به نام خدا

گزارش فاز ۱ پروژه

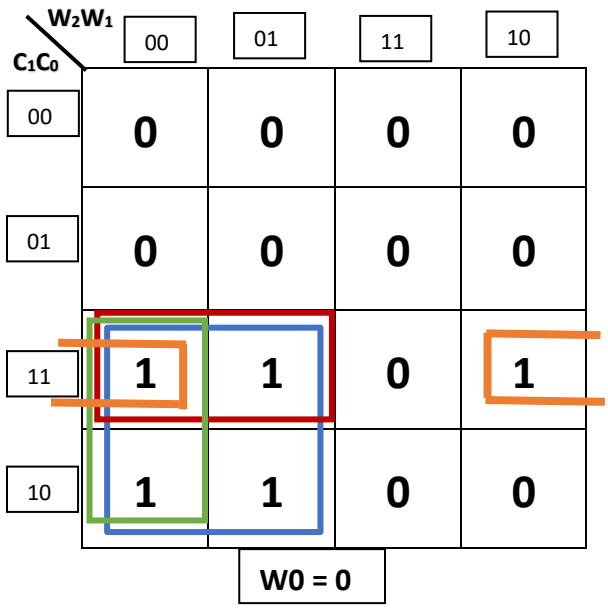
گروه پندار ربیعی رودســری(40331019) و دانیال ســیدی(40331022) از گروه ۴ آزمایشــگاه مدار منطقی استاد جوادزاده

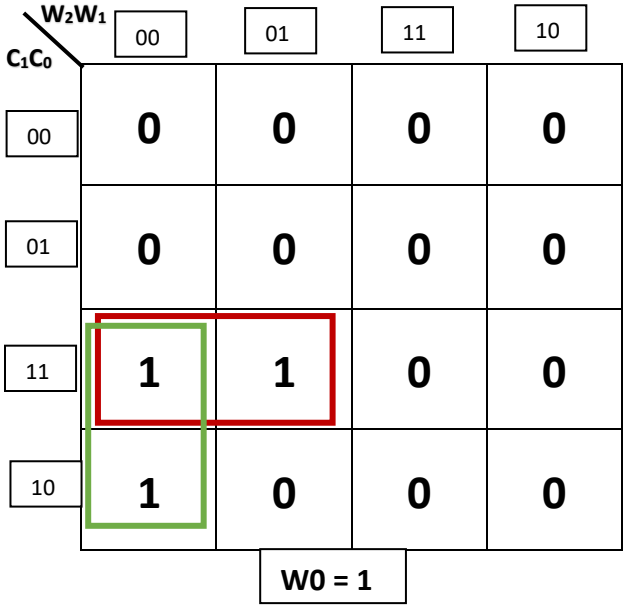
ابتدا جدول درستی مربوط به قسمت اول فرمول($\frac{Cal \times 60}{W}$) را رسم می کنیم که چون ۸ حالت برای ورودی W(مقدار وزن فرد) و ۴ حالت برای Cal(میزان کالری هدف) داریم، از مجموع این دو ورودی ۳۲ $= ۴ \times ۸$ حالت برای ورودی این قسمت خواهیم داشت. لذا باید یک جدول درستی با ۳۲ سطر رسم کنیم و پس از محاسبه مقدار نهایی طبق فرمول داده شده و تبدیل آن به نزدیکترین مقدار صحیح در صورت بدست آمدن اعشاری ، آن را به فرمت باینری(۸ بیتی چون بیشترین مقدار بدست آمده ۲۴۰ است که کمتر از ۲۵۶ می باشد.) تبدیل می کنیم. حال باید برای هر یک از ۸ بیت خروجی یک جدول کارنو ۵ متغیره رسم کنیم و پس از ساده سازی مدار ترکیبی مربوط به آن ها را رسم کنیم.

Cal(bits)	Cal	W(bits)	W(kg)	Cal×60 / W	Integer	Binary (8-bit)
00	50	000	50	60.00	60	00111100
01	100	000	50	120.00	120	01111000
10	150	000	50	180.00	180	10110100
11	200	000	50	240.00	240	11110000
00	50	001	60	50.00	50	00110010
01	100	001	60	100.00	100	01100100
10	150	001	60	150.00	150	10010110
11	200	001	60	200.00	200	11001000
00	50	010	70	42.86	43	00101011
01	100	010	70	85.71	86	01010110
10	150	010	70	128.57	129	10000001
11	200	010	70	171.43	171	10101011
00	50	011	80	37.50	38	00100110

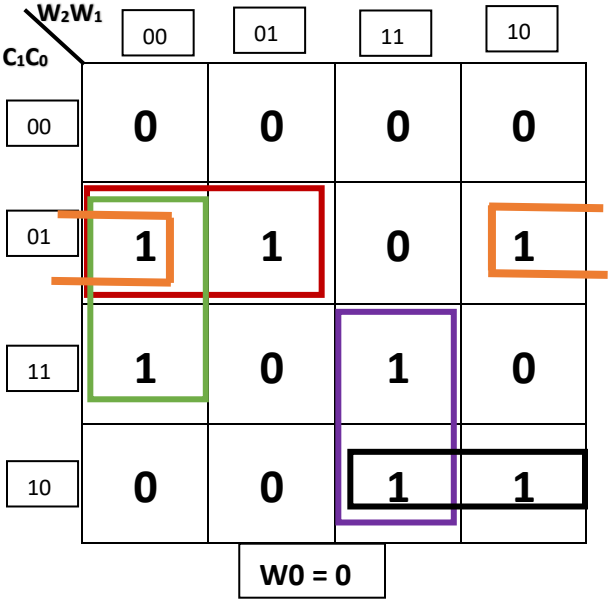
01	100	011	80	75.00	75	01001011
10	150	011	80	112.50	112	01110000
11	200	011	80	150.00	150	10010110
00	50	100	90	33.33	33	00100001
01	100	100	90	66.67	67	01000011
10	150	100	90	100.00	100	01100100
11	200	100	90	133.33	133	10000101
00	50	101	100	30.00	30	00011110
01	100	101	100	60.00	60	00111100
10	150	101	100	90.00	90	01011010
11	200	101	100	120.00	120	01111000
00	50	110	110	27.27	27	00011011
01	100	110	110	54.55	55	00110111
10	150	110	110	81.82	82	01010010
11	200	110	110	109.09	109	01101101
00	50	111	120	25.00	25	00011001
01	100	111	120	50.00	50	00110010
10	150	111	120	75.00	75	01001011
11	200	111	120	100.00	100	01100100

