Escola Universitària Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Grau en enginyeria electrònica industrial i automàtica

Grau en enginyeria mecànica

CONNEXIÓ A XARXA D'UN INVERSOR DE TRES NIVELLS AMB CONTROL PREDICTIU

Annexos

ALEJANDRO VIVARACHO PORCEL
PONENT: SALVADOR ALEPUZ MENÉNDEZ

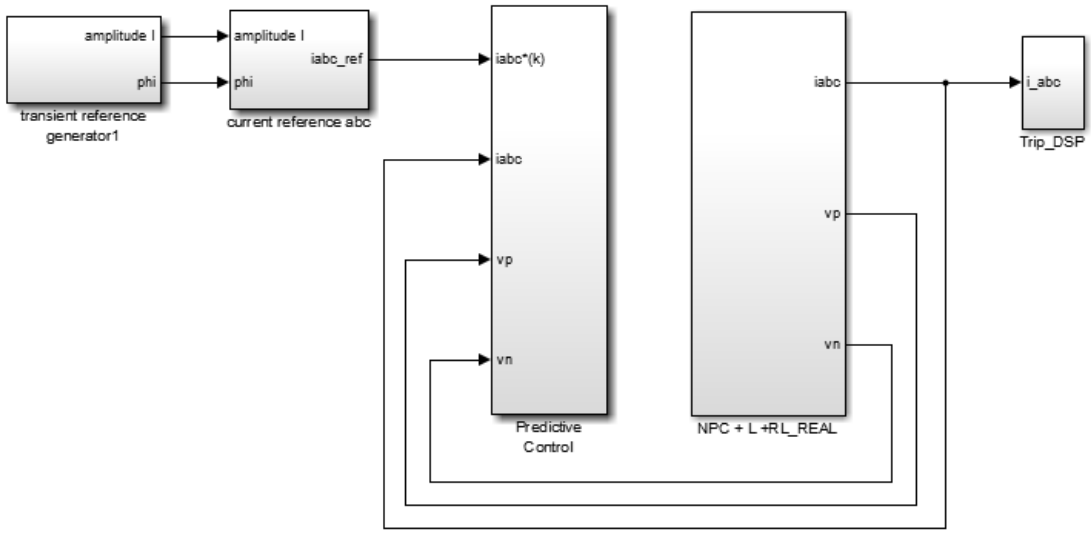
PRIMAVERA 2015

TecnoCampus Mataró-Maresme

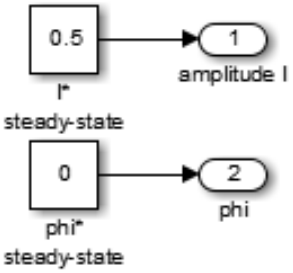
Índex.

Annex I. Programa de control del model experimental amb càrrega RL.

