بسمه تعالی

درس طراحی سستم های دیجیتال

دکتر احمدوند

پروژه نهایی

فرشاد سلامت

۹۷۰۴۷۲۳

علیرضا دلاوری

۹۹۰۷۵۳۳

دانشگاه صنعتی همدان

ترم ۹۹۲

راهبرد کلی:

این پروژه را سعی کرده ایم بر اساس رفتار سیستم نهایی پیاده سازی کنیم و در نتیجه موفق شدیم تا یک طراحی behavioral از این سیستم ارائه کنیم که در بخش های بعدی به توضیح جز به جز این طراحی می پردازیم.

توضیح پروژه:

در این پروژه قرار است در زمان های خاصی که به صورت ورودی(INST1 و INST2) به سیستم داده می شود ورودی را به صورت بیت به بیت خوانده و سپس پس از تبدیل این دو سیگنال ۸ بیت ای فرمتA-law به فرمت linear آن ها را با یکدیگر جمع کرده و در نهایت به صورت فرمت u-law در زمانی خاص به صورت بیت به بیت به خروجی بدهیم.

چگونگی برخورد با time-slot ها:

با درنظر گرفتن یک عدد صحیح به عنوان clock_counter تعداد کلاک ها را شمارش میکنیم و همانطورکه میدانیم هر هشت کلاک را به صورت یک slot در نظر میگیریم یعنی:

```
Courrent_TimeSlot <= (clock_counter + 1) / 8 ;
```

سپس در انتظار time-slot ورودی کاربر سیستم را نگه میداریم به این معنی که در هر کلاک Current_TimeSlot مربوطه را با ورودی کاربر مقایسه میکنیم و در صورت کاری انجام نمیدهیم. (خروجی z)

لازم به ذکر است که لازم است ابتدا دو سیگنال ورودی باید خوانده و عملیات مورد نیاز روی آن ها انجام شود و سپس بر روی خروجی مورد نیاز کاربر خروجی قرار میدهیم که این کار را با سیگنال ready انجام داده ایم.

نحوه خواندن بیت به بیت از ورودی:

زمانی که به اولین کلاک از ۸ کلاک تایم اسلات مورد نظر کاربر میرسیم در واقع بیت شماره صفر سیگنال را میخوانیم. پس در نتیجه بیت ها را از سمت چپ یکی یکی به سیگنال تعریف شده در برنامه شیفت میدهیم. که پس از گذشت ۸ بیت سیگنال ما آماده میشود.(اتمام عملیات خواندن سیگنال وکتور اول در متغیر بولینی ذخیره می شود.

```
IF (Courrent_TimeSlot = InST1_integer) THEN
        first_Wave <= DR & first_Wave(7 downto 1) ;
        IF (clock_counter mod 8 = 7) then
                flag_EndRead1 <= true;
        end IF;
```

نحوه قرار دادن بر خروجی:

پس از اعمال کار های مورد نیاز بر روی سیگنال های ورودی حالا وقت ان رسیده که این سیگنال را به صورت بیت به بیت بر روی تایم اسلات خواسته شده قرار دهیم. اولین بیت خروجی بیت شماره صفر است پس شیفت را از سمت چپ به راست انجام میدهیم.

```
elsif (Courrent_TimeSlot = OutST_integer AND ready = '1') then
        DX <= out_wave(0);
        out_wave <= '0' & out_wave(7 downto 1);
        IF (clock_counter mod 8 = 7) then
                ready <= '0';
        end IF;
```

نحوه تبدیل سیگنال وکتور های ورودی A-law به linear، جمع و سپس تبدیل به فرمت u-law:

این سری عملیات ها در طول ۵ کلاک و پس از مطمئن شدن از اتمام خواندن دو سیگنال انجام می شود. برای تبدیل به فرمت های خواسته شده از آرایه هایی استفاده کرده ایم که با دادن عدد صحیح شماره خانه آرایه، سیگنال معادل را به ما بر میگرداند.