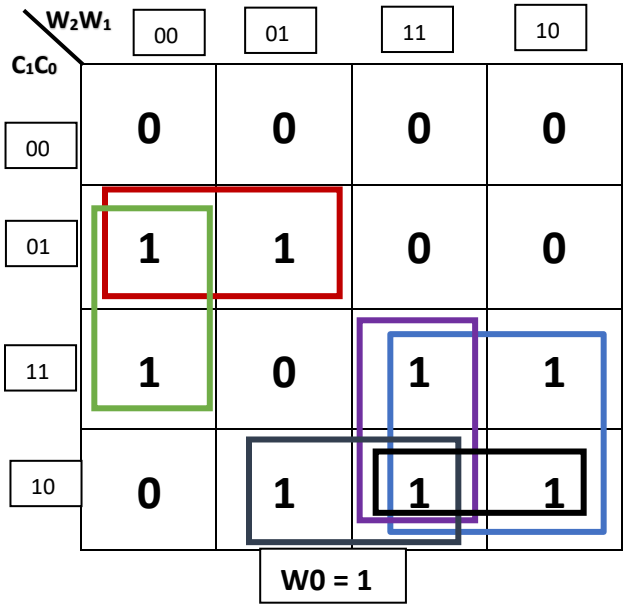
در ادامه تصاویر جدول کارنو های رسم شده برای بدست آوردن مدار ترکیبی مربوط به هر بیت از خروجی آمده است :
(بیت های خروجی به ترتیب از چپ به راست / پر ارزش به کم ارزش با O_8 تا O_0 نام گذاری شده اند) / $Cal = C_1C_0$
$W = W_2W_1W_0$



