2006年度 ハードウェア構成法 期末試験解説

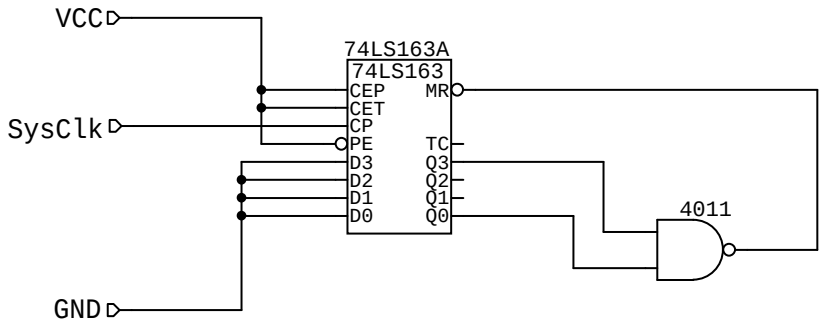
林崎 弘成

平成 19 年 6 月 15 日

1 問題 1: カウンタの出力波形

1.1 問題

次の Binary Synchronous Counter 74LS163A の出力波形を示してください。



CEP, CET 1 の場合、カウンタが動作する。

CP クロック。立ち上がり時に動作。

PE 0 の場合、データロード (D_0〜D_3 の値を Q_0〜Q_3 に読み込む)。

D_0〜D_3 データロード用の入力。

TC キャリー。Q_0〜Q_3 が全て 1 の場合に 1 となる。

MR リセット。0 の場合、次のクロックで Q_0〜Q_3 が 0 になる。

Q_0〜Q_3 カウンタの値。

PE, MR の所に ○ が書かれていますが、これは負論理である (入力が 0 の時に何かが起こる) ことを示すだけのもので、NOT ゲートではありません (問題 2 も同様)。つまり、PE の入力は 1、MR の入力は $\overline{Q_3 \cdot Q_0}$ です。

1.2 解法

信号線がたくさんありますが、このうち今回は考えなくていい信号線を省きます。

1. CEP, CET は 1 固定、つまり常に動作。
2. CP には普通にクロックが入っている。
3. PE は 1 固定、つまり常にデータロード「しない」。D_0〜D_3 は、データロードしないので無視。

4. TC も、Q_3〜Q_0 があれば決まるので省く。

すると、

1. カウンタは、 4 bit の正数 $[Q_3, Q_2, Q_1, Q_0]$ を保持する。(Q_3 の方が上位とする)

2. クロックの立ち上がりごとに、現在の $[Q_3, Q_2, Q_1, Q_0]$ に 1 を足したものを新しい $[Q_3, Q_2, Q_1, Q_0]$ とする。

3. 但し、$\overline{Q_3 \cdot Q_0} = 0$ だった場合には、新しい $[Q_3, Q_2, Q_1, Q_0]$ は $[0, 0, 0, 0]$ にリセットする。

となります。表と図にすると図 1 になります。

old					new			
Q_3	Q_2	Q_1	Q_0	$\overline{Q_3 \cdot Q_0}$	Q_3	Q_2	Q_1	Q_0
0	0	0	0	1	0	0	0	1
0	0	0	1	1	0	0	1	0
0	0	1	0	1	0	0	1	1
0	0	1	1	1	0	1	0	0
0	1	0	0	1	0	1	0	1
0	1	0	1	1	0	1	1	0
0	1	1	0	1	0	1	1	1
0	1	1	1	1	1	0	0	0
1	0	0	0	1	1	0	0	1
1	0	0	1	0	0	0	0	0
1	0	1	0	1	1	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	1	1	1	0	1
1	1	0	1	0	0	0	0	0
1	1	1	0	1	1	1	1	1
1	1	1	1	0	0	0	0	0