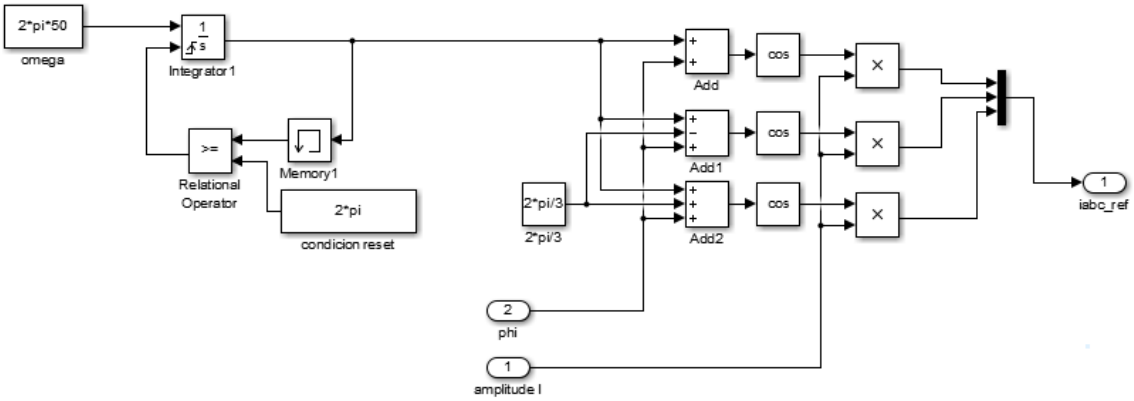
- **Vista general**



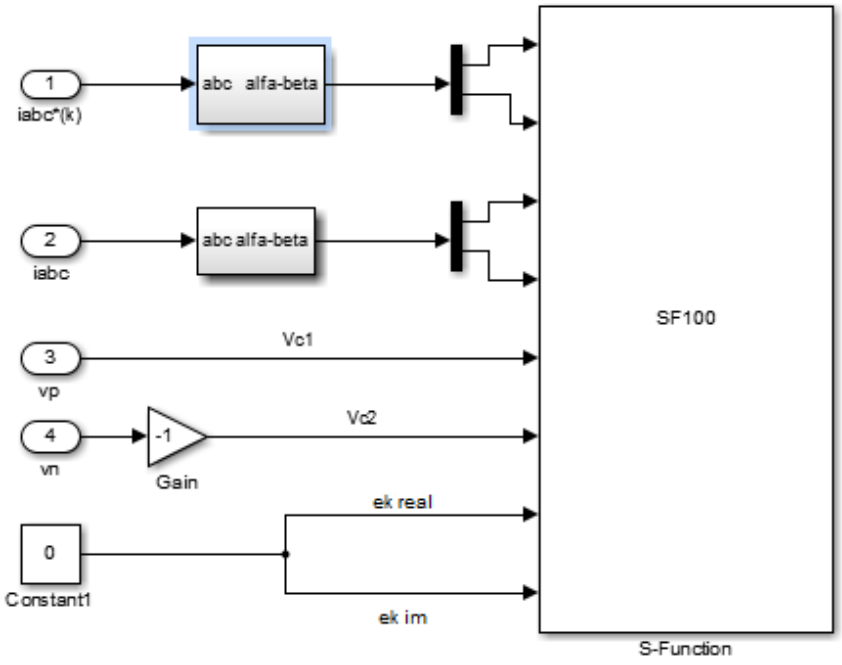
- **Transient reference generator**



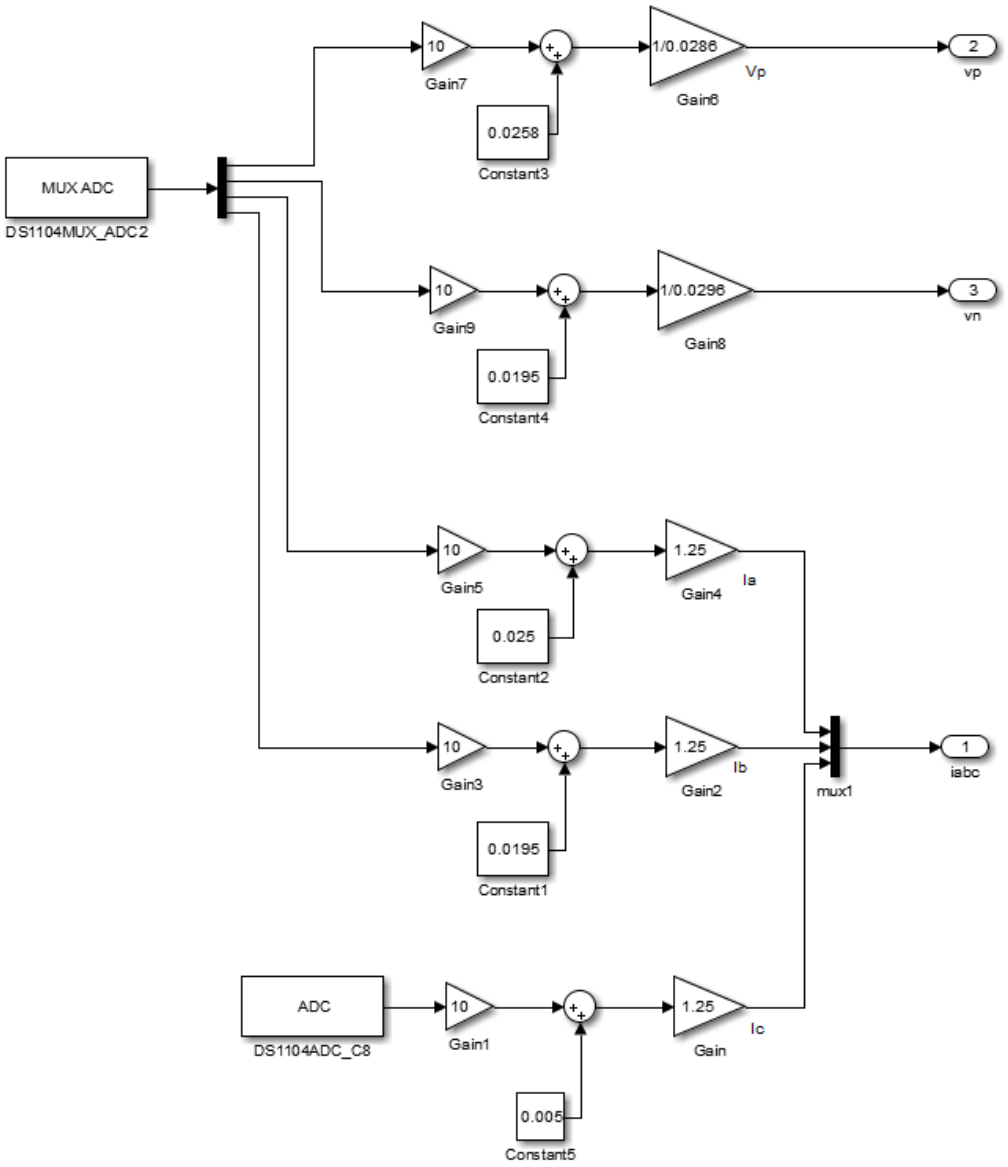
- **Current reference abc**



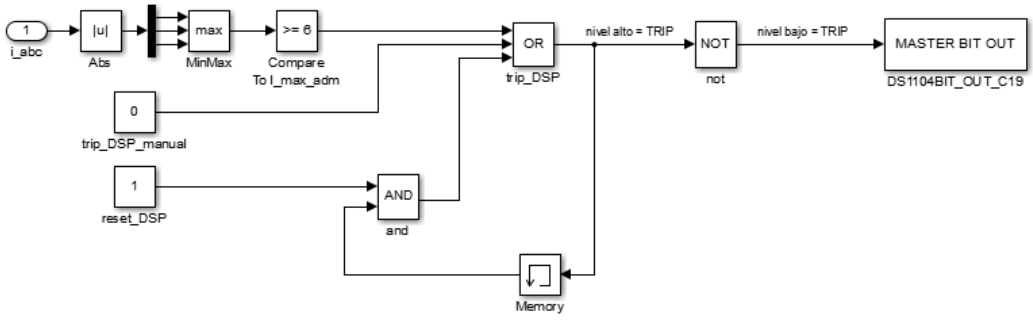
- **Predictive control**



- **NPC+L+RL_REAL**



- **TRIP_DSP**



- **Predictive Control \ SF100**

```
#define S_FUNCTION_NAME  SF100
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

/*Importació de llibreries*/
#ifndef MATLAB_MEX_FILE
# include <brtenv.h>
# include <ds1104.h>
# include <int1104.h>
# include <io1104.h>
# include <dsstd.h>
# include <math.h>
# include <assert.h>
# include <stdlib.h>
   #endif

/* DEFINICIÓ ENTRADES - SORTIDES */

static void mdlInitializeSizes(SimStruct *S)
{
   ssSetNumSFcnParams(S, 0);
   if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount (S)) {
   return;
  }
   /*8 INPUTS */
   if (!ssSetNumInputPorts(S,8)) return;
   {int_T i;
      for (i=0; i<8; i++)
      {
         ssSetInputPortWidth(S, i, 1);
         ssSetInputPortDirectFeedThrough (S, i, 1);
      }
   }

   /* 0 OUTPUTs */
   if (!ssSetNumOutputPorts(S, 0)) return;
   {  int_T i;
      for (i=0; i<0; i++)
      {
```

```
        ssSetOutputPortWidth(S, i, 1);
      }
   }
   ssSetNumSampleTimes(S, 1);
   ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE |
SS_OPTION_USE_TLC_WITH_ACCELERATOR |SS_OPTION_PLACE_ASAP);
}

/* INICIALITZACIÓ DEL TEMPS DE MOSTREIG    */

static void mdlInitializeSampleTimes(SimStruct *S)
   {
   ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
   ssSetOffsetTime(S,0,FIXED_IN_MINOR_STEP_OFFSET);
   }

#define MDL_START  /* Change to #undef to remove function */
#if defined(MDL_START)
static void mdlStart(SimStruct *S)
   {
   #ifndef MATLAB_MEX_FILE


   ds1104_bit_io_init( DS1104_DIO0_OUT | DS1104_DIO1_OUT | /* inicialització
d'inputs i outputs */
                  DS1104_DIO2_OUT | DS1104_DIO3_OUT |
                  DS1104_DIO4_OUT | DS1104_DIO5_OUT |
                  DS1104_DIO6_OUT | DS1104_DIO7_OUT |
                  DS1104_DIO8_OUT | DS1104_DIO9_OUT |
                  DS1104_DIO10_OUT | DS1104_DIO11_OUT |
                  DS1104_DIO12_OUT | DS1104_DIO13_OUT |
                  DS1104_DIO14_OUT | DS1104_DIO15_OUT |
                  DS1104_DIO16_OUT | DS1104_DIO17_OUT |
                  DS1104_DIO18_OUT | DS1104_DIO19_OUT );
   #endif
   }
   #endif /*  MDL_START */


static void mdlOutputs(SimStruct *S, int_T tid)
{
```

```
/* Constants */

   #define L 0.005
   #define R 5.0
   #define Ts 0.0001
   #define C1 0.00075
   #define C2 0.00075
   #define lambdaDC 1.0

   #define Treshold 0.5

   #define B00 1 /* declaració de bits per la paraula de sortida */
   #define B01 2
   #define B02 4
   #define B03 8
   #define B04 16
   #define B05 32

/* Declaració de variables*/
   real_T irefkre;
   real_T irefkim;
   real_T ikre;
   real_T ikim;
   real_T Vc1;
   real_T Vc2;
   real_T ekre;
   real_T ekim;
   real_T irefk1re;
   real_T irefk1im;
   real_T ik1re;
   real_T ik1im;
   real_T ia;
   real_T ib;
   real_T ic;
   real_T ip;
   real_T i0;
   real_T in;
   int_T contador;
   int_T n;
   real_T ic1;
   real_T ic2;
   real_T Vc11;
   real_T Vc21;
```

```
real_T g;
int_T iop;
int_T xop;
real_T gop;
real_T xcontrol;

real_T Sa;
real_T Sb;
real_T Sc;
real_T Sap;
real_T San;
real_T Sbp;
real_T Sbn;
real_T Scp;
real_T Scn;

int_T S0;
int_T S1;
int_T S2;
int_T S3;
int_T S4;
int_T S5;

/* Vectors de tensió */
float SV[19][2]=
        {
        {0.0,0.0}, /*V0*/
        {10.0,0.0}, /*V1*/
        {5.0,8.66}, /*V2*/
        {-5.0,8.66}, /*V3*/
        {-10.0,0.0}, /*V4*/
        {-5.0,-8.66}, /*V5*/
        {5.0,-8.66}, /*V6*/
        {20.0,0.0}, /*V7*/
        {15.0,8.66}, /*V8*/
        {10.0,17.32}, /*V9*/
        {0.0,17.32}, /*V10*/
        {-10.0,17.32}, /*V11*/
        {-15.0,8.66}, /*V12*/
        {-20.0,0.0}, /*V13*/
        {-15.0,-8.66}, /*V14*/
        {-10.0,-17.32}, /*V15*/
        {0.0,-17.32}, /*V16*/
```

```
                {10.0,-17.32}, /*V17*/
                {15.0,-8.66}, /*V18*/
                };

/* Estats IGBTs */
   int ST[27][3]=
                {
                {-1,-1,-1}, /*estado_0*/
                {0,0,0}, /*estado_1*/
                {1,1,1}, /*estado_2*/
                {1,0,0}, /*estado_3*/
                {0,-1,-1}, /*estado_4*/
                {1,1,0}, /*estado_5*/
                {0,0,-1}, /*estado_6*/
                {0,1,0}, /*estado_7*/
                {-1,0,-1}, /*estado_8*/
                {0,1,1}, /*estado_9*/
                {-1,0,0}, /*estado_10*/
                {0,0,1}, /*estado_11*/
                {-1,-1,0}, /*estado_12*/
                {1,0,1}, /*estado_13*/
                {0,-1,0}, /*estado_14*/
                {1,-1,-1}, /*estado_15*/
                {1,0,-1}, /*estado_16*/
                {1,1,-1}, /*estado_17*/
                {0,1,-1}, /*estado_18*/
                {-1,1,-1}, /*estado_19*/
                {-1,1,0}, /*estado_20*/
                {-1,1,1}, /*estado_21*/
                {-1,0,1}, /*estado_22*/
                {-1,-1,1}, /*estado_23*/
                {0,-1,1}, /*estado_24*/
                {1,-1,1}, /*estado_25*/
                {1,-1,0}, /*estado_26*/
                };

/* Input data adquisition through pointers */
   InputRealPtrsType uPtrs0 = ssGetInputPortRealSignalPtrs(S, 0);
   InputRealPtrsType uPtrs1 = ssGetInputPortRealSignalPtrs(S, 1);
   InputRealPtrsType uPtrs2 = ssGetInputPortRealSignalPtrs(S, 2);
   InputRealPtrsType uPtrs3 = ssGetInputPortRealSignalPtrs(S, 3);
   InputRealPtrsType uPtrs4 = ssGetInputPortRealSignalPtrs(S, 4);
   InputRealPtrsType uPtrs5 = ssGetInputPortRealSignalPtrs(S, 5);
```

```
    InputRealPtrsType uPtrs6 = ssGetInputPortRealSignalPtrs(S, 6);
    InputRealPtrsType uPtrs7 = ssGetInputPortRealSignalPtrs(S, 7);

/* Adjudicació del valor de variables dels punters*/
    irefkre= *uPtrs0[0];
    irefkim= *uPtrs1[0];
    ikre= *uPtrs2[0];
    ikim= *uPtrs3[0];
    Vc1= *uPtrs4[0];
    Vc2= *uPtrs5[0];
    ekre= *uPtrs6[0];
    ekim= *uPtrs7[0];

/* Funció */

/* Inicialització */
    contador=0;
    gop=10.0e+30;

/* Referencia extrapolació */
    irefk1re=irefkre;
    irefk1im=irefkim;
    xcontrol=ikim;

/* Avaluació i optimització de la funció de qualitat, g*/
    {int_T i;
    for (i=0; i<19; i++){
        if (i==0){
            n=3;}
        if ((i<7)&&(i>0)){
            n=2;}
        if (i>=7){
            n=1;}

/* Predicció de corrents */
    ik1re=(L/(R*Ts+L))*ikre+(Ts/(R*Ts+L))*(SV[i][0])-(Ts/(R*Ts+L))*ekre;
    ik1im=(L/(R*Ts+L))*ikim+(Ts/(R*Ts+L))*(SV[i][1])-(Ts/(R*Ts+L))*ekim;

/* Fase de predicció de corrents */
    ia=ik1re;
    ib=-0.5*ik1re+(sqrt(3)/2)*ik1im;
    ic=-0.5*ik1re-(sqrt(3)/2)*ik1im;
        {int_T j;
```

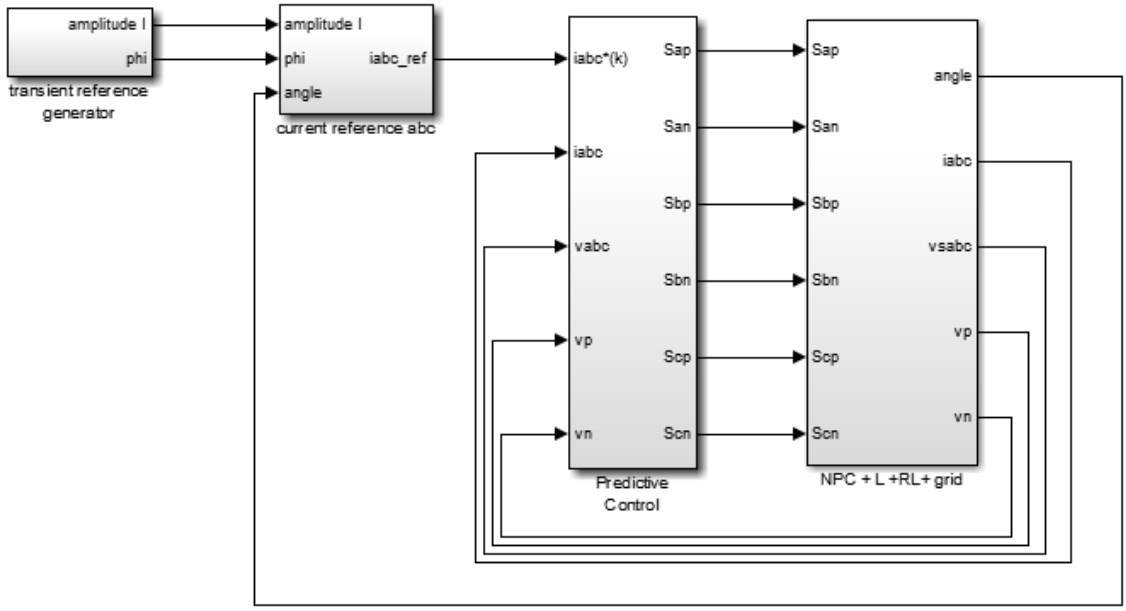
```
for (j=0; j<n; j++){

   if (i==0){                          /*V0*/
      i0=0;ip=0;in=0;}

   if (contador==3){                   /*V1*/
      ip=ia;i0=ic+ib;in=0;}

   if (contador==4){                   /*V1*/
      i0=ia;in=ib+ic;ip=0;}

   if (contador==5){                   /*V2*/
      ip=ia+ib;i0=ic;in=0;}

   if (contador==6){                   /*V2*/
      ip=0;i0=ia+ib;in=ic;}

   if (contador==7){                   /*V3*/
      ip=ib;i0=ia+ic;in=0;}

   if (contador==8){                   /*V3*/
      ip=0;i0=ib;in=ia+ic;}

   if (contador==9){                   /*V4*/
      ip=ib+ic;i0=ia;in=0;}

   if (contador==10){                  /*V4*/
      ip=0;i0=ib+ic;in=ia;}

   if (contador==11){                  /*V5*/
      ip=ic;i0=ia+ib;in=0;}

   if (contador==12){                  /*V5*/
      ip=0;i0=ic;in=ia+ib;}

   if (contador==13){                  /*V6*/
      ip=ia+ic;i0=ib;in=0;}

   if (contador==14){                  /*V6*/
      ip=0;i0=ia+ic;in=ib;}

   if (contador==15){                  /*V7*/
      ip=ia;i0=0;in=ib+ic;}
```

```
        if (contador==16){                    /*V8*/
      ip=ia;i0=ib;in=ic;}

    if (contador==17){                        /*V9*/
      ip=ia+ib;i0=0;in=ic;}

    if (contador==18){                        /*V10*/
      ip=ib;i0=ia;in=ic;}

    if (contador==19){                        /*V11*/
      ip=ib;i0=0;in=ia+ic;}

    if (contador==20){                        /*V12*/
      ip=ib;i0=ic;in=ia;}

    if (contador==21){                        /*V13*/
      ip=ib+ic;i0=0;in=ia;}

    if (contador==22){                        /*V14*/
      ip=ic;i0=ib;in=ia;}

    if (contador==23){                        /*V15*/
      ip=ic;i0=0;in=ia+ib;}

    if (contador==24){                        /*V16*/
      ip=ic;i0=ia;in=ib;}

    if (contador==25){                        /*V17*/
      ip=ia+ic;i0=0;in=ib;}

    if (contador==26){                        /*V18*/
      ip=ia;i0=ic;in=ib;}


  ic1=-ip;
  ic2=in;
  Vc11=Vc1+(1/C1)*ic1*Ts;
  Vc21=Vc2+(1/C2)*ic2*Ts;


/* Avaluació de la funció de qualitat */
  g=fabs((irefk1re)-(ik1re))+fabs((irefk1im)-(ik1im))+lambdaDC*fabs(Vc11-Vc21);
```