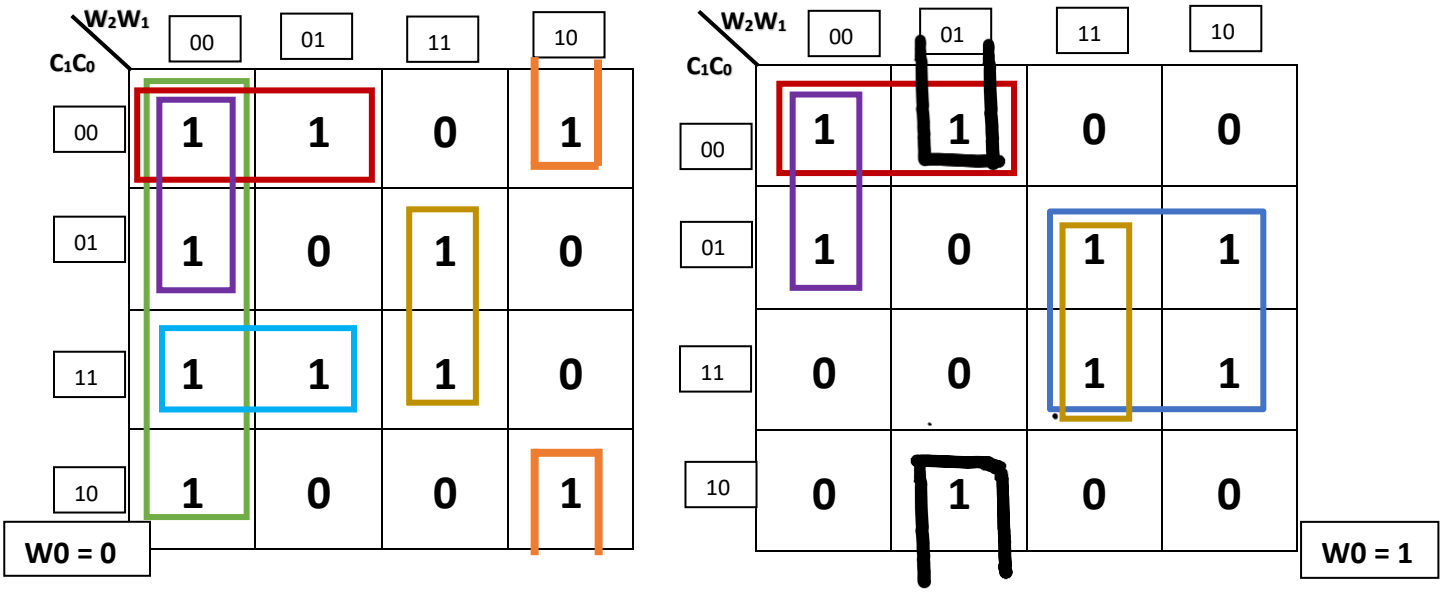


$$O_7 = C_1C_0W_2' + C_1W_2'W_0' + C_1W_2'W_1' + C_1C_0W_1'W_0'$$

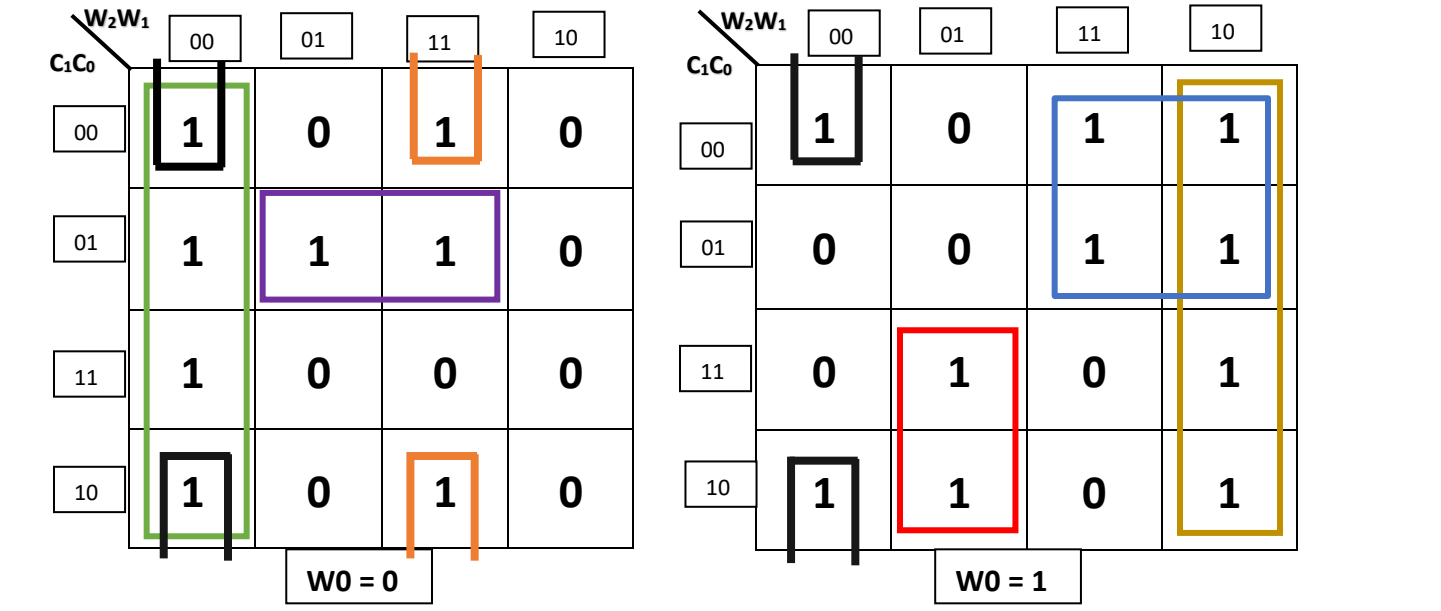




O_6 = C1'C0W2'+C1'C0W1'W0'+C0W2'W1'+C1W2W1+C1W2W0+C1C0'W2+C1C0'W1W0

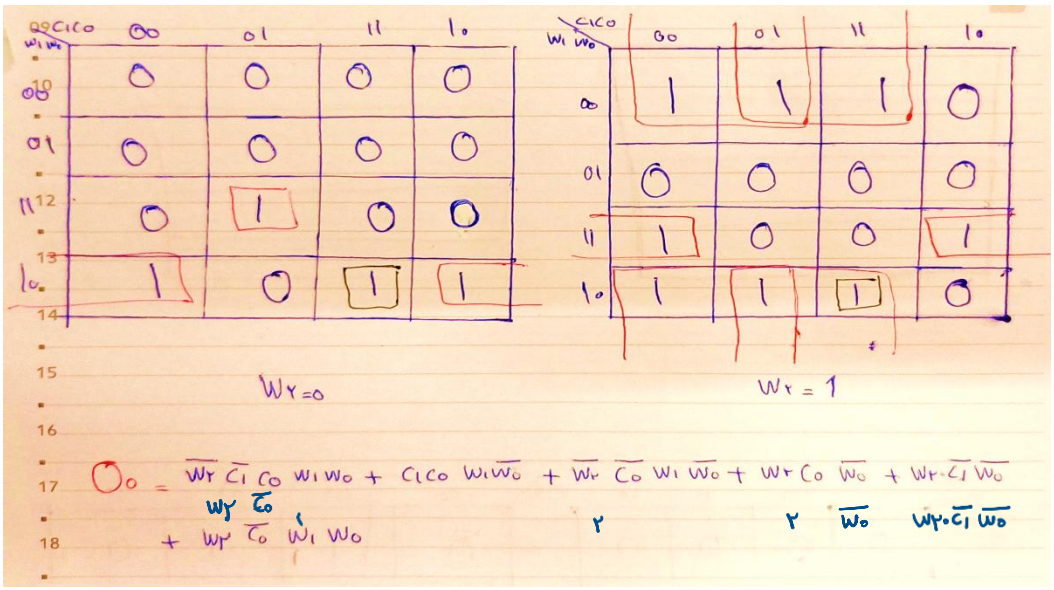


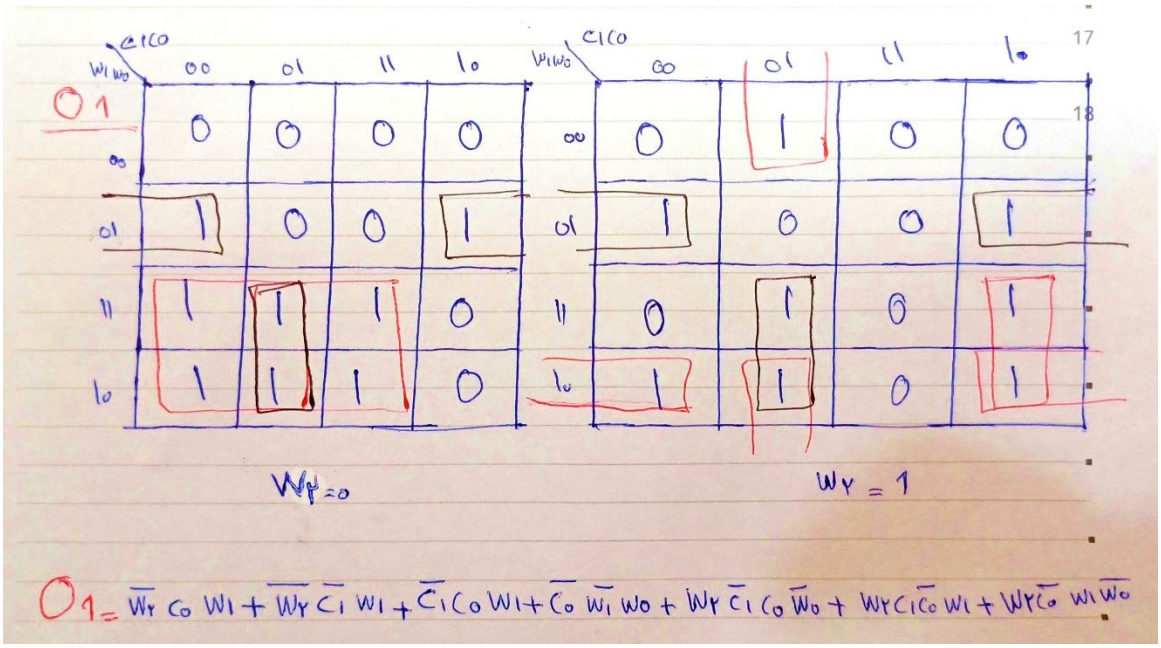
O_5=C1'C0'W2'+W2'W1'W0'+C1'W2'W1'+C1C0W2'W0'+C0W2W1+C0'W2W1'W0'+C0'W2'W1 W0+C0W2W0

