2005年度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 31 日

1　ジョンソンカウンタベースのステートマシン

問題 1

J_0 の出力波形を示してください。回路図は別紙。

重要なことなので確認です。この問題は状態遷移図を書けといっているのではなく、**波形を書け**といっています。ご注意を。

1.1　解法

まず、回路図とにらめっこをします。すると、この回路はシフトレジスタと帰還をかける論理回路という構成になっています。従って、J_0, J_1, J_2, J_3 から $NextJ_0$ を求めれば良いわけです。その後、状態遷移図を書いて波形図を書きます。

1.1.1　論理圧縮

どうも与えられた回路は NAND ゲートが多く、直感的に分かりずらいので論理圧縮をします。

$$\begin{aligned} NextJ_0 &= \overline{\overline{\overline{J_2} \wedge \overline{J_3} \wedge Search} \wedge \overline{J_0 \wedge \overline{J_3}}} \\ &= \overline{J_3} \wedge (J_0 \vee (\overline{J_2} \wedge Search)) \\ &= \overline{J_3} \wedge (J_0 \vee \overline{J_2}) \quad (Search = 1 \text{ より}) \end{aligned}$$

比較的簡単になったので状態遷移を追うことにします。

1.1.2　状態遷移図

あとは必死で状態遷移図を書きましょう。きっと図 1 のようになるはずです。

図中では $Search = 0$ の場合の遷移も書いています。これを見ると、$Search$ の値に関わらず放置しておけば Main Loop に帰ってくることが分かります。さらに $Search = 0$ の場合には 0000 で待機してくれることまで分かります。よくできた設計であるといえるでしょう。

