2004年ハードウェア構成法期末試験の解説

石井 康雄*

平成18年1月7日

実は、あまり書くべき内容がなかったりしますが…。

1 状態遷移図を書く

問題1

5ビットのシフトレジスタの第4、第5ビットの出力を2入力ＮＡＮＤゲートに入れて、その出力をシフトレジスタの先頭に戻したときの出力波形を示してください。

回路自身は非常に単純なカウンタ回路です。問題は、それがどのような出力波形を持つかですね。

1.1 解法

カウンタ回路なので、いくつかの状態をめぐって初期状態に戻るはずです。このようなループはいくつか存在することがあり、メジャーループ、マイナーループなどといいます。このループがいくつあるかを探します。

あとは、がんばって状態遷移図を書きましょう。最終的には図1のような遷移図がかけるはずです。なお、図中の数字は2進数での状態を示します。例えば11は01011を示し、左側が上位bitとなります。

遷移図が書けたら、あとはLoopの波形図を書くだけです。状態数9のメジャーループと状態数3のマイナーループの波形図を書けば、答えとなります。

2 ステートマシンを設計する

問題2

FPGAの中で使われるSSRAMのあるアドレスのデータを読んで、+1して、もとのアドレスに書き込むステートマシンを設計してください。+1の作業に1クロックかかるとします。
添付、ALTERA SSRAMの10P

2.1 解法

今回の問題はステートマシンの設計です。ポイントはSSRAMのデータシートを読むことでしょう。なお、ここに書くのはあくまで解答**例**です。仕様が満たせる設計は、ここに示した方法以外でもいくら

*yishii@is.s.u-tokyo.ac.jp

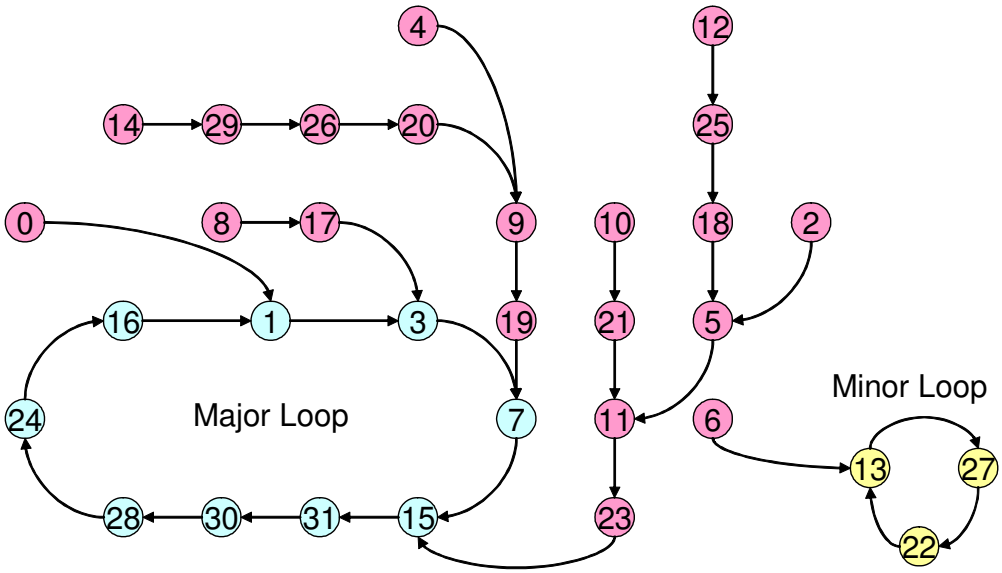

図 1: 状態遷移図

でもあります。自分らしい設計ができるように練習してみてください。

2.1.1 SSRAM の仕様

はじめに Altera の SSRAM の仕様書を読みます。資料の上半分には波形図、下半分にインタフェースが書かれています。

ここから、分かることは以下のようなものでしょう。

- 同期モードではアドレスを与えた次のクロックでデータが得られる
- 読み込みと書き込みが同時にできる (2 ポート構成)