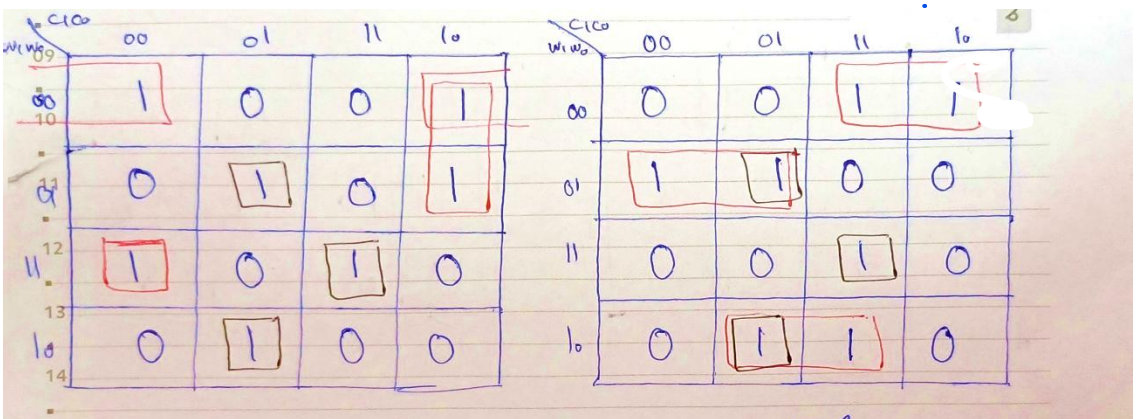


O_4 =

C0'W2'W1'+W2'W1'W0'+C1'C0W1W0'+C0'W2W1W0'+C1W2'W1W0+C1'W2W0+W2W1'W0

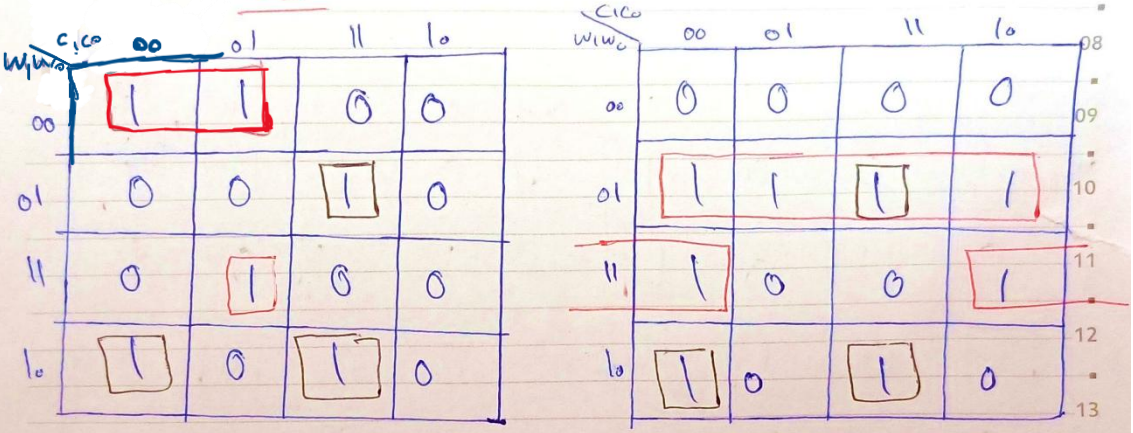






$$O_2 = \bar{C}_1 C_0 \bar{W}_1 W_0 + C_1 C_0 W_1 W_0 + \bar{C}_1 C_0 W_1 \bar{W}_0 + \bar{W}_4 C_1 C_0 \bar{W}_1 + \bar{W}_4 \bar{C}_0 \bar{W}_1 \bar{W}_0 + \bar{W}_4 \bar{C}_1 \bar{C}_0 W_1 W_0 + W_4 C_1 \bar{W}_1 \bar{W}_0 + W_4 C_0 W_1 \bar{W}_0 + W_4 \bar{C}_1 \bar{W}_1 W_0$$

O_3

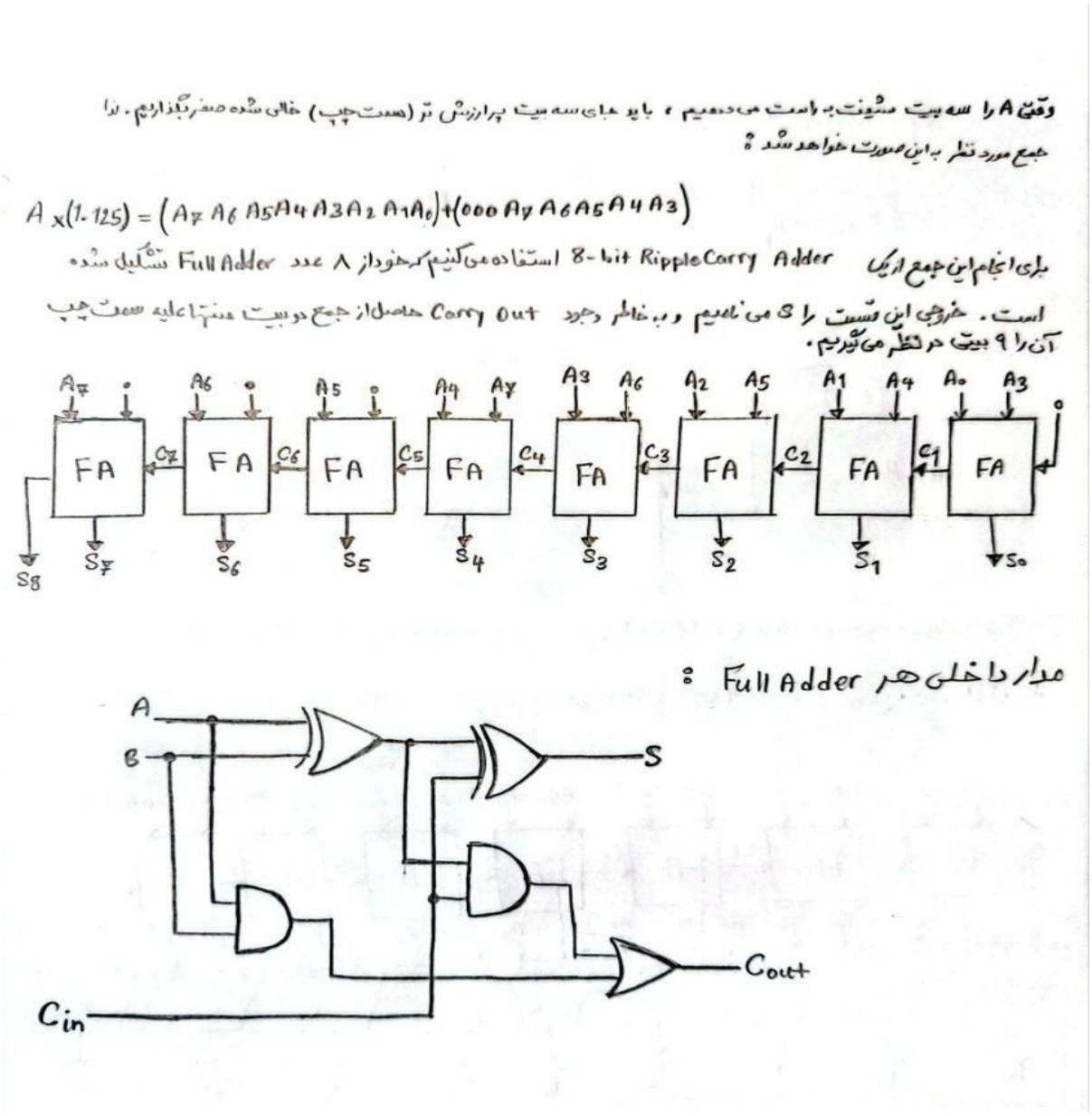


$$O_3 = W_4 \bar{W}_1 W_0 + W_4 \bar{C}_0 W_1 W_0 + C_1 C_0 \bar{W}_1 W_0 + C_1 C_0 W_1 \bar{W}_0 + \bar{C}_1 \bar{C}_0 W_1 W_0 + \bar{W}_4 \bar{C}_1 C_0 W_1 W_0 + \bar{W}_4 \bar{C}_1 \bar{W}_1 \bar{W}_0$$