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

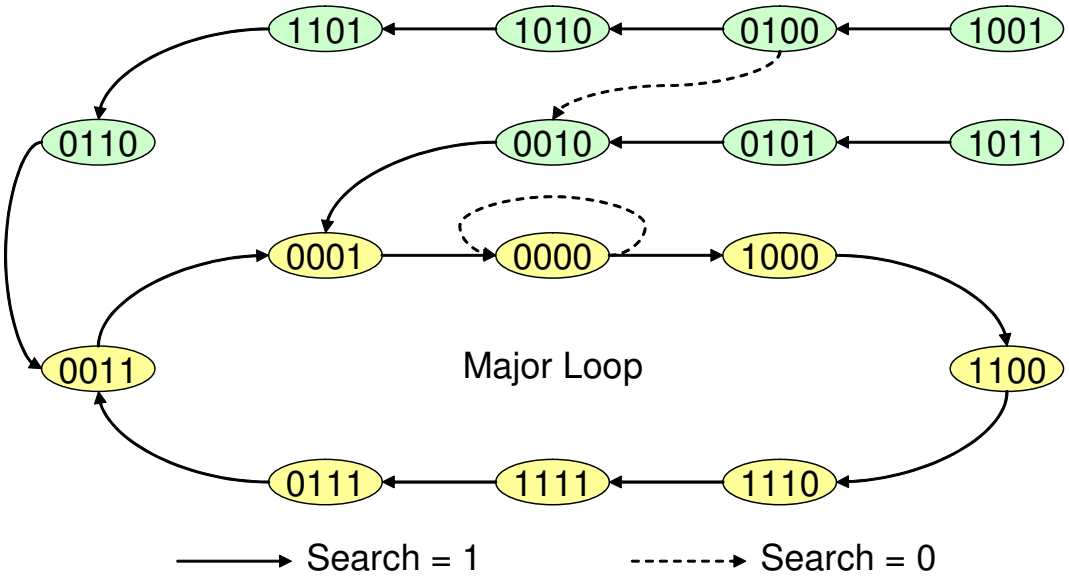

図 1: 状態遷移図

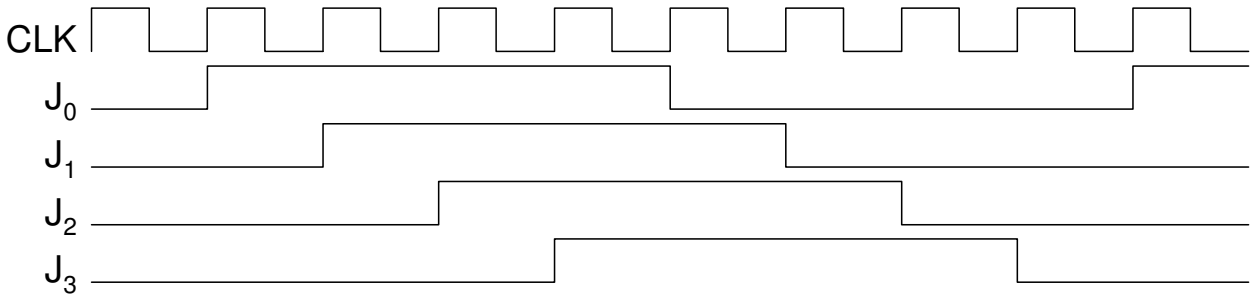

図 2: 波形図

1.1.3 波形図を書く

これを元に波形図を書きます。Major Loop だけ切り出して書けば十分なので、それを図 2 のように書きます。J_0 が求められていますのでそこを切り出せば答えになります。

2 ステートマシンを設計する

問題 2

SSRAM のアドレス A から A+3 までの連続したアドレスに 8 ビットの正数 4 つが含まれています。$Search = 1$ で探索を開始して最大値を発見して出力する回路を作ってください。ただし、A は 4 の倍数とします。

かなりの部分が、2004 年度の資料の使い回しです…。

```
 1  function SearchMax(A : integer)
 2  begin
 3    max := 0
 4    for i in 0..3 do
 5      D := SSRAM(A)
 6      A := A + 1
 7      if D > max then max := D end if
 8    end for
 9    return max
10  end function
```

図 3: 実現したい回路の機能

2.1 解法

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。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 2 クロック後でデータが利用できる
- 読み込みと書き込みが同時にできる (2 ポート構成)

この SSRAM には非同期モード (Flow Through) と同期モード (Pipeline) があります。ここでは Pipeline の場合の構成を考えます。

2.1.2 この回路の動作の確認

何がやりたいのかを確認しておきましょう。ソフトウェアに置き換えて考えると、この回路は配列のある領域からその最大値を探すという動作になります。その擬似コードは図 3 のようになります。

まあ、どうということのないプログラムですね。これをハードウェア上で実装するわけです。なんとなく、SSRAM からデータを読み出して (5 行目)、アドレスの値を変化させつつ (6 行目)、最大値を逐次求める (7 行目)、と理解してもらえれば十分です。

2.1.3 タイミングチャートを書く

前節で示したアルゴリズムをハードウェアで実現していきます。まず、やるべきことは状態数の把握です。そのためにタイミングチャートを書くことにします。タイミングチャートはステートマシンを構成するときに最も重要になるので、書く癖をつけるのが良いと思います。