なお、この SSRAM には非同期モードもあるのですが、試験では同期モードを使うように指示されたので無視します。

2.1.2 状態数を決める

ステートマシンの設計でもっとも重要なのが状態数を決めることです。今回与えられた仕様書から以下が分かっています。

- 読み出しにはアドレスを与えるサイクルとデータを受け取るサイクルがある (2 クロック)
- +1 する処理には 1 クロックかかる
- 書き込みはアドレスとデータを同時に与える (1 クロック)

従って、最低でも 4 状態は必要であることが分かります。多くの場合、状態数は処理の遅延時間 (レイテンシ) を決めます。今回の例ではレイテンシは 3 です。

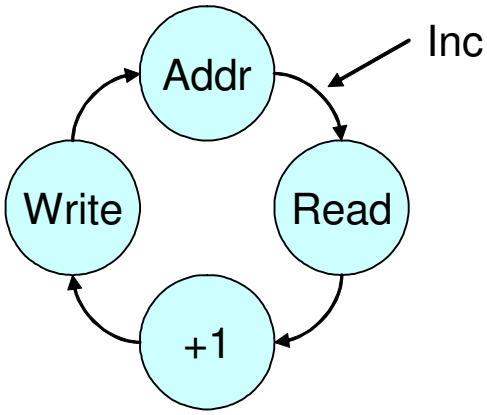

図 2: 状態遷移図

なお、後で先生に確認をしたところ、+1 処理と RAM からの READ が同じクロックでないことを期待していたとのことなので、Write と+1 は同じクロックにしてしまっても良いです。その場合には 3 状態になります。3 状態の場合にはレイテンシは 2 になります。

2.1.3 状態遷移図の作成

この 4 状態でステートマシンを組んだ場合の状態遷移を図にします。図 2 のようになります。Inc はこの回路への入力信号で、現在入力されているアドレスを+1 する必要がある場合に 1 になります。はじめのステート (Addr) で Inc が 1 になったときに同時に RAM へアドレスを入力し、次のステート (Read) で値を読み出します。3 番目のステート (+1) で読み出した値を+1 して、最後のステート (Write) で書き込みをします。

なお、ここでは Read、+1、Write のステートでは Inc を受け付けないという仕様にします。

2.1.4 回路の構成をする

あとは回路を実際に書きましょう。今回はシフトレジスタを利用して、ワンホット型のステートマシンを組むことにします。

さて、前節の議論からこの回路で保持するべき内容は書き込みのイネーブル信号ということになります。アドレスの制御もしても良いのですが、一度、ステートマシンが動作すると、自動的に Write まで進むのでシフトレジスタを構成して、アドレスに遅延を与れば十分です。

すると、ステートマシンで制御する信号は書き込み許可だけとなります。書き込み信号は Addr 状態の時にだけ受け付ける仕様なので、それ以外の時には自動的に 0 になるように適当な回路を挟みます。

今回の場合には、既に 1 がどこかにたっている場合には Addr 状態ではないので、0 にするといった処理で十分でしょう。色々言いましたが、回路にすると図 3 のようになります。