کلاک اول:

در این کلاک دو سیگنال ورودی را با استفاده از جدول به فرمت linear تبدیل میکنیم:

```
IF ( clk_1) THEN          ---------- To linear form ----------
        first_Wave_linear <= prom_AtoL(CONV_INTEGER(first_Wave));
        second_Wave_linear <= prom_AtoL(CONV_INTEGER(second_Wave));
        clk_1 <= false;
        clk_2 <= true;
```

کلاک دوم:

در این کلاک دو سیگنال به دست امده را برای جمع عدد صحیح تبدیل میکنیم.

```
elsif (clk_2) THEN      ------  integer form ------
        first_Wave_linear_int          <= to_integer(unsigned(first_Wave_linear));
        second_Wave_linear_int         <= to_integer(unsigned(second_Wave_linear));
        clk_2 <= false;
        clk_3 <= true;
```

کلاک سوم:

جمع دو عدد صحیح و ذخیره سازی در سیگنالی دیگر.

```
elsif (clk_3) THEN        --- SUM
        sum_Wave_int <= first_Wave_linear_int + second_Wave_linear_int;
        clk_3 <= false;
        clk_4 <= true;
```

کلاک چهارم:

تبدیل عدد صحیح به دست امده به فرم باینری

به این کلاک الزامی نیست اما برای اینکه شبیه سازی واضح تری داشته باشیم انجامش میدهیم.

```
elsif (clk_4) THEN        --- SUM to vector
        sum_Wave <= std_logic_vector(to_unsigned(sum_Wave_int, sum_Wave'length));
        clk_4 <= false;
        clk_5 <= true;
```

کلاک پنجم:

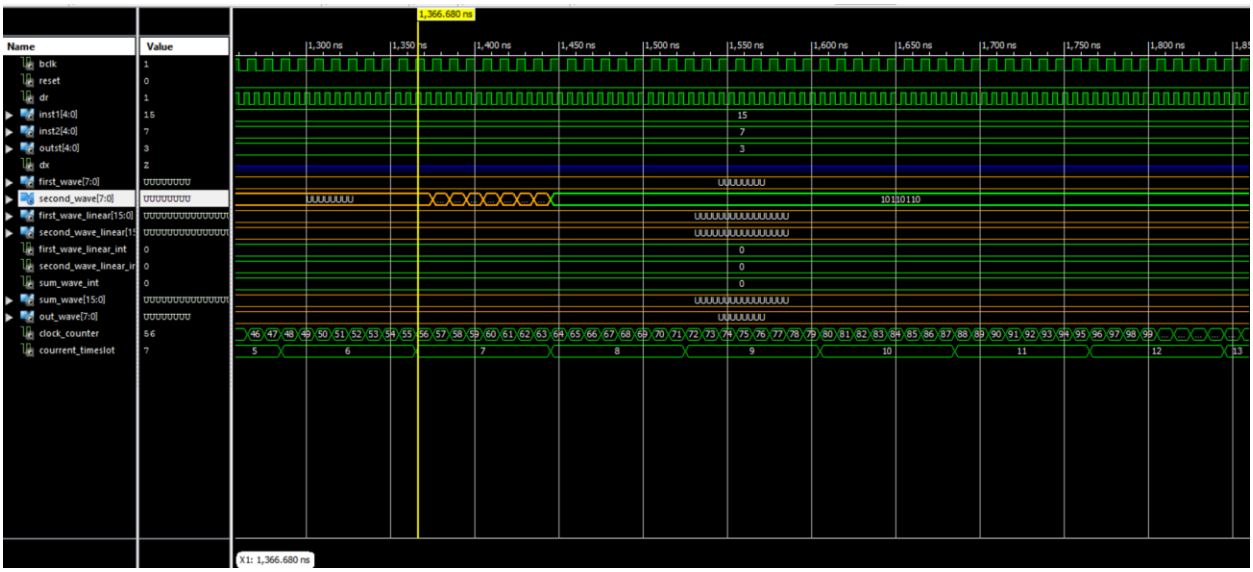
تبدیل عدد به دست امده به فرمت u-law طبق جدول.

لازم به ذکر است به دلیل نا کامل بودن جدول داده ی استفاده شده ورودی ها را به شرطی محدود کرده ایم.