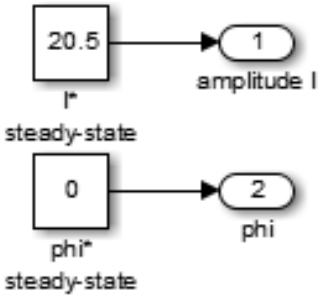
```
    if (g<gop){
          iop=i;          /* vector òptim */
          xop=contador;   /* estat òptim */
          gop=g;
          }
          contador=contador+1;
    }
    }
}

    Sa=ST[xop][0];
    Sb=ST[xop][1];
    Sc=ST[xop][2];

/*  Funció switch treshold */
    if (Sa>=Treshold) {Sap=1;} else {Sap=0;}
    if (Sa<-Treshold) {San=1;} else {San=0;}
    if (Sb>=Treshold) {Sbp=1;} else {Sbp=0;}
    if (Sb<-Treshold) {Sbn=1;} else {Sbn=0;}
    if (Sc>=Treshold) {Scp=1;} else {Scp=0;}
    if (Sc<-Treshold) {Scn=1;} else {Scn=0;}

    S0=Sap;     /* interruptor S1 */
    S1=1-San;   /* interruptor S22 */
    S2=Sbp;     /* interruptor S3 */
    S3=1-Sbn;   /* interruptor S44 */
    S4=Scp;     /* interruptor S5 */
    S5=1-Scn;   /* interruptor S66 */

    #ifndef MATLAB_MEX_FILE      /* escriptura sortida digital DSP */
       ds1104_bit_io_write(B00*S0+B01*S1+B02*S2+B03*S3+B04*S4+B05*S5);
    #endif
    } /* end mdlOutputs */
}
    static void mdlTerminate(SimStruct *S)
    {
    }

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

Annex II: Programa de control del model de simulació amb sincronització a xarxa.

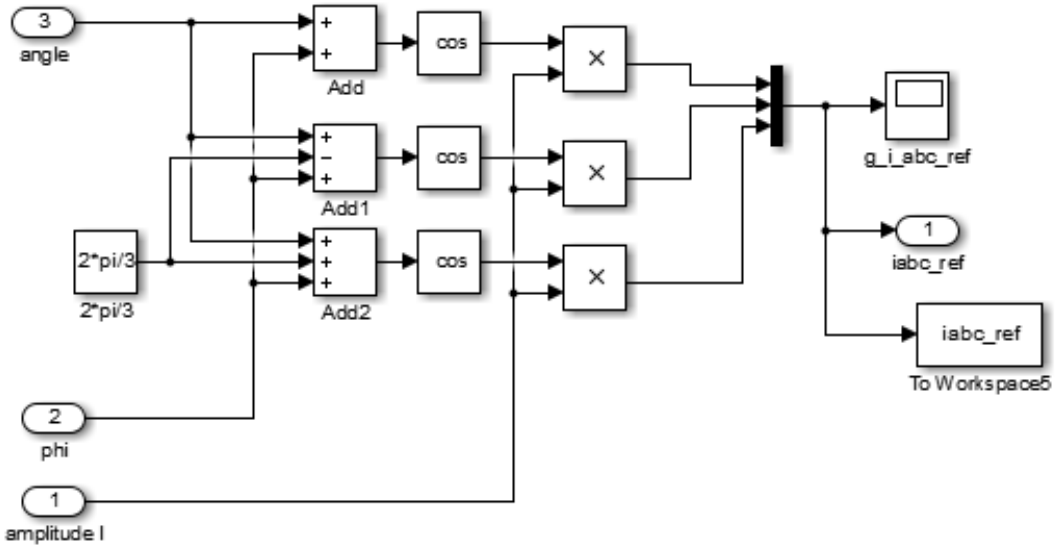
- **Vista general**



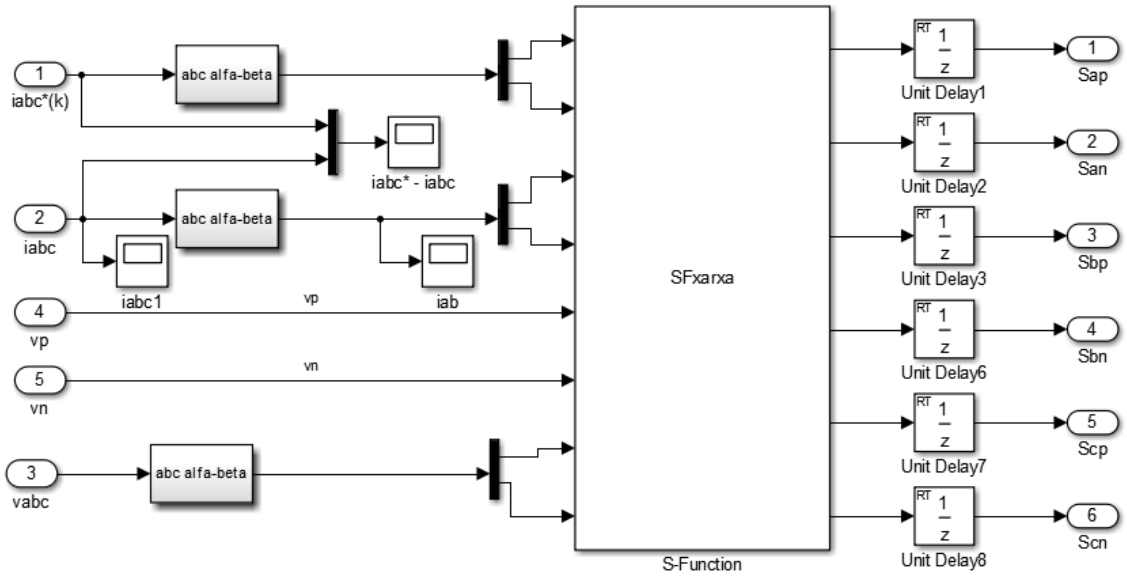
- **Transient reference generator**



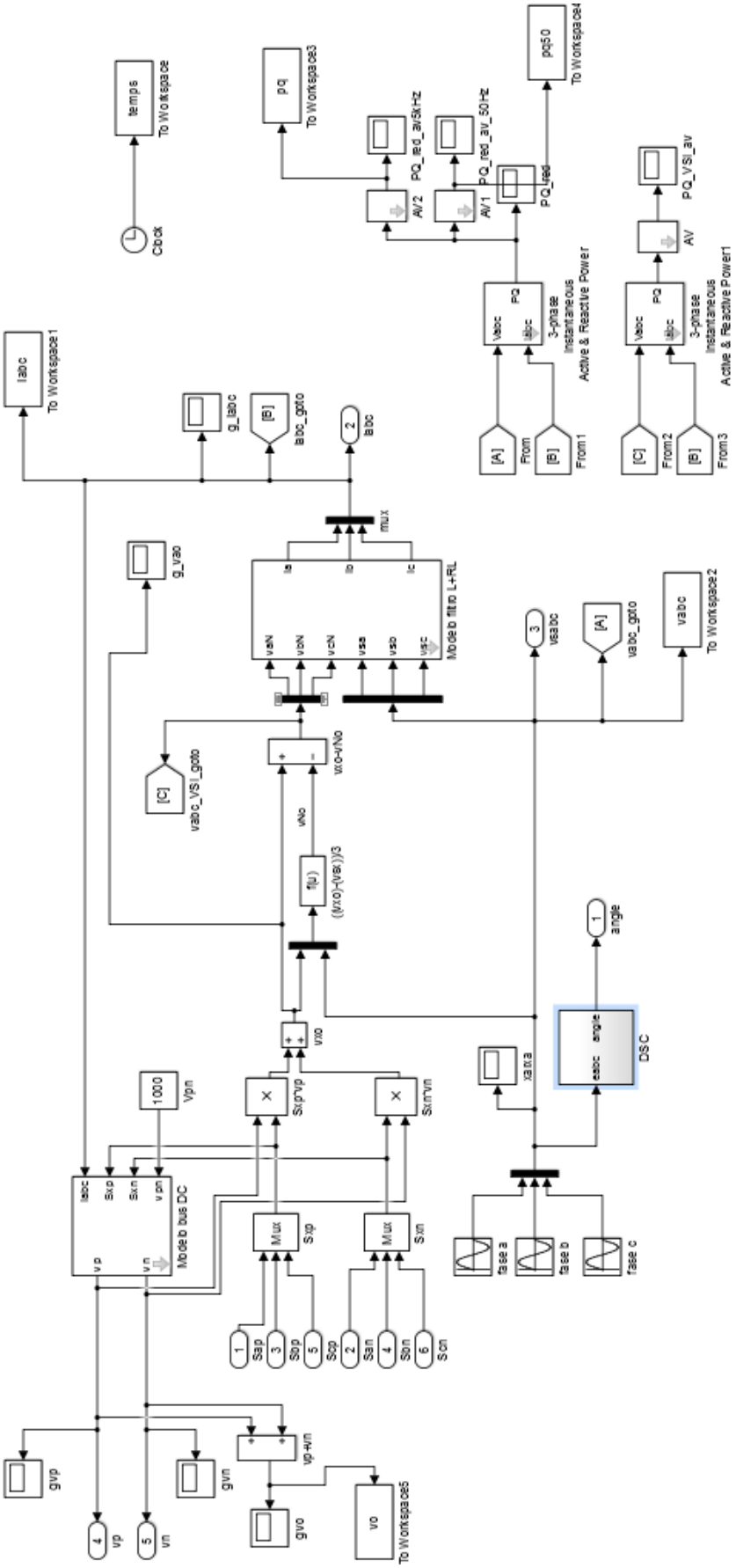
- **Current reference abc**



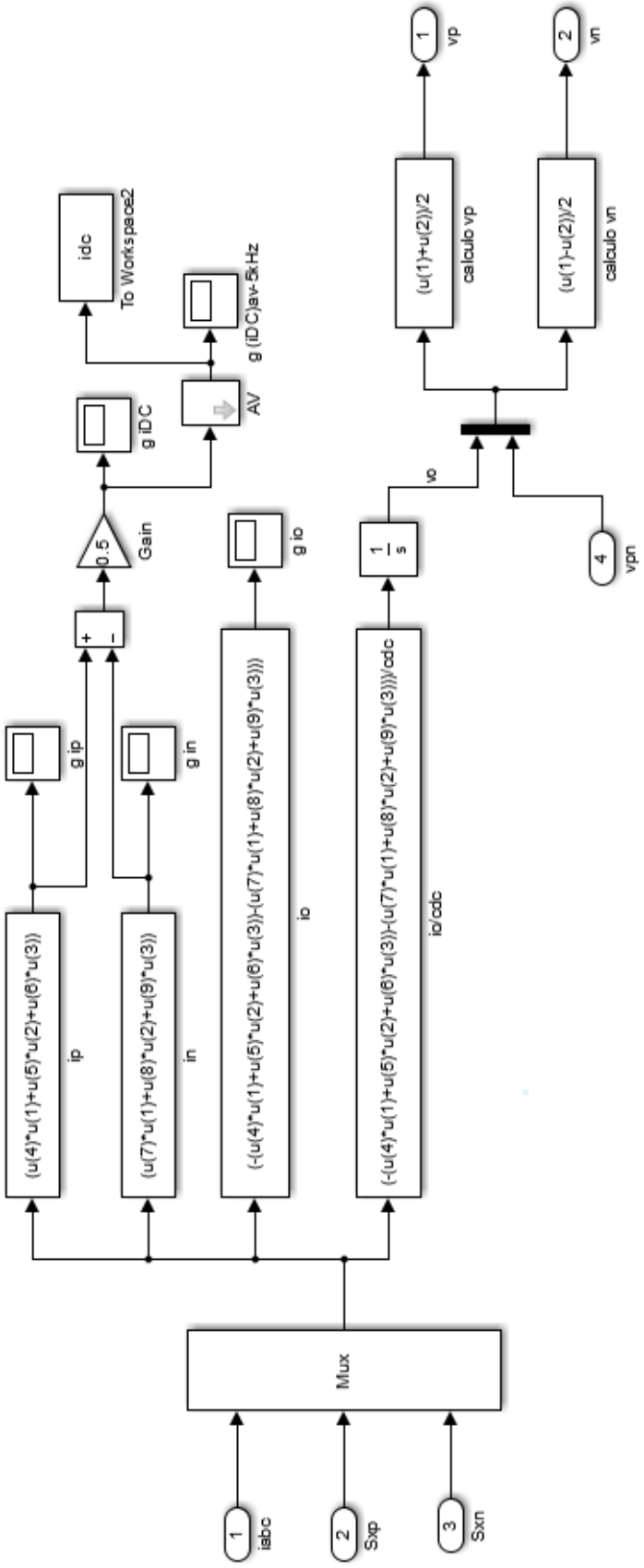
- **Predictive Control**



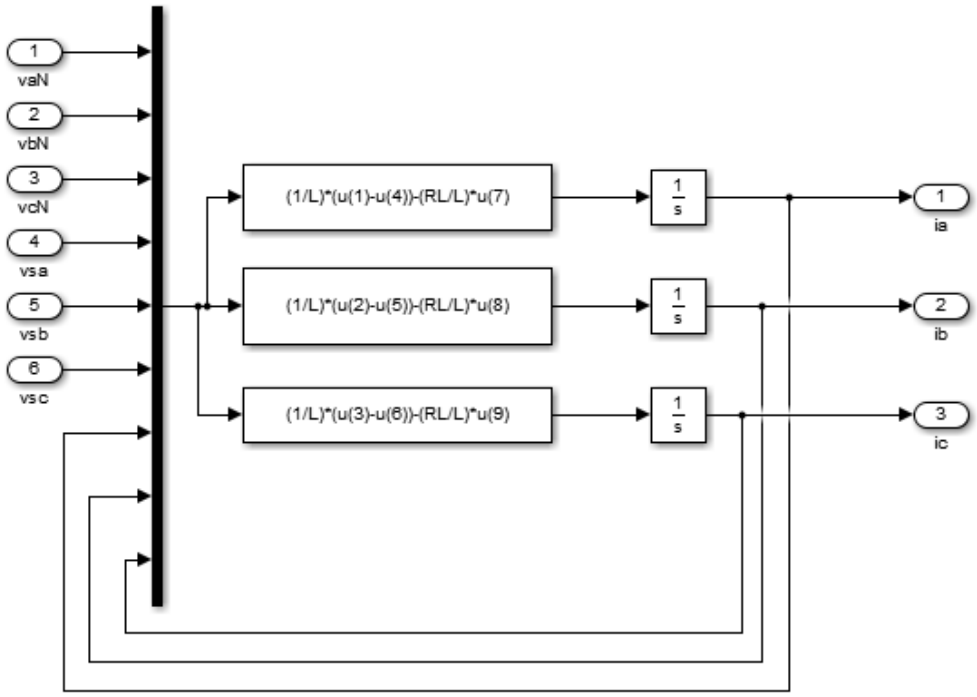
- **NPC+L+RL+grid**


temps
To Workspace
Clock
pq
To Workspace3
PQ_red_av5kHz
PQ_red_av_50Hz
AV2
AV1
PQ_red
pq50
To Workspace4
PQ_VSI_av
AV
Vabc
PQ
Iabc
3-phase
Instantaneous
Active & Reactive Power
Vabc
PQ
Iabc
3-phase
Instantaneous
Active & Reactive Power1
[A]
From
[B]
From1
[C]
From2
[B]
From3
Iabc
To Workspace1
g_iabc
[B]
Iabc_goto
2
Iabc
mux
g_vao
Ia
Ib
Ic
Modelo filtro L+RL
VaN
VbN
VcN
Vas
Vbs
Vcs
3
Vabc
[A]
Vabc_goto
Vabc
To Workspace2
[C]
Vabc_VSI_goto
Vxo-vNo
vNo
f(u)
((Vxo)-(Vas))/3
1
angle
Vxo
Vana
Iabc
Vabc
angle
DSC
1000
Vpn
Sxp*vp
Sxn*vn
Modelo bus DC
Iabc
Sxp
Sxn
vpn
vp
vn
Mux
Sxp
Mux
Sxn
1
Sap
3
Sbp
5
Scp
2
San
4
Sbn
6
Scn
fase a
fase b
fase c
gvp
gvn
vp+vn
gvo
vo
To Workspace5
4
vp
5
vn

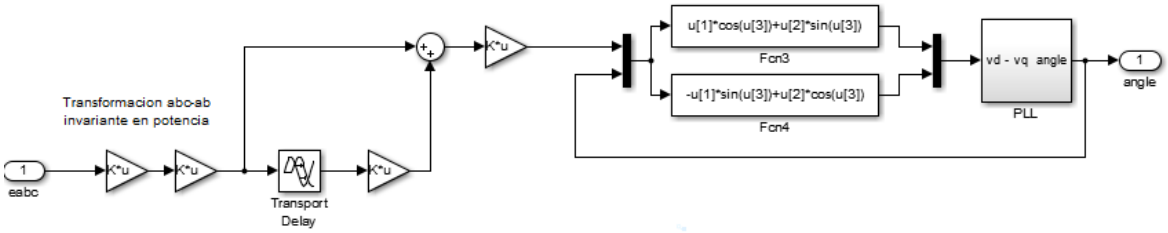
- **NPC+L+RL+grid \ Modelo bus DC**



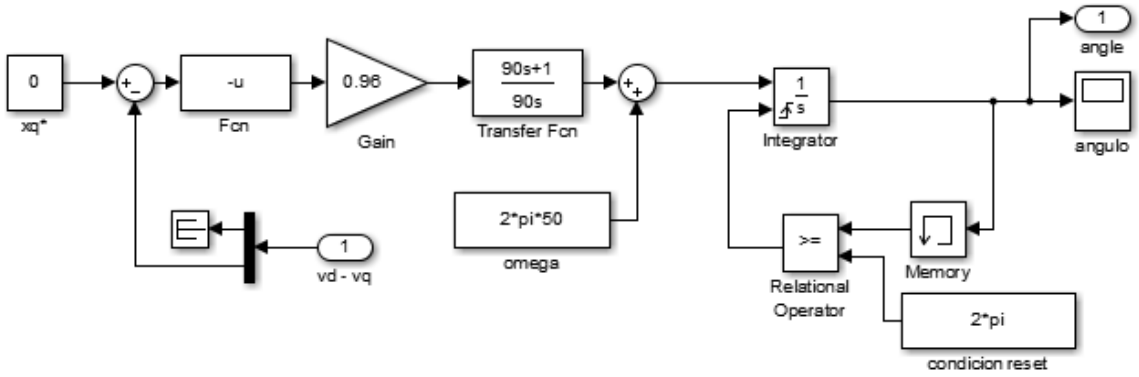
- **NPC+L+RL+grid \ Modelo filtro L+RL**



- **NPC+L+RL+grid \ DSC**



- **NPC+L+RL+grid \ DSC \ PLL**



- **Predictive Control \ SFxarxa**