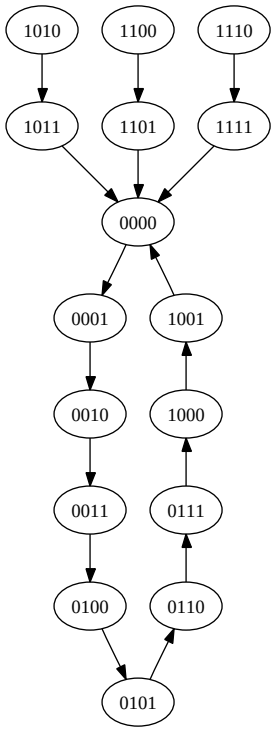


図 1: ステート遷移

ここで、 old の欄はクロック立ち上がり前の値で、new がクロック立ち上がり後の値です。
メジャーループの波形は図 2 となります。

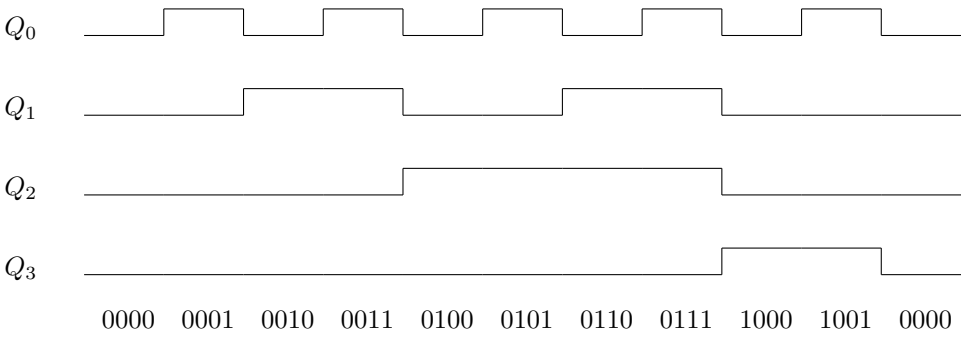
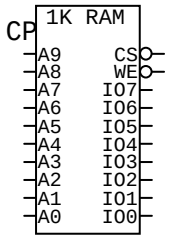


図 2: 出力波形 (メジャーループ)

2 問題 2: SRAM と最大値

2.1 問題

Synchronous SRAM のアドレス A (8 の倍数) から A + 7 までの内容の最大値を求める回路を示してください。



信号線の意味は以下の通りです (先生の板書より)。

CP クロック。

A_0〜A_9 アドレス。

IO_0〜IO_7 データ。IO_7 が最上位ビット、IO_0 が最下位ビット。

WE Write Enable。0 の場合、書き込みを行う。1 の場合、読み込みを行う。

CS Chip Select。0 の場合、動作を行う。

2.2 SRAM の挙動

データシートが配られたと思いますが、今回最低限必要な部分は「アドレス」「データ出力」の 2 つの信号線だけです (クロック立ち上がり時に動作、とかも読み取れますが)。データシートの「アドレス」「データ出力」が、それぞれ問題図中の A_0〜A_9、IO_0〜IO_7 に対応します。まとめると、以下のようになります。

1. 「アドレス」信号線から、アドレス A1 を入れる (2 クロック目)
2. そのアドレスに対応するデータ (Q1) が、1 クロック後に「データ出力」から出てくる (3 クロック目)

信号線のうち、CP にはクロックを入れます。WE には常に 1 を入れて、常に読み込みモードにしておきます。後は、A_0〜A_9、IO_0〜IO_7、CS に適切な信号を入れることを考えます。

2.3 入出力信号の定義

まず、作りたい回路を回路 X とします。回路 X の中に、SRAM のチップが入っています。この回路 X にアドレスを入れると、8 個の要素の最大値が出力されます。よって、回路 X の入出力として必要なものは、アドレス入力と最大値の出力、それと処理開始を指示する信号です。

Go 処理開始を伝える入力信号。これが 1 になったら処理開始。

$Addr_3$〜$Addr_9$ アドレス入力。アドレスは 8 の倍数限定 = 下位 3 bit は常に 0 なので、$Addr_0$〜$Addr_2$ はありません (要りません)。

Max_0〜Max_7 最大値出力。SRAM のアドレス [$Addr_9$,...,$Addr_3$,0,0,0]〜[$Addr_9$,...,$Addr_3$,1,1,1] に格納されている値の最大値を返します。

2.4 タイミングチャートの作成

最初に、全体のタイミングチャートを図 3 に、回路図概要を図 4 に示します。タイミングチャート中、右の方の青い線はとりあえず無視してください (2.6 で述べます)。以下これを見ながら、順を追って解説していきます。

