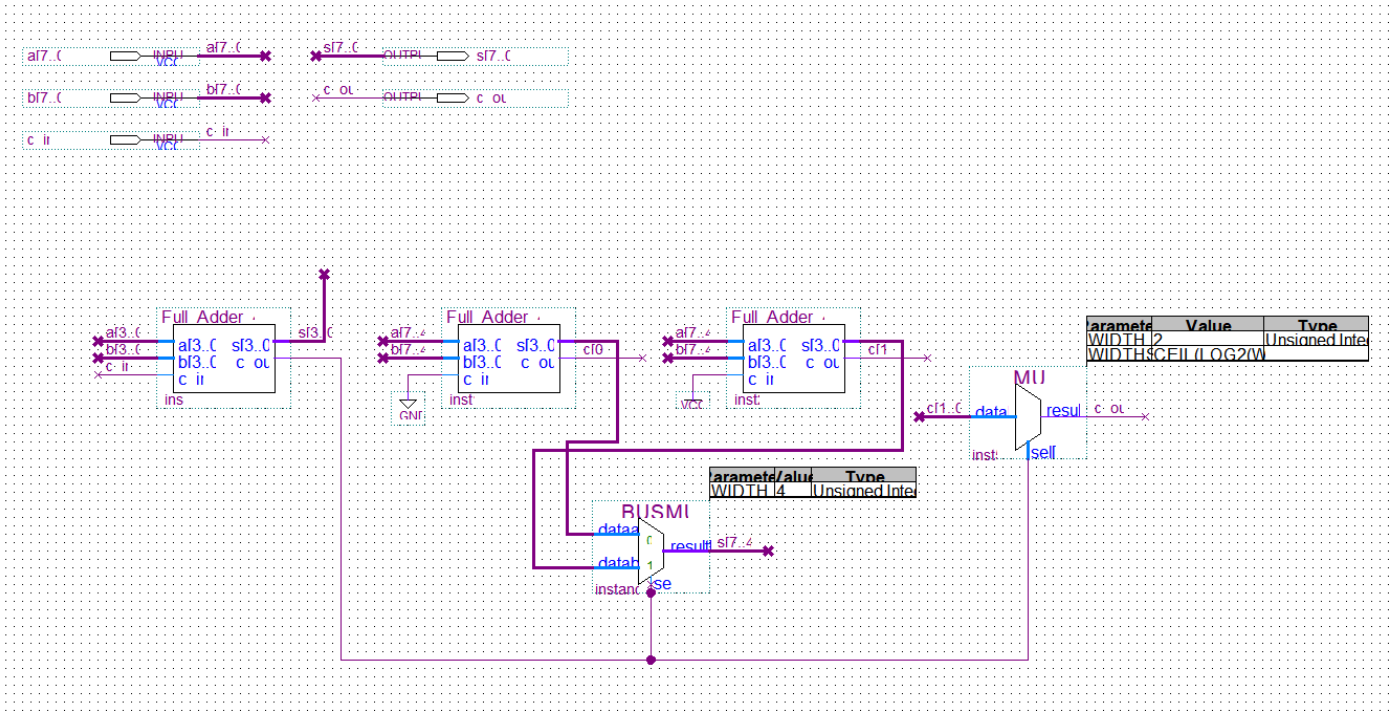


(د) add با $opcode = ۰۱۰۰$

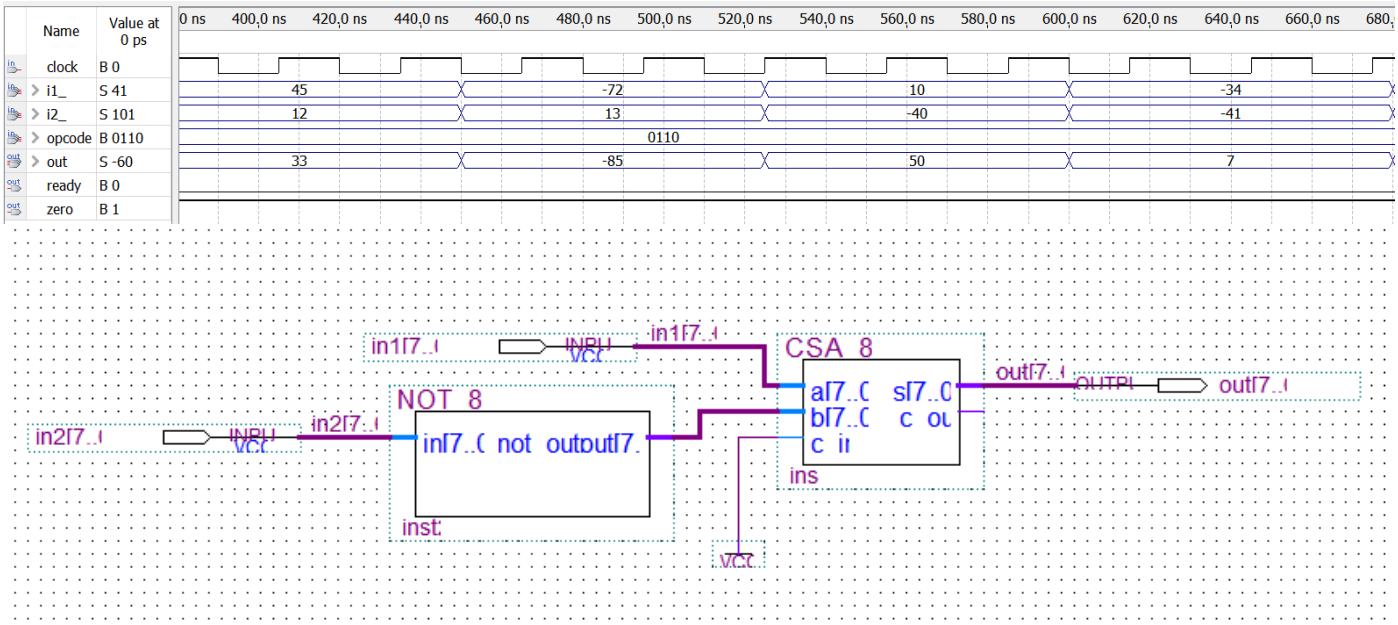




همانطور که گفته شده بود، جمع کننده ی ما با استفاده از carry save adder ساخته شده است.