なお、図中ではクロックの信号を省略しています。そして、黄色の四角は D ラッチ (D-FF) をあらわします。単純な回路が構成できていることが分かると思います。

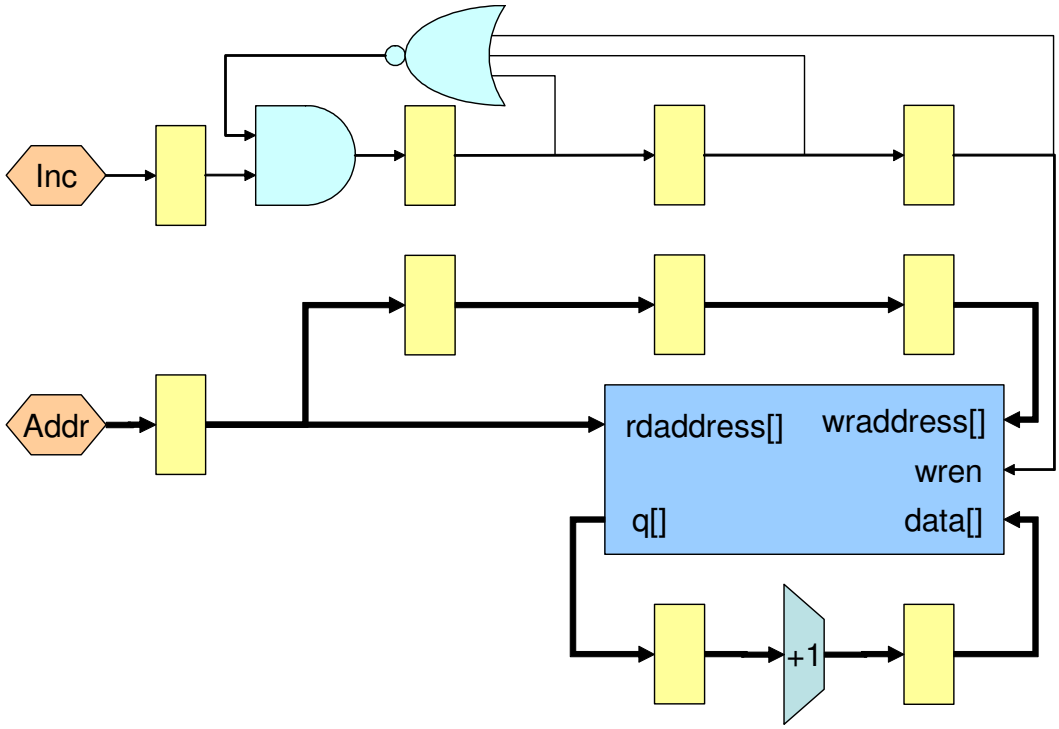

図 3: 解答例

2.1.5 高速化

前節までの内容が書ければ満点です。ここから先は芸術点 (?) を狙った話をします。

前節で作った回路を高速化することを考えます。前節の回路では RAM の Read ポートと Write ポートが 4 クロックに一度しか利用されていません。これは非常にもったいないです。

全てのクロックで両方のポートが利用できるような構成に作り変えます。アイデアとしては、図 4 のような回路を構成することを考えます。この場合には、毎クロック RAM の両方のポートにアクセスできます。しかし、この構成では同一のアドレスが連続で来た場合に仕様を満たせなくなります。これは値を読み出した後に書き込みが行なわれることが原因で RAW(Read After Write) ハザードなどと呼ばれます。

この問題を解決した回路が図 5 です。連続したアドレスが来た場合でも大丈夫なように Forwarding という処理を行なっています。Forwarding は RAW ハザードを解消する最も有力な手段の一つです。

回路としては、セレクタを挟む分、動作周波数が上がらなくなりますが、回路全体のスループットはおそらく 2 倍以上にはなるので高速化が果たせたことになります。

また、Read したデータを直接セレクタに入力していますが、+1 回路よりもセレクタの方が回路の遅延時間が短いため、仕様違反にはならないと判断しました。実際の設計では CAD の出力した遅延時間と要求される性能から最終的な回路構成を決める必要があります。

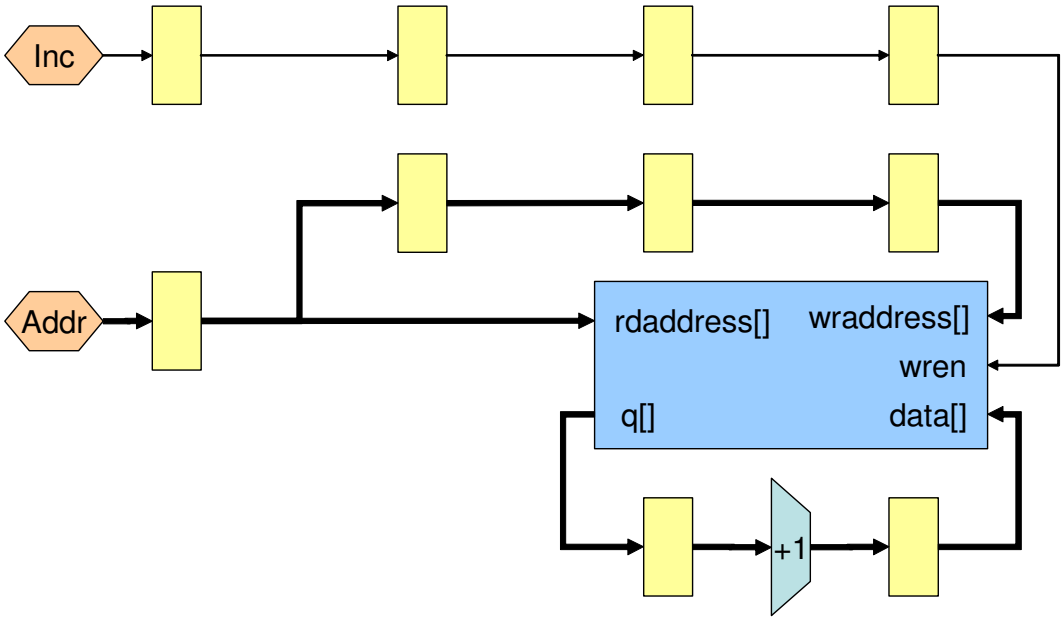

図 4: 高速化した不完全な回路

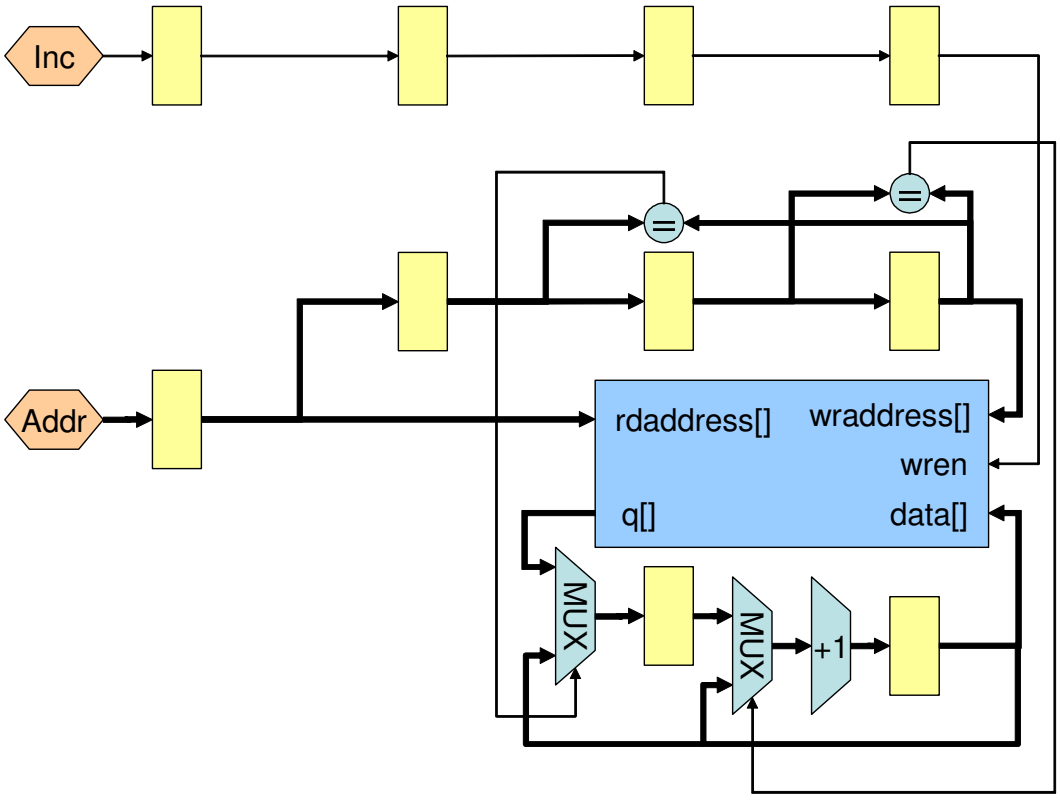

図 5: 高速化した回路