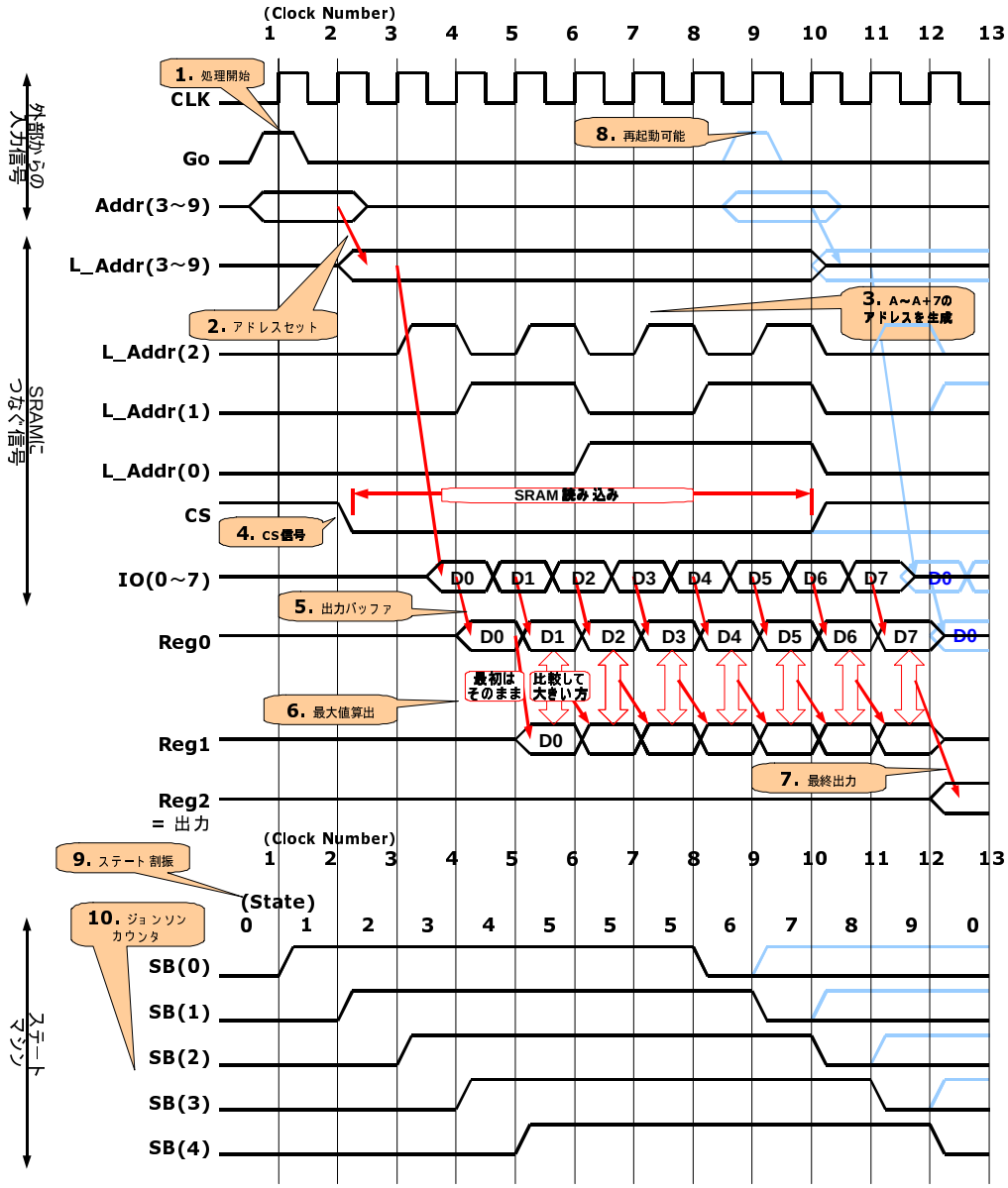


図 3: タイミングチャート

2.4.1 処理開始〜SRAM 読み出し

1. [**処理開始**] まず、Go が 1 になります (クロック 1; 以下、特に断らない限り立ち上がり時点を指す)。これを受けて、処理が開始されます。

2. [**アドレス読み込み**] 次の クロック 2 でアドレス入力 ($Addr_3$〜$Addr_9$) を読んで、内部のラッチ (L_Addr_3〜L_Addr_9) に入れます。