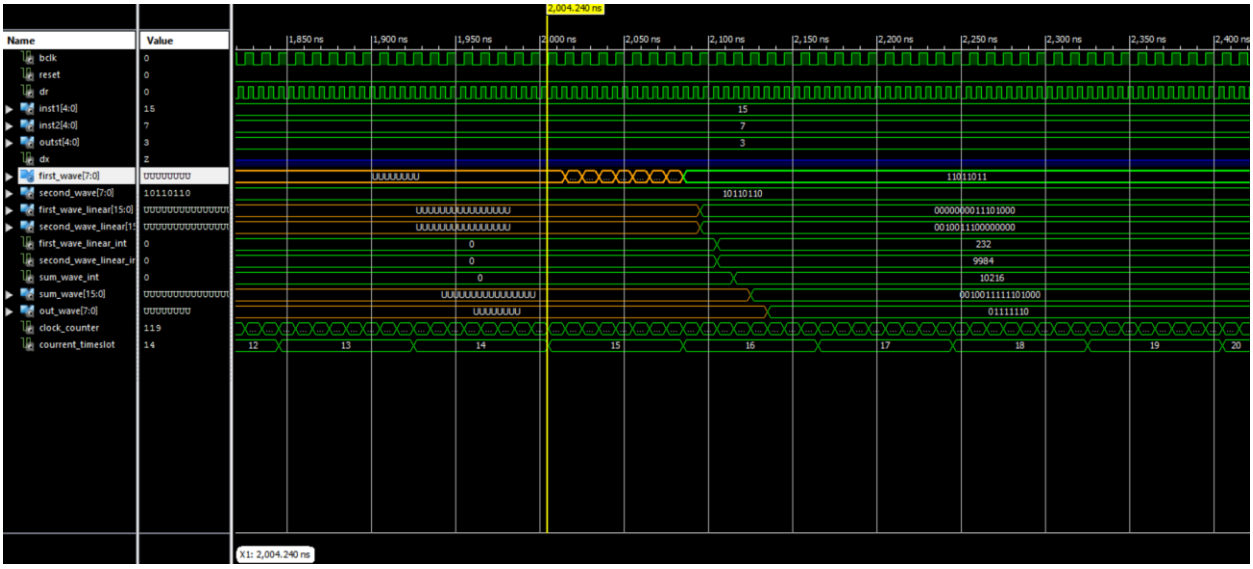
```
elsif (clk_5) THEN
        if (CONV_INTEGER(sum_wave) > 8191) then  -- NOT OK
                out_wave <= prom_U2law(8191);       -- be khater kochaki size dade vorodi jadval
        else -- OK
                out_wave <= prom_U2law(CONV_INTEGER(sum_wave));
        end if;
        ready <= '1';
        flag_EndRead1 <= false;                    -- baraye in ke dobar ein if ejra nashe
        flag_EndRead2 <= false;
```

تصاویر از محیط شبیه سازی:

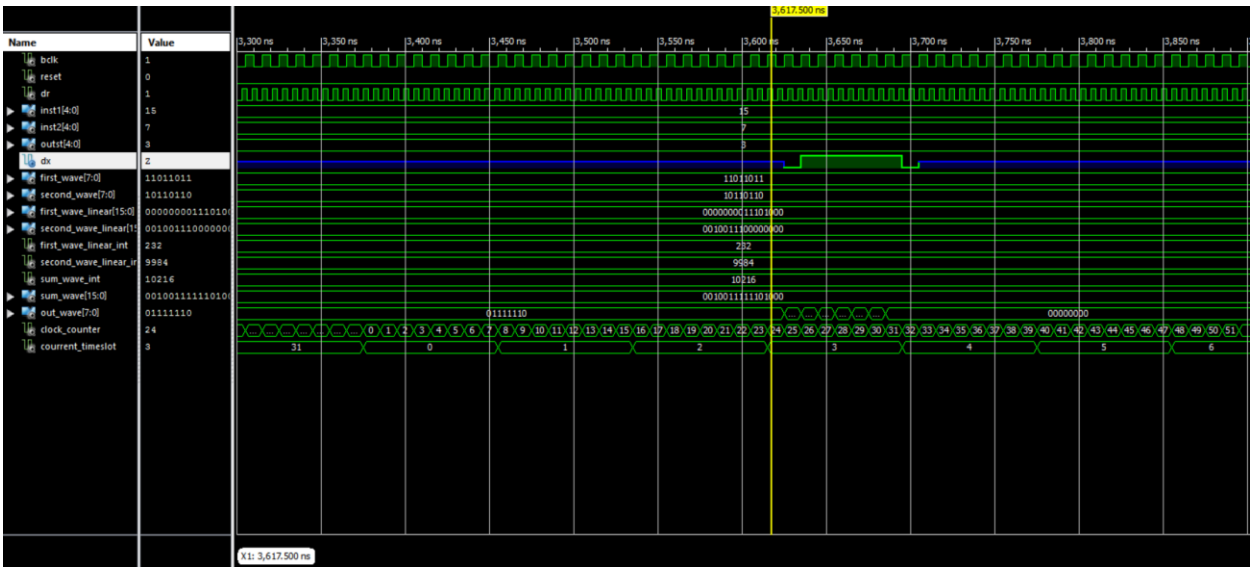
این تصاویر به پیوست الصاق شده اند



تصویر اول خواندن از ورودی شماره ۲:



تصویر دوم خواندن از ورودی شماره ۱:



تصویر سوم نمایش خروجی dx: