REFERENCES

[1] E Seevinck, Project Proposal Report, University of Pretoria, January 1995.

[2] T-H Joubert, E Seevinck, M du Plessis, "A CMOS Reduced-Area SRAM Cell", *Proceedings of the IEEE 2000 International Symposium on Circuits and Systems*, pp III-335-8, 28-31 May 2000, Geneva, Switzerland.

[3] S Yamamoto, N Tanimura, K Nagasawa *et al*, "A 256K CMOS SRAM with Variable Impedance Data-Line Loads", *IEEE Journal of Solid-State Circuits*, Vol. 20, No. 5, pp 924-8, October 1985.

[4] N Okazaki, T Komatsu, N Hoshi *et al*, "A 16ns 2K X 8 Bit Full CMOS SRAM", *IEEE Journal of Solid-State Circuits*, Vol. 19, No. 5, pp 552-6, October 1984.

[5] R Hollingsworth, A Ipri, C Kim, "A CMOS/SOS 4K Static RAM", *IEEE Journal of Solid-State Circuits*, Vol. 13, No. 5, pp 664-9, October 1978.

[6] H Levy, E Daniel, T McGill, "A Transistorless-Current-Mode Static RAM Architecture", *IEEE Journal of Solid-State Circuits*, Vol. 33, No. 4, pp 669-72, April 1998.

[7] K Takeda, Y Aimoto, N Nakamura, *et al*, "A 16-Mb 400-MHz Loadless CMOS Four-Transistor SRAM Macro", *IEEE Journal of Solid-State Circuits*, Vol. 35, No. 11, pp 1631-39, November 2000.

[8] K Noda, K Matsui, K Imai, *et al*, "A 1.9-μm^2 Loadless CMOS Four-Transistor SRAM Cell in a 0.18μm Logic Technology", *IEDM Dig. Tech. Papers*, pp 22.8.1-4, 1998.

[9] N Weste, K Eshraghian, *Principles of CMOS VLSI Design, A Systems Perspective*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1985.

[10] E Seevinck, F List, J Lohstroh, "Static-Noise Margin Analysis of MOS SRAM Cells", *IEEE Journal of Solid-State Circuits*, Vol. 22, No. 5, pp 748-54, October 1987.

[11] T Hirose, H Kuriyama, S Murakami, *et al*, "A 20-ns 4-Mb CMOS SRAM with Hierarchical Word Decoding Architecture", *IEEE Journal of Solid-State Circuits*, Vol. 25, No. 5, pp 1068-73, October 1990.

[12] J Rabaey, *Digital Integrated Circuits, A Design Perspective*, Prentice-Hall Inc., New Jersey, 1996.

[13] P Gray, R Meyer, *Analysis and Design of Analog Integrated Circuits, Third Edition*, John Wiley and Sons, Inc., New York, 1993.

[14] K Anami, M Yoshimoto, H Shinohara, *et al*, "Design Considerations of a Static Memory Cell", *IEEE Journal of Solid-State Circuits*, Vol. 18, No. 4, pp 414-7, August 1983.

[15] J Uyemura, *Fundamentals of MOS Digital Integrated Circuits*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1988.

[16] Austria Mikro Systeme International AG, *0.6μm CMOS Design Rules*, Document No. 9931025, Revision 2.0, October 1998.

[17] J Lohstroh, E Seevinck, J De Groot, "Worst-Case Static Noise Margin Criteria for Logic Circuits and Their Mathematical Equivalence", *IEEE Journal of Solid-State Circuits*, Vol. 18, No. 6, pp 803-7, December 1983.

[18] J Lohstroh, "Static and Dynamic Noise Margins of Logic Circuits", *IEEE Journal of Solid-State Circuits*, Vol. 14, No. 3, pp 591-8, June 1979.

[19] Austria Mikro Systeme International AG, *0.6μm CMOS CUP Process Parameters*, Document No. 9933011, Revision B, October 1998.

[20] M Yoshimoto, K Anami, H Hirofumi, *et al*, "A Divide Word-Line Structure in the Static RAM and Its Application to a 64K Full CMOS RAM", *IEEE Journal of Solid-State Circuits*, Vol. 18, No. 5, pp 479-85, October 1983.

[21] D Johns, K Martin, *Analog Integrated Circuit Design*, John Wiley and Sons, Inc., New York, 1997.

[22] Gregorian, G Temes, *Analog MOS Integrated Circuits for Signal Processing*, John Wiley and Sons, Inc., New York, 1986.

[23] P Gray, R Meyer, "MOS Operational Amplifier Design - A Tutorial Overview", *IEEE Journal of Solid-State Circuits*, Vol. 17, No. 6, pp 969-82, December 1982.

[24] K Ishibashi, K Komiyaji, S Morita, *et al*, "A 12.5-ns 16-Mb CMOS SRAM with Common-Centroid-Geometry-Layout Sense Amplifiers", *IEEE Journal of Solid-State Circuits*, Vol. 29, No. 4, pp 411-7, April 1994.

[25] S Kayano, K Ichinose, Y Kohno, *et al*, "25-ns 256Kx1/64Kx4 CMOS SRAM's", *IEEE Journal of Solid-State Circuits*, Vol. 21, No. 5, pp 686-91, October 1986.

[26] E Seevinck, P van Beers, H Ontrop, "Current-Mode Techniques for High-Speed VLSI Circuits with Application to Current Sense Amplifier for CMOS SRAM's", *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 4, pp 525-36, April 1991.

[27] Y Seng, S Rofail, "1.5V High Speed Low Power CMOS Current Sense Amplifier" *Electronics Letters*, Vol. 31, No. 23, pp 1991-3, November 1995.

[28] G Lahiji, A Sodagar, "High-Speed Current-Mode Sense Amplifier" *Electronics Letters*, Vol. 30, No. 17, pp 1371-2, August 1994.

[29] P Chee, P Liu, L Siek, "High-Speed Hybrid Current-Mode Sense Amplifier for CMOS SRAMs" *Electronics Letters*, Vol. 28, No. 9, pp 871-3, April 1992.

[30] L Kim, R Dutton, "Metastability of CMOS Latch/Flip-Flop" ", *IEEE Journal of Solid-State Circuits*, Vol. 25, No. 4, pp 942-951, April 1990.

[31] T Blalock, R Jaeger, "A High-Speed Clamped Bit-Line Current-Mode Sense Amplifier" ", *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 4, pp 542-8, April 1991.

[32] H Veendrick, *Deep-Submicron CMOS ICs*, Kluwer BedrijfsInformatie b.v. - Deventer, The Netherlands, 1998.

[33] R Howe, C Sodini, *Microelectronics, An Integrated Approach*, Prentice-Hall Inc., New Jersey, 1997.

ADDENDUM A: C-CODE FOR NOISE MARGIN ANALYSIS

A.1 FOUR-TRANSISTOR CELL NOISE MARGIN ANALYSIS

```
#include <stdio.h>
#include <time.h>
#define MAXV 501
#define MAXQ 70
#define MAXR 70
#define One_Over_Root2 0.707106781
int main(void){
  // Declarations
  char filenameinn[12];
  char filenameinp[12];
  char filenameoutr[12];
  char filenameoutq[12];
  char filenameswitch[12];
  char indata[200];
  float rStart = 0.5;
  float rEnd = 5;
  float rSpacing = 0.5;
  float qStart = 0.5;
  float qEnd = 5;
  float qSpacing = 0.5;
  float VStart = 0;
  float VEnd = 5.0;
  float VSpacing = 0.01;
  register int i,j,k,l,r,q,v;
  register float temp, tempa;
  register float err, erri, errj;
  register float m,c;
  float NMMax, NMMin;
  int rlength, qlength, vlength;
  float Vin[MAXV], R[MAXR], Q[MAXQ];
  float Vout1[MAXR][MAXQ][MAXV], Vout2[MAXR][MAXQ][MAXV];
  float SNM[MAXR][MAXQ][2];
  char Switch[MAXR][MAXQ];
  int bar = 0;
  int barcount = 0;
  float w[MAXV], u[MAXV], s[MAXV], t[MAXV];
  FILE *in;
  FILE *out;
  time_t dt;
  // Initialisation
  printf("\e[J");
  printf("Welcome to SNM - Static Noise Margin Analysis!!!!!\n\n");
  //Creating Vectors
  printf("\n\nCreating Data Vectors\n");
  rlength = 0;
  i = 1;
  while (i == 1){
    temp = rStart+(rSpacing*rlength);
    if (temp <= rEnd){
      R[rlength] = temp;
      rlength++;}
    else{
      i = 0;}}
  qlength = 0;
```

```
i = 1;
while (i == 1){
  temp = qStart+(qSpacing*qlength);
  if (temp <= qEnd){
    Q[qlength] = temp;
    qlength++;}
  else{
    i = 0;}}
vlength = 0;
i = 1;
while (i == 1){
  temp = VStart+(VSpacing*vlength);
  if (temp <= VEnd){
    Vin[vlength] = temp;
    vlength++;}
  else{
    i = 0;}}
// Read Data from input files
printf("Reading the LO data input file\n");
i = qlength*rlength;
if ((in = fopen("Invlo.csd", "rt")) == NULL){
  fprintf(stderr, "Cannot open LO data input file\n");
  return 1;}
indata[0] = '#';
indata[1] = 'H';
r=0;
q=0;
while (i > 0){
  while ((indata[0] != '#') || (indata[1] != 'C')){
    fgets(indata, 200, in);}
  v=0;
  while ((indata[0] == '#') && (indata[1] == 'C')){
    fgets(indata, 200, in);
    sscanf(indata, "%f", &Vout1[r][q][v]);
    v++;
    fgets(indata, 200, in);}
  r++;
  if (r==rlength){
    r=0;
    q++;}
  i--;}
fclose(in);
printf("Reading the HI data input file\n");
i = qlength*rlength;
if ((in = fopen("Invhi.csd", "rt")) == NULL){
  fprintf(stderr, "Cannot open HI data input file\n");
  return 1;}
indata[0] = '#';
indata[1] = 'H';
r=0;
q=0;
while (i > 0){
  while ((indata[0] != '#') || (indata[1] != 'C')){
    fgets(indata, 200, in);}
  v=0;
  while ((indata[0] == '#') && (indata[1] == 'C')){
    fgets(indata, 200, in);
    sscanf(indata, "%f", &Vout2[r][q][v]);
    v++;
```

```
      fgets(indata, 200, in);}
    r++;
    if(r==rlength){
      r=0;
      q++;}
    i--;}
  fclose(in);
  //Analysing the Data
  printf("Analysing Noise Margins\n");
  printf("[                          ]\n\e[A[");
  for(r=0;r<rlength;++r){
    for(q=0;q<qlength;++q){
      //Translate Coordinate systems
      for (v=0;v<vlength;++v){
        u[v] = One_Over_Root2*(Vout1[r][q][v] + Vin[v]);
        w[v] = One_Over_Root2*(Vout1[r][q][v] - Vin[v]);
        s[v] = One_Over_Root2*(Vin[v] + Vout2[r][q][v]);
        t[v] = One_Over_Root2*(Vin[v] - Vout2[r][q][v]);}
      // Noise Margin Algorithm
      NMMin = 0;
      NMMax = 0;
      for (v=0;v<vlength;++v){
        i=0;
        j=0;
        erri=1000;
        errj=1000;
        temp = w[v];
        // Scan for closest values
        for (k=0;k<vlength;++k){
          if((temp <= t[vlength-1]) && (temp >= t[0])){
            tempa = t[k];
            err = temp-tempa;
            if ((err >=0) && (err <= erri)){
              erri = err;
              i = k;}
            err = tempa-temp;
            if ((err >=0) && (err <= errj)){
              errj = err;
              j = k;}}
          else{
            k = vlength;
            i = -1;
            j = -1;}}
        // Calculate the noise margin
        if (i != j){
          m = (t[i]-t[j])/(s[i]-s[j]);
          c = t[i]-m*s[i];
          temp = u[v]-((w[v]-c)/m);}
        else{
          if (i != -1){
            temp = u[v] - s[i];}
          else{
            temp = 0;}}
        if (temp > NMMax){
          NMMax = temp;}
        if (temp < NMMin){
          NMMin = temp;}}
      SNM[r][q][0] = -NMMin*One_Over_Root2;
      SNM[r][q][1] = NMMax*One_Over_Root2;
```

```
      if ((NMMin >= -0.01) || (NMMax <= 0.01)){
        Switch[r][q] = '0';}
      else{
        Switch[r][q] = '1';}
      bar++;
      temp = (float)barcount/77.0;
      tempa = (float)bar/(rlength*qlength);
      if (temp < tempa){
        barcount = 0;
        while(temp < tempa){
          printf("*");
          barcount++;
          temp = (float)barcount/77.0;}
        printf("\n\e[A[");}}}
printf("\n");
bar = 0;
barcount = 0;
// Write the Switch data output file
time(&dt);
strcpy(indata,ctime(&dt));
printf("\nWriting the Switch data output file\n");
if ((out = fopen("Swit.txt", "wt")) == NULL){
  fprintf(stderr, "Cannot open switch data output file\n");
  return 1;}
fprintf(out,"Switch Data output File\n");
fprintf(out,"Written by 4TCell SNM Analysis %s\n", indata);
fprintf(out,"Vertical: q Value\n");
fprintf(out,"Horizontal: r Value\n\n");
fprintf(out,"       ");
for (i=0;i<rlength;++i){
  fprintf(out,"%1.2f  ",R[i]);}
fprintf(out,"\n\n");
for (j=(qlength-1);j >= 0; --j){
  fprintf(out,"%1.2f  ", Q[j]);
  for (i=0;i<rlength;++i){
    fprintf(out,"   %c  ",Switch[i][j]);}
  fprintf(out,"  %1.2f\n", Q[j]);}
fprintf(out,"\n       ");
for (i=0;i<rlength;++i){
  fprintf(out,"%1.2f  ",R[i]);}
fprintf(out,"\n");
fclose(out);
//Write the X Data output file
printf("Writing the R data output file\n");
if ((out = fopen("Outwr.csd", "wt")) == NULL){
  fprintf(stderr, "Cannot open R data output file\n");
  return 1;}
for(q=0;q<qlength;++q){
  fprintf(out,"#H\n");
  fprintf(out,"SOURCE='SNM Analysis' VERSION='1.0 (June 2001)'\n");
  fprintf(out,"TITLE='** Static Noise Margin of 4TSRAM Cell '\n");
  fprintf(out,"SUBTITLE='Step parameter Q = %1.3E'\n", Q[q]);
  fprintf(out,"TIME='%c%c:%c%c:%c%c' DATE='%c%c%c/%c%c/%c%c'
       TEMPERATURE='27'\n",indata[11], indata[12], indata[14],
       indata[15], indata[17], indata[18], indata[4], indata[5],
       indata[6], indata[8],indata[9], indata[22], indata[23]);
  fprintf(out,"ANALYSIS='DC Sweep' SERIALNO='00001'\n");
  if(q==0){
    i = 3;}
```

```
    else{
      i = 2;}
    fprintf(out,"ALLVALUES='YES' COMPLEXVALUES='NO' NODES='%i'\n",i);
    fprintf(out,"SWEEPVAR='r' SWEEPMODE='LINEAR'\n");
    fprintf(out,"XBEGIN='%1.3E' XEND='%1.3E'\n",rStart,rEnd);
    fprintf(out,"FORMAT='0 VOLTSorAMPS;EFLOAT : NODEorBRANCH;NODE  '\n");
    fprintf(out,"DGTLDATA='NO'\n");
    fprintf(out,"#N\n");
    if (q==0){
      fprintf(out,"'V(Min_Noise_Margin)' 'V(Max_Noise_Margin)'
          'Q(Cell_Trigger)'\n");}
    else{
      fprintf(out,"'V(Min_Noise_Margin)' 'V(Max_Noise_Margin)'\n");}
    for (r=0;r<rlength;++r){
      if (q==0){
        fprintf(out,"#C %1.3E 3\n",R[r]);
        i = 0;
        for (j=(qlength-1);j>=0;--j){
          if (Switch[r][j] == '0'){
            i = j;}}
        fprintf(out,"%1.3E:1 %1.3E:2
          %1.3E:3\n",SNM[r][q][0],SNM[r][q][1], Q[i]);}
      else{
        fprintf(out,"#C %1.3E 2\n",R[r]);
        fprintf(out,"%1.3E:1 %1.3E:2\n",SNM[r][q][0], SNM[r][q][1]);}}
    fprintf(out,"#;\n");}
  fclose(out);
  //Write the Y Data output file
  printf("Writing the Q data output file\n");
  if ((out = fopen("Outwq.csd", "wt")) == NULL){
    fprintf(stderr, "Cannot open Q data output file\n");
    return 1;}
  for(r=0;r<rlength;++r){
    fprintf(out,"#H\n");
    fprintf(out,"SOURCE='SNM Analysis' VERSION='1.0 (June 2001)'\n");
    fprintf(out,"TITLE='** Static Noise Margin of 4TSRAM Cell '\n");
    fprintf(out,"SUBTITLE='Step parameter R = %1.3E'\n", R[r]);
    fprintf(out,"TIME='%c%c:%c%c:%c%c' DATE='%c%c%c/%c%c/%c%c'
          TEMPERATURE='27'\n",indata[11], indata[12], indata[14],
          indata[15], indata[17], indata[18], indata[4], indata[5],
          indata[6], indata[8],indata[9], indata[22], indata[23]);
    fprintf(out,"ANALYSIS='DC Sweep' SERIALNO='00001'\n");
    if(r==0){
      i = 3;}
    else{
      i = 2;}
    fprintf(out,"ALLVALUES='YES' COMPLEXVALUES='NO' NODES='%i'\n",i);
    fprintf(out,"SWEEPVAR='q' SWEEPMODE='LINEAR'\n");
    fprintf(out,"XBEGIN='%1.3E' XEND='%1.3E'\n",qStart,qEnd);
    fprintf(out,"FORMAT='0 VOLTSorAMPS;EFLOAT : NODEorBRANCH;NODE  '\n");
    fprintf(out,"DGTLDATA='NO'\n");
    fprintf(out,"#N\n");
    if (r==0){
    fprintf(out,"'V(Min_Noise_Margin)' 'V(Max_Noise_Margin)'
          'R(Cell_Trigger)'\n");}
    else{
      fprintf(out,"'V(Min_Noise_Margin)' 'V(Max_Noise_Margin)'\n");}
    for (q=0;q<qlength;++q){
      if (r==0){
```

```
        fprintf(out,"#C %1.3E 6\n",Q[q]);
        i = 0;
        for (j=(rlength-1);j>=0;--j){
          if (Switch[j][q] == '0'){
            i = j;}}
        fprintf(out,"%1.3E:1 %1.3E:2
         %1.3E:3\n",SNM[r][q][0],SNM[r][q][1], R[i]);}
      else{
        fprintf(out,"#C %1.3E 2\n",Q[q]);
        fprintf(out,"%1.3E:1 %1.3E:2\n",SNM[r][q][0], SNM[r][q][1]);}}
    fprintf(out,"#;\n");}
  fclose(out);
  return 0;}
```

A.2 SIX-TRANSISTOR CELL NOISE MARGIN ANALYSIS

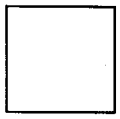
```
#include <stdio.h>
#include <time.h>
#define MAXV 501
#define MAXQ 70
#define MAXR 70
#define One_Over_Root2 0.707106781

int main(void){
  // Declarations
  char filenamein[12];
  char filenameoutr[12];
  char filenameoutq[12];
  char indata[200];
  float rStart = 1;
  float rEnd = 3;
  float rSpacing = 0.1;
  float qStart = 1;
  float qEnd = 3;
  float qSpacing = 0.1;
  float VStart = 0;
  float VEnd = 5.0;
  float VSpacing = 0.01;
  register int i,j,k,l,r,q,v;
  register float temp, tempa;
  register float err, erri, errj;
  register float m,c;
  float NMMax, NMMin;
  int rlength, qlength, vlength;
  float Vin[MAXV], R[MAXR], Q[MAXQ];
  float Vout[MAXR][MAXQ][MAXV];
  float SNM[MAXR][MAXQ][2];
  int bar = 0;
  int barcount = 0;
  float w[MAXV], u[MAXV], s[MAXV], t[MAXV];
  FILE *in;
  FILE *out;
  time_t dt;
  // Initialisation
  printf("\e[J");
  printf("Welcome to SNM - Static Noise Margin Analysis!!!!!\n\n");
  //Creating Vectors
  printf("\n\nCreating Data Vectors\n");
```

```
rlength = 0;
i = 1;
while (i == 1) {
  temp = rStart+(rSpacing*rlength);
  if (temp <= rEnd){
    R[rlength] = temp;
    rlength++;}
  else{
    i = 0;}}
qlength = 0;
i = 1;
while (i == 1){
  temp = qStart+(qSpacing*qlength);
  if (temp <= qEnd){
    Q[qlength] = temp;
    qlength++;}
  else{
    i = 0;}}
vlength = 0;
i = 1;
while (i == 1){
  temp = VStart+(VSpacing*vlength);
  if (temp <= VEnd){
    Vin[vlength] = temp;
    vlength++;}
  else{
    i = 0;}}
// Read Data from input files
printf("Reading the HI data input file\n");
i = qlength*rlength;
if ((in = fopen("Invhi.csd", "rt")) == NULL){
  fprintf(stderr, "Cannot open HI data input file\n");
  return 1;}
indata[0] = '#';
indata[1] = 'H';
r=0;
q=0;
while (i > 0){
  while ((indata[0] != '#') || (indata[1] != 'C')){
    fgets(indata, 200, in);}
  v=0;
  while ((indata[0] == '#') && (indata[1] == 'C')){
    fgets(indata, 200, in);
    sscanf(indata, "%f", &Vout[r][q][v]);
    v++;
    fgets(indata, 200, in);}
  r++;
  if(r==rlength){
    r=0;
    q++;}
  i--;}
fclose(in);
//Analysing the Data
printf("Analysing Noise Margins\n");
printf("[                                                    ]\n\e[A[");
for(r=0;r<rlength;++r){
  for(q=0;q<qlength;++q){
    //Translate Coordinate systems
    for (v=0;v<vlength;++v){
```