   クロック 1 で読むような回路も作れますが、ここでは余裕を見てクロック 2 で読むことにします (以下同様、

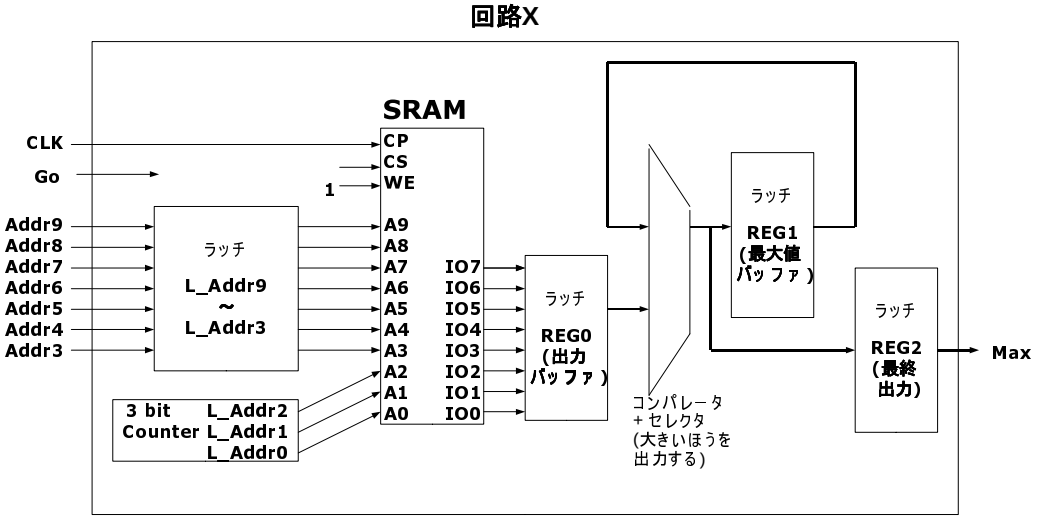


図 4: 回路図概要

がんばれば or 場合によっては 1 クロック速くできる箇所がまだありますが、特に断りません)。

3. [**SRAM 読み込みアドレス生成**] クロック 3 から、$\mathrm{L\_Addr}_3$〜$\mathrm{L\_Addr}_9$ の値が使えるようになるので、それを使って、クロック 3〜10 でそれぞれアドレス [Addr_9,...,Addr_3,0,0,0]〜[Addr_9,...,Addr_3,1,1,1] の値を SRAM から読み出します。

   上位 7 bit は、クロック 2 でアドレスを読み込んだ $\mathrm{L\_Addr}_3$〜$\mathrm{L\_Addr}_9$ をそのまま SRAM の A_3 〜 A_9 信号線につなぎます。

   下位 3 bit は、$[0,0,0],[0,0,1],...,[1,1,1]$ という信号が必要なので、3-bit バイナリカウンタ $\mathrm{L\_Addr}_0$〜$\mathrm{L\_Addr}_2$ を作って、その出力を SRAM の A_0 〜 A_2 信号線につなぎます。

4. [**CS 信号セット**] CS 信号は、クロック 3〜10 で 0 としたいので、クロック 2 の時点で 0 に切り替えて、クロック 10 の時点で 1 に切り替えます。

5. [**SRAM から出てきたデータをバッファに格納**] データはアドレスを入れた 1 クロック後に出てきます (D0, D1, ..., D7。それぞれ、アドレス $\mathrm{Addr}+0,...,\mathrm{Addr}+7$ に対応)。そこで、クロック 4〜11 で、SRAM の IO 信号線を読んで出力バッファのラッチ REG0 (8 bit) に格納します。REG0 の値が使えるようになるのはクロック 5〜12 です。

2.4.2 最大値の算出

6. [**最大値の算出**] ラッチ REG1 (8 bit) に、現時点での最大値を格納します。

   最初のクロック (クロック 5) では最初の要素 D0 がそのまま最大値となるので、REG1 = 最初の要素 (D0) = REG0 の値 とします。

   それ以降のクロックでは、それまでの最大値 = REG1 と、次の要素の値 = REG0 とを比較して、大きい方を REG1 に格納します。

7. [**最終出力**] クロック 12 で、最後の要素 D7 が来るので、ここで REG0 と REG1 を比較して大きい方を取れば、全体の最大値が算出完了となります。

   この値を、(REG1 ではなく) 最終結果バッファ REG2 に格納します。

   この REG2 の出力を Max につなげば、クロック 12 の立ち上がり後から Max に最大値が出力されます。

2.4.3 ステートの設計

タイミングと波形の設計が済んだので、次はそれを実現するためのステートマシンを設計します。

単純に、各クロックごとにステート 1, 2, 3, ... とステートを割り振って、ジョンソンカウンタなどを回しても動きます。

ただ、そうすると実は無駄なステートが含まれてしまいます。ここでは、必要なステートの数を数えて、その分だけステートを割り振る方法を説明します。

まず、どこを区別する必要があるか考えます。そのクロックでしか行われない動作があれば、区別しないとその動作が他のクロックでも行われてしまいます。以下に、他とは区別すべきクロックの立ち上がりと、そこでしか行われない動作を列挙します。