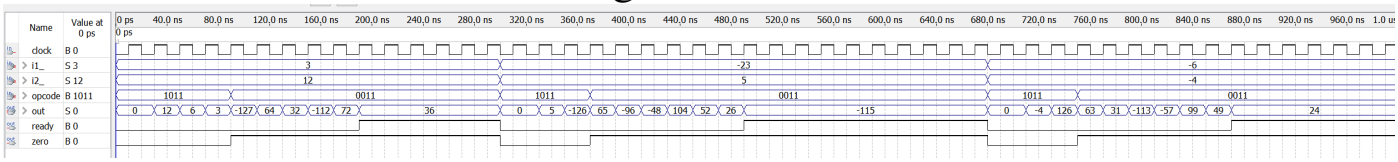


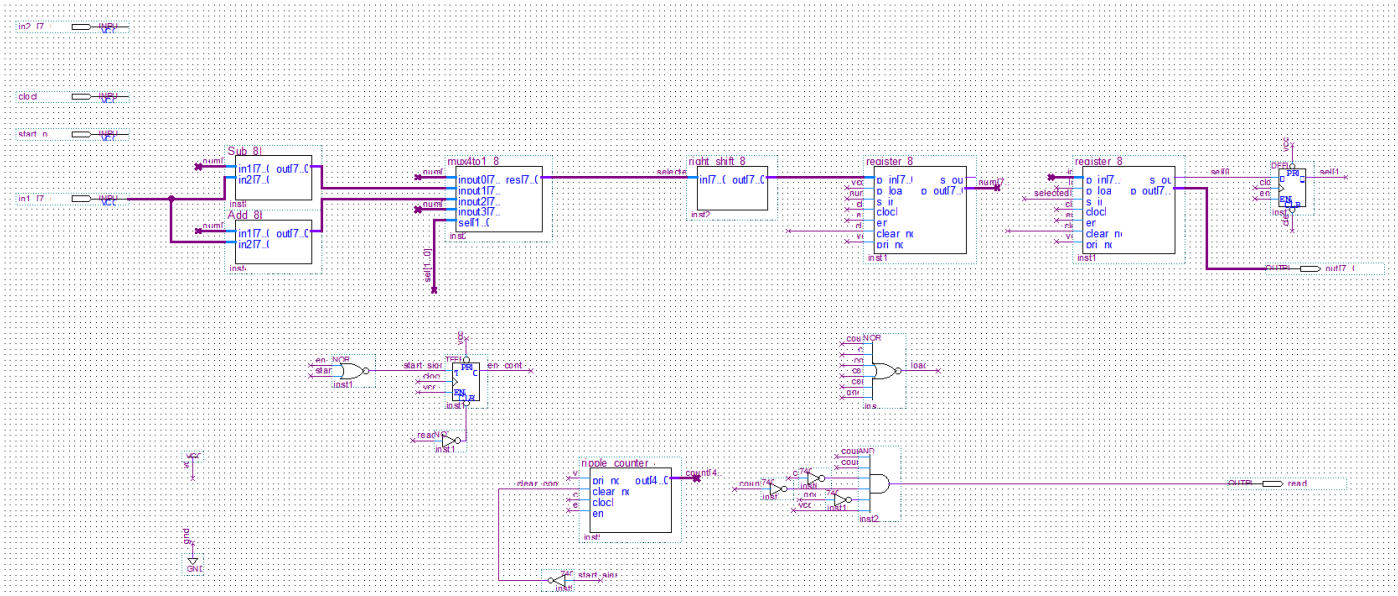
(ه) sub با $opcode = ۰۱۱۰$



تفریق کننده مان نیز عدد دوم را مکمل می کند و از همان جمع کننده ی قسمت قبل استفاده می کند.

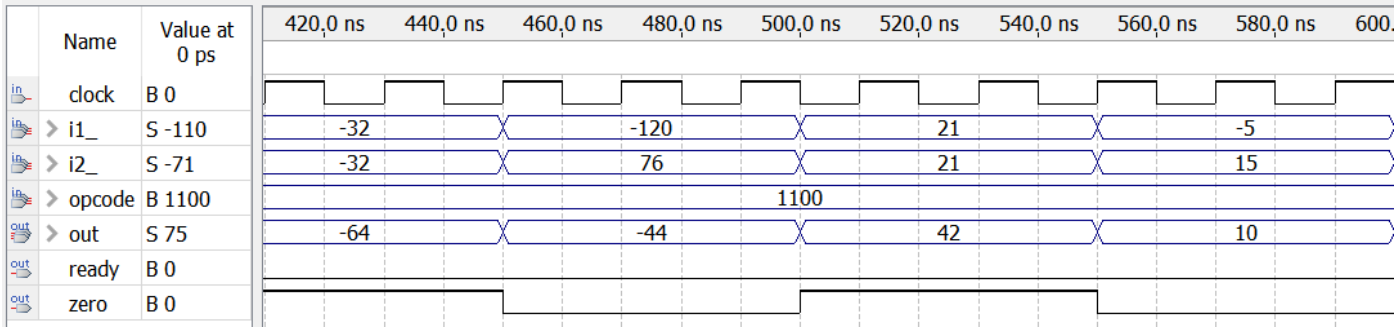
(و) multiply with booth's algorithm با $opcode = ۱۰۱۱$ و برای دیدن پاسخ $opcode = ۰۰۱۱$