```
#define S_FUNCTION_NAME  SFxarxa
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

/*Importació de llibreries*/
#ifndef MATLAB_MEX_FILE
# include <brtenv.h>
# include <stdio.h>
# include <dsstd.h>
# include <math.h>
# include <assert.h>
# include <stdlib.h>
   #endif


/* DEFINICIÓ VARIABLES GLOBALS */

real_T vxm1a;
real_T vxm1b;

   float SV[1][3]=
               {
               {0.0,0.0,0.0},
   };

/* DEFINICIÓ ENTRADES - SORTIDES */

static void mdlInitializeSizes(SimStruct *S)
{
   ssSetNumSFcnParams(S, 0);
   if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount (S)) {
   return;
  }
   /*8 INPUTS */
   if (!ssSetNumInputPorts(S,8)) return;
   {int_T i;
      for (i=0; i<8; i++)
      {
         ssSetInputPortWidth(S, i, 1);
         ssSetInputPortDirectFeedThrough (S, i, 1);
```

```
      }
   }
   /* 6 OUTPUTs */
   if (!ssSetNumOutputPorts(S, 6)) return;
   {  int_T i;
      for (i=0; i<6; i++)
      {
        ssSetOutputPortWidth(S, i, 1);
      }
   }
   ssSetNumSampleTimes(S, 1);
   ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE |
SS_OPTION_USE_TLC_WITH_ACCELERATOR |SS_OPTION_PLACE_ASAP);
}

/* INICIALITZACIÓ DEL TEMPS DE MOSTREIG    */
 static void mdlInitializeSampleTimes(SimStruct *S)
   {
   ssSetSampleTime(S, 0, 100e-6);
   ssSetOffsetTime(S,0,0);
   }

#define MDL_START  /* Change to #undef to remove function */
#if defined(MDL_START)
static void mdlStart(SimStruct *S)
   {
   #ifndef MATLAB_MEX_FILE

   #endif
   }
   #endif /*  MDL_START */

static void mdlOutputs(SimStruct *S, int_T tid)
{

/* Constants */
   #define Vdc 1000.0
   #define L 10.0e-3
   #define R 0.1
   #define Ts 100.0e-6
   #define C1 750.0e-6
   #define C2 750.0e-6
   #define lambdaDC 1.0
```

```
#define Treshold 0.5

#define B00 1 /* declaracio de bits per la paraula de sortida */
#define B01 2
#define B02 4
#define B03 8
#define B04 16
#define B05 32

/* Declaració de variables*/
real_T vpk;
real_T vnk;
real_T gop;
real_T Saop;
real_T Sbop;
real_T Scop;
real_T irefk1a;
real_T irefka;
real_T irefk1b;
real_T irefkb;
real_T ik1a;
real_T ika;
real_T ik1b;
real_T ikb;
real_T iak1;
real_T ibk1;
real_T ick1;
real_T ik1c;
real_T iok1;
real_T vpk1;
real_T vnk1;
real_T va;
real_T vb;
real_T vc;
real_T vxa;
real_T vxb;
real_T eka;
real_T ekb;
real_T iak2;
real_T ik2a;
real_T ibk2;
real_T ik2b;
real_T ick2;
```

```
    real_T iok2;
    real_T vp;
    real_T vn;
    real_T Sm1a;
    real_T Sm1b;
    real_T Sm1c;
    real_T iok2a;
    real_T iok2b;
    real_T iok2c;
    real_T Sa;
    real_T Sb;
    real_T Sc;
    int_T Sap;
    int_T San;
    int_T Sbp;
    int_T Sbn;
    int_T Scp;
    int_T Scn;
    real_T vpk2;
    real_T vnk2;
    real_T g;
    int_T S0;
    int_T S1;
    int_T S2;
    int_T S3;
    int_T S4;
    int_T S5;

/* Adquisició de dades d'entrada per punters */
    InputRealPtrsType uPtrs0 = ssGetInputPortRealSignalPtrs(S, 0);
    InputRealPtrsType uPtrs1 = ssGetInputPortRealSignalPtrs(S, 1);
    InputRealPtrsType uPtrs2 = ssGetInputPortRealSignalPtrs(S, 2);
    InputRealPtrsType uPtrs3 = ssGetInputPortRealSignalPtrs(S, 3);
    InputRealPtrsType uPtrs4 = ssGetInputPortRealSignalPtrs(S, 4);
    InputRealPtrsType uPtrs5 = ssGetInputPortRealSignalPtrs(S, 5);
    InputRealPtrsType uPtrs6 = ssGetInputPortRealSignalPtrs(S, 6);
    InputRealPtrsType uPtrs7 = ssGetInputPortRealSignalPtrs(S, 7);

/* Adjudicació de sortides per punters */
    real_T *y0 = ssGetOutputPortRealSignal(S,0);
    real_T *y1 = ssGetOutputPortRealSignal(S,1);
    real_T *y2 = ssGetOutputPortRealSignal(S,2);
    real_T *y3 = ssGetOutputPortRealSignal(S,3);
```

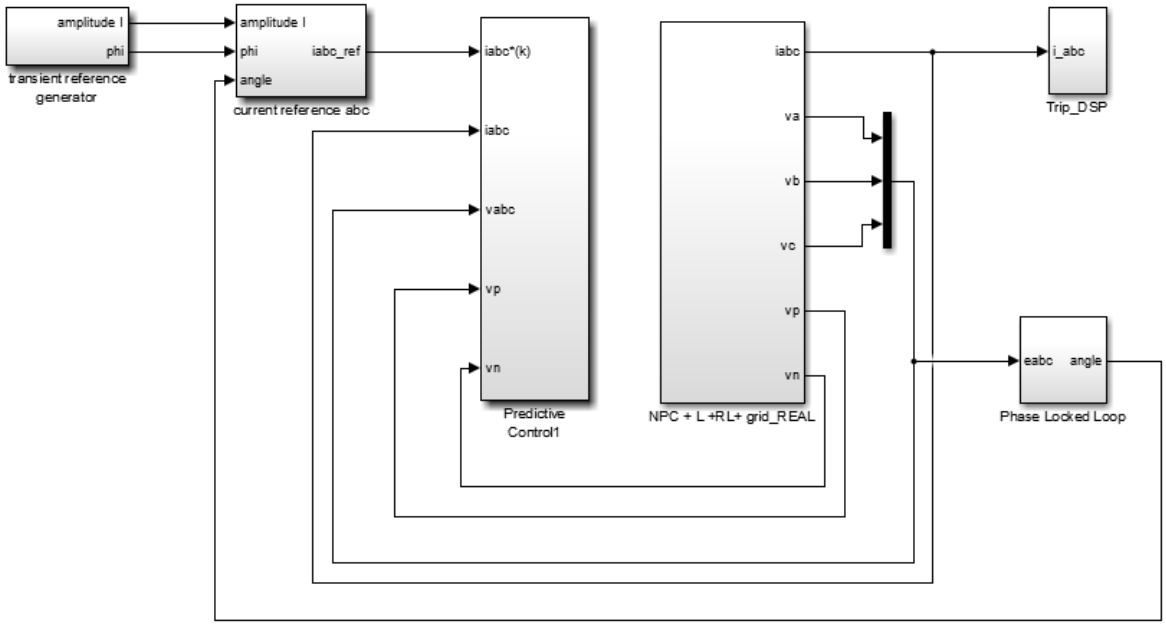
```
    real_T *y4 = ssGetOutputPortRealSignal(S,4);
    real_T *y5 = ssGetOutputPortRealSignal(S,5);

/* Adjudicació de valor de variables per punters */
    irefka= *uPtrs0[0];
    irefkb= *uPtrs1[0];
    ika= *uPtrs2[0];
    ikb= *uPtrs3[0];
    vp= *uPtrs4[0];
    vn= *uPtrs5[0];
    eka= *uPtrs6[0];
    ekb= *uPtrs7[0];

/* Funció */

/* Inicialització */
    vpk=vp;
    vnk=vn;
    gop=10.0e100;
    Saop=0.0;
    Sbop=0.0;
    Scop=0.0;

/* Current reference */
    irefk1a=irefka;
    irefk1b=irefkb;

/* Current calculated for k+1 */
    ik1a=(1.0-(R*Ts/L))*ika+(Ts/L)*(vxm1a-eka);
    ik1b=(1.0-(R*Ts/L))*ikb+(Ts/L)*(vxm1b-ekb);

/* Phase current prediction for k+1 */
    iak1=ik1a;
    ibk1=-0.5*ik1a+(sqrt(3.0)/2.0)*ik1b;
    ick1=-0.5*ik1a-(sqrt(3.0)/2.0)*ik1b;
 /* Balanç de voltatge vp i vn per k+1 */
    if ((SV[0][0])==0.0) {Sm1a=iak1;} else {Sm1a=0.0;}
    if ((SV[0][1])==0.0) {Sm1b=ibk1;} else {Sm1b=0.0;}
    if ((SV[0][2])==0.0) {Sm1c=ick1;} else {Sm1c=0.0;}
    iok1=Sm1a+Sm1b+Sm1c;
    vpk1=vpk+(1.0/(2.0*C1))*iok1*Ts;
    vnk1=vnk+(1.0/(2.0*C2))*iok1*Ts;
```

```
/*Avaluació i optimització de g pels 27 estats*/
 for (Sa=-1.0; Sa<=1.0; Sa=Sa+1.0) {
   for (Sb=-1.0; Sb<=1.0; Sb=Sb+1.0) {
      for (Sc=-1.0; Sc<=1.0; Sc=Sc+1.0) {

         /*vectors*/
         va=Sa*Vdc;
         vb=Sb*Vdc;
         vc=Sc*Vdc;

         /*vx=(valpha)+(vbeta)*/
         vxa=((1.0/3.0)*(va-0.5*vb-0.5*vc));
         vxb=(1.0/3.0)*((sqrt(3.0)/2.0)*vb-(sqrt(3.0)/2.0)*vc);

         /*Predicció de current per k+2*/
         ik2a=(1-(R*Ts/L))*ik1a+(Ts/L)*(vxa-eka);
         ik2b=(1-(R*Ts/L))*ik1b+(Ts/L)*(vxb-ekb);

         /* Predicció del corrent de fase per k+2 */
         iak2=ik2a;
         ibk2=-0.5*ik2a+(sqrt(3.0)/2.0)*ik2b;
         ick2=-0.5*ik2a-(sqrt(3.0)/2.0)*ik2b;


         /* Balanç de voltatge vp i vn per k+2 */
         if (Sa==0.0) {iok2a=iak2;} else {iok2a=0.0;}
         if (Sb==0.0) {iok2b=ibk2;} else {iok2b=0.0;}
         if (Sc==0.0) {iok2c=ick2;} else {iok2c=0.0;}
         iok2=iok2a+iok2b+iok2c;
         vpk2=vpk1+(1.0/(2.0*C1))*iok2*Ts;
         vnk2=vnk1+(1.0/(2.0*C2))*iok2*Ts;
    /*Avaluació de g*/
         g=(pow((irefk1a-ik2a),2))+(pow((irefk1b-
ik2b),2))+lambdaDC*(pow((vpk2+vnk2),2));
         if (g<gop)
         {
            gop=g;
            vxm1a=vxa;
            vxm1b=vxb;
            Saop=Sa;
            Sbop=Sb;
            Scop=Sc;
         }
```

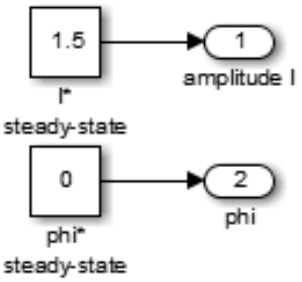
```
      }  //end for Sc
   } //end for Sb
} //end for Sa

/* Estat pel pròxim període*/
   SV[0][0]=Saop;
   SV[0][1]=Sbop;
   SV[0][2]=Scop;

/*  Funció switch treshold */
   if (Saop>=Treshold) {Sap=1;} else {Sap=0;}
   if (Saop<-Treshold) {San=1;} else {San=0;}
   if (Sbop>=Treshold) {Sbp=1;} else {Sbp=0;}
   if (Sbop<-Treshold) {Sbn=1;} else {Sbn=0;}
   if (Scop>=Treshold) {Scp=1;} else {Scp=0;}
   if (Scop<-Treshold) {Scn=1;} else {Scn=0;}

   #ifndef MATLAB_MEX_FILE
   #endif

   *y0=Sap;
   *y1=San;
   *y2=Sbp;
   *y3=Sbn;
   *y4=Scp;
   *y5=Scn;

   } /* end mdlOutputs */

    static void mdlTerminate(SimStruct *S)
   {
   }

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

Annex III. Programa de control del model experimental amb sincronització a xarxa.