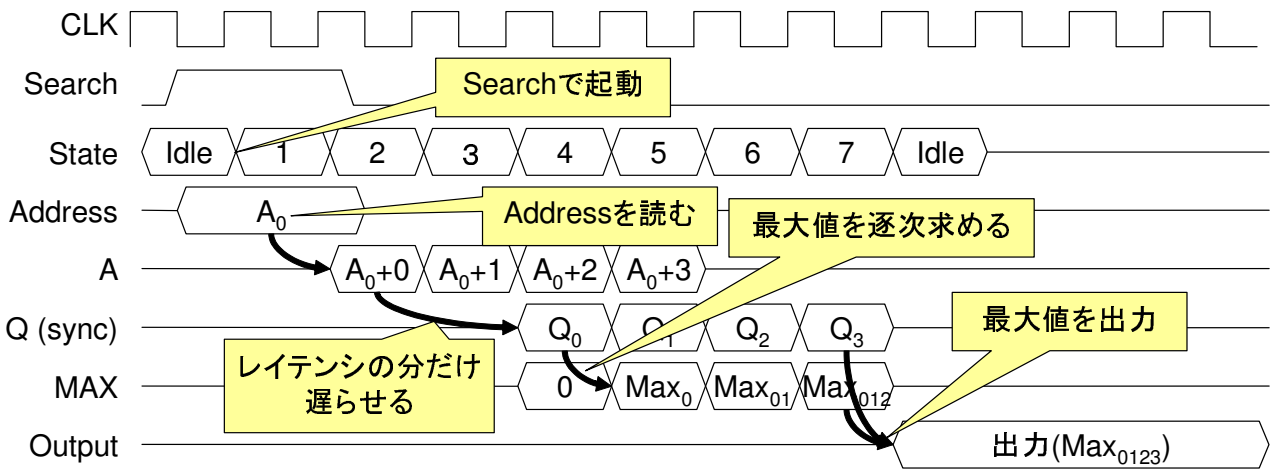

図 4: タイミングチャート

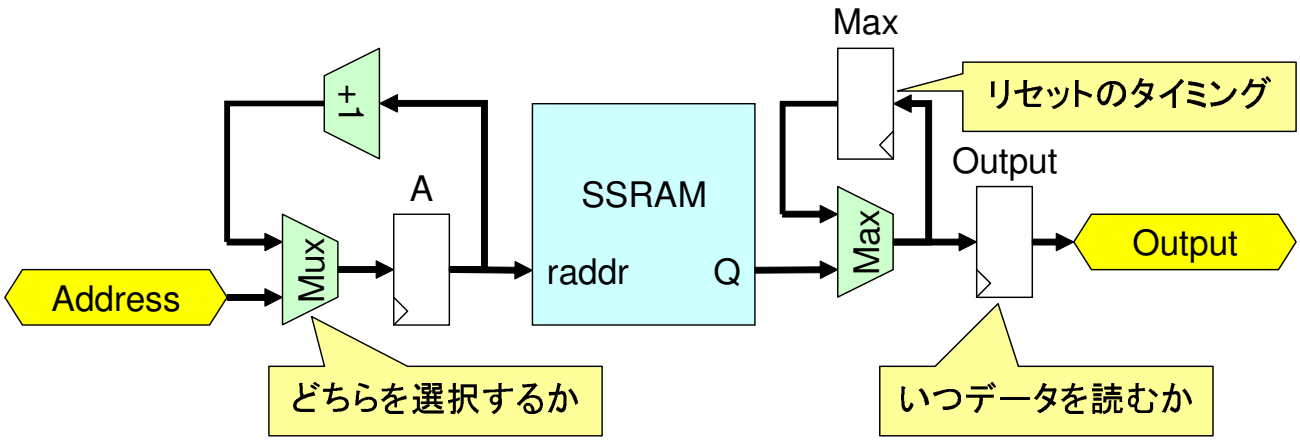

図 5: データパスの設計

今回の解答例のタイミングチャートを図 4 に書きます。SSRAM はレイテンシが 2 あるのでアドレス (A) を与えてから出力 (Q) が得られるまで 2 クロック間を空けます (図 3 の 5 行目)。また、A は必要に応じてインクリメントします (図 3 の 6 行目)。Q はクロックの立ち上がりとほぼ同時に出力されるので、それを以前までの最大値 (Max) と比較して新しい Max として、最後まで生き残ったものを $Output$ として出力します (図 3 の 7 行目)。状態数は 8 個あれば十分でしょう。$Search$ の立ち上がりから 8 クロック後に出力があるので、この回路のレイテンシは 8 です。

2.1.4 データパスの作成

実際に回路を構成していってみましょう。まずは、データパスを構成していきます。データパスとは、制御と関わらない信号の通る回路のことをいい、一般に非常に大きなものになります。しかし、単純なものが多く、ここでミスをすると設計を職業にしている人の場合は怒られるみたいです。

図 5 に今回の例のデータパスを示します。各 FF の上に書いてあるラベルは前節のタイミングチャート上の名前と一致させています。データの流れを書いているだけなので制御は今のところ書いていませ

表 1: 各ステートで必要な制御信号

STATE	0	1	2	3	4	5	6	7
アドレスを+1 (INC)	-	0	1	1	1	-	-	-
Max のリセット (MR)	-	-	-	1	0	0	0	-
$Output$ の読み込み (OE)	0	0	0	0	0	0	0	1

表 2: 実際の制御信号

STATE	0	1	2	3	4	5	6	7
アドレスを+1 するか (INC)	0	0	1	1	1	1	0	0
Max のリセット (MR)	1	1	1	1	0	0	0	0
$Output$ の読み込み (OE)	0	0	0	0	0	0	0	1