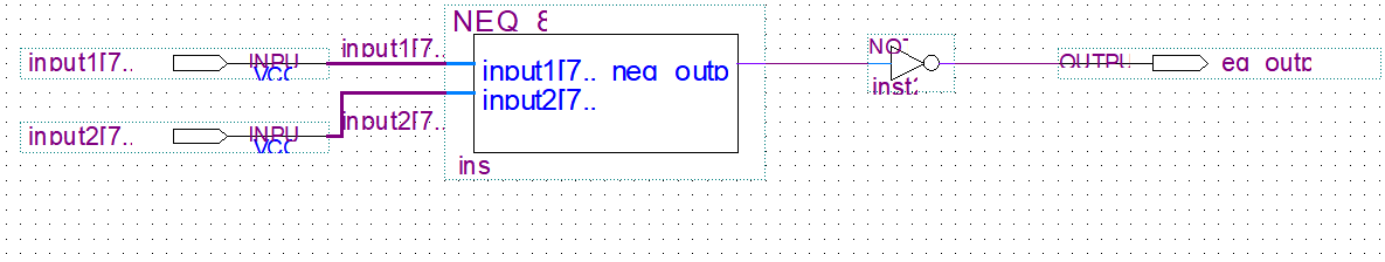




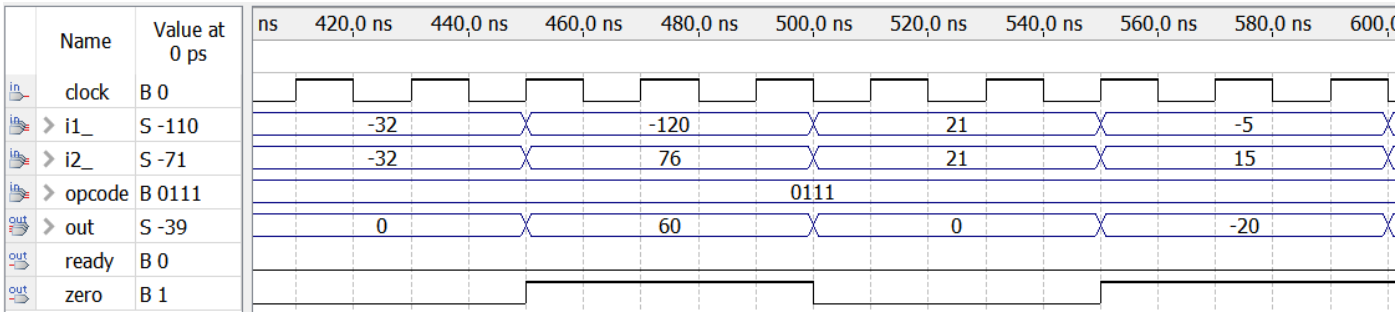
ضرب کننده ی ما با استفاده از الگوریتم booth زده شده است. برای ضرب کردن دو عدد، به شیفت رجیستر، ripple counter و جمع کننده و تفریق کننده ی قسمت های قبل نیاز داشتیم. این ضرب کننده وقتی سیگنال شروع را دریافت کنند، شروع به ضرب می کند تا زمانی که حاصل آماده شود و سیگنال ready فعال شود. وقتی این سیگنال فعال شد می توانیم ۸ بیت کم ارزش جواب را با فعال کردن $opcode = ۰۰۱۱$ بگیریم.