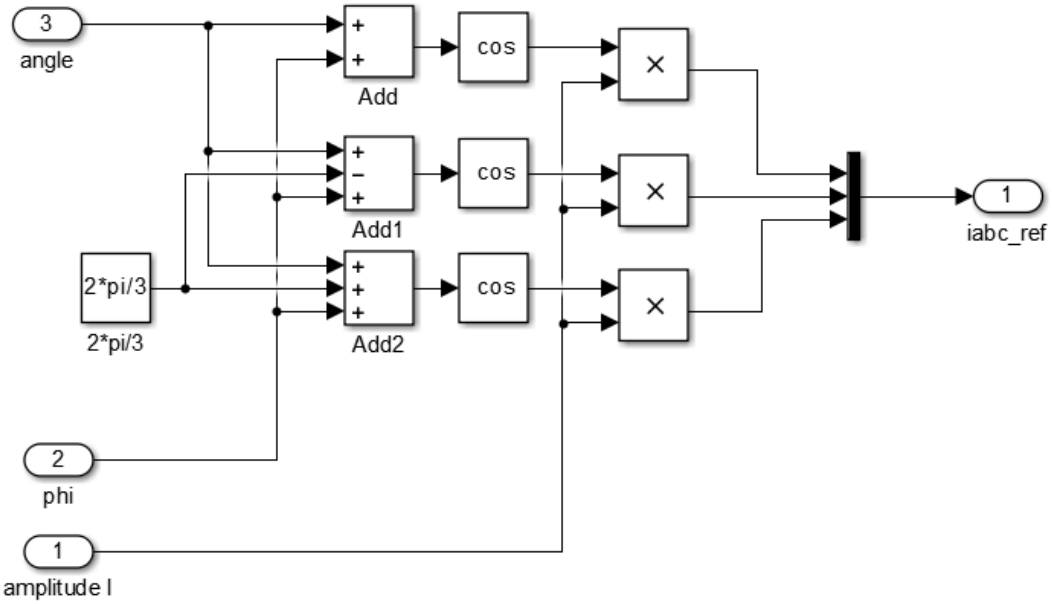
- **Vista general**



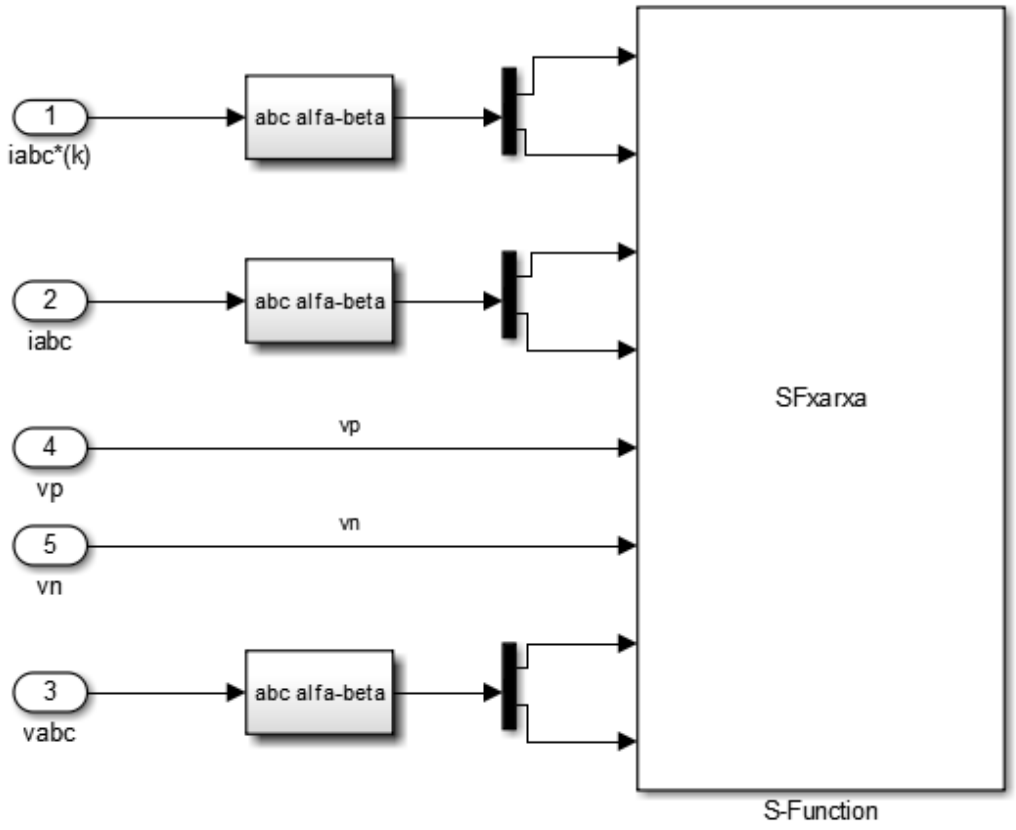
- **Transient reference generator**



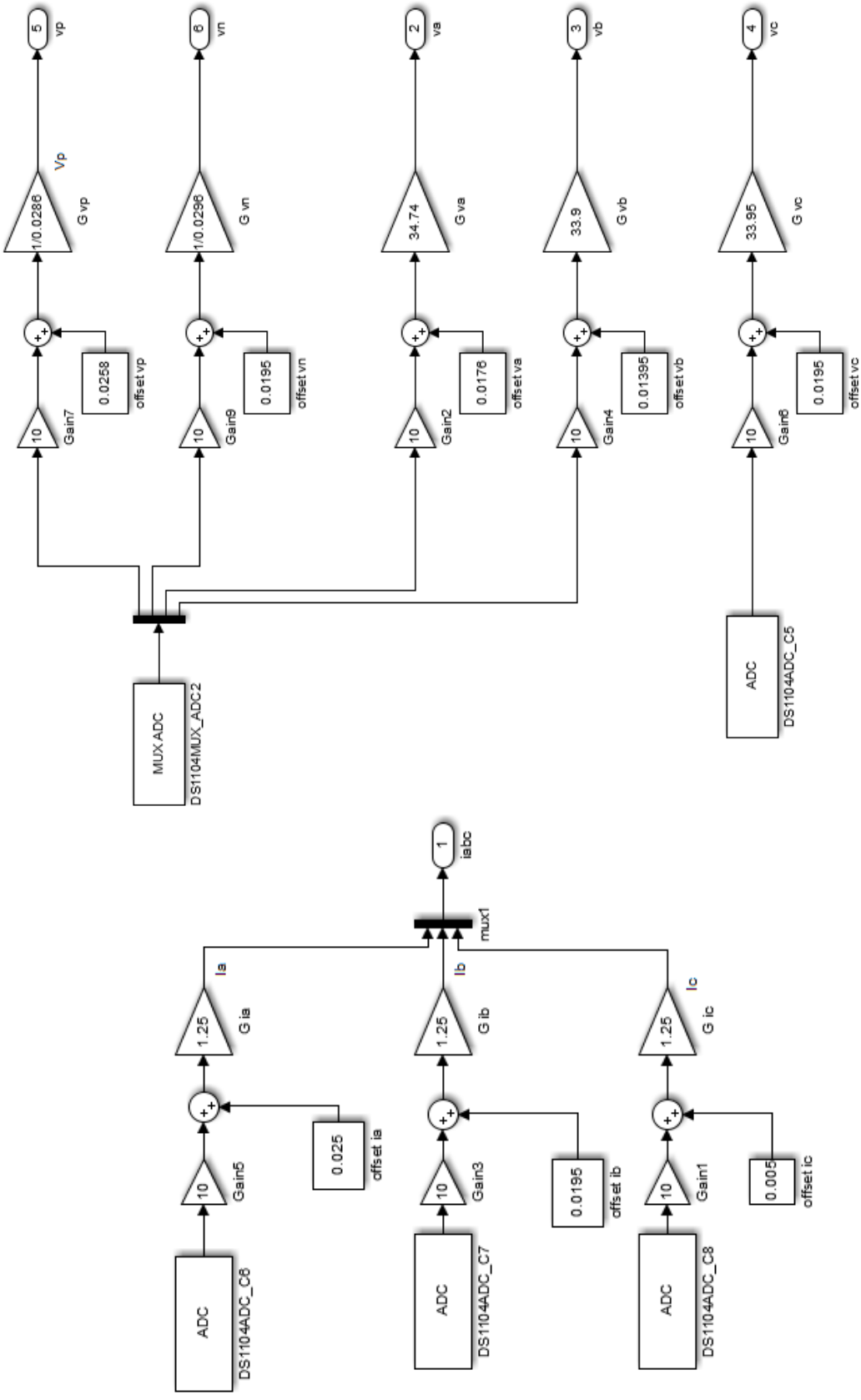
- **Current reference abc**



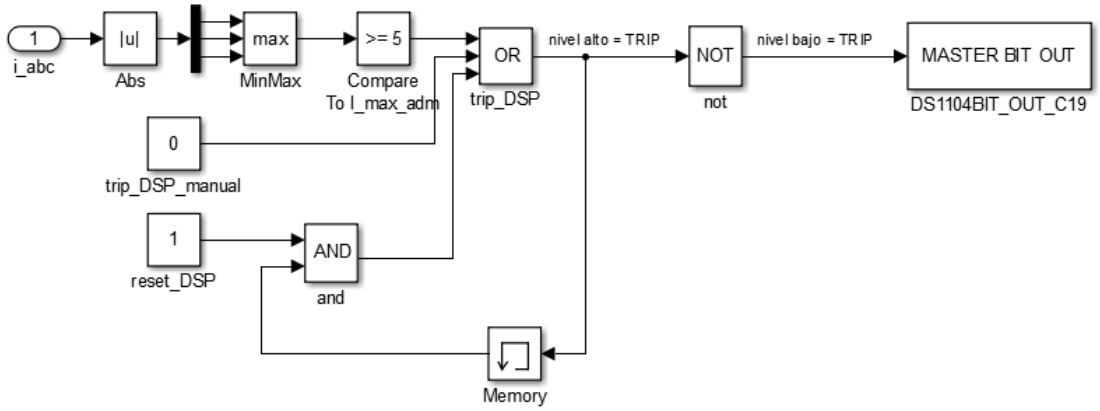
- **Predictive Control**



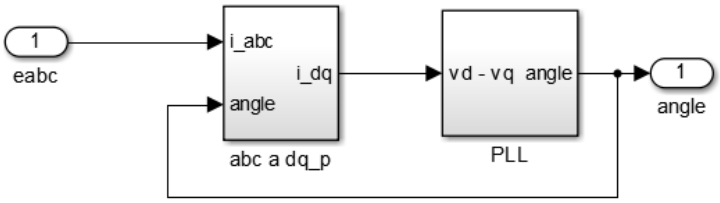
- **NPC+L+RL+grid_REAL**



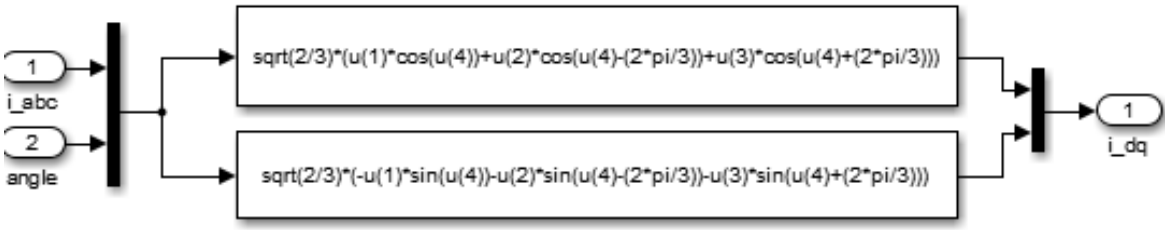
- **TRIP_DSP**



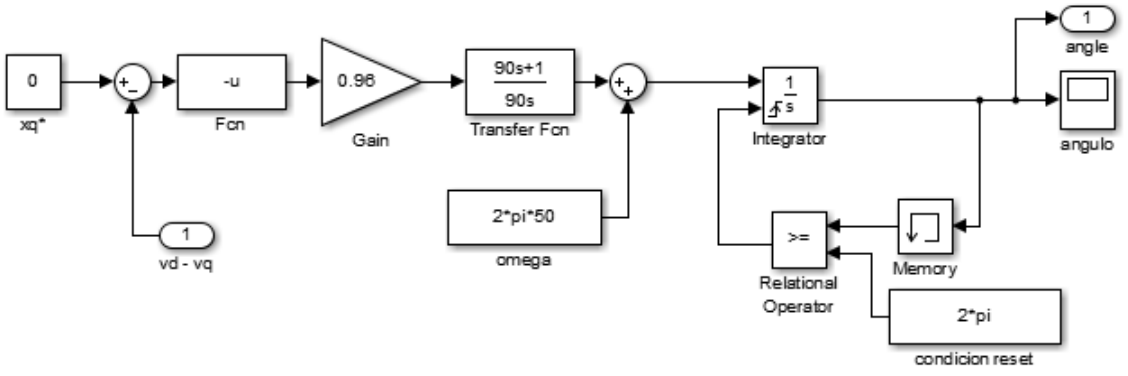
- **Phase Locked Loop**



- **Phase Locked Loop \ abc a dq_p**



- **Phase Locked Loop \ PLL**



- **Predictive Control \ SFxarxa**

```
#define S_FUNCTION_NAME  SFxarxa
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

 /*Importació de llibreries*/
#ifndef MATLAB_MEX_FILE
# include <brtenv.h>
# include <stdio.h>
# include <dsstd.h>
# include <math.h>
# include <assert.h>
# include <stdlib.h>
# include <ds1104.h>
# include <int1104.h>
# include <io1104.h>
#endif

/* Declaració de variables globals */

real_T vxm1a;
real_T vxm1b;
float SV[1][3]=  {
                   {0.0,0.0,0.0},
                 };

/*inicialització d'entrades I sortides*/
static void mdlInitializeSizes(SimStruct *S)
{
   ssSetNumSFcnParams(S, 0);
   if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount (S)) {
   return;
  }

   /* inicialització d'inputs del bloc, 8 */
   if (!ssSetNumInputPorts(S,8)) return;
   {int_T i;
      for (i=0; i<8; i++)
      {
         ssSetInputPortWidth(S, i, 1); //amplitud dels imputs
         ssSetInputPortDirectFeedThrough (S, i, 1); //inputs utilitzats en mdlOutputs
```

```
      }
   }

   /* inicialització d'outputs del bloc, 0*/
   if (!ssSetNumOutputPorts(S, 0)) return;
   {  int_T i;
      for (i=0; i<0; i++)
      {
        ssSetOutputPortWidth(S, i, 1);
      }
   }
   ssSetNumSampleTimes(S, 1); // número de diferents temps de mostreig
   ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE |
SS_OPTION_USE_TLC_WITH_ACCELERATOR |SS_OPTION_PLACE_ASAP);
//opcions per la millora de la velocitat d'execució d'algunes S-Functions
} //end mdlInitializeSizes

/* inicialització del temps de mostreig */
static void mdlInitializeSampleTimes(SimStruct *S)
   {
   ssSetSampleTime(S, 0, 150e-6); //temps de mostreig, 150us
   ssSetOffsetTime(S,0,0); //temps d'offset, 0
   }

#define MDL_START
#if defined(MDL_START)
static void mdlStart(SimStruct *S) //funció que només s'executa una vegada
   {
   #ifndef MATLAB_MEX_FILE

   /* inicialització d' inputs i outputs de la DS1104 */
    ds1104_bit_io_init( DS1104_DIO0_OUT | DS1104_DIO1_OUT |
                  DS1104_DIO2_OUT | DS1104_DIO3_OUT |
                  DS1104_DIO4_OUT | DS1104_DIO5_OUT |
                  DS1104_DIO6_OUT | DS1104_DIO7_OUT |
                  DS1104_DIO8_OUT | DS1104_DIO9_OUT |
                  DS1104_DIO10_OUT | DS1104_DIO11_OUT |
                  DS1104_DIO12_OUT | DS1104_DIO13_OUT |
                  DS1104_DIO14_OUT | DS1104_DIO15_OUT |
                  DS1104_DIO16_OUT | DS1104_DIO17_OUT |
                  DS1104_DIO18_OUT | DS1104_DIO19_OUT );
   #endif
   }
```

```
   #endif // fi de MDL_START

/* Codi del programa */

static void mdlOutputs(SimStruct *S, int_T tid) //funció que s'executa cada temps de
mostreig
{

/* declaració de constants */
   #define Vdc 100.0
   #define L 0.005
   #define R 0.1
   #define Ts 150.0e-6
   #define C1 470e-6
   #define C2 470e-6
   #define lambdaDC 1.0
   #define Treshold 0.5

   #define B00 1 // declaracio de bits per la paraula de sortida
   #define B01 2
   #define B02 4
   #define B03 8
   #define B04 16
   #define B05 32

/* Declaració de variables locals*/
   real_T vpk;
   real_T vnk;
   real_T gop;
   real_T Saop;
   real_T Sbop;
   real_T Scop;
   real_T irefk1a;
   real_T irefka;
   real_T irefk1b;
   real_T irefkb;
   real_T ik1a;
   real_T ika;
   real_T ik1b;
   real_T ikb;
   real_T iak1;
   real_T ibk1;
   real_T ick1;
```

```
real_T ik1c;
real_T iok1;
real_T vpk1;
real_T vnk1;
real_T va;
real_T vb;
real_T vc;
real_T vxa;
real_T vxb;
real_T eka;
real_T ekb;
real_T iak2;
real_T ik2a;
real_T ibk2;
real_T ik2b;
real_T ick2;
real_T iok2;
real_T vp;
real_T vn;