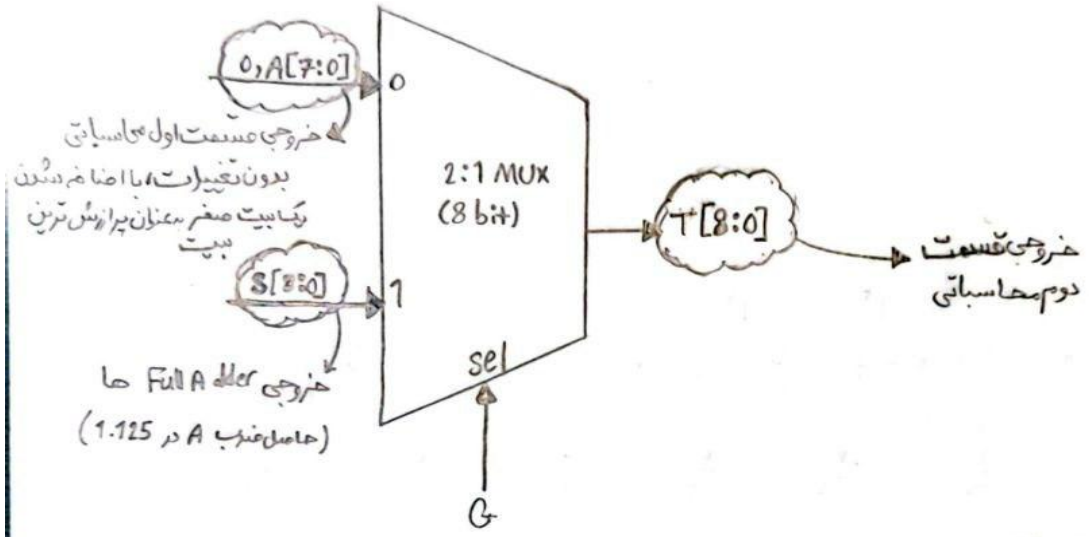
تا به اینجا با رســـم جدول کارنو مربوط به هریک از ۸ بیت ورودی رابطه مربوط به هریک از بیت های خروجی را به فرم SOP بدست آوردیم.

حال به سراغ طراحی مدار مربوط به بخش های محاسباتی دوم(ضرب خروجی بخش اول در G) و سوم(ضرب خروجی قسمت دوم در $\frac{1}{MET}$) می رویم :

خروجی قسمت محاسباتی اول را که ۸ بیت دارد ، A(از A[0] تا A[7]) می نامیم.

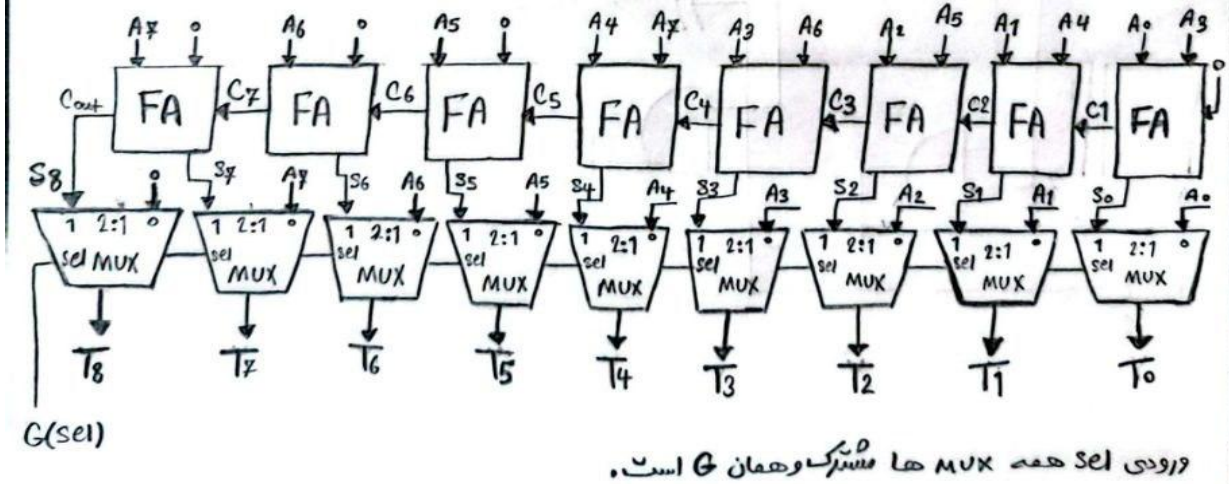


حال برای تولید حالت های بین $G=0$ و $G=1$ از یک 2:1 MUX استفاده می کنیم :



در اصل برای تولید هر بیت خروجی یک 2:1 MUX (در مجموع 9 تا 2:1 MUX) که ما همه این MUX ها را به صورت یک 2:1 MUX بزرگ در نظر گرفته ایم که ورودی هایش به صورت آرایه های 9 بیتی هستند.

مدار کامل به صورت زیر خواهد بود :