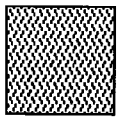
```
      u[v] = One_Over_Root2*(Vout[r][q][v] + Vin[v]);
      w[v] = One_Over_Root2*(Vout[r][q][v] - Vin[v]);
      s[v] = One_Over_Root2*(Vin[v] + Vout[r][q][v]);
      t[v] = One_Over_Root2*(Vin[v] - Vout[r][q][v]);}
    // Noise Margin Algorithm
    NMMin = 0;
    NMMax = 0;
    for (v=0;v<vlength;++v){
      i=0;
      j=0;
      erri=1000;
      errj=1000;
      temp = w[v];
      // Scan for closest values
      for (k=0;k<vlength;++k){
        if ((temp <= t[vlength-1]) && (temp >= t[0])){
          tempa = t[k];
          err = temp-tempa;
          if ((err >=0) && (err <= erri)){
            erri = err;
            i = k;}
          err = tempa-temp;
          if ((err >=0) && (err <= errj)){
            errj = err;
            j = k;}}
        else{
          k = vlength;
          i = -1;
          j = -1;}}
      // Calculate the noise margin
      if (i != j){
        m = (t[i]-t[j])/(s[i]-s[j]);
        c = t[i]-m*s[i];
        temp = u[v]-((w[v]-c)/m);}
      else{
        if (i != -1){
          temp = u[v] - s[i];}
        else{
          temp = 0;}}
      if (temp > NMMax){
        NMMax = temp;}
      if (temp < NMMin){
        NMMin = temp;}}
    SNM[r][q][0] = -NMMin*One_Over_Root2;
    SNM[r][q][1] = NMMax*One_Over_Root2;
    bar++;
    temp = (float)barcount/77.0;
    tempa = (float)bar/(rlength*qlength);
    if (temp < tempa){
      barcount = 0;
      while(temp < tempa){
        printf("*");
        barcount++;
        temp = (float)barcount/77.0;}
      printf("\n\e[A[");}}
  printf("\n");
  bar = 0;
  barcount = 0;}
//Write the R Data output file
```

```
time(&dt);
strcpy(indata,ctime(&dt));
printf("Writing the R data output file\n");
if ((out = fopen("Outrr.csd", "wt")) == NULL){
  fprintf(stderr, "Cannot open R data output file\n");
  return 1;}
for(q=0;q<qlength;++q){
  fprintf(out,"#H\n");
  fprintf(out,"SOURCE='SNM Analysis' VERSION='1.0 (June 2001)'\n");
  fprintf(out,"TITLE='** Static Noise Margin of 6TSRAM Cell '\n");
  fprintf(out,"SUBTITLE='Step parameter Q = %1.3E'\n", Q[q]);
  fprintf(out,"TIME='%c%c:%c%c:%c%c' DATE='%c%c%c/%c%c/%c%c'
       TEMPERATURE='27'\n",indata[11], indata[12], indata[14],
       indata[15], indata[17], indata[18], indata[4], indata[5],
       indata[6], indata[8],indata[9], indata[22], indata[23]);
  fprintf(out,"ANALYSIS='DC Sweep' SERIALNO='00001'\n");
  fprintf(out,"ALLVALUES='YES' COMPLEXVALUES='NO' NODES='1'\n",i);
  fprintf(out,"SWEEPVAR='r' SWEEPMODE='LINEAR'\n");
  fprintf(out,"XBEGIN='%1.3E' XEND='%1.3E'\n",rStart,rEnd);
  fprintf(out,"FORMAT='0 VOLTSorAMPS;EFLOAT : NODEorBRANCH;NODE  '\n");
  fprintf(out,"DGTLDATA='NO'\n");
  fprintf(out,"#N\n");
  fprintf(out,"'V(Noise_Margin)'\n");
  for (r=0;r<rlength;++r){
    fprintf(out,"#C %1.3E 1\n",R[r]);
    fprintf(out,"%1.3E:1\n",SNM[r][q][0]);}
  fprintf(out,"#;\n");}
fclose(out);
//Write the Y Data output file
printf("Writing the Q data output file\n");
if ((out = fopen("Outrq.csd", "wt")) == NULL){
  fprintf(stderr, "Cannot open Q data output file\n");
  return 1;}
for(r=0;r<rlength;++r){
  fprintf(out,"#H\n");
  fprintf(out,"SOURCE='SNM Analysis' VERSION='1.0 (June 2001)'\n");
  fprintf(out,"TITLE='** Static Noise Margin of 6TSRAM Cell '\n");
  fprintf(out,"SUBTITLE='Step parameter R = %1.3E'\n", R[r]);
  fprintf(out,"TIME='%c%c:%c%c:%c%c' DATE='%c%c%c/%c%c/%c%c'
       TEMPERATURE='27'\n",indata[11], indata[12], indata[14],
       indata[15], indata[17], indata[18], indata[4], indata[5],
       indata[6], indata[8],indata[9], indata[22], indata[23]);
  fprintf(out,"ANALYSIS='DC Sweep' SERIALNO='00001'\n");
  fprintf(out,"ALLVALUES='YES' COMPLEXVALUES='NO' NODES='1'\n",i);
  fprintf(out,"SWEEPVAR='q' SWEEPMODE='LINEAR'\n");
  fprintf(out,"XBEGIN='%1.3E' XEND='%1.3E'\n",qStart,qEnd);
  fprintf(out,"FORMAT='0 VOLTSorAMPS;EFLOAT : NODEorBRANCH;NODE  '\n");
  fprintf(out,"DGTLDATA='NO'\n");
  fprintf(out,"#N\n");
  fprintf(out,"'V(Noise_Margin)'\n");
  for (q=0;q<qlength;++q){
    fprintf(out,"#C %1.3E 1\n",Q[q]);
    fprintf(out,"%1.3E:1\n",SNM[r][q][0]);}
  fprintf(out,"#;\n");}
fclose(out);
return 0;}
```

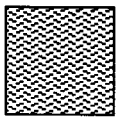
ADDENDUM B: LAYOUT LEGEND



N-Well



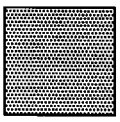
N-Active



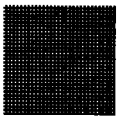
P-Active

Poly 1

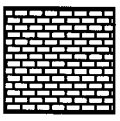
Contact



Metal 1



Via



Metal 2

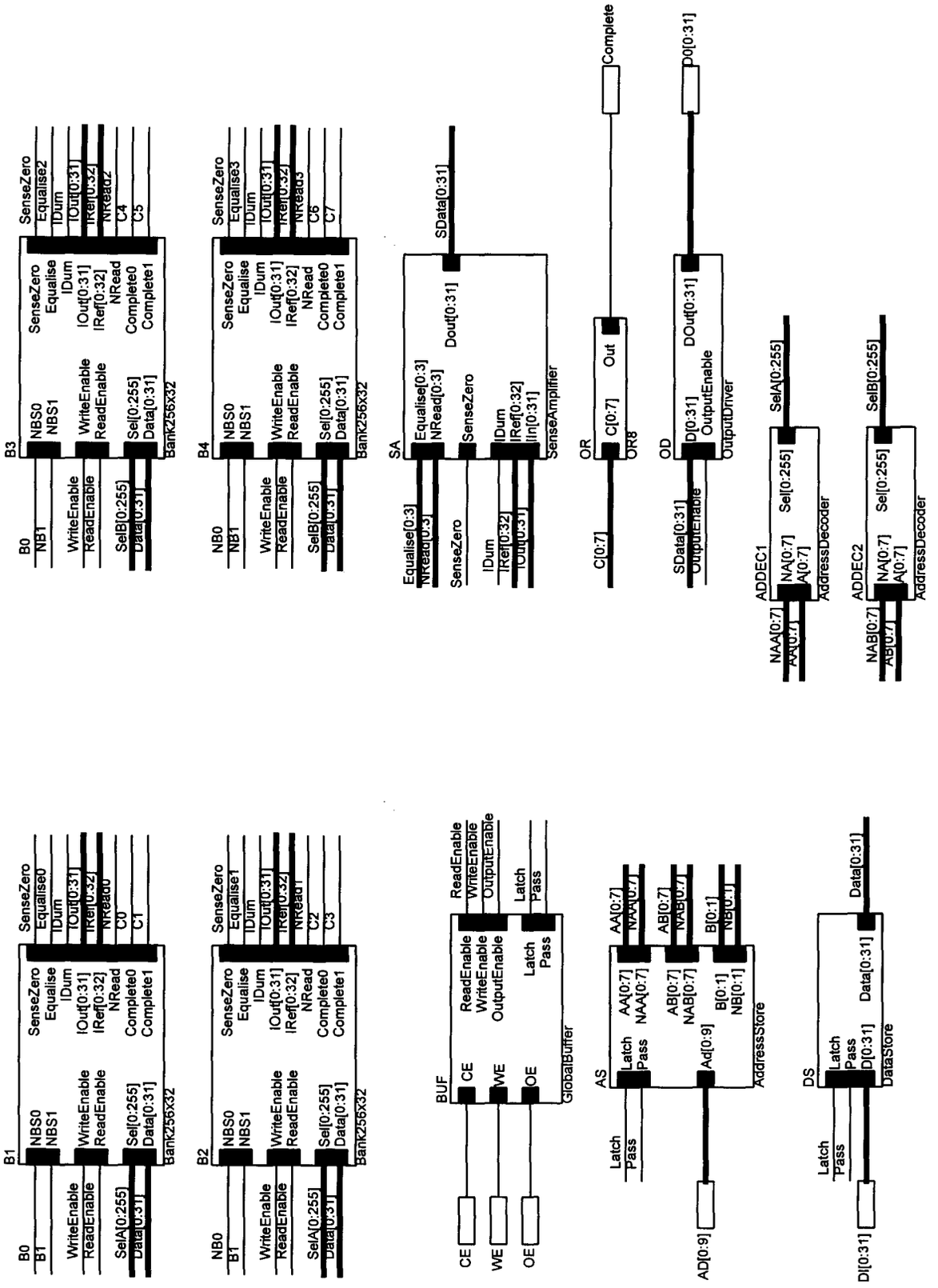


Figure C.1 SRAM System: Top level of the four-transistor SRAM cell system.

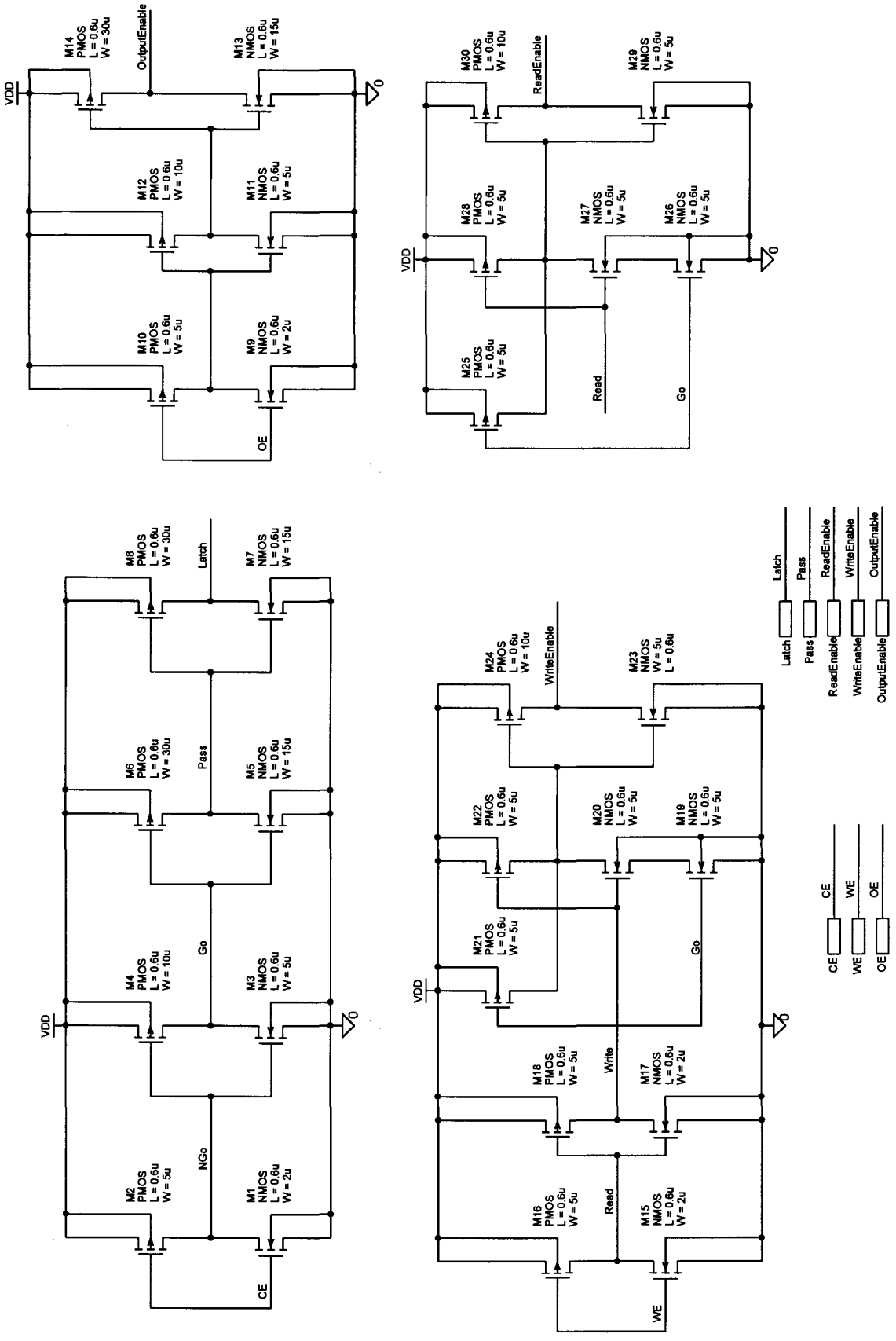


Figure C.2 Global Buffer: Buffering system for the input control signals.

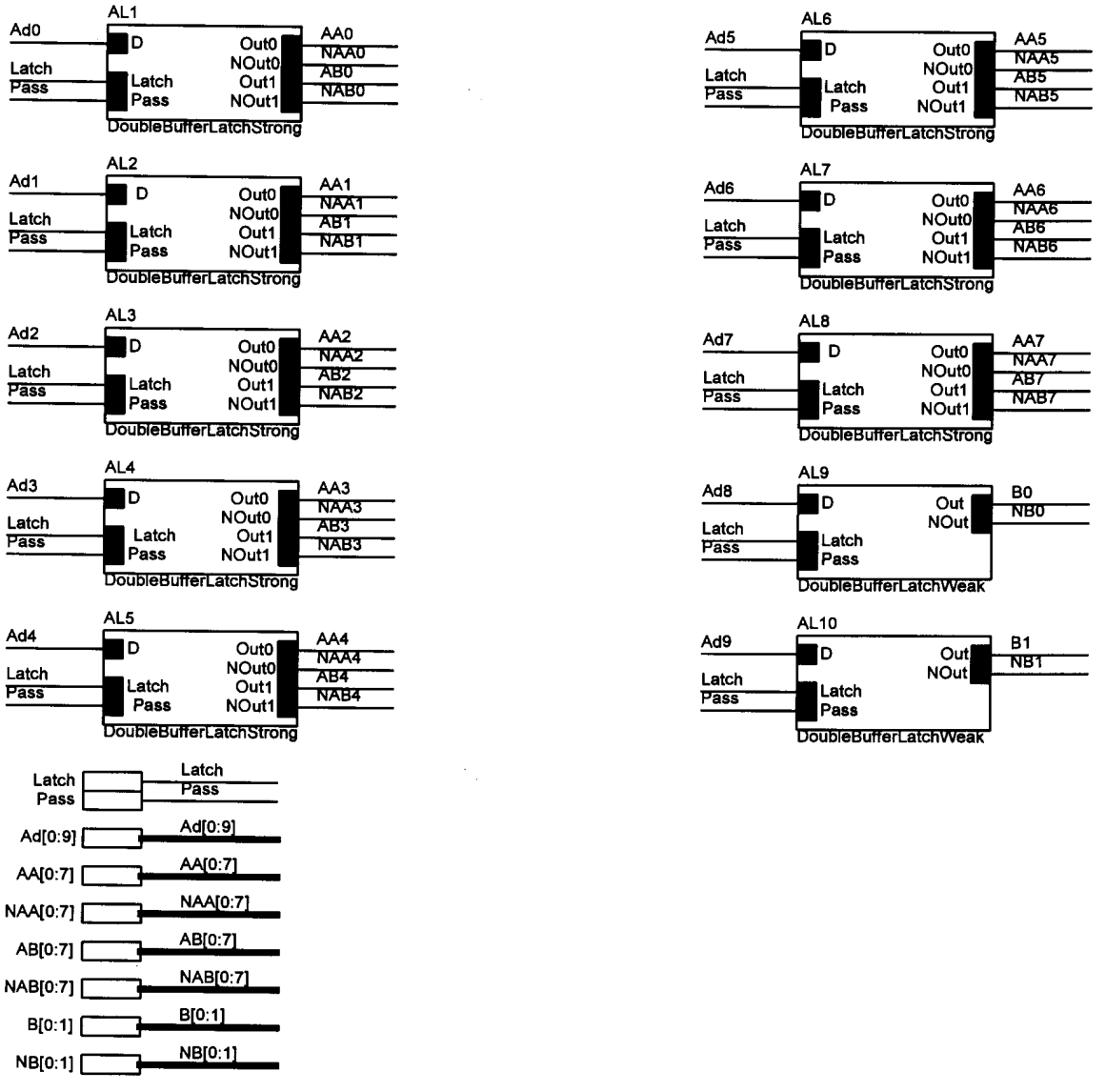


Figure C.3 Address Store: Latch system for the address inputs.

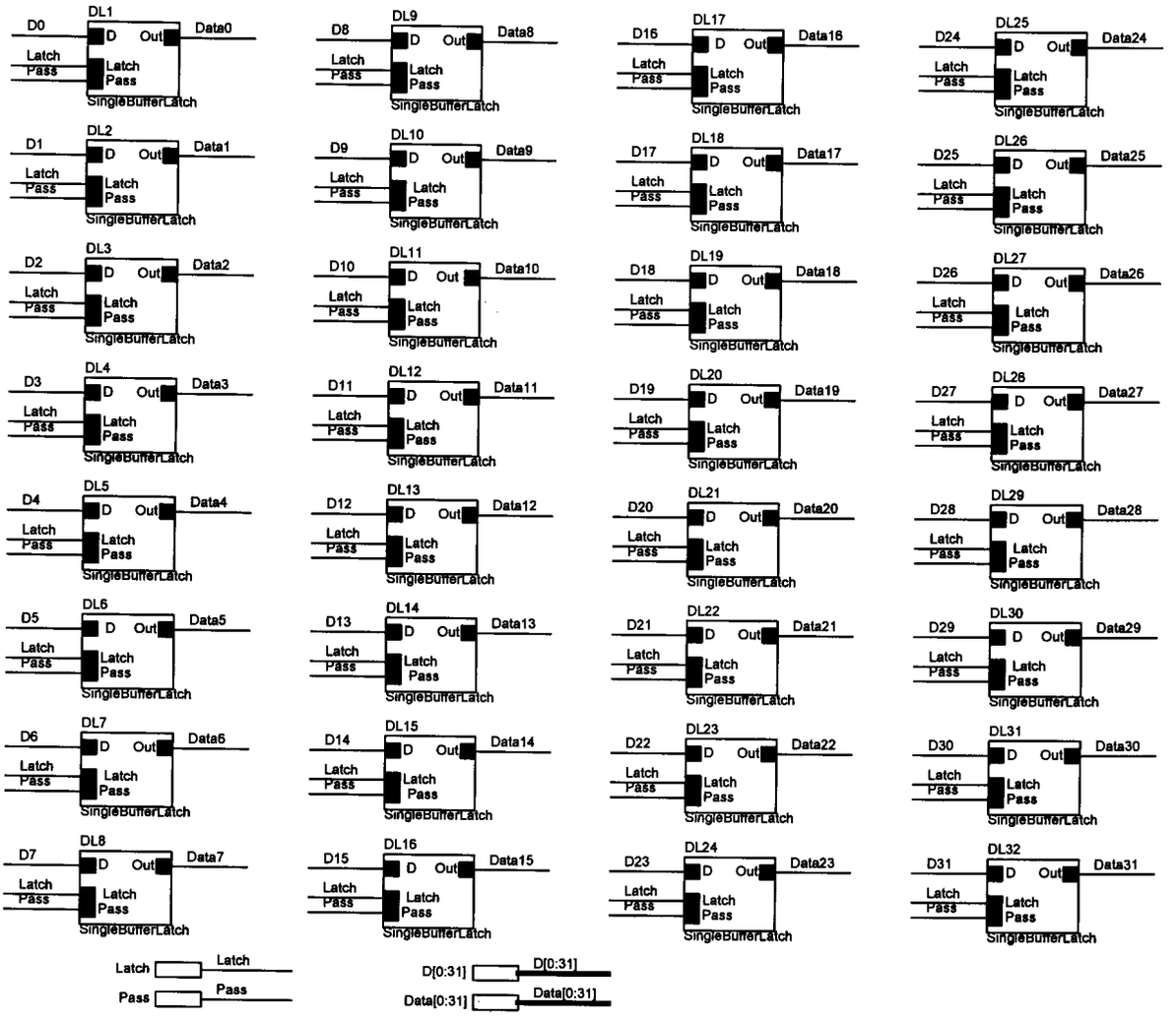


Figure C.4 Data Store: Latch system for the data inputs.

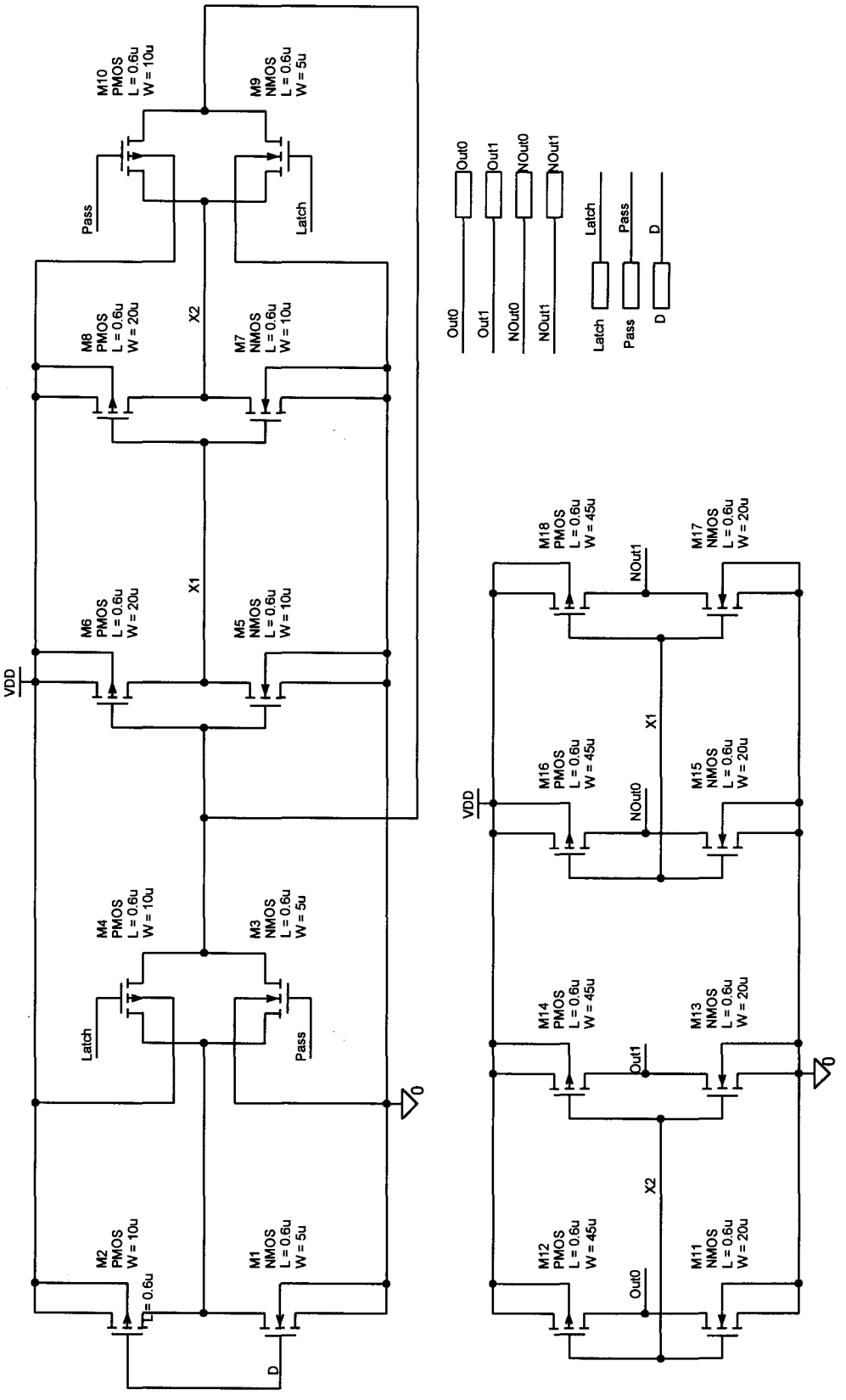


Figure C.5 Double Buffer Latch Strong: Transparent latch with double strong complementary output buffers.

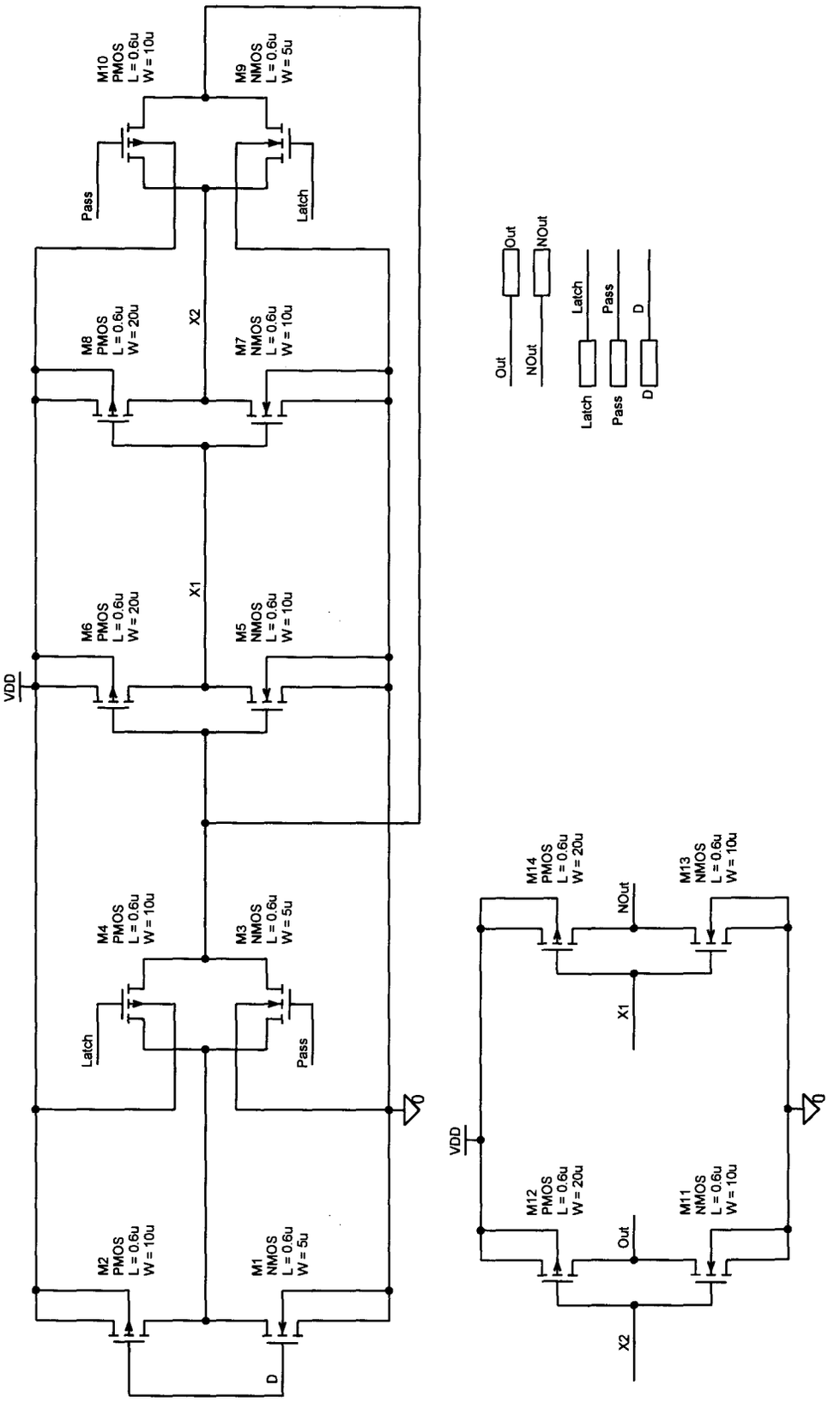


Figure C.6 Double Buffer Latch Weak: Transparent latch with weak complementary output buffers.

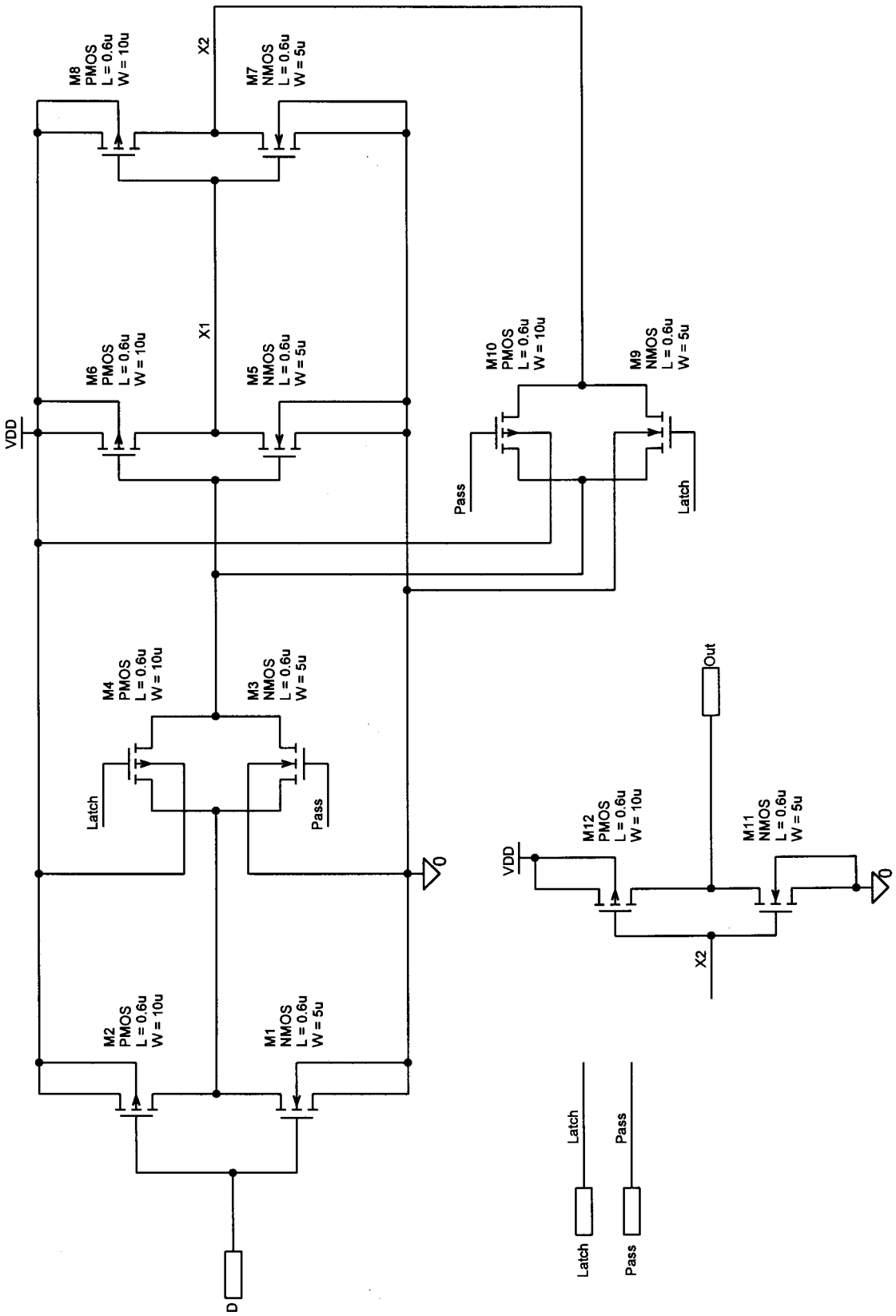


Figure C.7 Single Buffer Latch: Transparent latch with output buffer.

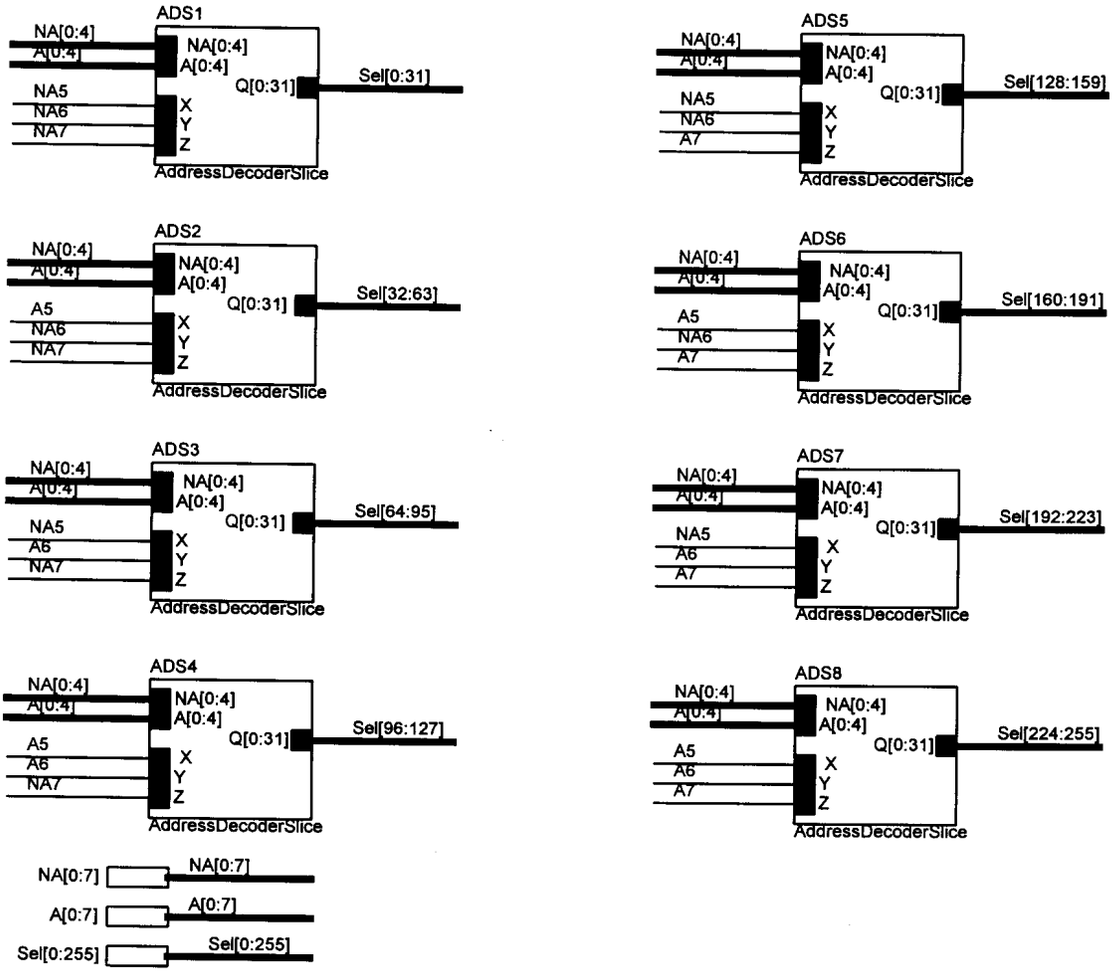


Figure C.8 Address Decoder: 8-256 line address decoder.

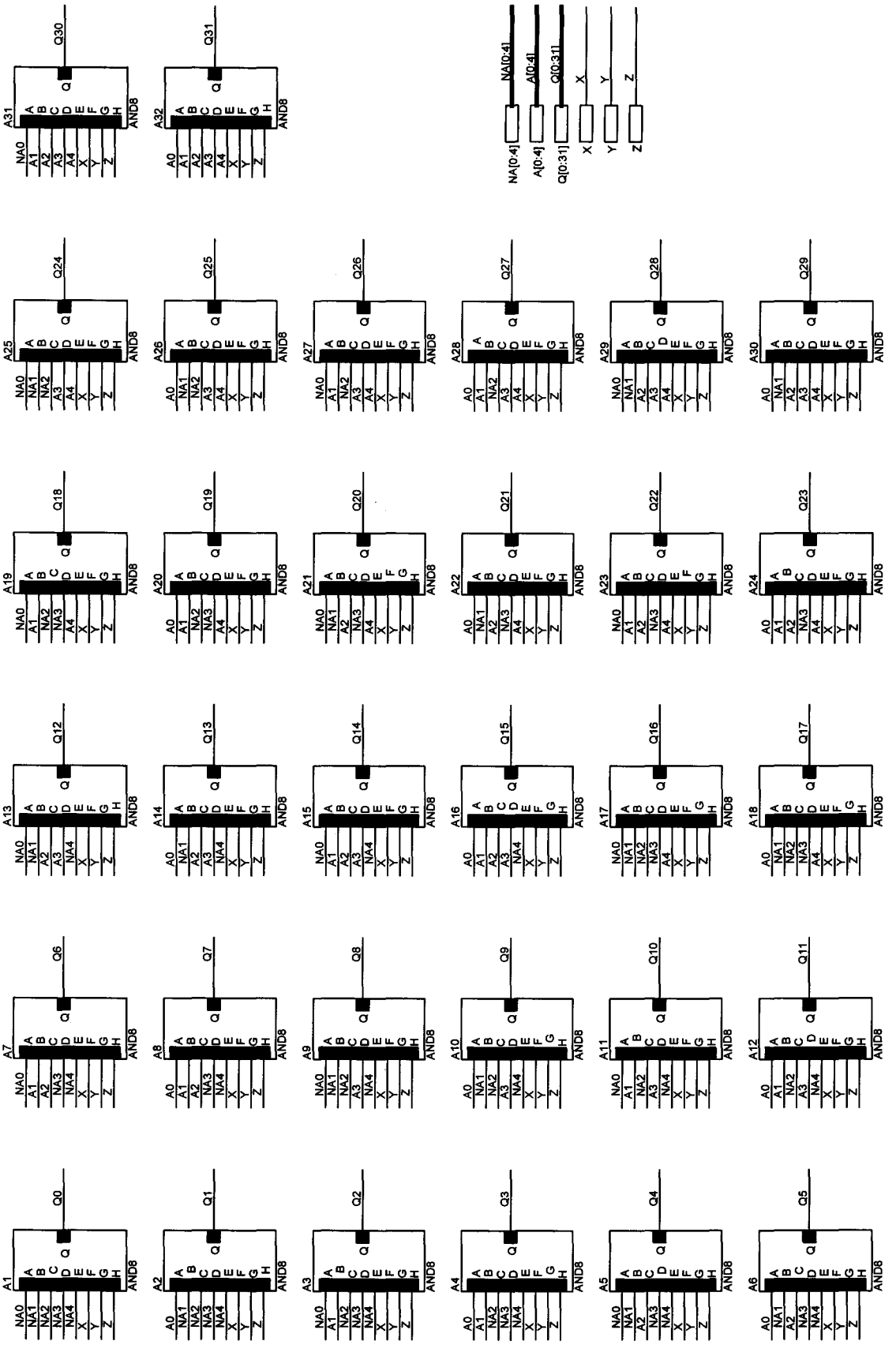


Figure C.9 Address Decoder Slice: 1/8 of the 8 - 256 line address decoder.

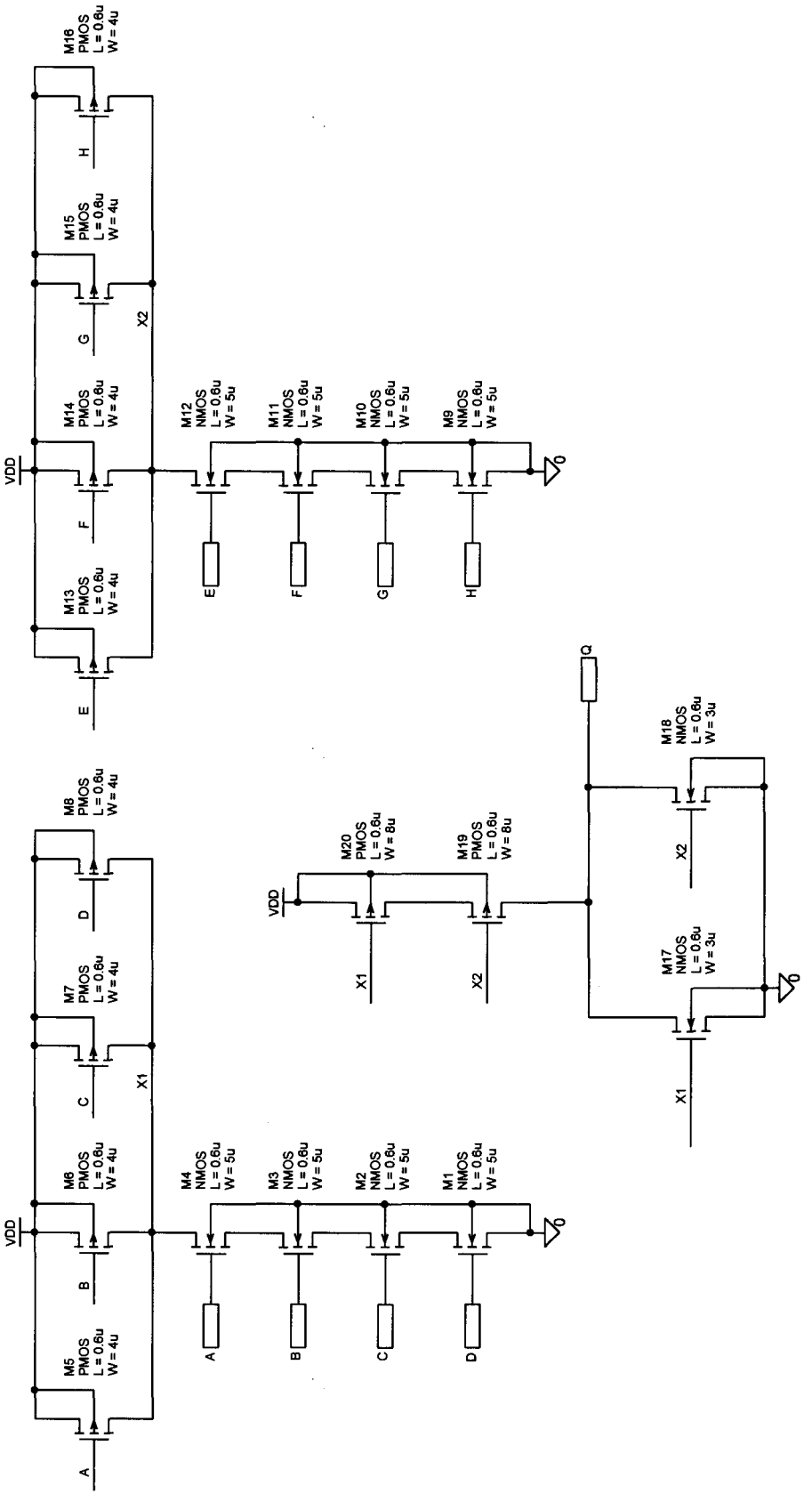


Figure C.10 And8: Eight input And-gate.

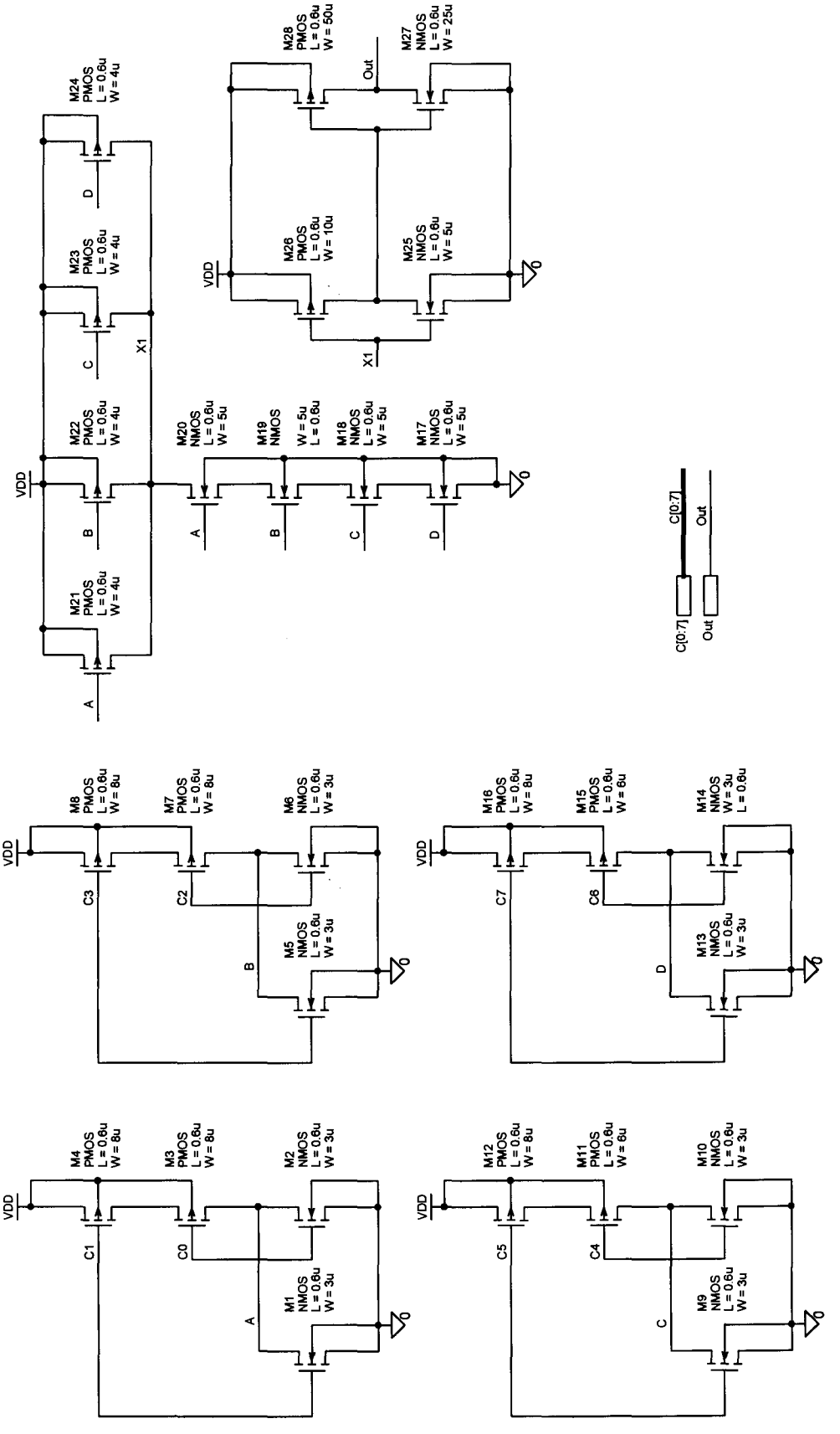


Figure C.11 Or8: Eight input Or-gate with buffered output.

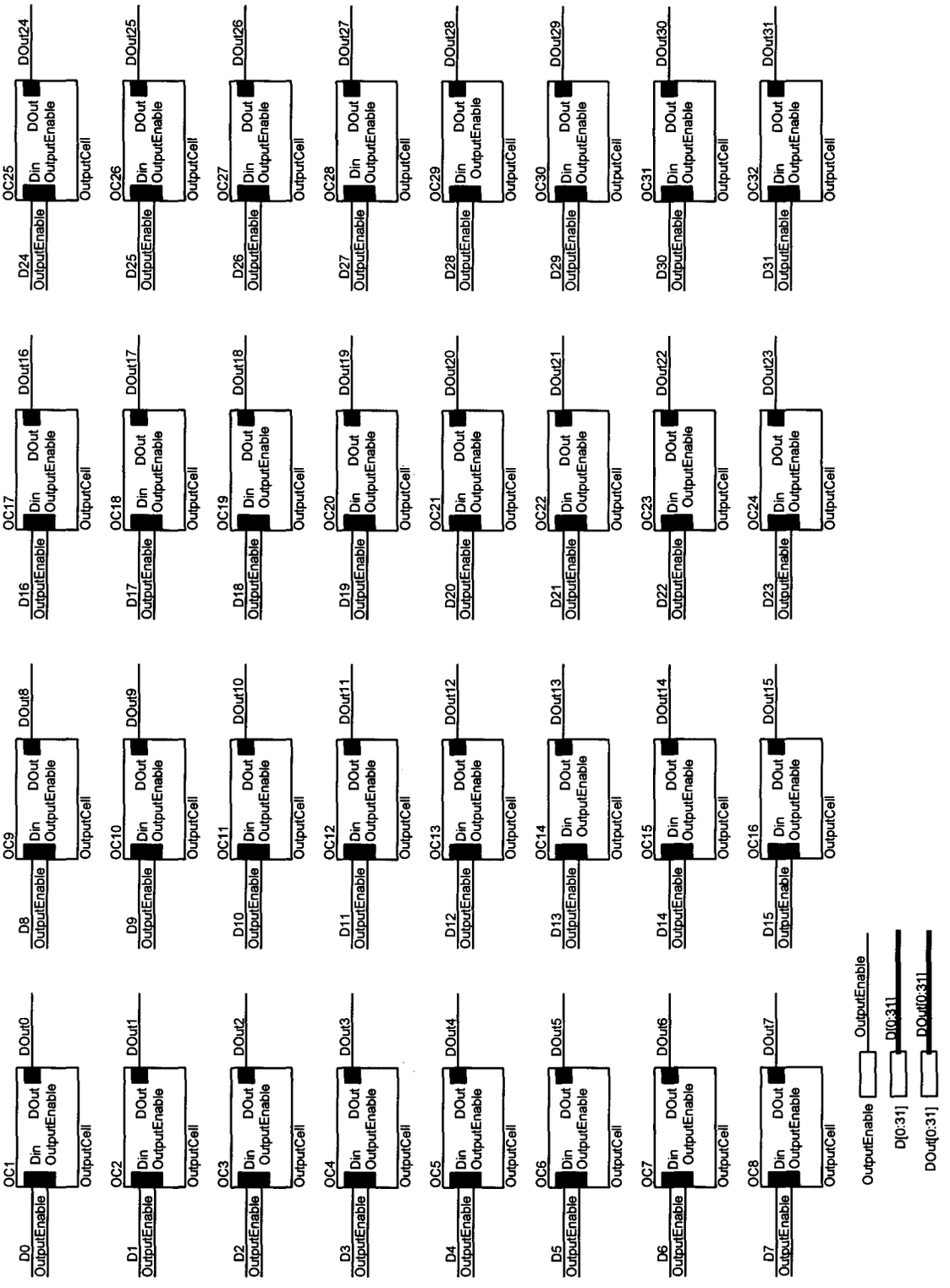


Figure C.12 Output Driver: Tri-state data output driver circuit array.

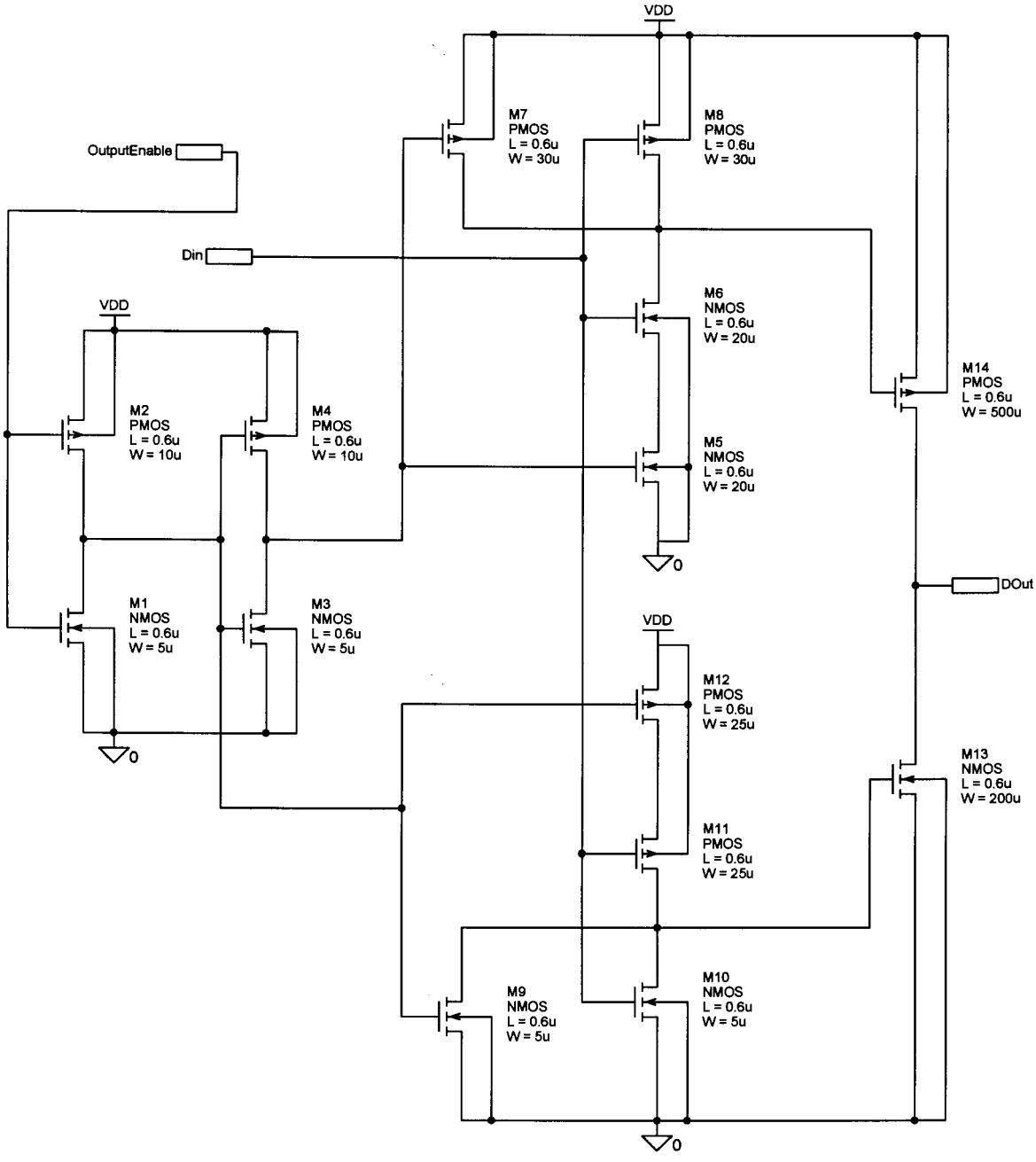


Figure C.13 Output Cell: Tri-state data output driver circuit.

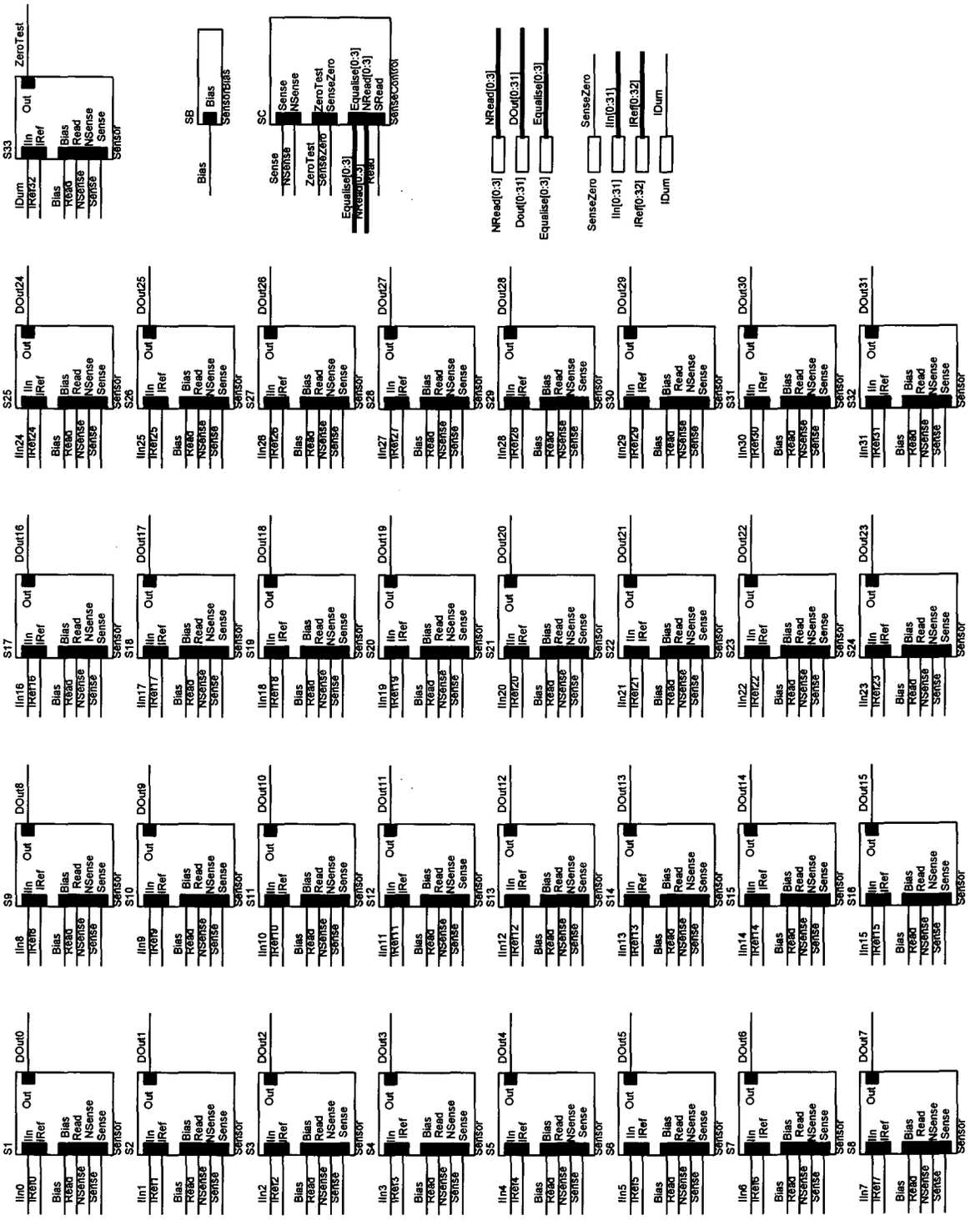


Figure C.14 Sense Amplifier: Complete sense amplifier system.

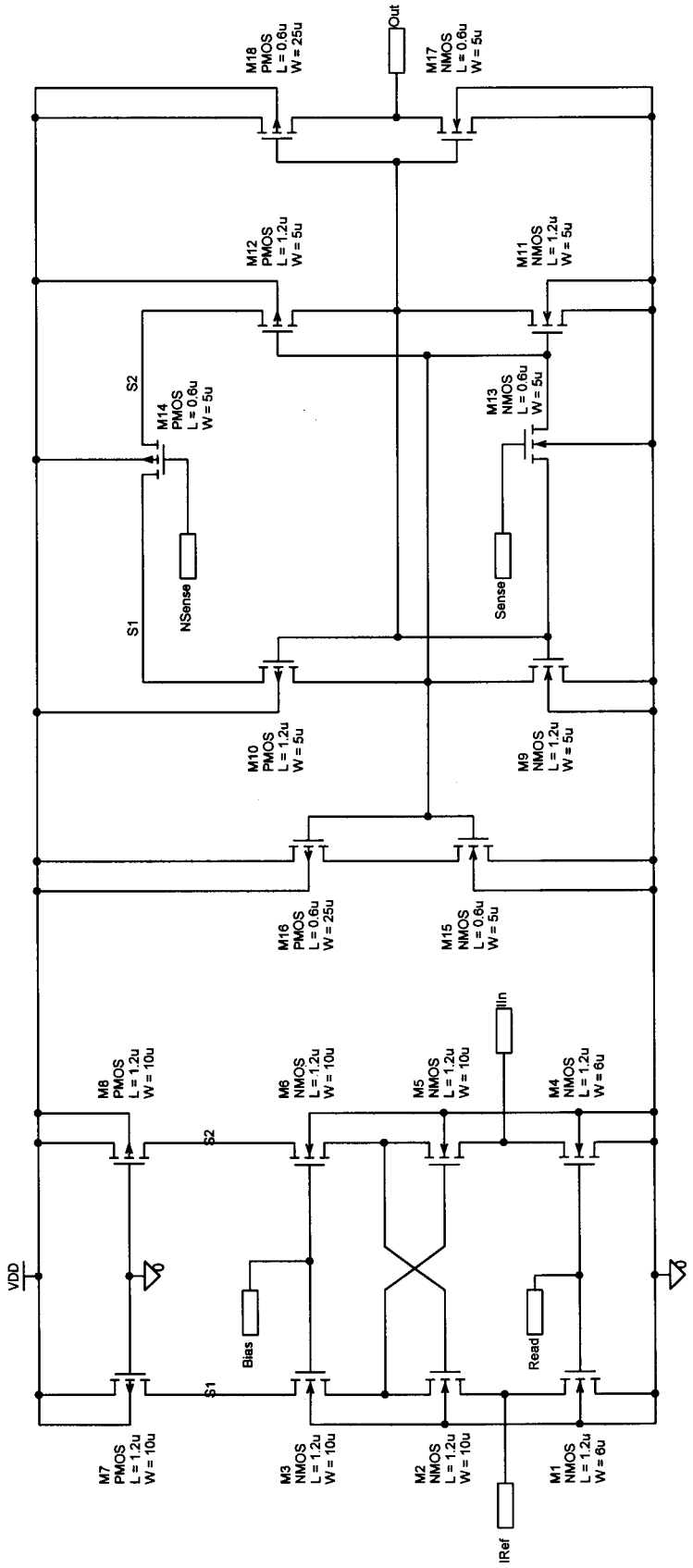


Figure C.15 Sensor: Current conveyor and clamped bit line sense amplifier.

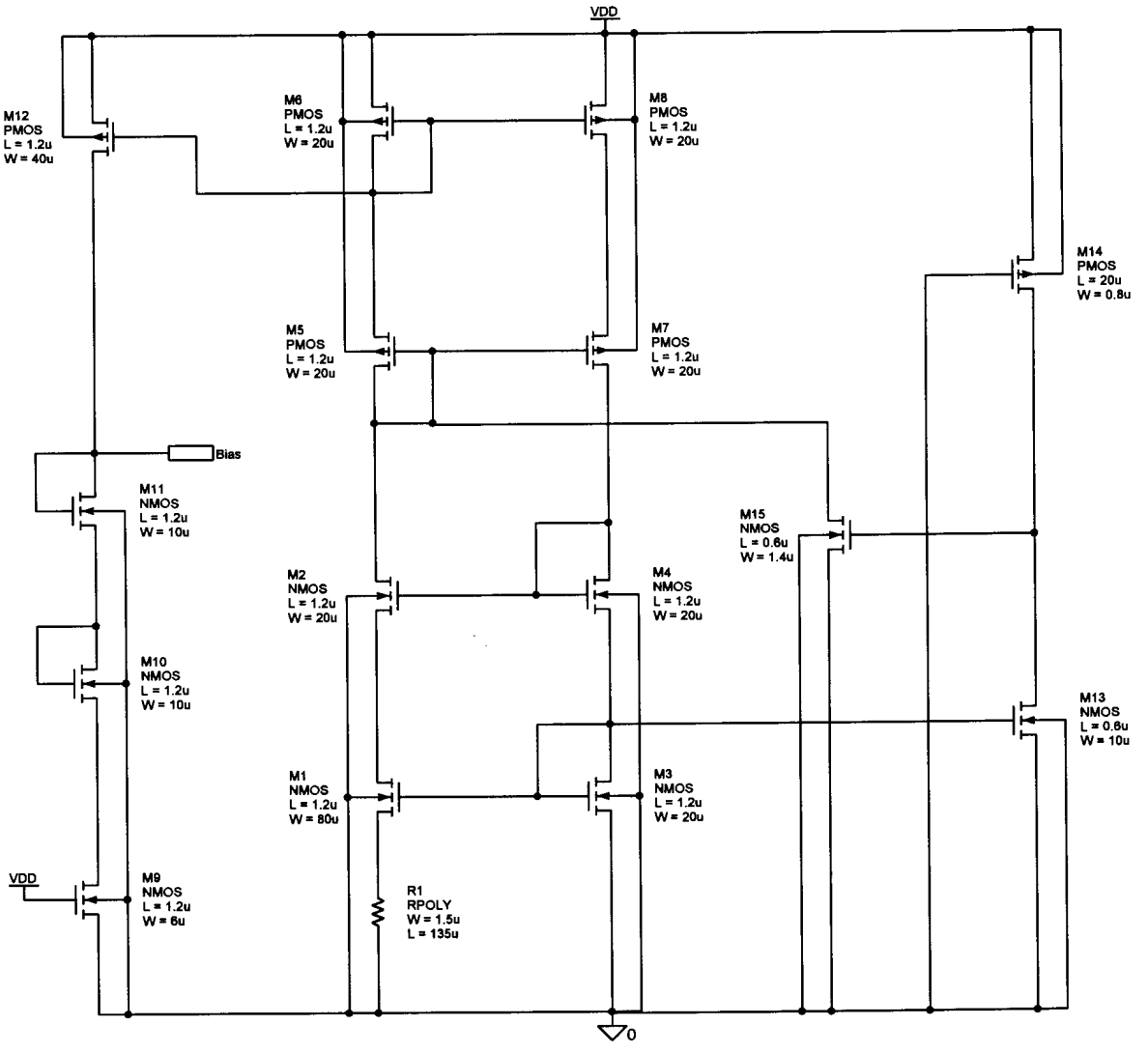


Figure C.16 Sensor Bias: Constant transconductance bias network for the current sense amplifier.

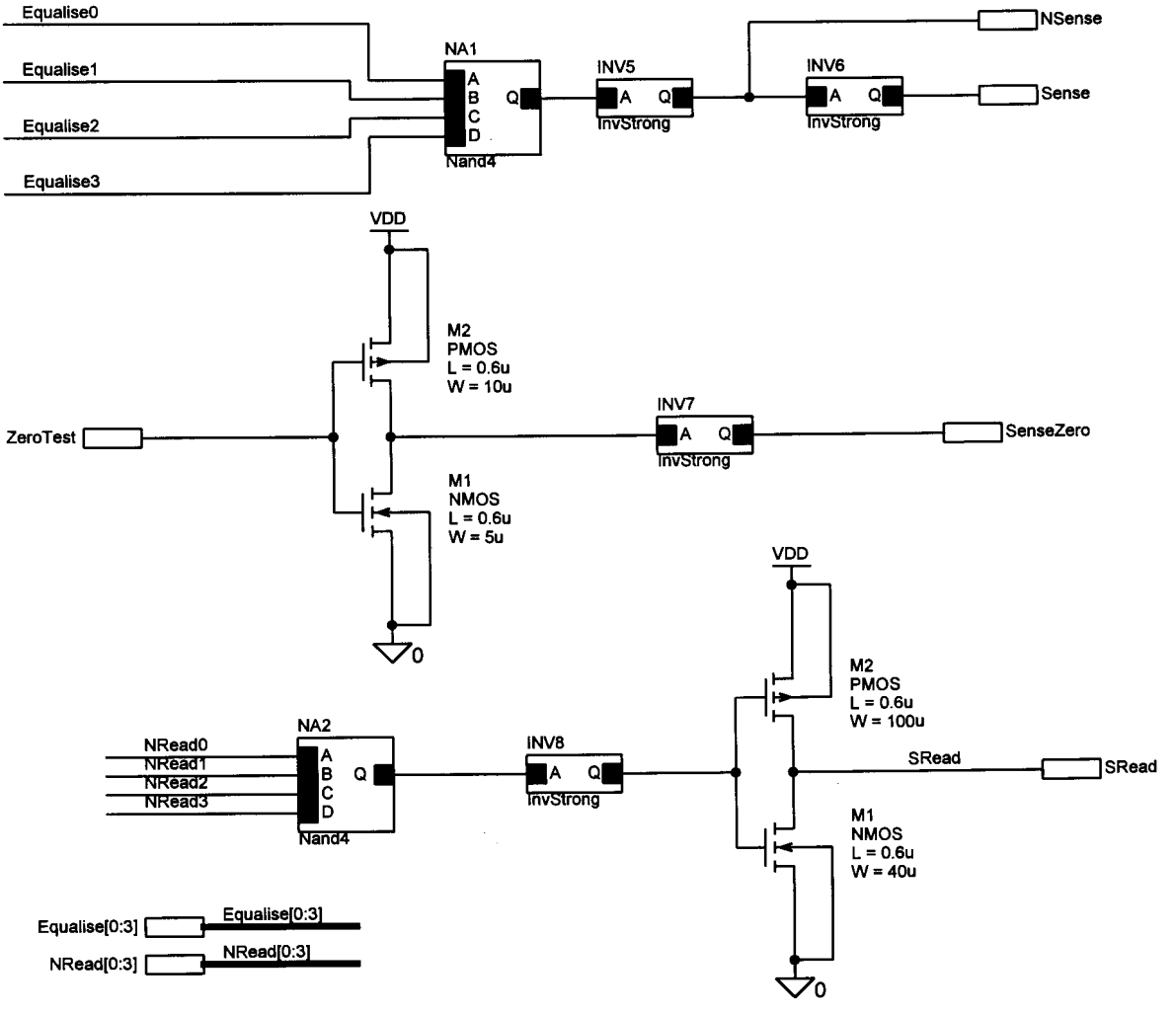


Figure C.17 Sense Control: Sense amplifier peripherals and control circuits.

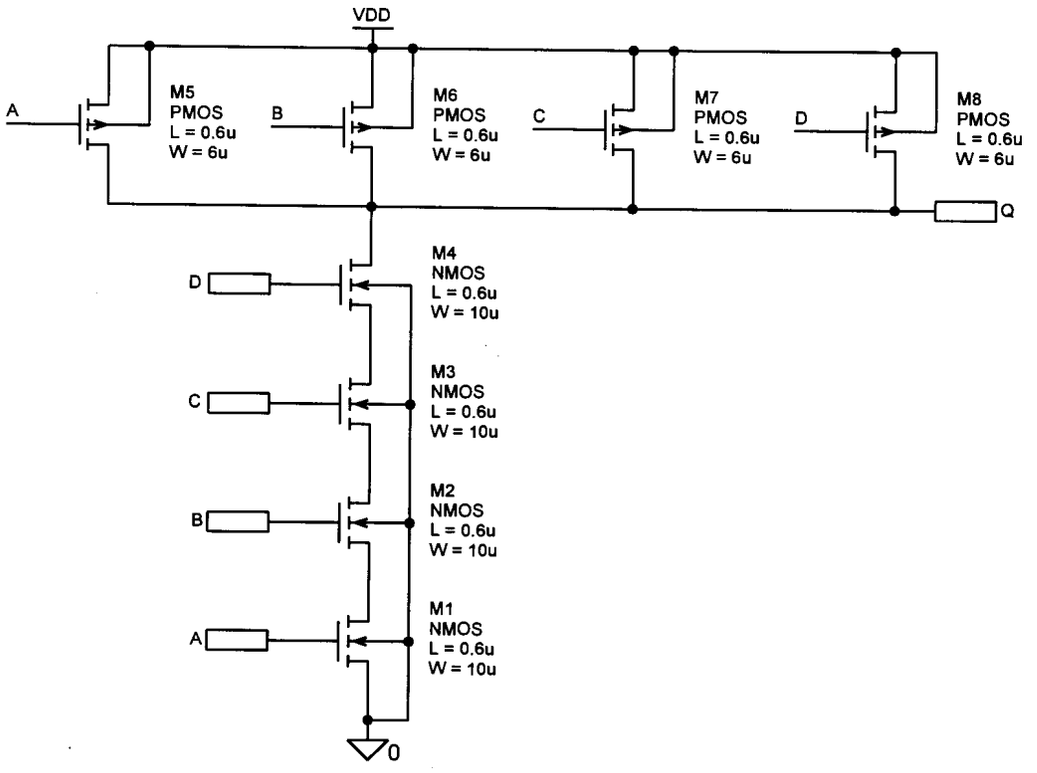


Figure C.18 Nand4: Four-input Nand-gate.

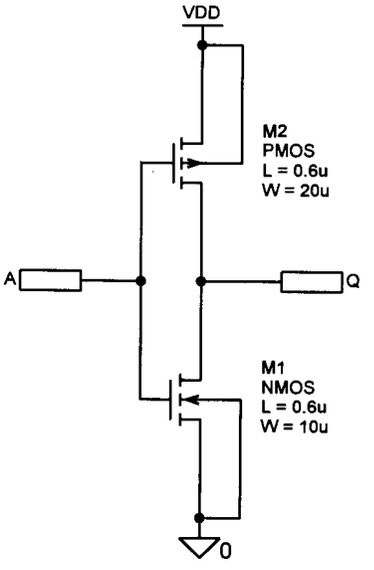


Figure C.19 Inv Strong: High driving strength inverter.

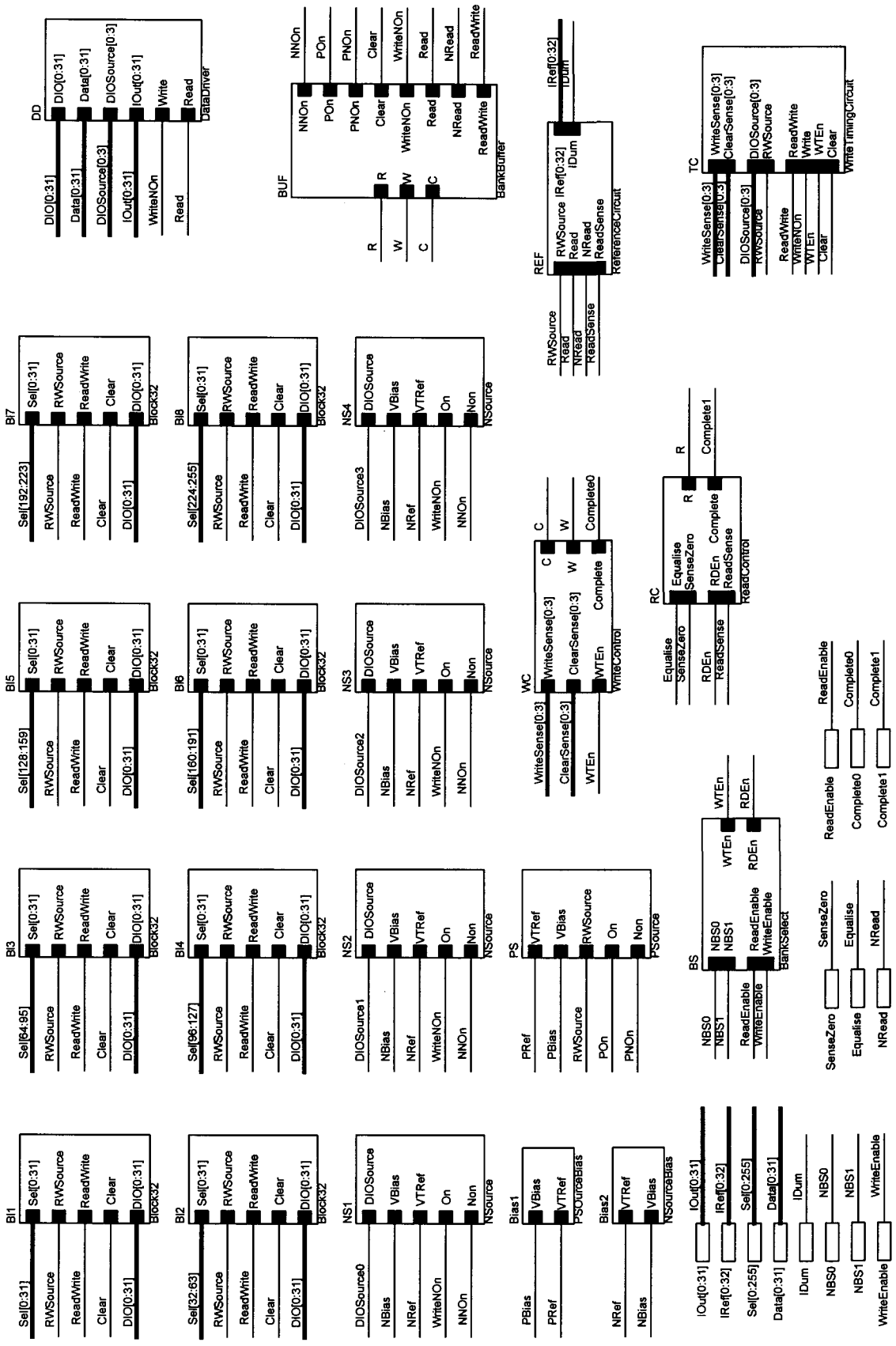


Figure C.20 Bank 256x32: One memory bank with all the associated peripheral circuits.

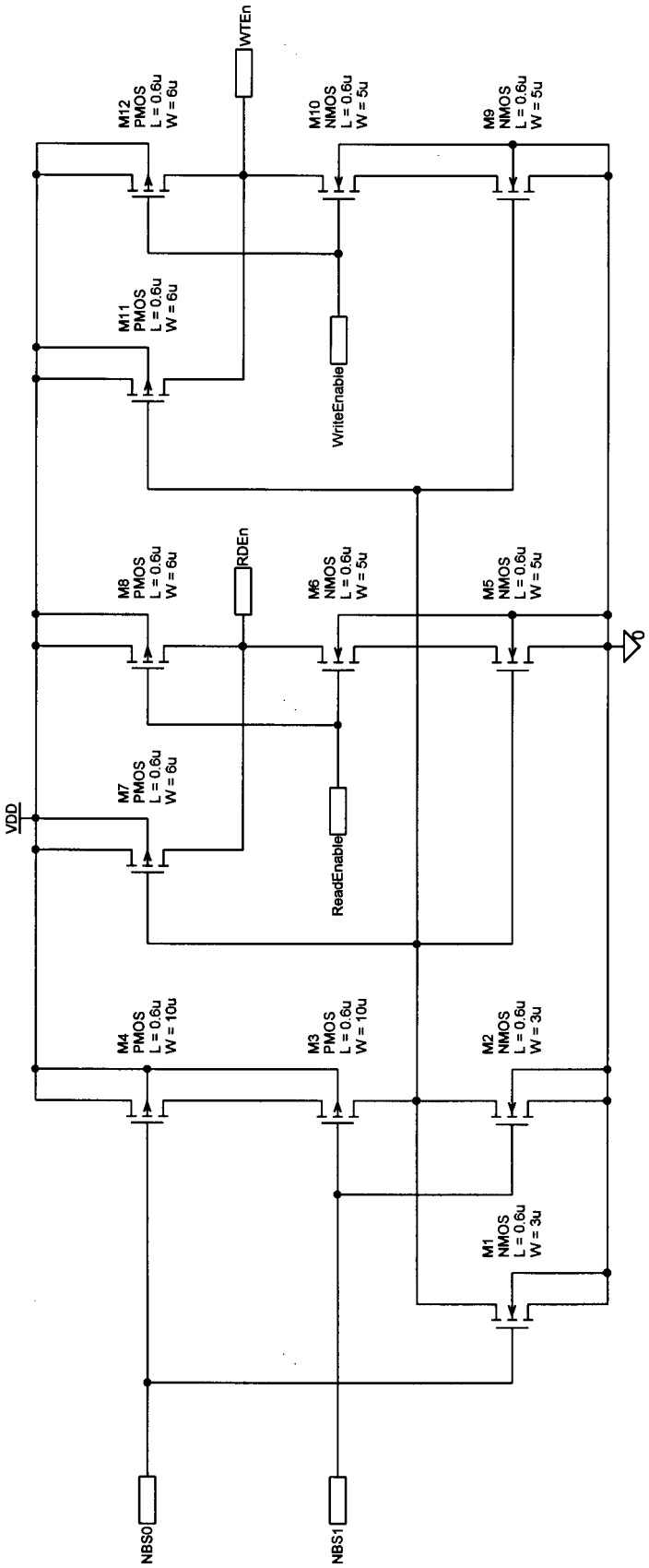


Figure C.21 Bank Select: Bank selection via control signal masking.

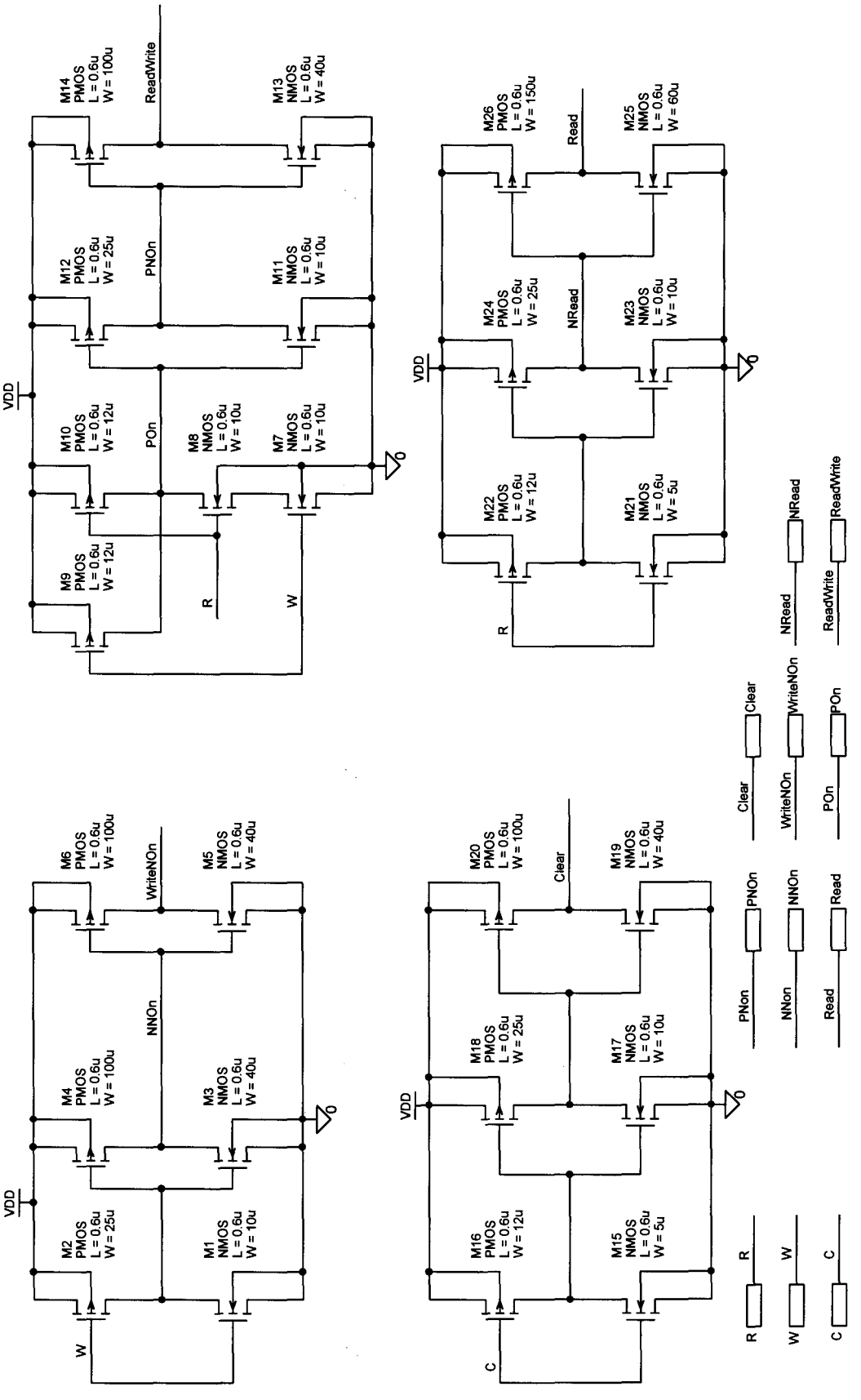


Figure C.22 Bank Buffer: Buffering system for the outputs of the control circuits.

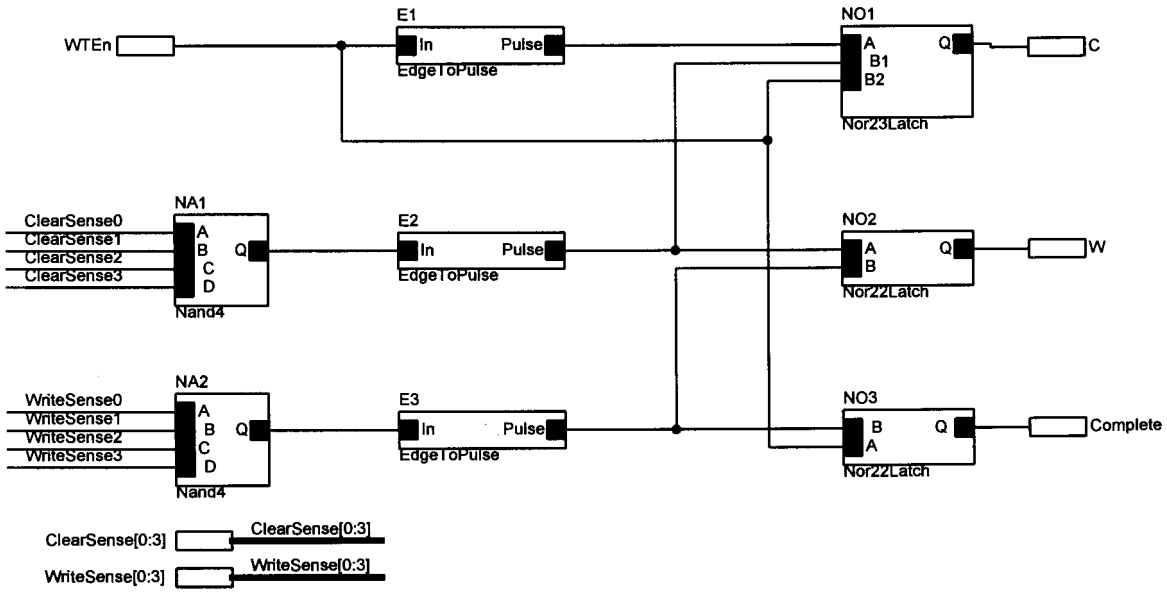


Figure C.23 Write Control: Write cycle control signal sequencer.

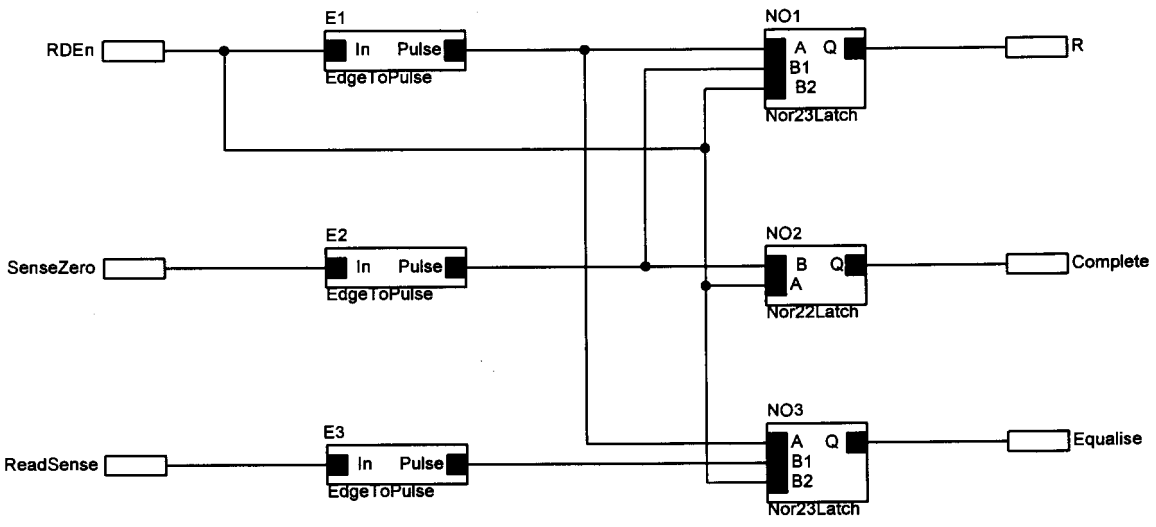


Figure C.24 Read Control: Read cycle control signal sequencer.

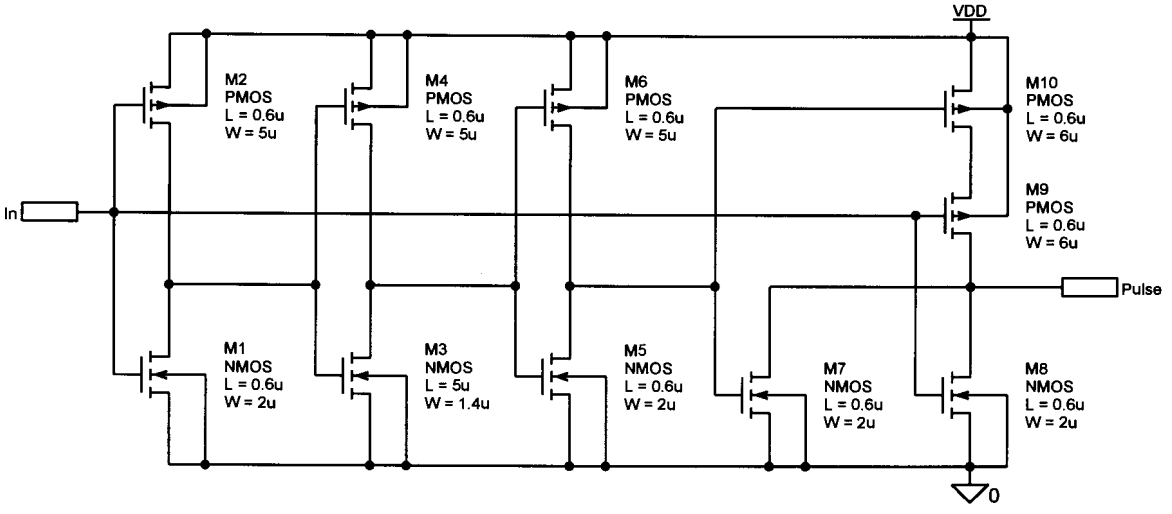


Figure C.25 Edge To Pulse: Falling edge to positive pulse converter.

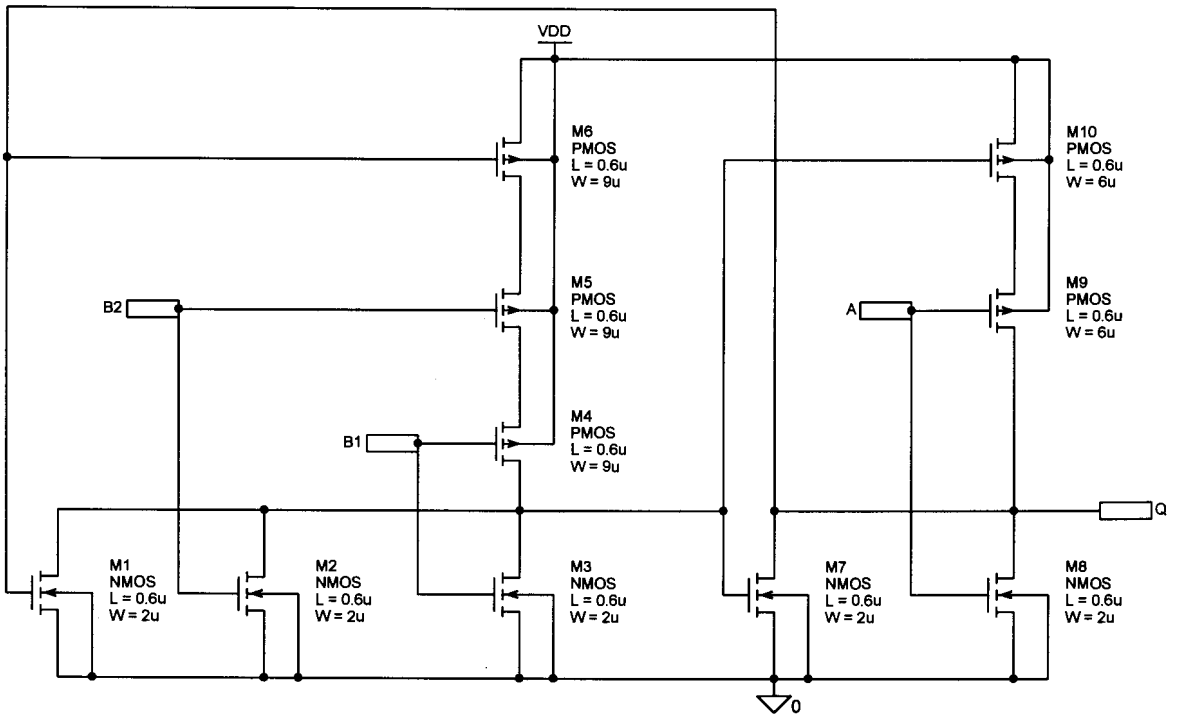


Figure C.26 Nor23 Latch: Set-reset latch with one set and two reset inputs.

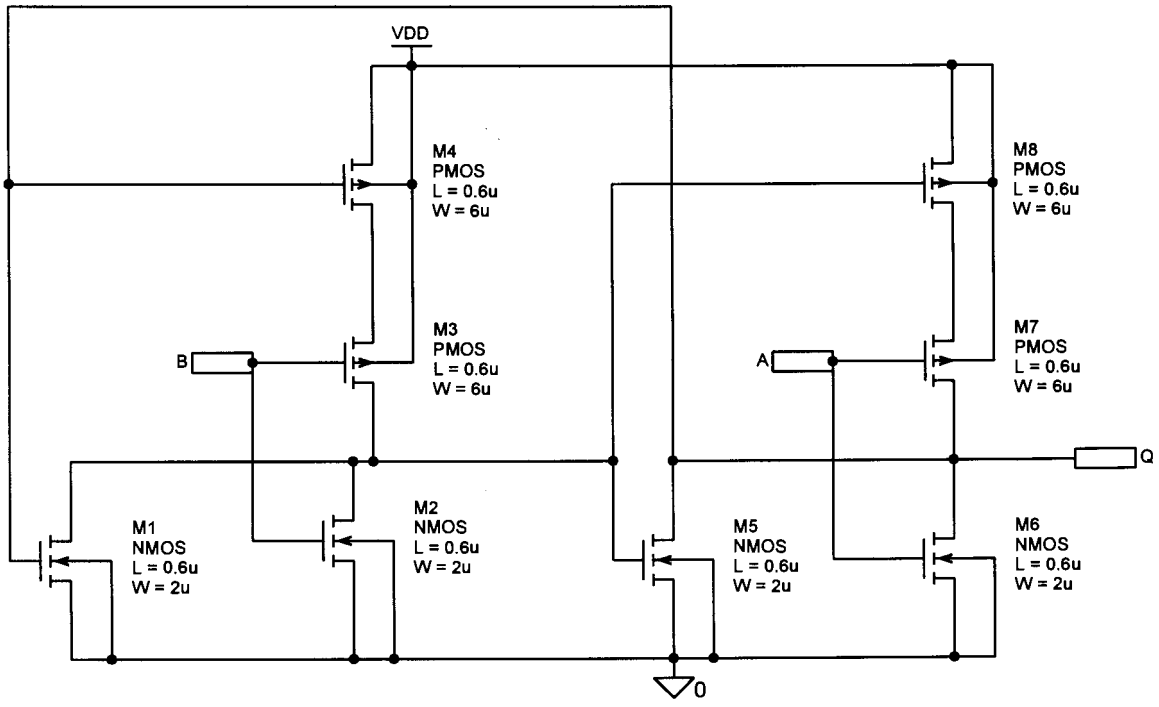


Figure C.27 Nor22 Latch: Set-reset latch with one set and one reset input

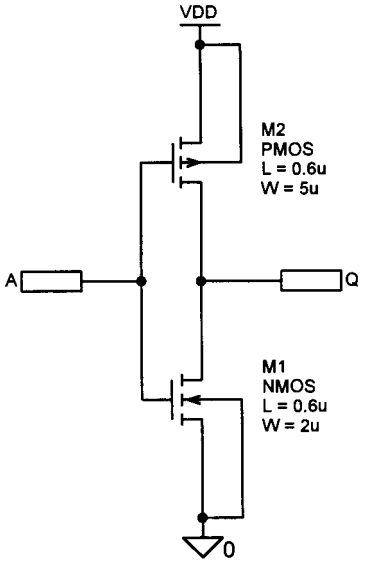


Figure C.28 Inv Weak: Low driving strength inverter.

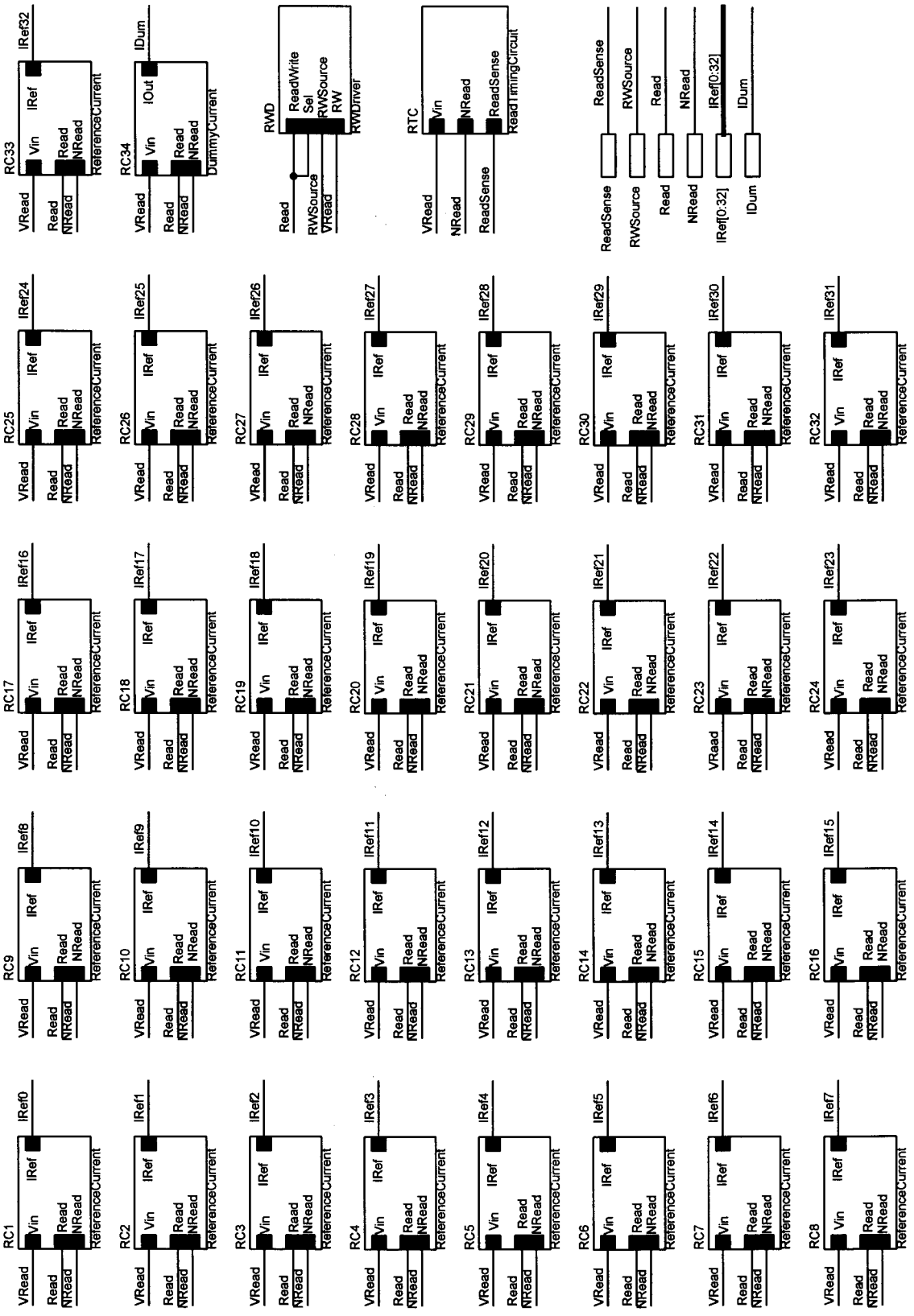


Figure C.29 Reference Circuit: Array of reference current generators with driving circuit.

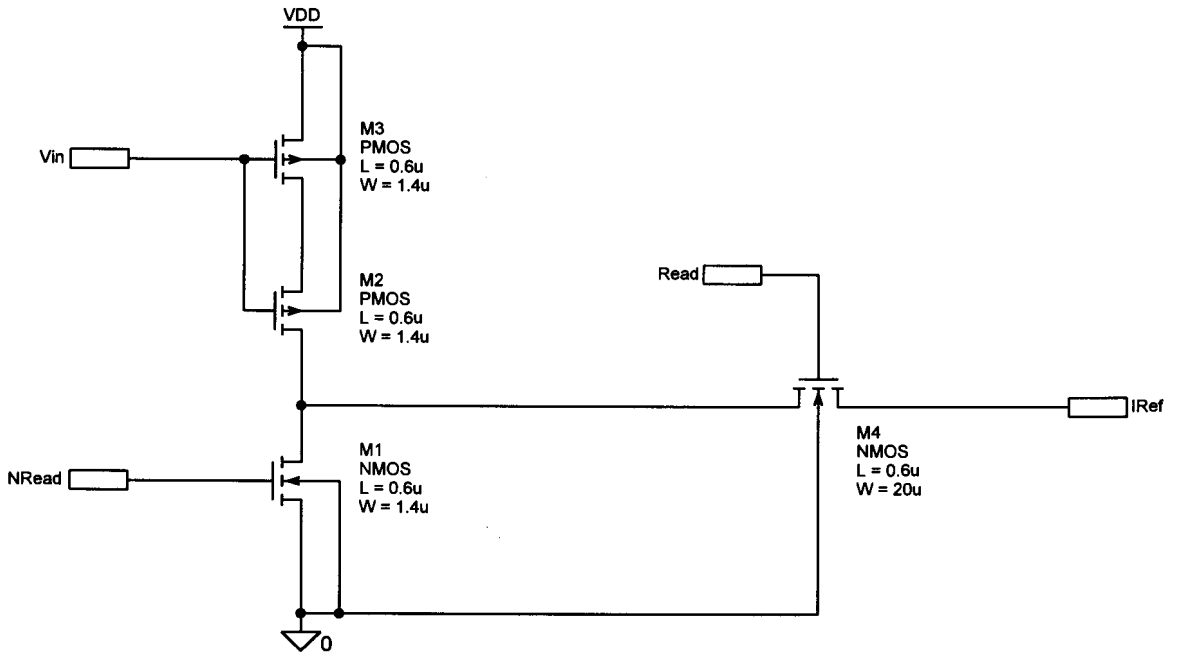


Figure C.30 Reference Current: Reference current generator with current mode multiplexing access device added.

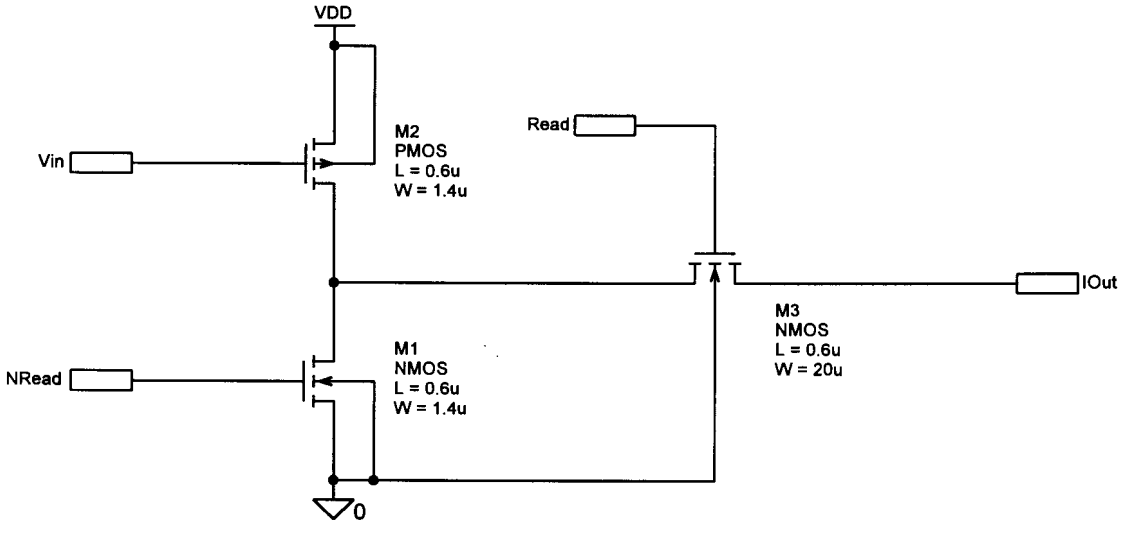


Figure C.31 Dummy Current: Dummy read current generator with current mode multiplexing access device added.

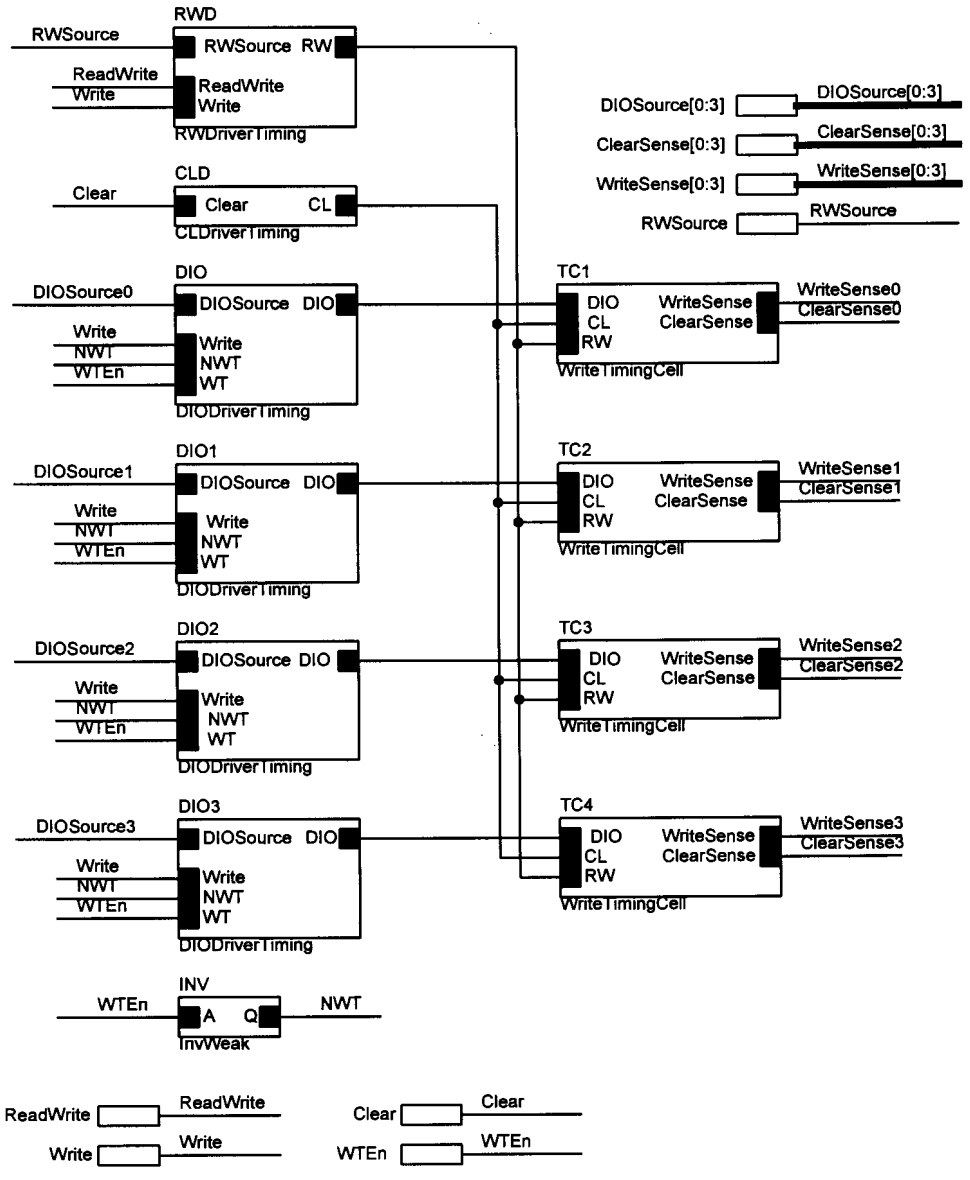


Figure C.32 Write Timing Circuit: Circuit used for sensing the timing of the write cycle.

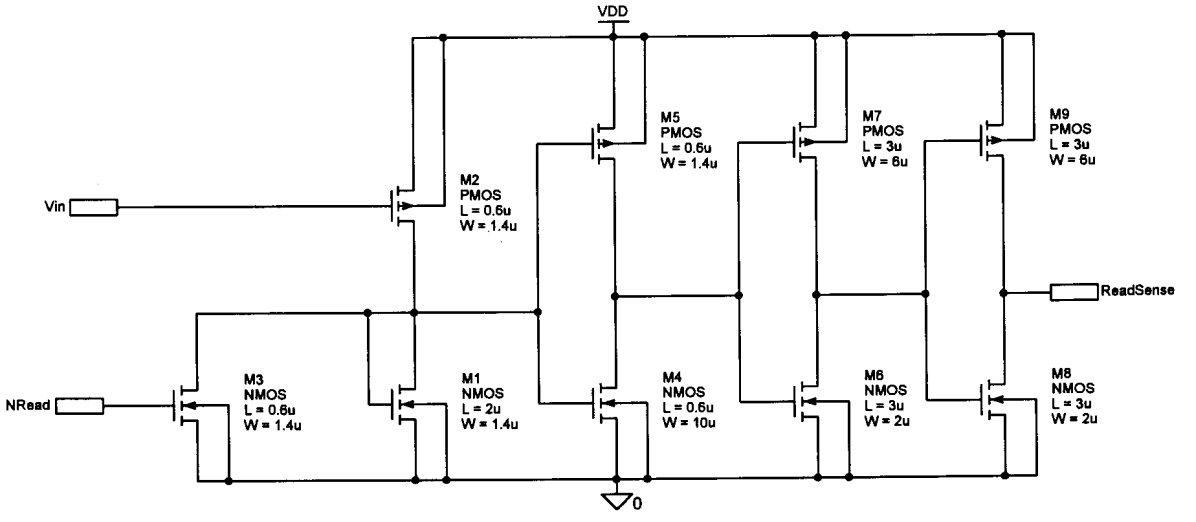


Figure C.33 Read Timing Circuit: Circuit to sense the completion of the initial read cycle phase.

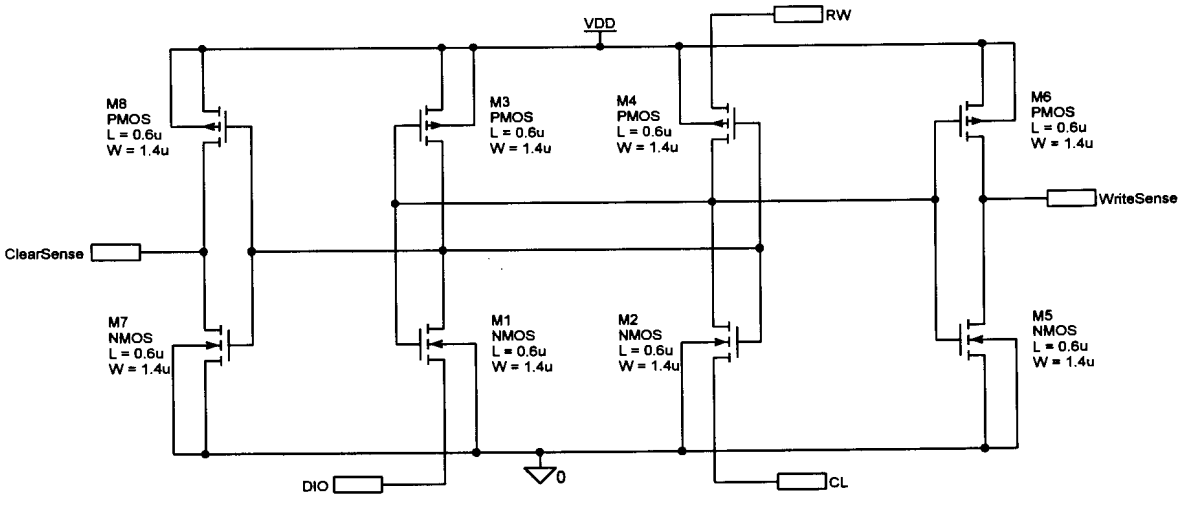


Figure C.34 Write Timing Cell: Dummy cell with access inverters to sense the write phases.

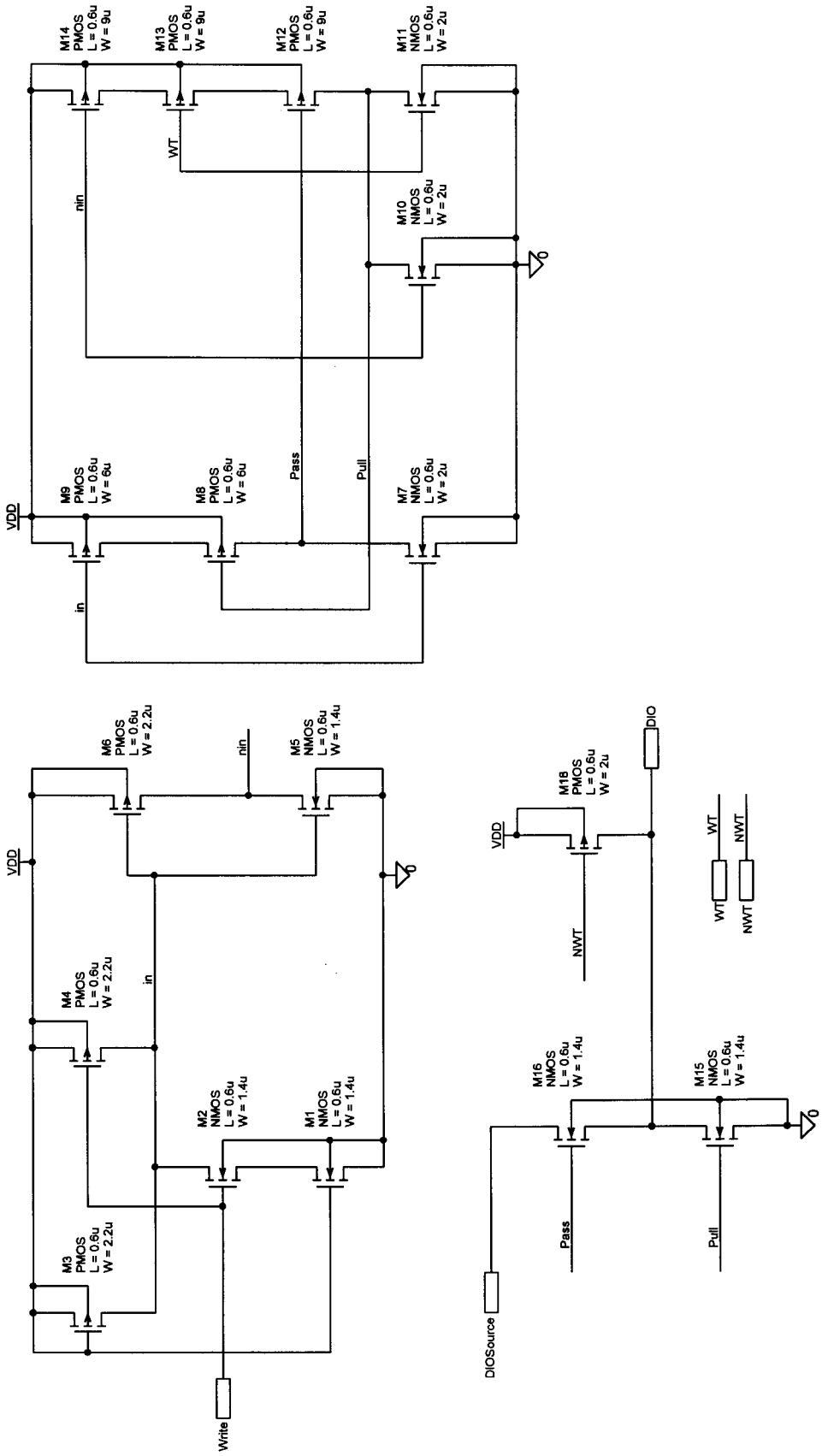


Figure C.35 DIO Driver Timing: *DIO*-line driver for the write cycle timing cells.

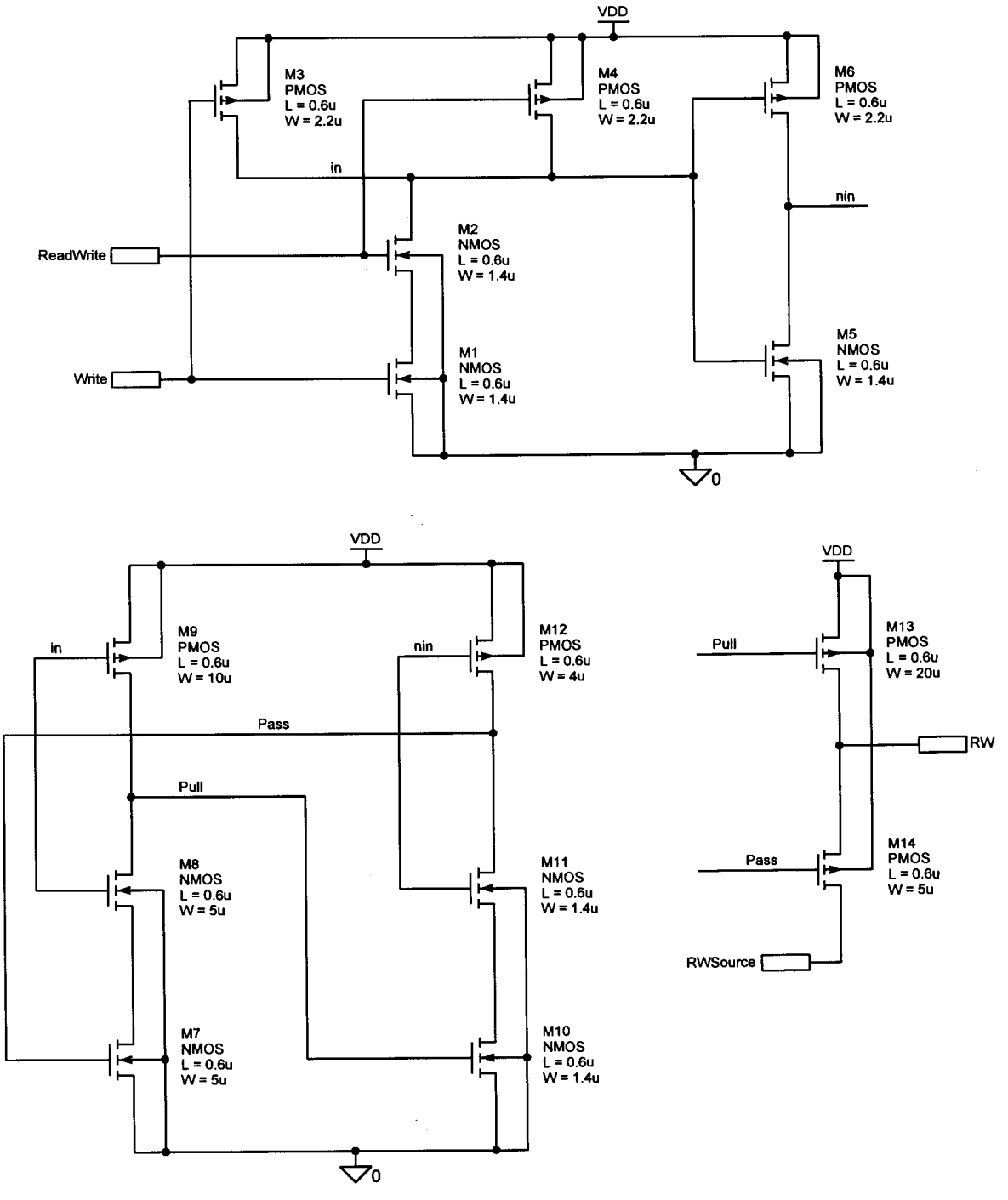


Figure C.36 RW Driver Timing: *RW*-line driver for the write cycle timing cells.

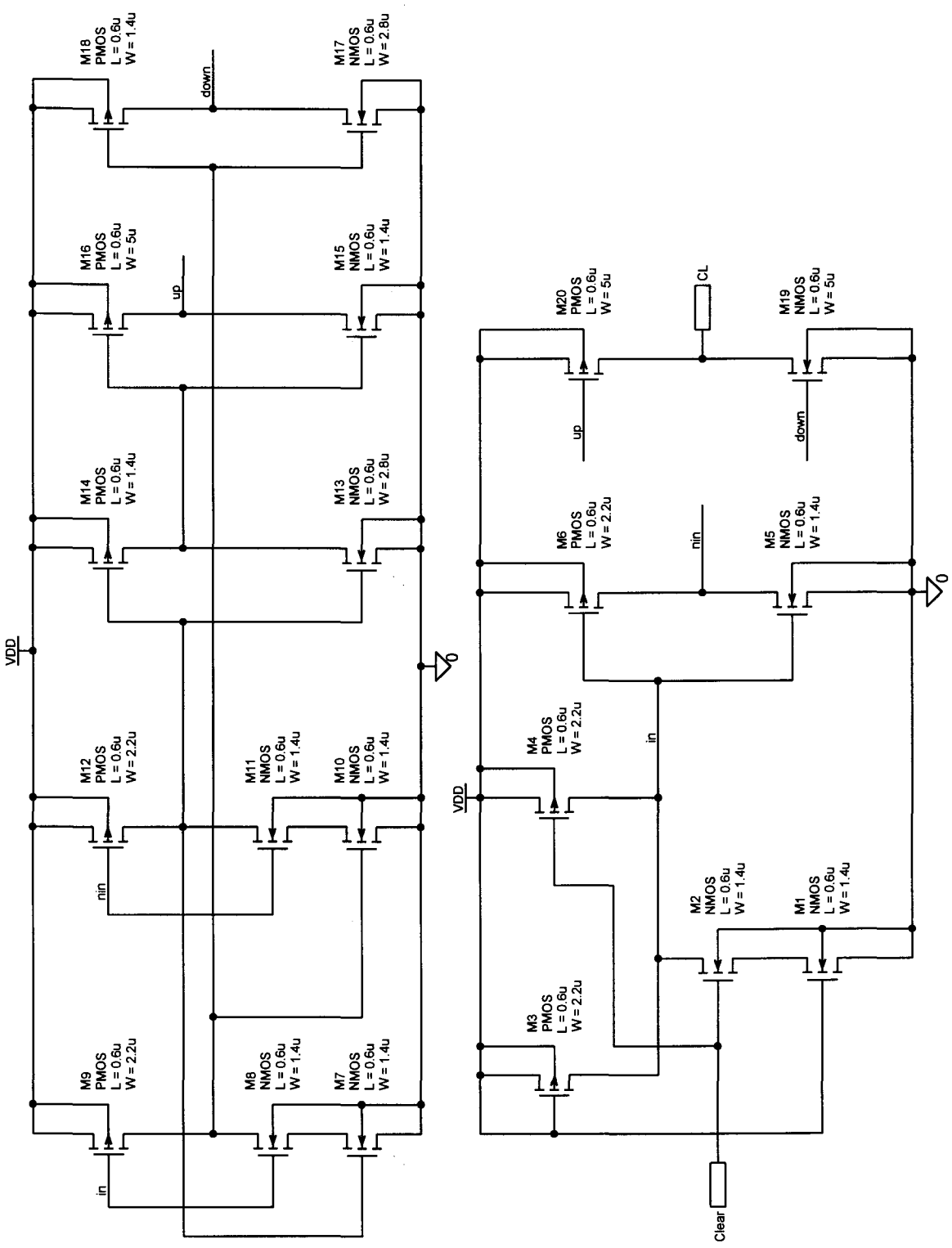


Figure C.37 CL Driver Timing: *CL*-line driver for the write cycle timing cells.

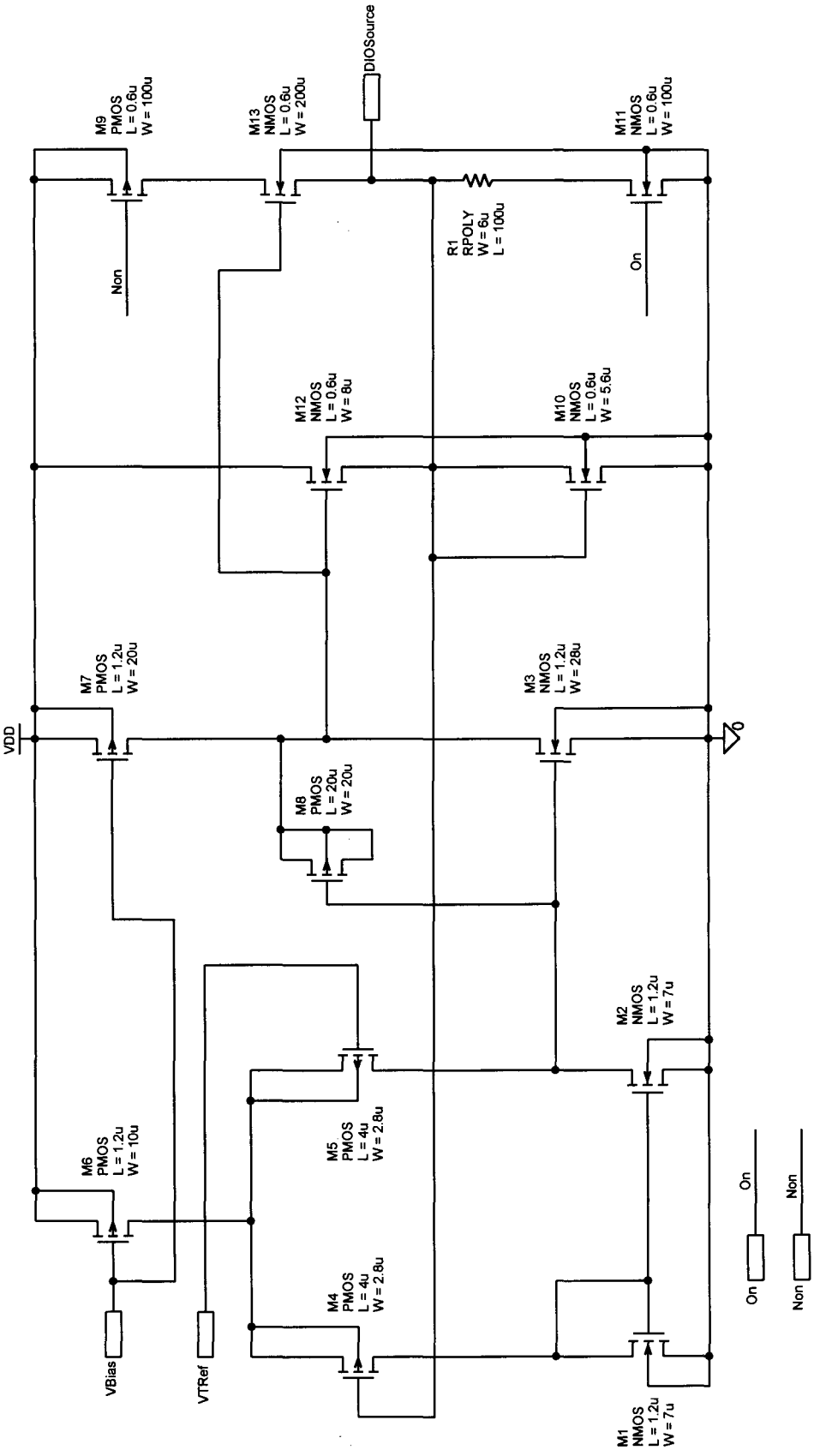


Figure C.38 NSource: *DIO*-line driver op-amp and low-impedance driver circuit.

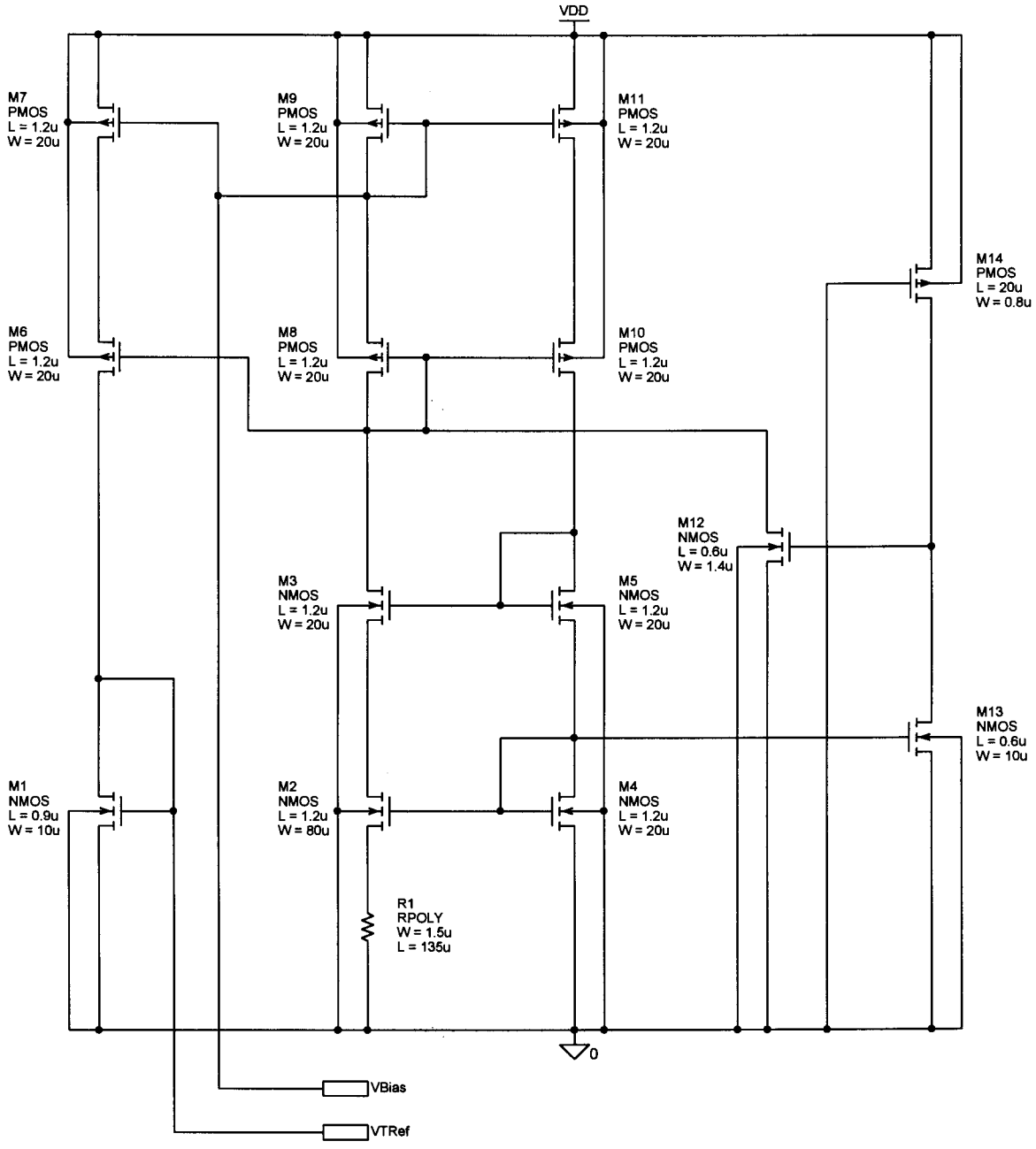


Figure C.39 NSource Bias: Reference voltage generator and bias network for the *DIO*-line driver circuit.

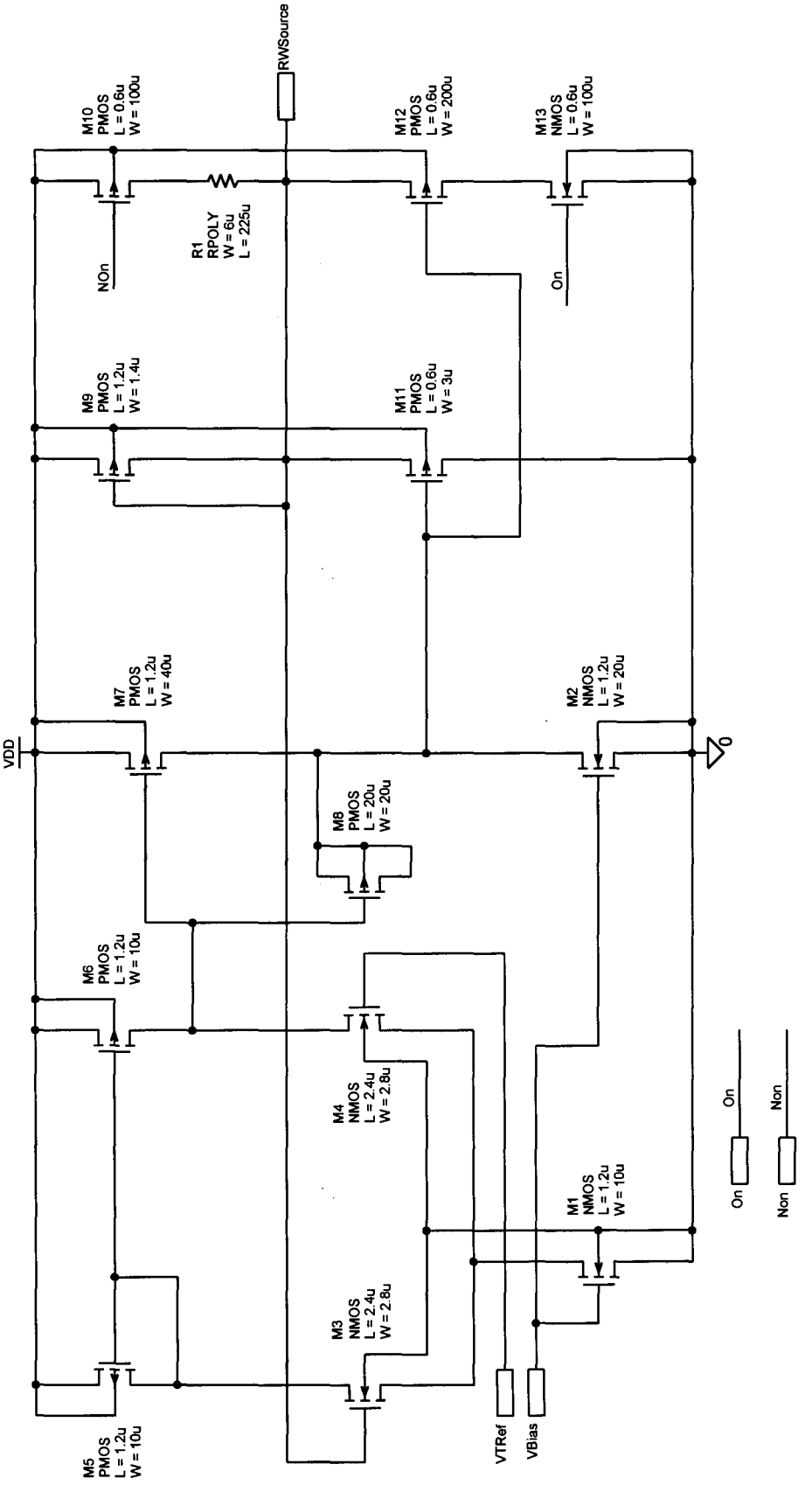


Figure C.40 PSource: *RW*-line driver op-amp and low-impedance driver circuit.

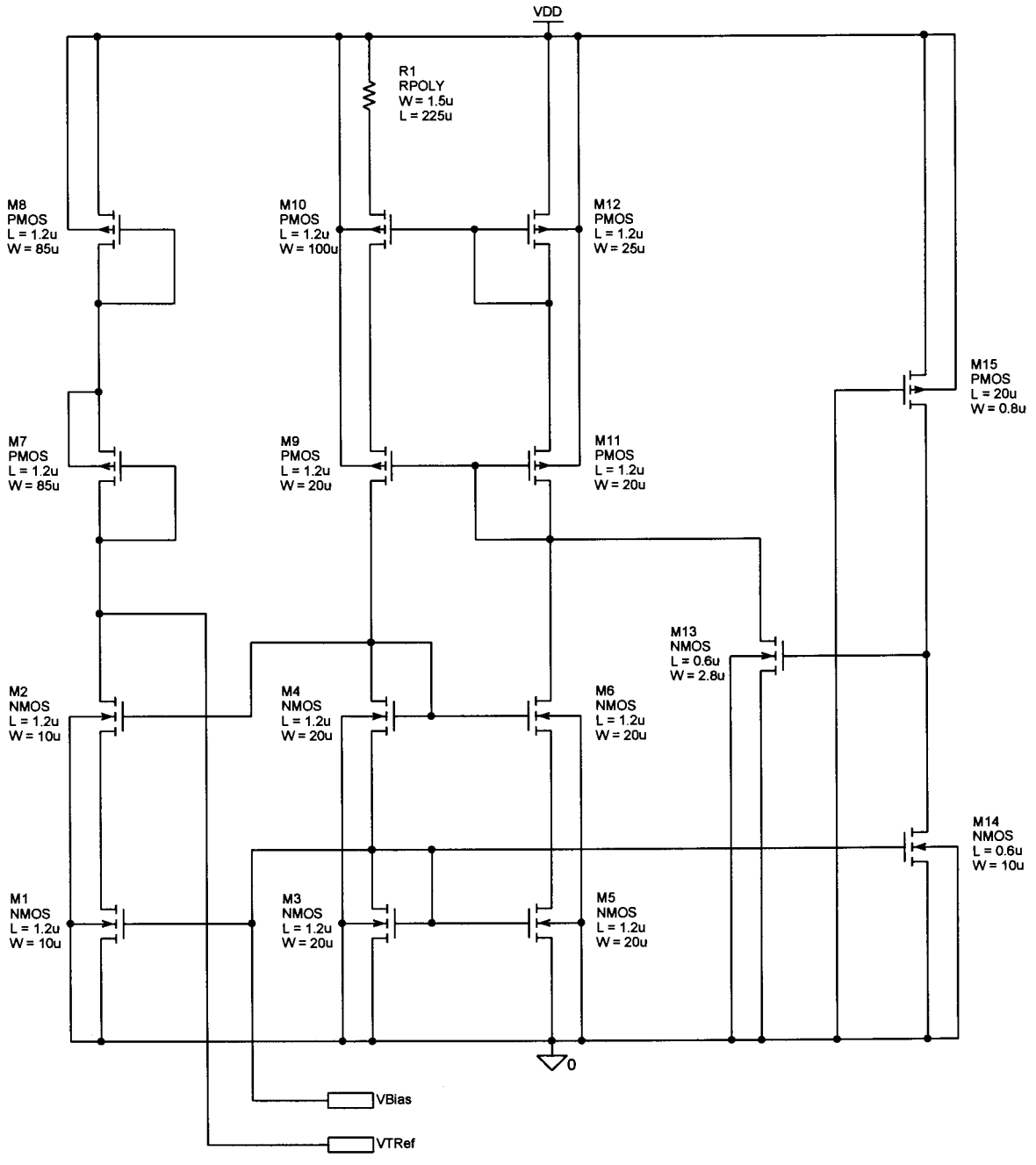


Figure C.41 PSource Bias: Reference voltage generator and bias network for the *RW*-line driver circuit.