real_T Sm1a;
real_T Sm1b;
real_T Sm1c;
real_T iok2a;
real_T iok2b;
real_T iok2c;
real_T Sa;
real_T Sb;
real_T Sc;
int_T Sap;
int_T San;
int_T Sbp;
int_T Sbn;
int_T Scp;
int_T Scn;
real_T vpk2;
real_T vnk2;
real_T g;
int_T S0;
int_T S1;
int_T S2;
int_T S3;
int_T S4;
int_T S5;
```

```
/* Adquisició de dades dels inputs i assignació a punters */
   InputRealPtrsType uPtrs0 = ssGetInputPortRealSignalPtrs(S, 0);
   InputRealPtrsType uPtrs1 = ssGetInputPortRealSignalPtrs(S, 1);
   InputRealPtrsType uPtrs2 = ssGetInputPortRealSignalPtrs(S, 2);
   InputRealPtrsType uPtrs3 = ssGetInputPortRealSignalPtrs(S, 3);
   InputRealPtrsType uPtrs4 = ssGetInputPortRealSignalPtrs(S, 4);
   InputRealPtrsType uPtrs5 = ssGetInputPortRealSignalPtrs(S, 5);
   InputRealPtrsType uPtrs6 = ssGetInputPortRealSignalPtrs(S, 6);
   InputRealPtrsType uPtrs7 = ssGetInputPortRealSignalPtrs(S, 7);

/* Adjudicació de punters a variables */
   irefka= *uPtrs0[0];
   irefkb= *uPtrs1[0];
   ika= *uPtrs2[0];
   ikb= *uPtrs3[0];
   vp= *uPtrs4[0];
   vn= *uPtrs5[0];
   eka= *uPtrs6[0];
   ekb= *uPtrs7[0];

/* Control predictiu*/

/* Inicialtització */
   vpk=vp;
   vnk=vn;
   gop=10.0e100;
   Saop=0.0;
   Sbop=0.0;
   Scop=0.0;

/* Current de referència */
   irefk1a=irefka;
   irefk1b=irefkb;

/* Current calculat per k+1 */
   ik1a=(1.0-(R*Ts/L))*ika+(Ts/L)*(vxm1a-eka);
   ik1b=(1-(R*Ts/L))*ikb+(Ts/L)*(vxm1b-ekb);


/* Predicció de current de fase per k+1 */
   iak1=ik1a;
   ibk1=-0.5*ik1a+(sqrt(3)/2)*ik1b;
```

```
    ick1=-0.5*ik1a-(sqrt(3)/2)*ik1b;

/* Predicció de balanç de tensió continua per k+1 */
    if ((SV[0][0])==0.0) {Sm1a=iak1;} else {Sm1a=0.0;}
    if ((SV[0][1])==0.0) {Sm1b=ibk1;} else {Sm1b=0.0;}
    if ((SV[0][2])==0.0) {Sm1c=ick1;} else {Sm1c=0.0;}
    iok1=Sm1a+Sm1b+Sm1c;
    vpk1=vpk+(1.0/(2.0*C1))*iok1*Ts;
    vnk1=vnk+(1.0/(2.0*C2))*iok1*Ts;

 /* Avaluació i optimització de g pels 27 estats */
   for (Sa=-1.0; Sa<=1.0; Sa=Sa+1.0) {
    for (Sb=-1.0; Sb<=1.0; Sb=Sb+1.0) {
       for (Sc=-1.0; Sc<=1.0; Sc=Sc+1.0) {

          /*vectors*/
          va=Sa*Vdc;
          vb=Sb*Vdc;
          vc=Sc*Vdc;

          /*vx=(valpha)+(vbeta)*/
          vxa=((1.0/3.0)*(va-0.5*vb-0.5*vc));
          vxb=(1.0/3.0)*((sqrt(3.0)/2.0)*vb-(sqrt(3.0)/2.0)*vc);

          /* Predicció de current per k+2*/
          ik2a=(1-(R*Ts/L))*ik1a+(Ts/L)*(vxa-eka);
          ik2b=(1-(R*Ts/L))*ik1b+(Ts/L)*(vxb-ekb);

          /* Predicció del current de fase per k+2 */
          iak2=ik2a;
          ibk2=-0.5*ik2a+(sqrt(3)/2)*ik2b;
          ick2=-0.5*ik2a-(sqrt(3)/2)*ik2b;

          /* Predicció de balanç de tensió continua per k+2 */
          if (Sa==0.0) {iok2a=iak2;} else {iok2a=0.0;}
          if (Sb==0.0) {iok2b=ibk2;} else {iok2b=0.0;}
          if (Sc==0.0) {iok2c=ick2;} else {iok2c=0.0;}
          iok2=iok2a+iok2b+iok2c;
          vpk2=vpk1+(1.0/(2.0*C1))*iok2*Ts;
          vnk2=vnk1+(1.0/(2.0*C2))*iok2*Ts;
```

```
          /* Avaluació de la funció de qualitat */
          g=(pow((irefk1a-ik2a),2))+(pow((irefk1b-
ik2b),2))+lambdaDC*(pow((vpk2+vnk2),2));

          if (g<gop)
          {
             gop=g;
             vxm1a=vxa;
             vxm1b=vxb;
             Saop=Sa;
             Sbop=Sb;
             Scop=Sc;
          }
       }  //end for Sc
    }  //end for Sb
 }  //end for Sa

/* Canvi d'estat pel proper temps de mostreig */
    SV[0][0]=Saop;
    SV[0][1]=Sbop;
    SV[0][2]=Scop;
 /*  Funció de commutració */
    if (Sa>=Treshold) {Sap=1;} else {Sap=0;}
    if (Sa<-Treshold) {San=1;} else {San=0;}
    if (Sb>=Treshold) {Sbp=1;} else {Sbp=0;}
    if (Sb<-Treshold) {Sbn=1;} else {Sbn=0;}
    if (Sc>=Treshold) {Scp=1;} else {Scp=0;}
    if (Sc<-Treshold) {Scn=1;} else {Scn=0;}

    S0=Sap;     /* interruptor S1 */
    S1=1-San;   /* interruptor S22 */
    S2=Sbp;     /* interruptor S3 */
    S3=1-Sbn;   /* interruptor S44 */
    S4=Scp;     /* interruptor S5 */
    S5=1-Scn;   /* interruptor S66 */

    #ifndef MATLAB_MEX_FILE
       ds1104_bit_io_write(B00*S0+B01*S1+B02*S2+B03*S3+B04*S4+B05*S5); //
escriptura sortida digital DSP
    #endif
    } // fi mdlOutputs
```