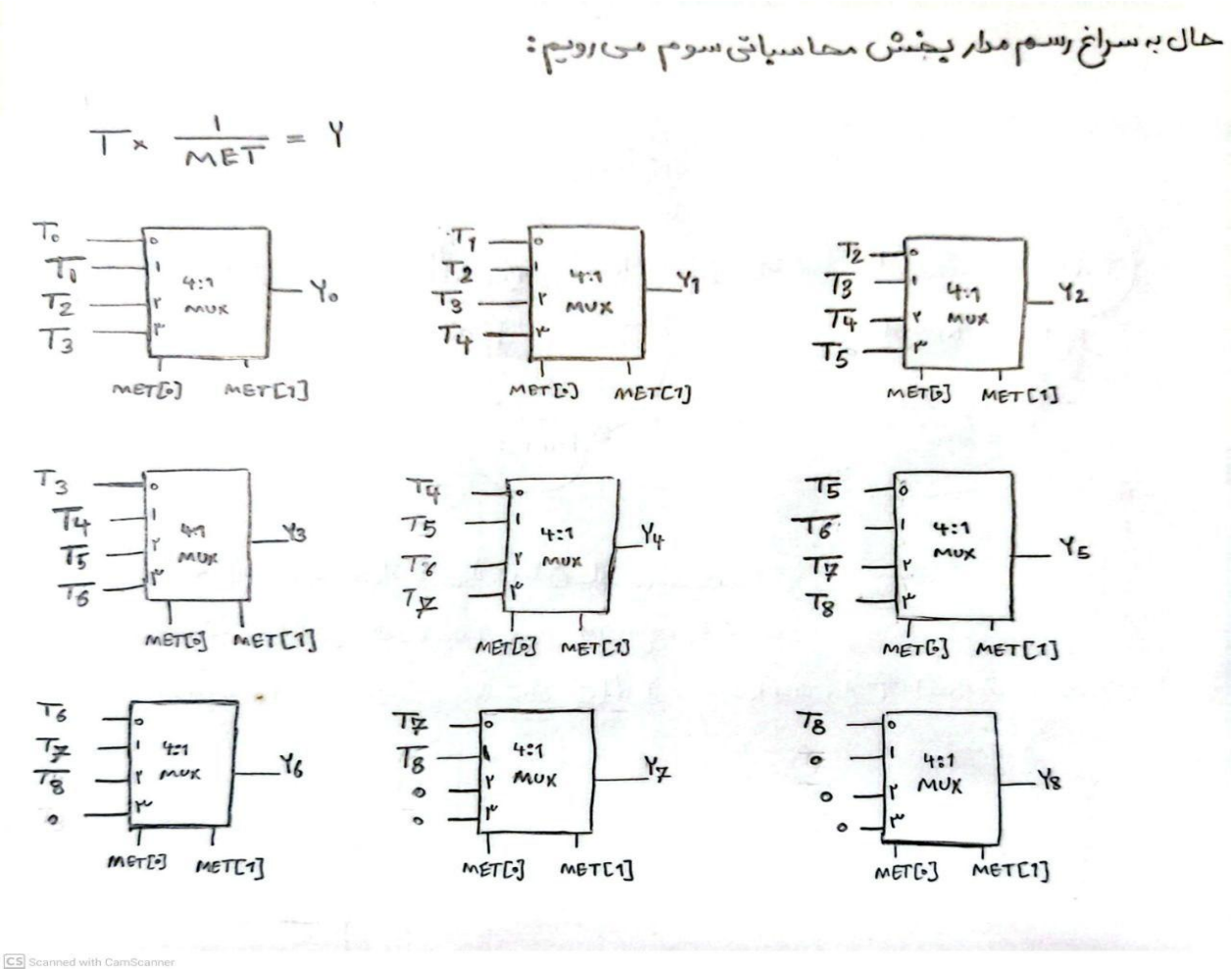
ورودی sel همه MUX ها مشترک و همان G است.

برای مدار قسمت محاسباتی سوم($\times \frac{1}{MET}$) ، همانطور که در توضیحات زیر آمده از ۹ MUX (یک MUX برای هر بیت)استفاده می کنیم :

(در کد وریلاگ مربوط به این قسمت خروجی T ، ۸ بیتی در نظر گرفته شده و فرض شده که ورودی که عدد بیشتر از ۸ بیت تولید کند داده نمی شود.)

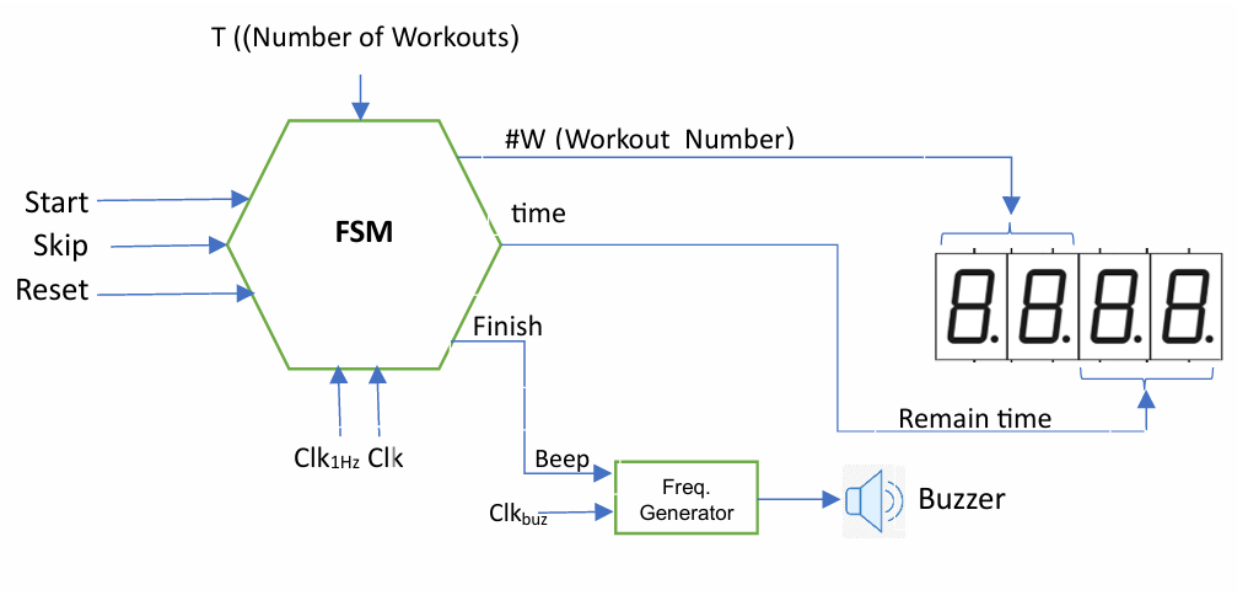
خروجی این بخش را T نامیده ایم(از T_0 تا T_8) :

عملگر شیفت	MET
بدون شیفت	**00**
یک بار شیفت به راست	**01**
دو بار شیفت به راست	**10**
سه بار شیفت به راست	**11**