ん。しかし、制御するべき回路がどれなのかということは意識する必要があるのでそれをコメントで書いておきました。

2.1.5 制御回路の作成

前々節のチャートは状態遷移図も兼ねていると考えて良いでしょう。そのチャートから 8 状態用意すれば十分ということは容易に読み取れると思います。

さて、各ステートで行なうべきことを整理してみます。それが表 1 です。なお、Idle ステートをステート 0 として書いています。それぞれの行が制御信号に対応します。-は don't care、つまり 1 でも 0 でもどちらでも良い信号です。$Output$ の制御に関しては、ずっと 1 でも良いのですが、違う結果を外に出すのもどうかと思うのでこういう制御にしました。

あとは、8 状態のステートマシンを用意すればよいわけです。丁度、前問の解答が状態数 8 のステートマシンなのでそれを利用します。これは J_0, J_1, J_2, J_3 の 4 つの信号が利用できます。仕様に反しないように $INC = J_1$、$MR = \overline{J_3}$、$OE = \overline{J_2} \wedge J_3$ あたりで制御信号を割り振ります。Don't Care が多いので適当な割り振りでも仕様を満たせますね。まとめると、表 2 のようになります。

2.1.6 回路の構成をする

あとは回路を実際に書きましょう。既にデータパスも制御信号も設計済みなのでつなぐだけです。図 6 のような回路ができるはずです。

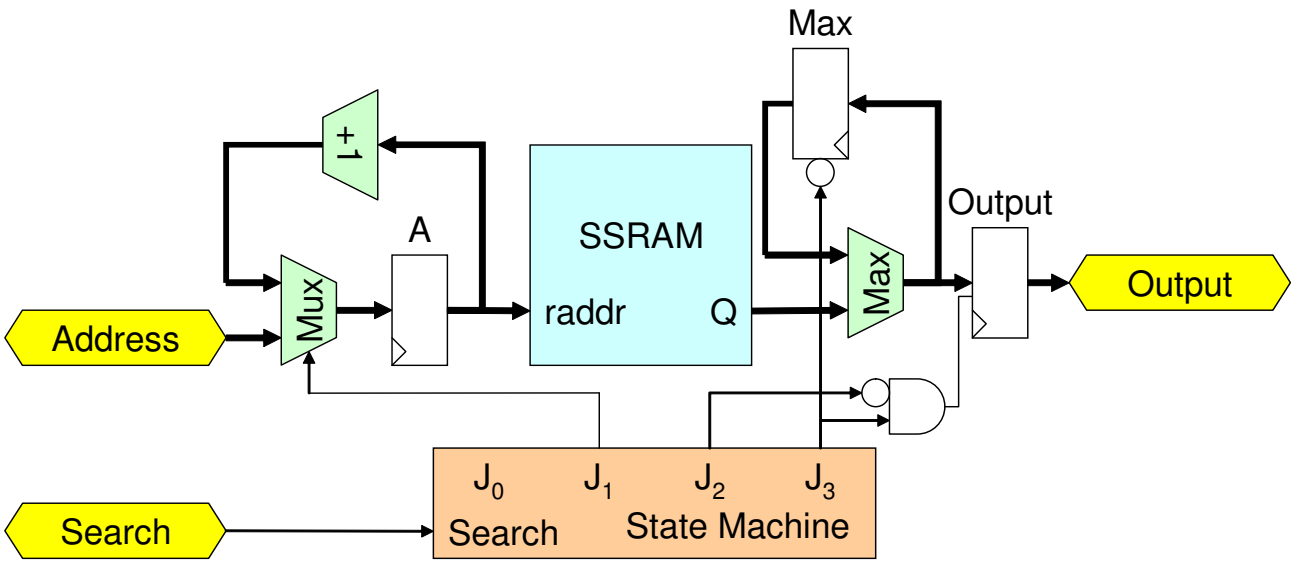

図 6: データパスの設計

2.1.7 高速化

前節までの内容が書ければ満点です。ここからは高速化、効率化の話をします。前節の解答例は SSRAM に対して 2 クロックに 1 度しかアクセスがされていません。これは非常にもったいない[1]です。全てのクロックで SSRAM が利用できるような構成に作り変えます。

しかし、8 状態のステートマシンを用いて、一つの動作として読み出しと最大値計算を行なう場合には常に SSRAM にアクセスできる構成は実現できません。今回は、機能の分割によるスループットの増加を目指します。前回の回路の機能を以下の 3 つに分割します。