- クロック 1: Go が立ったのを検出して動作を開始する。
- クロック 2: Addr を読み込む。
- クロック 3: L_Addr(2〜0) の カウントアップを開始。
- クロック 4: REG0 に IO からの出力を入れ始める。
- クロック 5: REG1 に REG0 の内容をそのまま入れる。
- クロック 9: これ以降、次の動作を開始できる。つまり、クロック 9 の時点で Go を立てて、次のアドレスを投入できます (図 3 中「8. 再起動可能」)。これよりも以前には再起動はできません (そもそも、SRAM から 8 個のデータを読み出す必要があるので、前回の起動 (クロック 1) から 8 クロック経過していないと無理)。詳しくは 2.6 で述べます。
- クロック 10: CS 信号を 1 に戻す。
- クロック 12: REG2 に出力を書き込む。

逆に言うと、クロック 6〜8 は区別する必要がない、ということです。そこで、以下のようにステート番号を振ります (図 3 中「9. ステート割振」)。

- 〜クロック 1: ステート 0。
- クロック 2, 3, 4, 5: それぞれステート 1, 2, 3, 4。
- クロック 6, 7, 8: ステート 5。
- クロック 9, 10, 11, 12: それぞれステート 6, 7, 8, 9。
- クロック 13〜: ステート 0。

10 ステートですので、5 bit ジョンソンカウンタを作ります (図 3 中「10. ジョンソンカウンタ」)。

2.5 回路の作成

タイミングチャートとステート設計に基づいて、回路に落とします。といっても、後はステート関係の線をつなぐだけです。

2.5.1 ジョンソンカウンタ

まず、SB0 について考えます。

まず、起動時 (クロック 1) に SB0 を 0 から 1 にします。つまり、SB0 = 0 の場合に Go = 1 となった場合、SB0 を 1 にセットします。

逆に、クロック 8 で SB0 が 1 から 0 になっています。これを実現するために、L_Addr(2) · L_Addr(0) ならば 1 から 0 に切り替えることにします。

残りの SB1 〜 SB4 は、単純に SB0 から 1 クロックずつ遅らせるだけです。

回路図にすると図 5 のようになります。

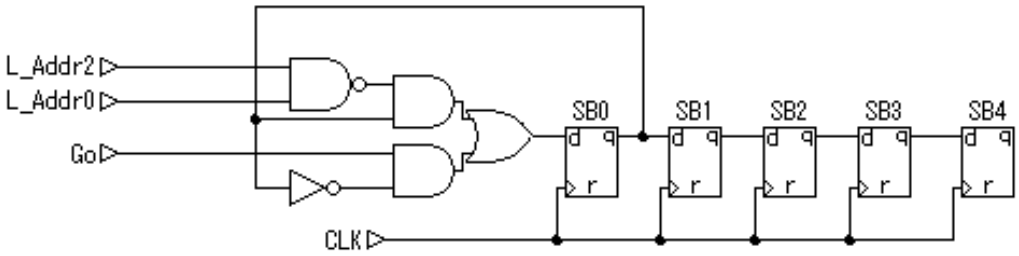


図 5: ジョンソンカウンタ回路図

2.5.2 その他

1. [**ラッチ L_Addr9〜L_Addr3**] クロック 2 (ステート 1) の時だけ、入力 Addr を読み込んで格納し、それ以外の時は何もしません。

   よって、入力を読み込む条件として $SB0 \cdot \overline{SB1}$ をセットします。(この信号は、ステート 1 の時だけ 1 になります。)

2. [**カウンタ L_Addr2〜L_Addr0**] クロック 3〜10 (ステート 2〜7) の時だけ、カウントアップ (+1) します。それ以外の時は何もしません。

   よって、カウントアップの条件として $SB1 + SB2$ をセットします。

   また、最初に 0 にリセットする必要があります。

3. [**CS 信号**] クロック 3〜10 (ステート 2〜7) の時だけ、0 にします。

   よって、$\overline{SB1} \cdot \overline{SB2}$ をセットします。

4. [**ラッチ REG1 への入力**] クロック 5 (ステート 4) の時だけ、REG0 の内容をそのまま入れ、それ以外の場合には REG0 と REG1 の大きいほうを入れます。