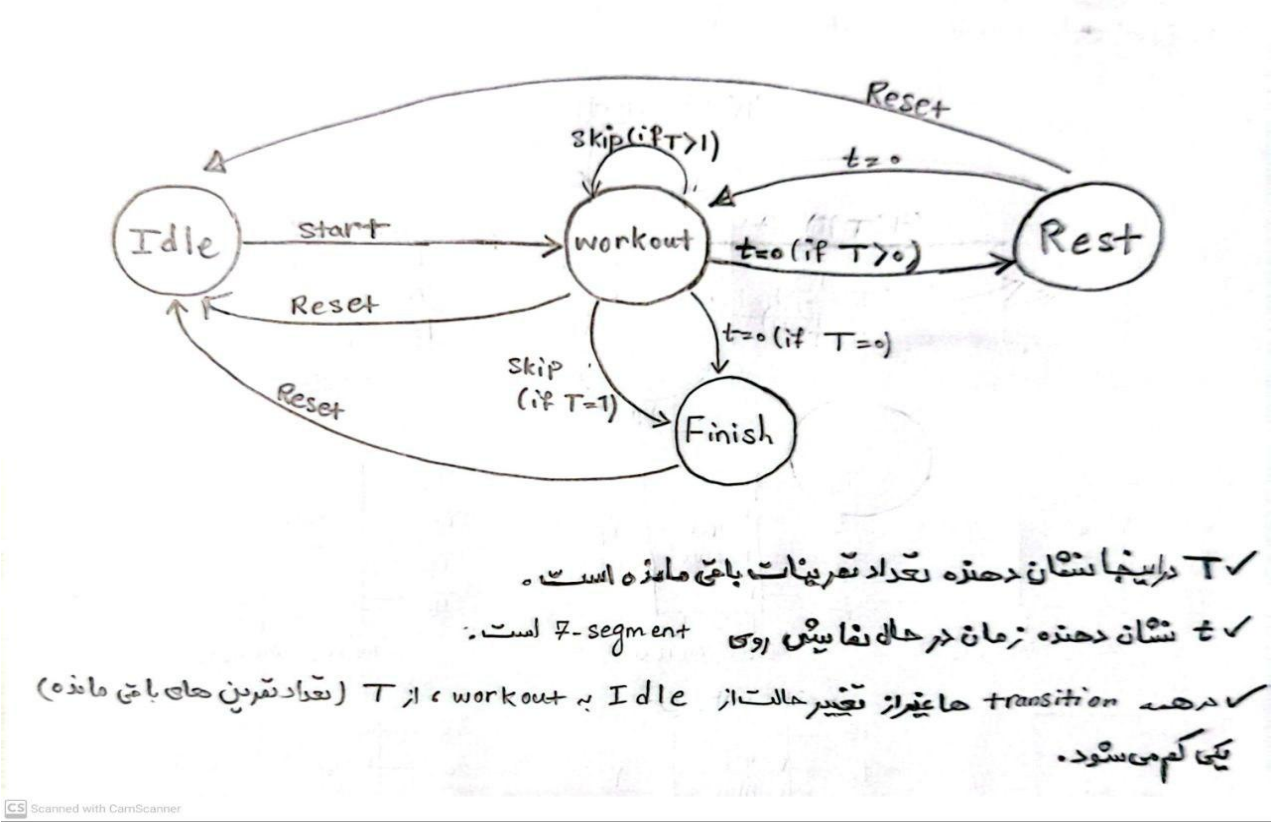


همانطور که در توضیحات پروژه گفته شده ، ضرب در 1.125 را با جمع عدد با سه بار شیفت به راست شده خودش و ضرب در $\frac{1}{MET}$ که همه توان های دو هستند را با اعمال تعداد مناسب شیفت به راست اعمال کرده ایم.

حال به سراغ بخش زمان بندی ترتیبی می رویم :



شکل 3 - ماشین حالت



در این ماشین حالت ۴ حالت(State) داریم که شامل Idle(در انتظار زدن دکمه start برای شروع) ، Workout(حالت شمارش ۴۵ ثانیه مربوط به هر تمرین) ، Rest(حالت شمارش ۱۵ ثانیه در استراحت بین تمرین ها) و Finish(حالت اتمام همه T تمرین برنامه ریزی شده).

توضیحات مربوط به نحوه کار ماشین حالت :

در شروع کار ماشین ، ماشین به صورت خودکار در حالت Idle قرار می گیرد. سپس اگر کاربر دکمه start را فشار دهد ماشین تعداد حرکات ورزشی مورد نیاز(T) را از کاربر گرفته و به حالتworkout می رود.

در حالت Workout شمارش ۴۵ ثانیه مربوط به تمرین آغاز می شود. اگر کاربر قبل از اتمام زمان تمرین دکمه Skip را بزند و تمرینی که آن را skip کرده آخرین تمرین نبوده باشد به تمرین بعدی می رود و شمارش ۴۵ ثانیه دوباره آغاز شده و از تعداد تمرین های باقی مانده یکی کم می شود و اگر دکمه skip را بزند و تمرین که آن را skip کرده آخرین تمرین بوده باشد(T=1) مستقیما به حالت Finish می رود.

اگر زمان تمرین به پایان برسد(t=0) و هنوز تمرینی باقی مانده باشد(T>0) ، ماشین به حالت Rest می رود و شمارش ۱۵ ثانیه مربوط به استراحت شروع می شود. ولی اگر این حالت رخ دهد و تمرین دیگری نیز باقی نمانده باشد(T=0) ماشین به حالت Finish می رود و جلسه تمرینی به اتمام می رسد.

در حالت Rest ، به علت اینکه وقتی به حالت Rest رفته ایم همیشه به این معنا است که حداقل یک تمرین دیگر نیز باقی مانده است ، ماشین با تمام شدن تایمر استراحت(t=0)، همواره به حالت Workout بر می گردد و شمارش ۴۵ ثانیه مربوط به تمرین بعدی را آغاز می کند.

همچنین اگر ماشین در هر یک از حالات Rest،Workout یا Finish باشد ، با زدن دکمه Reset توسط کاربر دوباره به حالت Idle بر می گردد و دوباره منتظر فشردن دکمه start توسط کاربر برای شروع می ماند.

کد وریلاگ قسمت های خواسته شده به همراه این فایل گزارش pdf در فایل فشرده آپلود شده آورده شده است.

توضیحات مربوط به کد وریلاگ combinational circuit :

کد این قسمت مربوط به بخش مدار ترکیبی تولید کننده مقدار T بر اساس ۸ بیت ورودی دارای مقادیر W وCal و MET و G است.

W			Cal		MET		G
0	0	1	0	1	1	0	1

در اینجا این ورودی ۸ بیتی را input_bits نامیده ایم و خروجی این قسمت را نیز T3 در نظر گرفته ایم.

ابتدا مقادیر مربوط به هریک از ۴ متغیر نامبرده شـده از ورودی ۸ بیتی مورد نظر اسـتخراج می شـود و در wire های مربوطه ریخته می شود.

```
wire w2 = input_bits[7];
wire w1 = input_bits[6];
wire w0 = input_bits[5];
wire c1 = input_bits[4];
wire c0 = input_bits[3];
wire m1 = input_bits[2];
wire m0 = input_bits[1];
wire G  = input_bits[0];
```

سپس T1 نیز از نوع wire تعریف می شود که خروجی قسمت اول محاسباتی($\frac{Cal \times 60}{W}$) را در خود نگاه می دارد.(آن را ۸ بیتی در نظر گرفته ایم ، چون مقدار خروجی این قسمت در مبنای ۱۰ از ۰ تا ۲۴۰ تغییر می کند و ۸ بیت می تواند از ۰ تا ۲۵۵ را پوشش دهد که آن را به گزینه ای مناسب تبدیل می کند.)