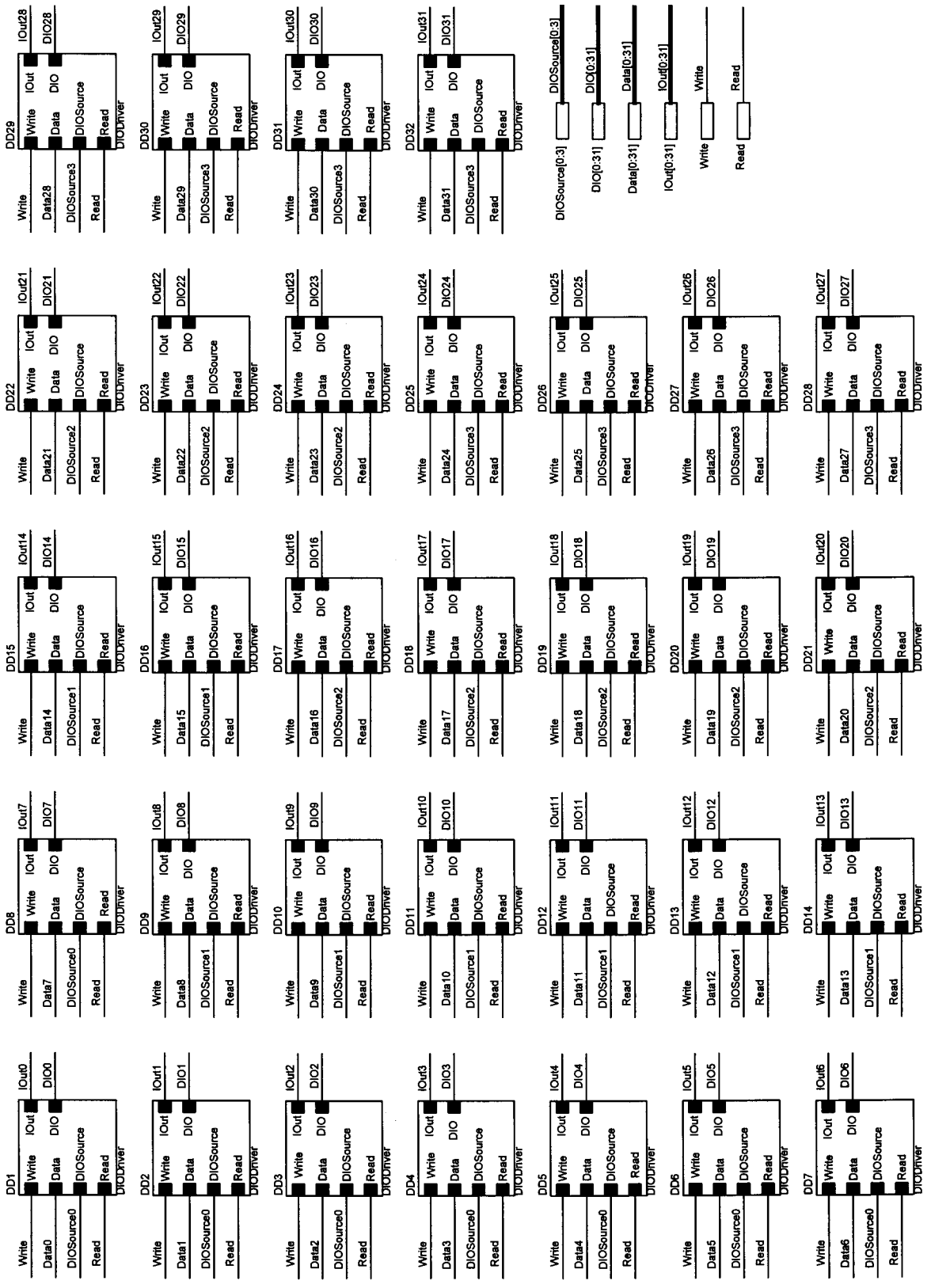


Figure C.42 Data Driver: Array of *DIO*-line driver switching circuits.

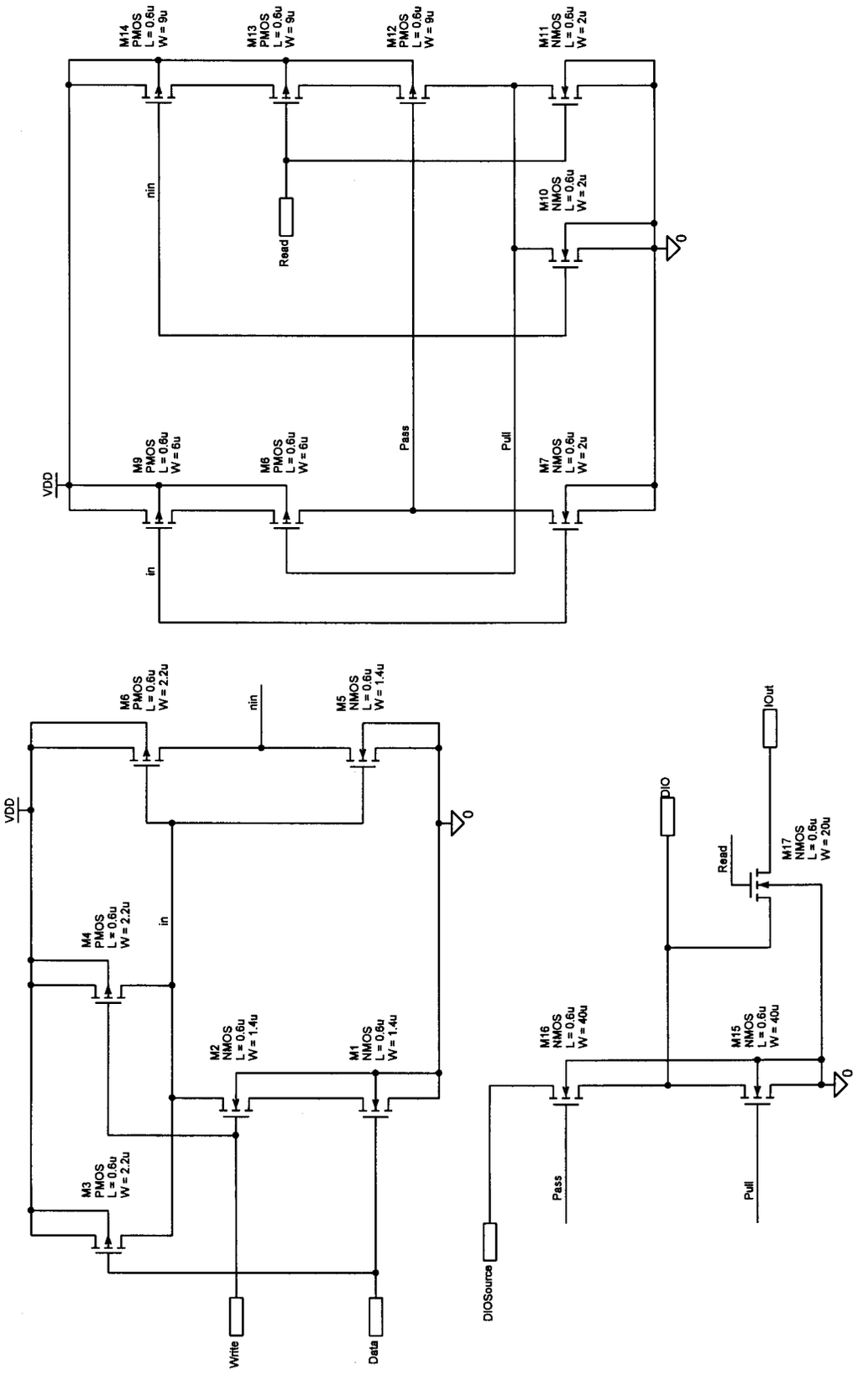


Figure C.43 DIO Driver: *DIO*-line driver switching circuit with build in current-mode multiplexing device.

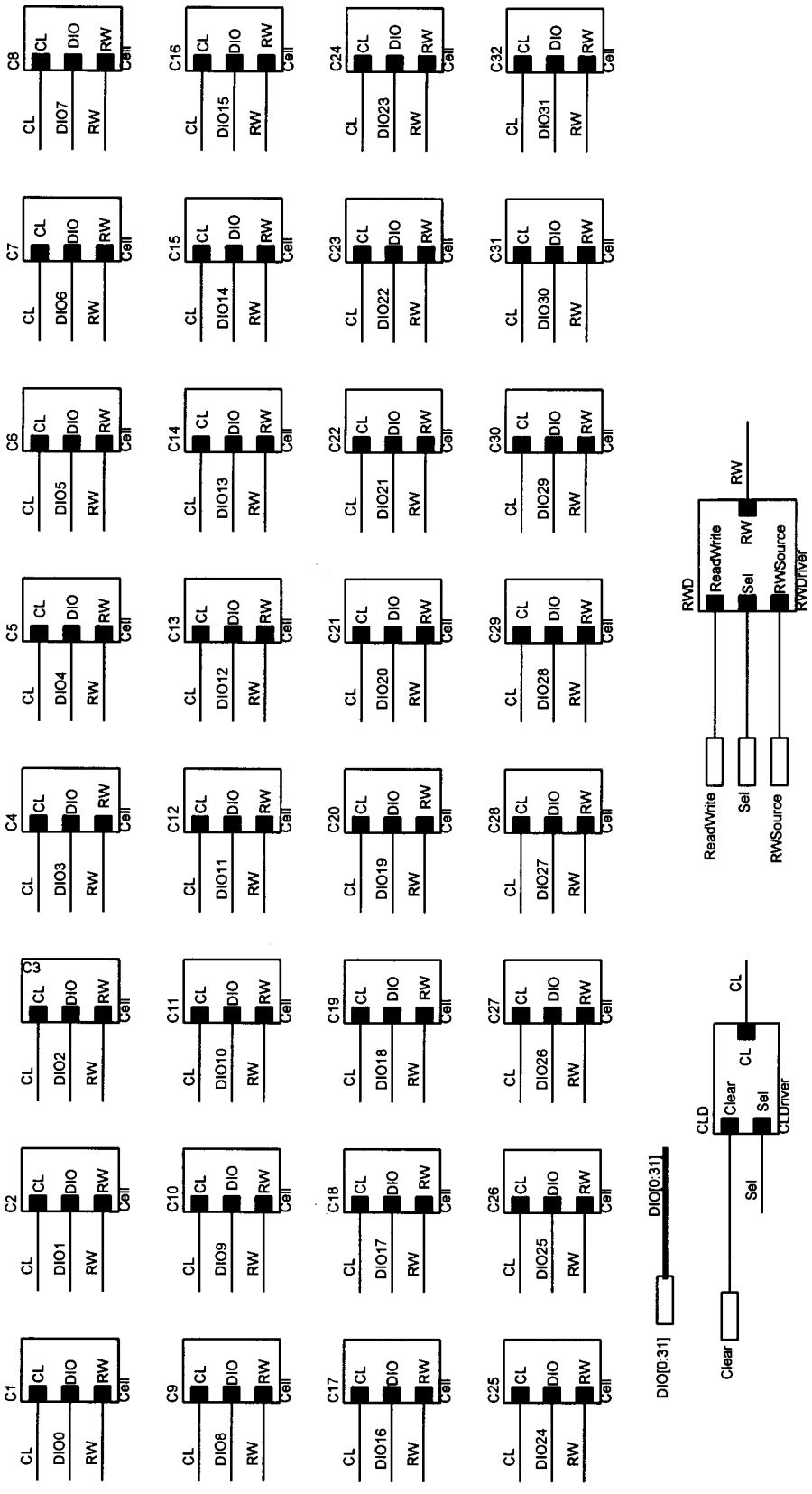


Figure C.45 Row: One row of 32 cells with the *RW*-line driver switching circuit and the *CL*-line driver circuit.

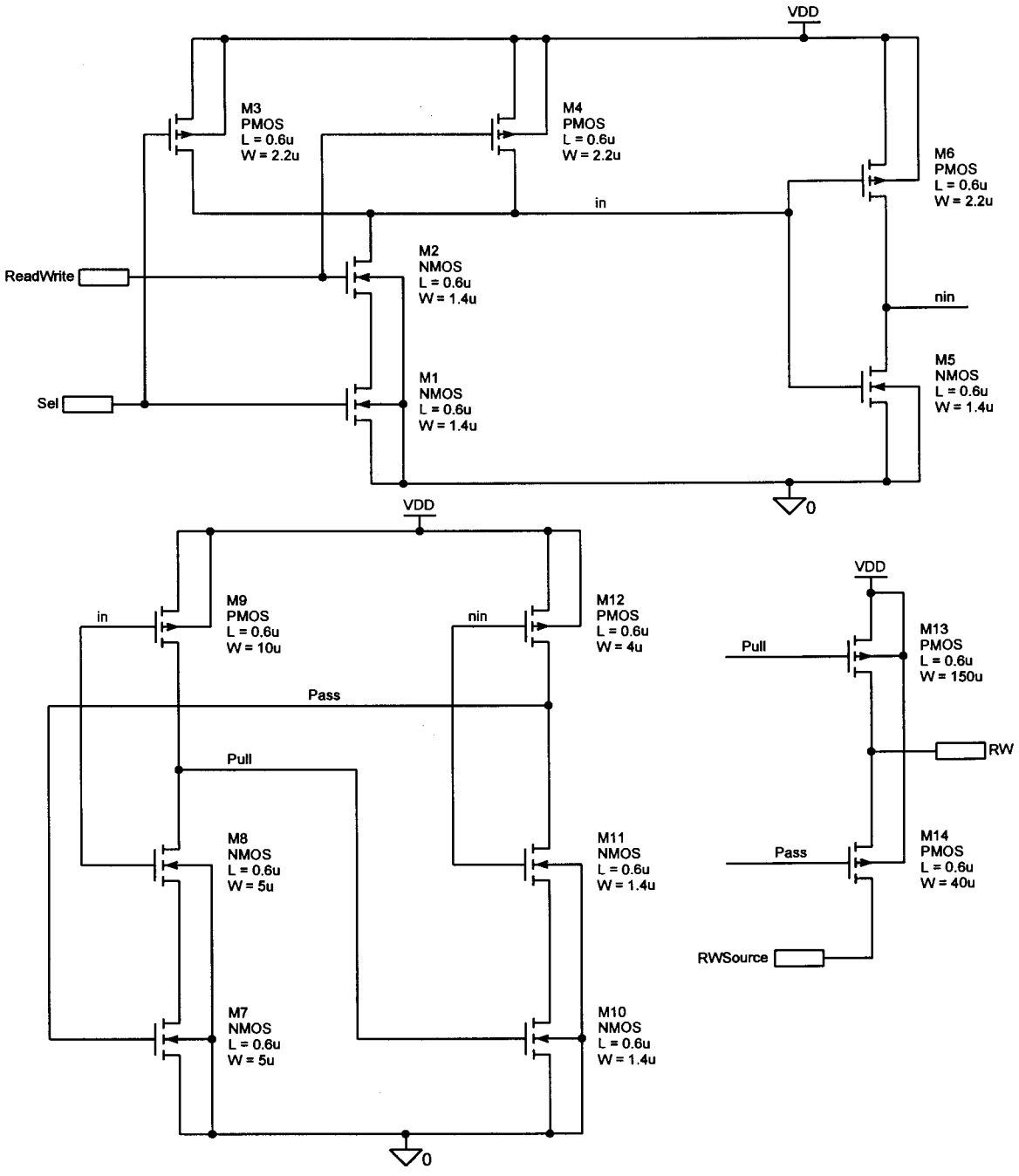


Figure C.46 RW Driver: *RW*-line driver switching circuit.

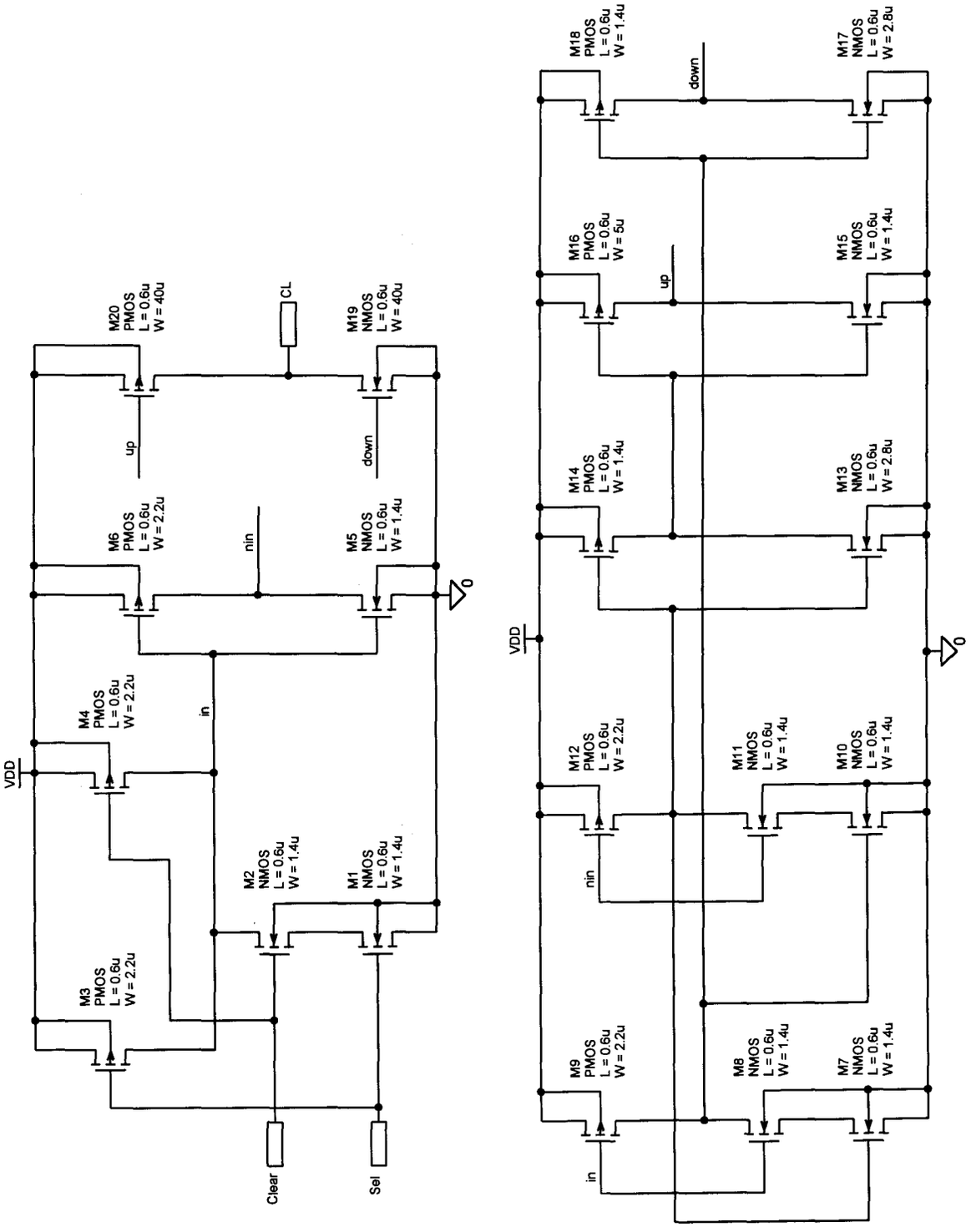


Figure C.47 CL Driver: *CL*-line driver circuit.

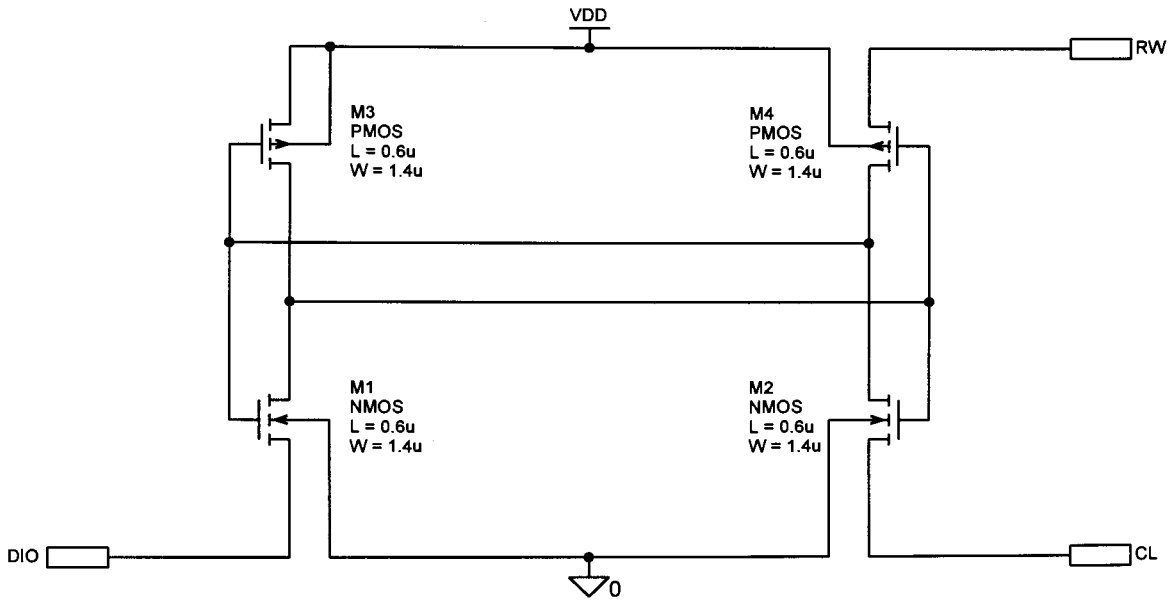


Figure C.48 Cell: Four-transistor SRAM cell.