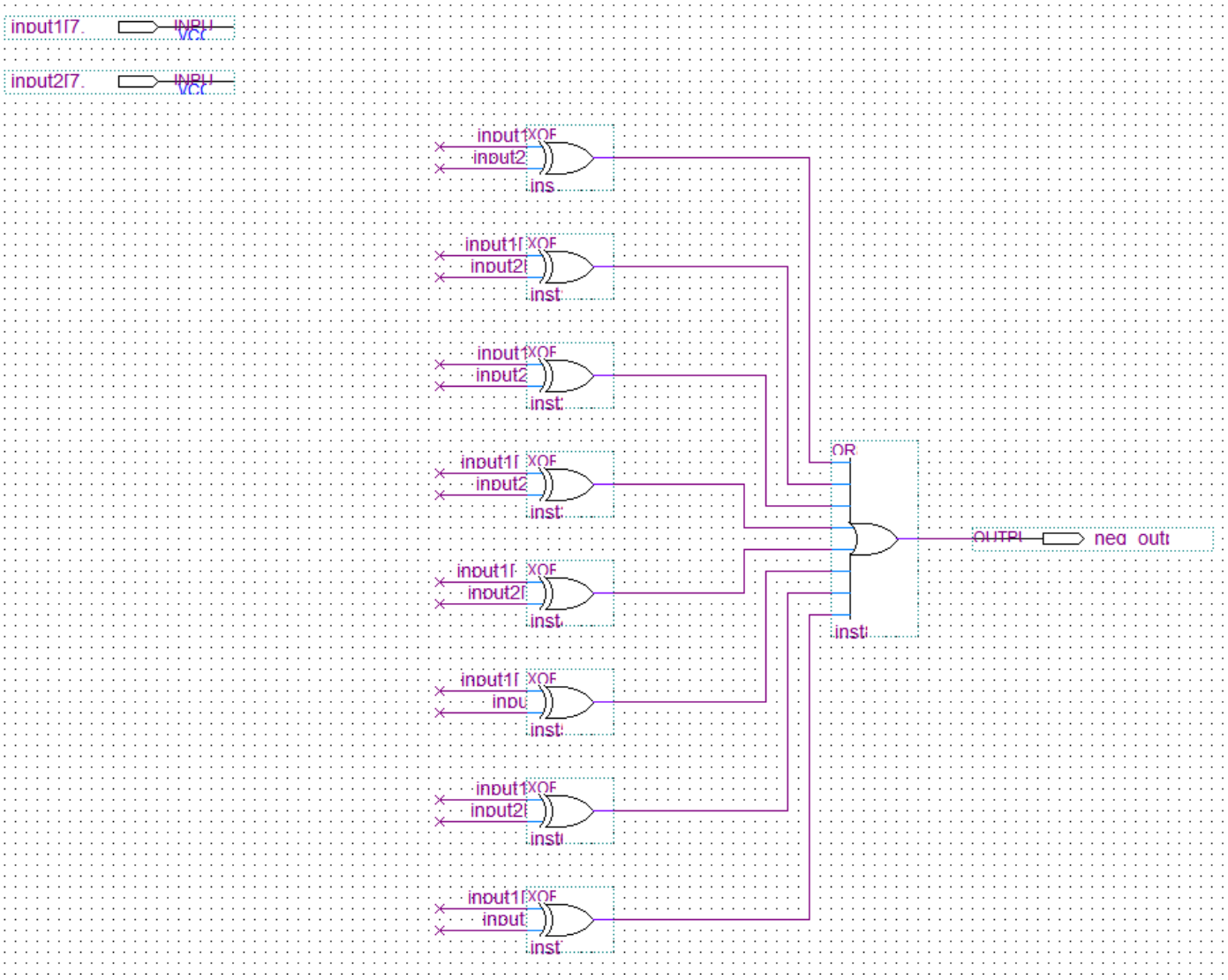
ز) equal با $opcode = ۱۱۰۰$





ح) not equal با $opcode = ۰۱۱۱$




input1[7..
INPU
VCC
input2[7..
INPU
VCC
input1
XOR
input2
ins
input1[
XOR
input2[
inst
input1
XOR
input2
inst
input1[
XOR
input2
inst
input1[
XOR
input2[
inst
input1[
XOR
inpu
inst
input1
XOR
input2[
inst
input1[
XOR
input
inst
OR
inst
OUTPU
neg_outi