سـپس مدار های به فرم SOP مربوط به هریک از ۸ بیت مربوط به این قسـمت را که در اول گزارش با اسـتفاده از جدول های کارنو ۵ متغیره بدست آمده بود را با استفاده از & و | برای گیت های and و or شبیه سازی می کنیم.

پس از مقدار دهی T1[0] تا T1[7] به سـراغ قسـمت دوم محاسـباتی(G×) می رویم. برای حالت G=1(حالت مربوط به بانوان) باید به جای ضـرب در 1.125، عدد مورد نظر را سـه واحد به راسـت شـیفت می دادیم و سـپس با خودش جمع می کردیم.

عدد سه بار به راست شیفت داده شده را در T1_shifted ذخیره کرده ایم و به جای سه بیت سمت چپ(پر ارزش تر) آن صفر قرار داده ایم.

همچنین خروجی مربوط به حالت بانوان این قسمت(که باید در ۱.۱۲۵ ضرب شود) را در T1_woman ریخته ایم.

در carry نیز carry-out های مربوط به جمع هر بیت توسـط Full Adder را ریخته ایم و چون تعداد carry-out ها از تعداد بیت های جمع شونده ها یکی بیشتر می شود ، carry را ۹ بیتی در نظر گرفته ایم.

سـپس با اسـتفاده از مدل full_adder ای که در پایین تعریف کرده ایم ، این ۸ بیت را با ۸ بار اسـتفاده از full_adder با هم جمع کرده ایم.

در نهایت خروجی مربوط به این قسـمت دوم محاسـباتی را با اسـتفاده از MUX های دو ورودی برای هر بیت(که در اینجا به جای تعریف module مربوط به MUX دو ورودی از and وor برای پیاده سـازی دسـتی آن اسـتفاده کرده ایم) ایجاد

کرده ایم و در نهایت آن ها را در T2 ذخیره کرده ایم.(به این صـــورت که اگر G=1 باشـــد ، خروجی ذخیره شـــده در T1_woman منتقل می شـود(که همان T1 بود که در 1.125 ضـرب شـده بود) و اگر G=0 باشـد همان ورودی T1 به صورت مستقیم به خروجی این قسمت انتقال داده می شود چرا که با ضرب شدن در 1 تغییری در مقدار ان ایجاد نمی شد.

در ادامه خروجی T2 نیز برای قسمت محاسباتی سوم($\times \frac{1}{MET}$) ، به ترتیب در wire های shift0،shift1،shif2،shift3 برای حالت های MET=1 و MET=2 و MET=4 و MET=8 ذخیره می شود(به جای استفاده از عملگر ضرب این کار با انجام تعداد مناسب شیفت به راست روی ورودی انجام می شود.)

در نهایت همانطور که در مدار رسم شده برای این قسمت پیشتر نشان داده شد(هر چند که آنجا تعداد بیت ها را ۹ تا در نظر گرفته بودیم برخلاف اینجا که ۸ تا در نظر گرفته ایم.)، ورودی های مناسب این قسمت را به ۸ تا MUX ۴ به ۱ داده ایم و m1 و m0 را نیز که همان MET[1] و MET[0] هستند را نیز به عنوان ورودی select همه MUX ها داده ایم.

در آخر کد این قسمت نیز همان module های مربوط به Full Adder و MUX ۴ به ۱ که پیشتر از آنها استفاده کرده ایم تعریف شده است.

حال به سراغ توضیحات مربوط به کد بخش دوم(fsm) می رویم :

در این برنامه ابتدا ورودی های clk, start, skip, reset, time_done و T تعریف شده اند که به ترتیب مربوط به نشان دهنده پایان زمان شـمارش ، دکمه های reset و skip و start برای تعامل کاربر با ماشـین و ورودی سـاعت مربوط به مدار ترتیبی و همچنین ورودی T تولید شـده توسـط مدار قسـمت قبلی که نشـان دهنده تعداد حرکات مورد نیاز برای انجام است.

همچنین output ها نیز در این قسمت به صـورت reg تعریف شـده اند که در طول ادامه توضـیحات به توضـیح آنها نیز خواهیم پرداخت.

ابتدا ۴ حالتی که در state diagram ترسـیم شـده در گزارش تعریف کرده بودیم ، اینجا نیز به صـورت enum تعریف شـــده اند(که شـــامل START , WORKOUT, REST, FINISH هســـتند) و دو state با نام های current_state و next_state نیز از نوع این enum تعریف شـده اند تا به ترتیب حالتی که ماشـین اکنون در آن قرار دارد و حالت بعدی که قرار است به آن برود را ذخیره می کنند.

سپس reg با نام count نیز برای شمارش تعداد حرکات باقی مانده تعریف شده است(با همان ۸ بیت)

در ادامه در یک always block که به تغییرات همه ورودی ها حسـاس اسـت، transition های همه state ها به همان صـورت که پیشتر در توضیح state diagram آمده بود انجام شده است. به این صـورت که روی این که هم اکنون روی کدام state هسـتیم(current_state مان چیسـت) حالت بندی کرده ایم و همه transition های ممکن را در نظر گرفته ایم.

برای توضـیحات مربوط به این قسـمت از کد خواننده را به توضـیحات مربوط به نحوه کار state daigram ارجاع می دهیم.

در ادامه نیز یک always block دیگر حســاس به لبه clock و reset آورده شـــده که بررســـی می کند که در هر لبه بررســـی می کند که اگر دکمه reset فشـــار داده شـــده بود به حالت IDLE بر گردد و تعداد تمرین ها را دوباره به T بر گرداند. همچنین در ادامه گفته شده که با هر بار پایان یک تمرین(چه با فشردن دکمه skip و چه با اتمام زمان تمرین) یکی از تعداد تمرین های باقی مانده(count) کم شود.

در ادامه نیز با آوردن یک always block دیگر ، ابتدا همه خروجی ها را صـفر می کنیم و state خروجی را نیز برابر بـا متغیرcurrent_state قرار می دهیم. و اگر در state هـای WORKOUT یـا REST بـاشـــیم ، مقـدار متغیر هـای start_timer و show_time را ۱ می کنیم (یعنی تایمر شروع به کار کرده و زمان در حال نمایش است) و با پایان هر تمرین و رفتن به حالت REST مقدار متغیر workout_done_beep را ۱ می کنیم که یعنی صـــدای beep مربوط به پایان یک حرکت باید پخش شـود و با هربار برگشـت دوباره به حالت WORKOUT مقدار این متغیر را دوباره صـفر نی کنیم. همچنین اگر به حالت FINISH رفته باشـیم مقدار متغیر های finish_beep وdone را ۱می کنیم که نشـان می دهد تمرین های تمام شده و صدای beep مربوط به پایان کل تمرین ها باید از buzzer پخش شود.