1. A を受け取る
2. $A, A+1, A+2, A+3$ の 4 つの連続したアドレスのデータを SSRAM から読み出す
3. Q_0, Q_1, Q_2, Q_3 の 4 つのデータの最大値を求める

この 3 つの機能をステートマシンをオーバーラップ動作させることにより実現します。

タイミングチャートは図 7 のような形になります。この例では 3 つのステートマシン (STATE0,1,2) が独立に動作することにより、高スループットを実現できるわけです。ここまでできれば、あとはデータパスと制御回路です。実は、データパスは前節と同じです[2]。従って、制御信号に関してチェックしましょう。各ステートに関して制御する信号が別ですので適宜チェックしていきましょう。今回は Don't Care が一つもないですね。高効率にするほど Don't Care は減る傾向にあります。そして、回路は複雑になる傾向にあるので、設計者の腕の見せ所となるわけです。

ステートマシンはジョンソンカウンタでもワンホットでも OK です。あとは前節と同じく、状態と信号を対応させれば完成です。ステートマシンから別のステートマシンに対する起動の信号も忘れないようにしてください。

[1] SSRAM が高価なわけは次節で解説します。
[2] 今回はデータパスの効率化を高速化と表現しました

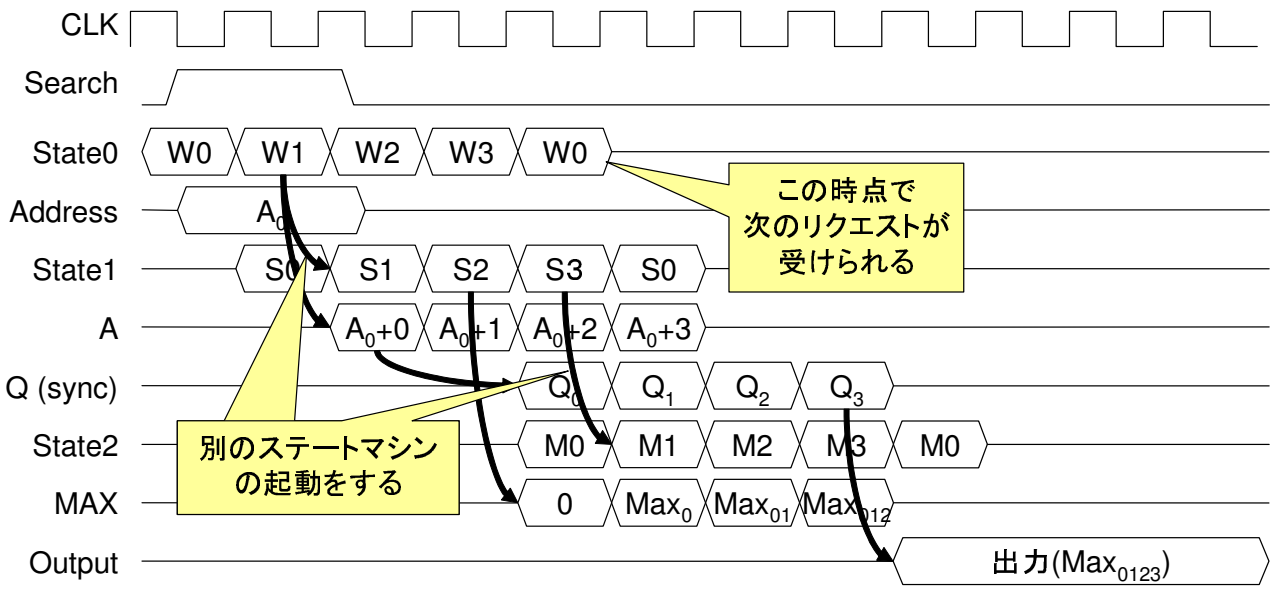

図 7: 高速版のタイミングチャート

表 3: 高速版の回路の制御信号

STATE0	W0	W1	W2	W3
アドレスを+1 (INC)	1	0	1	1
STATE1 の起動 ($Start1$)	0	1	0	0
STATE1	S0	S1	S2	S3
Max のリセット (MR)	0	0	1	0
STATE2 の起動 ($Start2$)	0	0	0	1
STATE3	M0	M1	M2	M3
$Output$ の読み込み (OE)	0	0	0	1

最終的にできた例を図 8 に示します。一つ目のステートマシンから次のシフトレジスタに対して起動をかけていることが分かると思います。また、シフトレジスタは入力の制御をしていないので、上流のステートマシンからの信号により、いつでも起動されるので高速化が達成できます。