   よって、セレクタを用意して、$SB3 \cdot \overline{SB4}$ が 1 だった場合 (つまりステート 4 の場合) には REG0 を、0 だった場合には max(REG0, REG1) が入るようにします。

5. [**ラッチ REG2**] クロック 12 (ステート 9) の時だけ、値を読み込みます。

   よって、読み込み条件として $SB4 \cdot \overline{SB3}$ をセットします。

2.6 再起動に関する考察

2.4.3 で軽く触れましたが、この回路はクロック 9 で再起動できます。

図 3 に描かれている青い線は、クロック 9 で再起動した場合の信号変化を示しています。クロック 9 で Go が再び 1 になり、以下それに従って信号の変化が起こっています。ぎりぎりですが、クロック 1 で起動した分の信号 (黒い線) と衝突はしていないことが分かると思います。

ステートに関しては、図 6 のように遷移します (SB0, SB1, ..., SB4 の順でステートビットを並べています)。黒い矢印で書かれた遷移が、図 3 の黒線で書かれた普通の遷移です。00000 が起動待ちステートで、そこに Go 信号が入ると 10000 に遷移し、後はそのままぐるっと一周します。

クロック 9 で再起動が起こった場合 (図 3 の青線) は、途中 01111 から 10111 に分岐し、 11011, 11101, 11110 と遷移して普通の遷移コースに戻ります (図 6 の青矢印)。ステート番号だけ見ると、普通の場合と全然違う遷移をしているのですが、2.5.2 節で書いたジョンソンカウンタのデコード方法だと問題なく処理できます。

例えば、2.5.2 節の 1 に出てくる $SB0 \cdot \overline{SB1}$ という条件ですが、これは 10--- というステートにマッチします (- は Don't Care、つまりなんでもいい)。よって、普通の遷移の場合には 10000 にマッチし、クロック 9 で再起動した場合には 10111 にマッチします。同様に、 11000, 11100 にマッチする条件は、それぞれ 11011, 11101 にもマッチします。よって、ステート 00000 から起動して 10000 → 11000 → 11100 と遷移した場合も、クロック 9 で再起動して 10111 → 11011 → 11101 と遷移した場合も、同じように処理が進みます。

また、00111, 00011, 00001 にマッチする条件はそれぞれ 10111, 11011, 11101 にもマッチし、クロック 9 で再起動して 10111 → 11011 → 11101 と遷移した場合も、再起動せず 00111 → 00011 → 00001 と遷移して 00000 まで戻る場合も、同様に処理が行われます。

クロック 9 以降で再起動した場合も同様です (図 6 の淡緑矢印)。

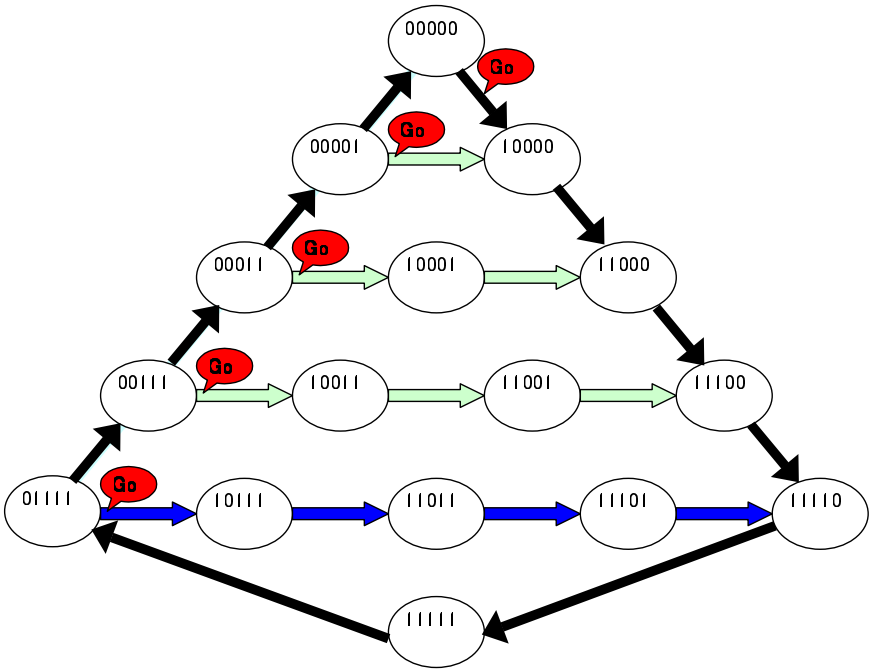


図 6: 再起動時のステート遷移