2.1.8 完成した回路を見積もる:SSRAM が高価なわけ

前節で SSRAM が高価だから遊ばせるのはもったいない、という文章があります。これは利用しているゲートの数に起因しています。一般に SRAM は FF のアレイで実現されていて SRAM 中の FF は大体 1 ゲート分くらいのトランジスタを利用します。従って、N bit の SRAM は N ゲート分くらいの回路が利用されているわけです。

例えば、アルテラ社の FPGA[3]の SSRAM の容量は 4k bit です。Xilinx 社の FPGA では 18k bit のも

[3]テストの配布資料はこの会社の FPGA のマニュアルです。

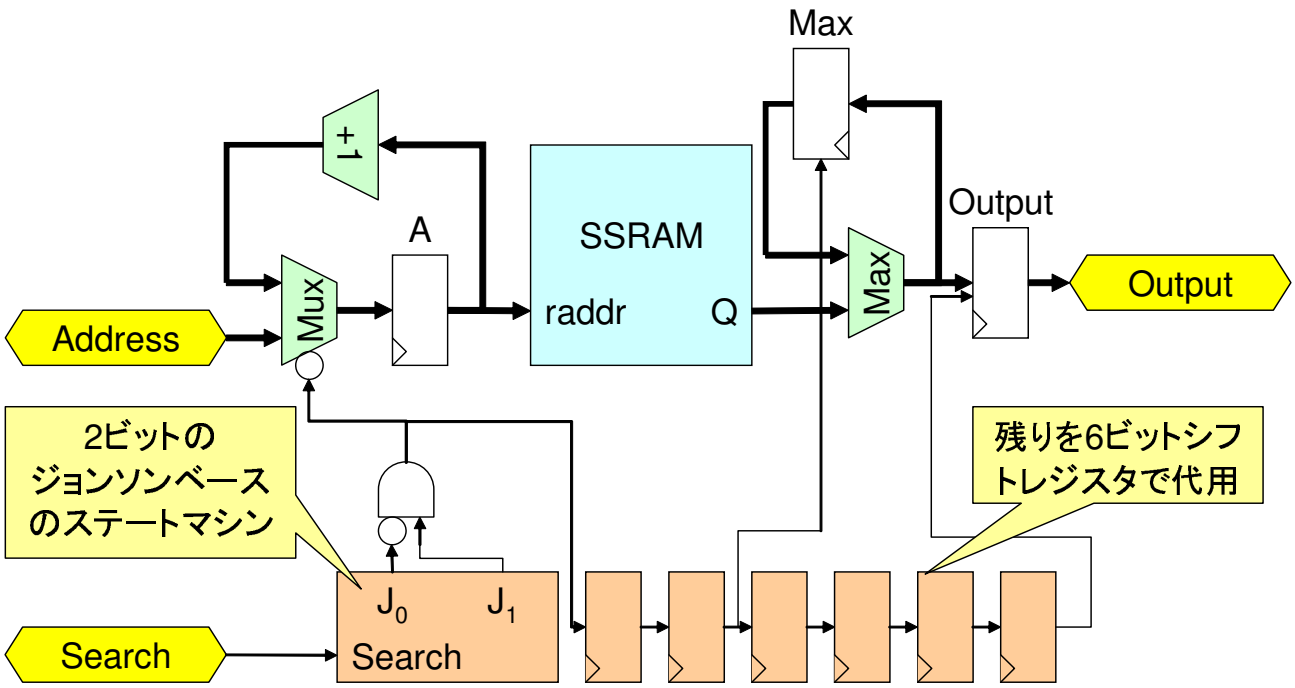

図 8: 高速版の回路例

のもあります。

つまり、実用の SSRAM の場合には最低でも数千から数万ゲート分くらいの資源がそこに利用されています。対して 6 の SSRAM 以外の部分の主な回路は以下の通りです。

- FF(アドレス用に 1 つ、データ用に 2 つ)
- +1 回路 (アドレス用)
- 最大値回路 (データ用)
- セレクタ (アドレス用)
- ジョンソンカウンタ (ステートマシン)

と、全部足しても 100 ゲート行くか行かないかくらいです。多少、回路を増やしてでも SSRAM の利用効率を上げたい理由はここから来るわけです。