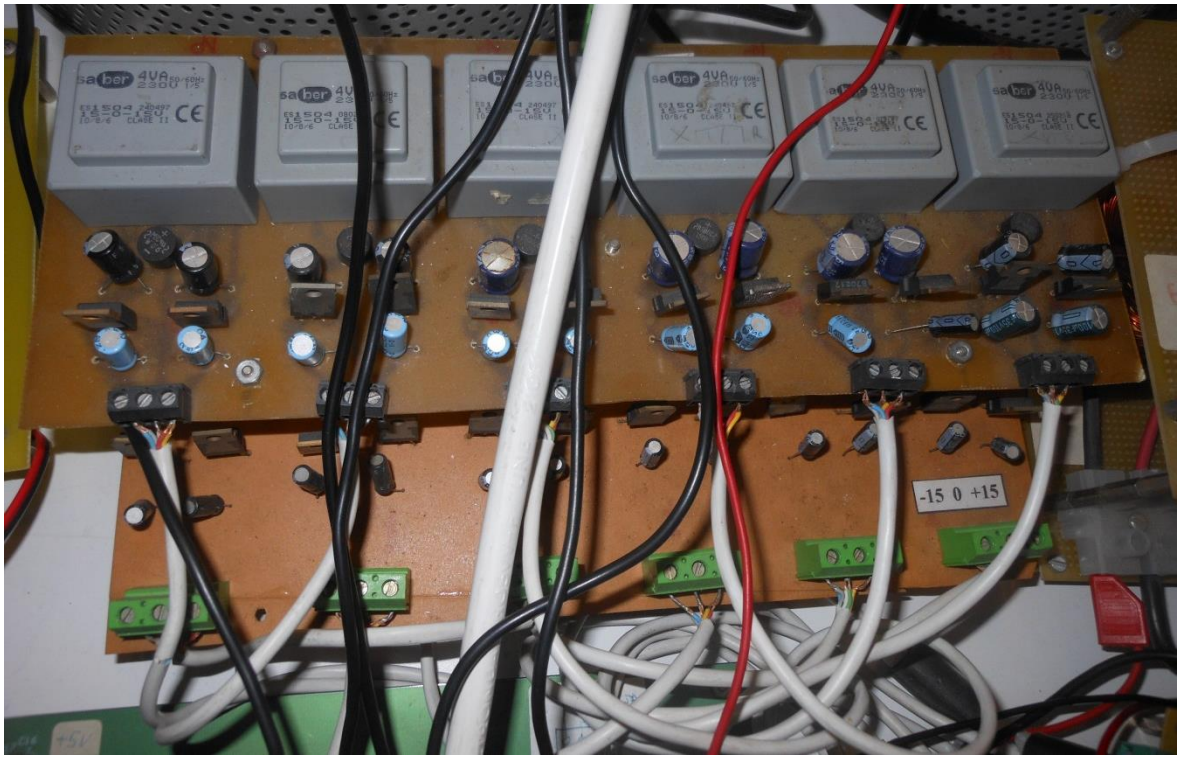
```
   static void mdlTerminate(SimStruct *S) //funció executada al final del programa
   {
   }

/* final estructura MEX S-Function */
#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

Annex IV. Fotografies de hardware.

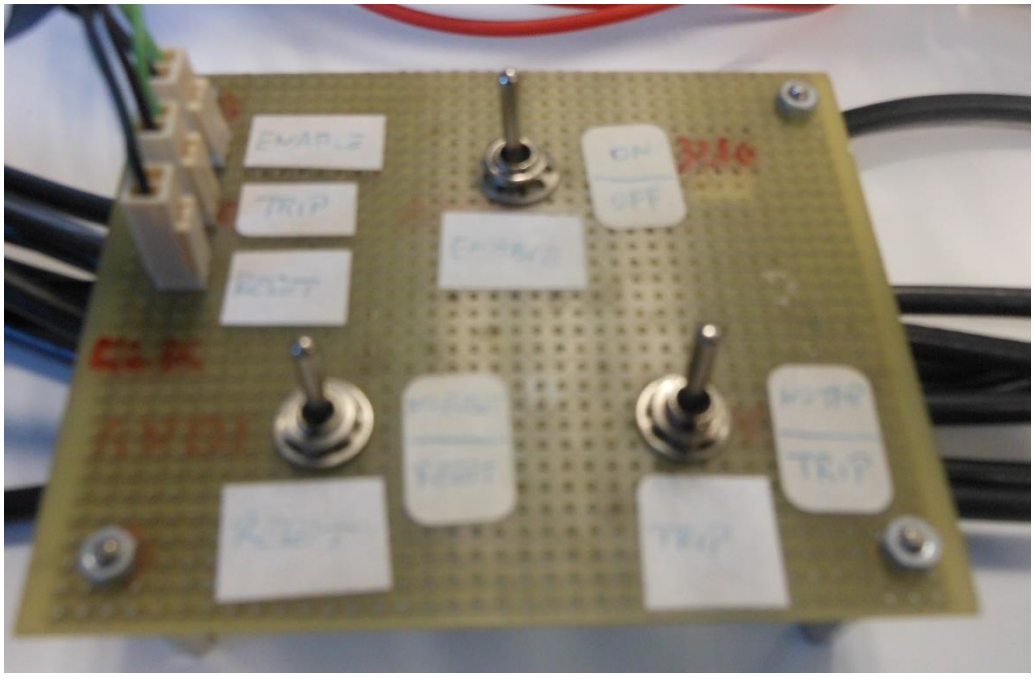
- **DSP DS1104**

- **Convertidor CC/CA de tres nivells**

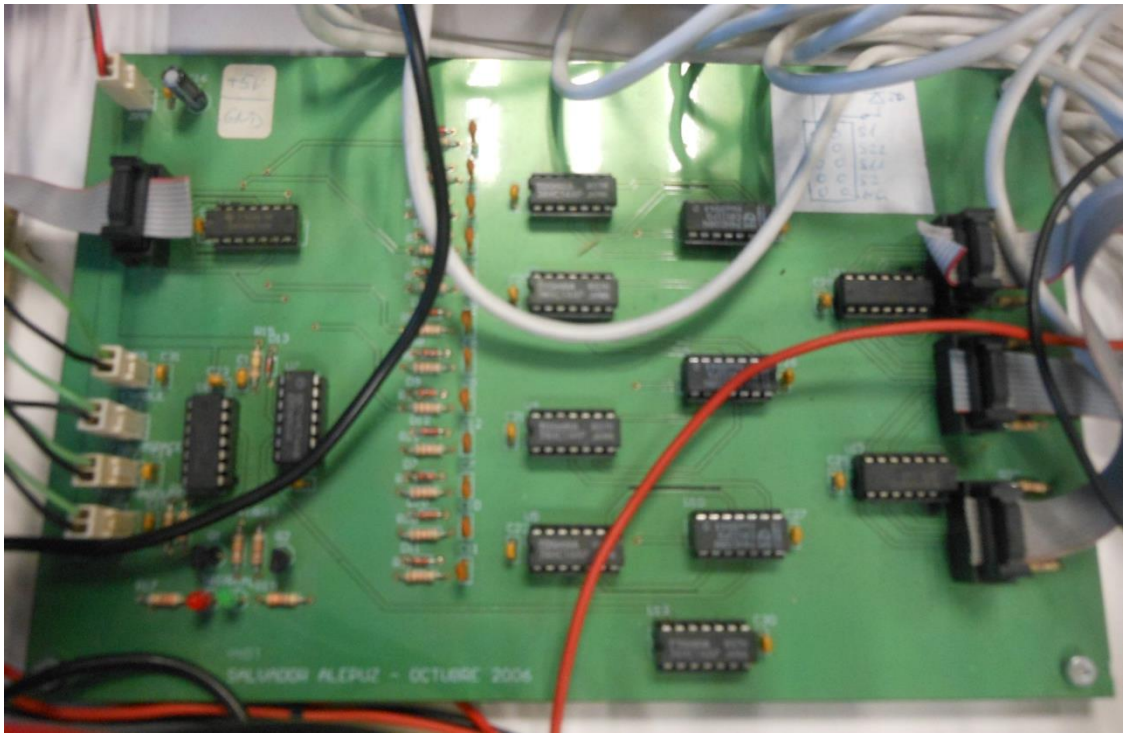
- **Alimentació**



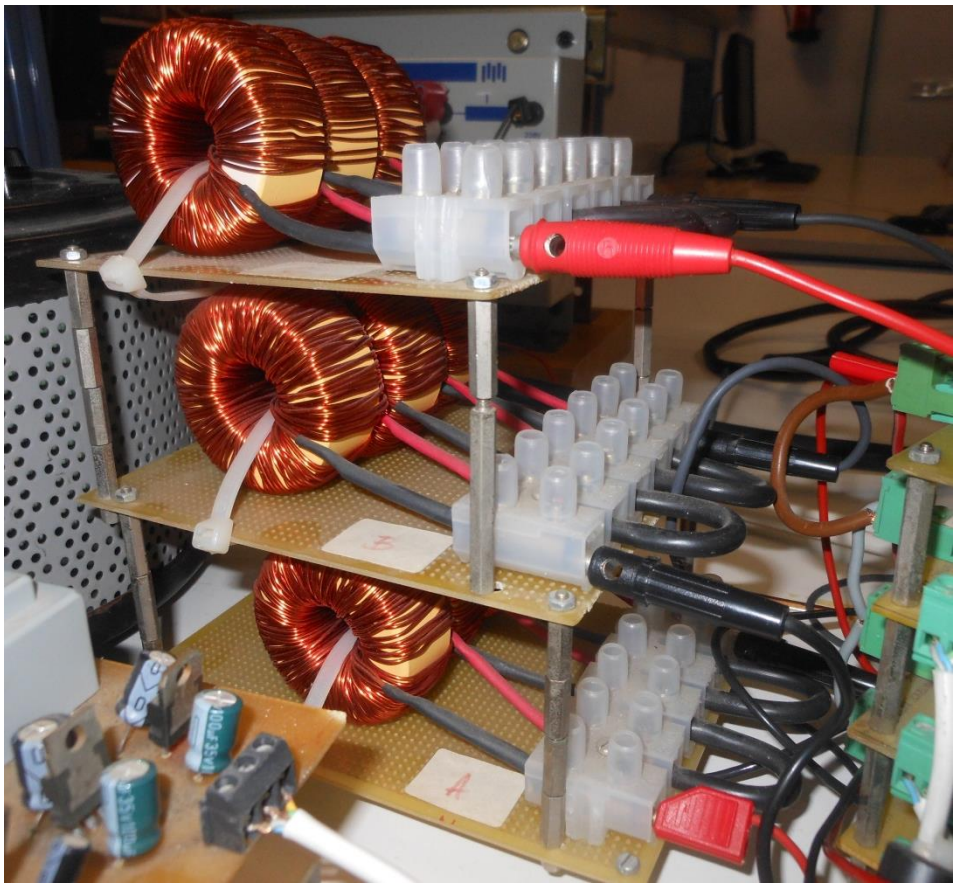
- **Interruptors Enable, Reset i Trip**



- **PCB control Reset, Trip i Enable**

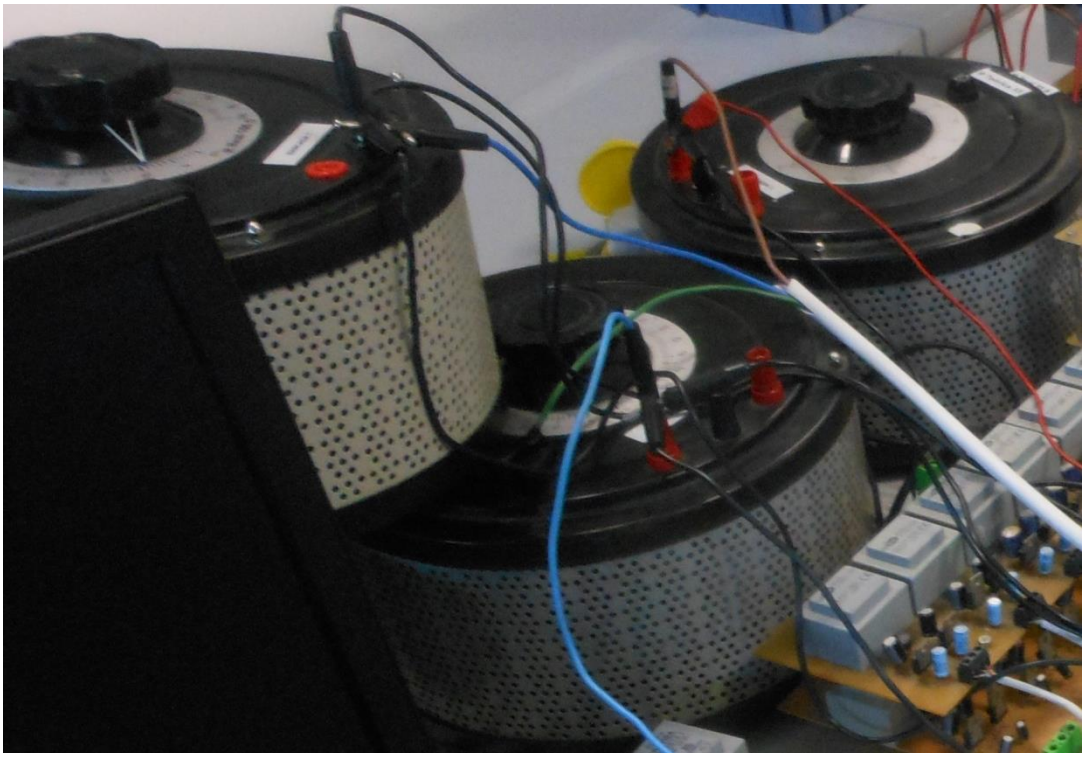


- **Bobines**

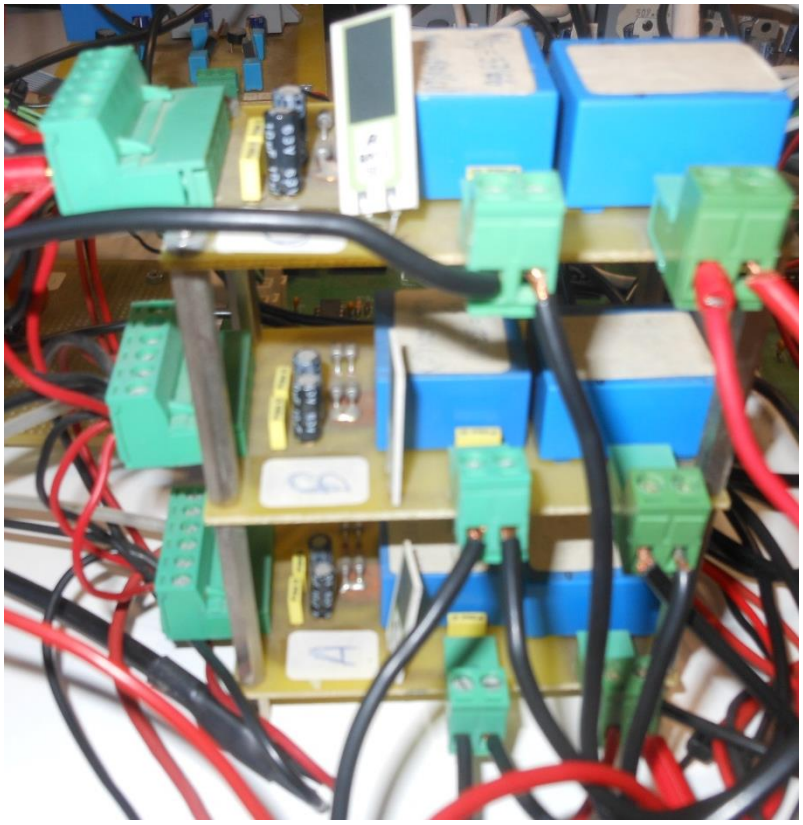


- **Font de tensió v_p i oscil·loscopi**

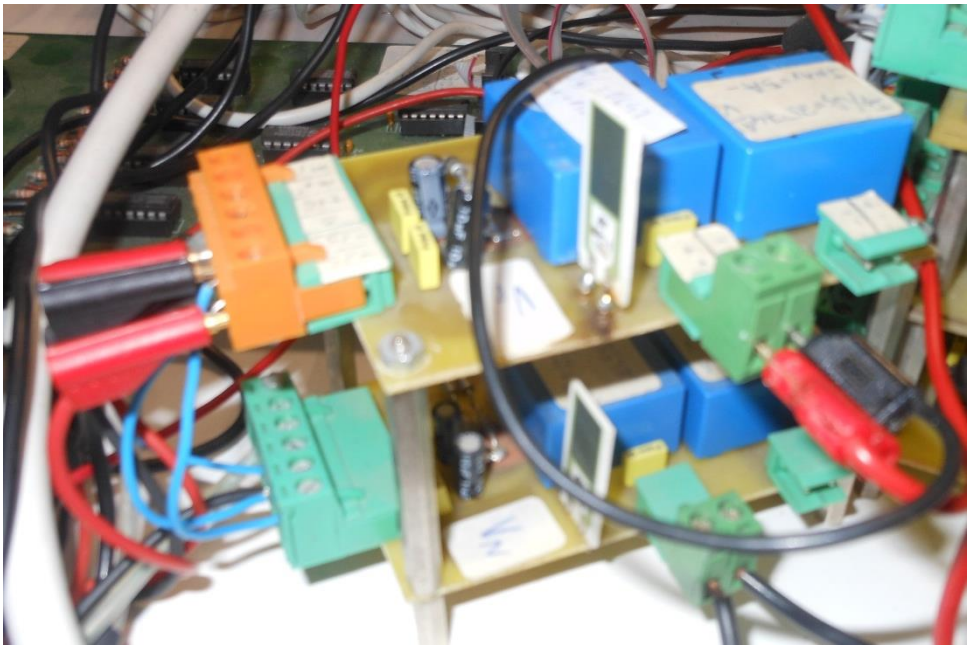
- **Reòstats de càrrega**



- **Sensors de tensió i corrent de i_{abc} i v_{abc}**



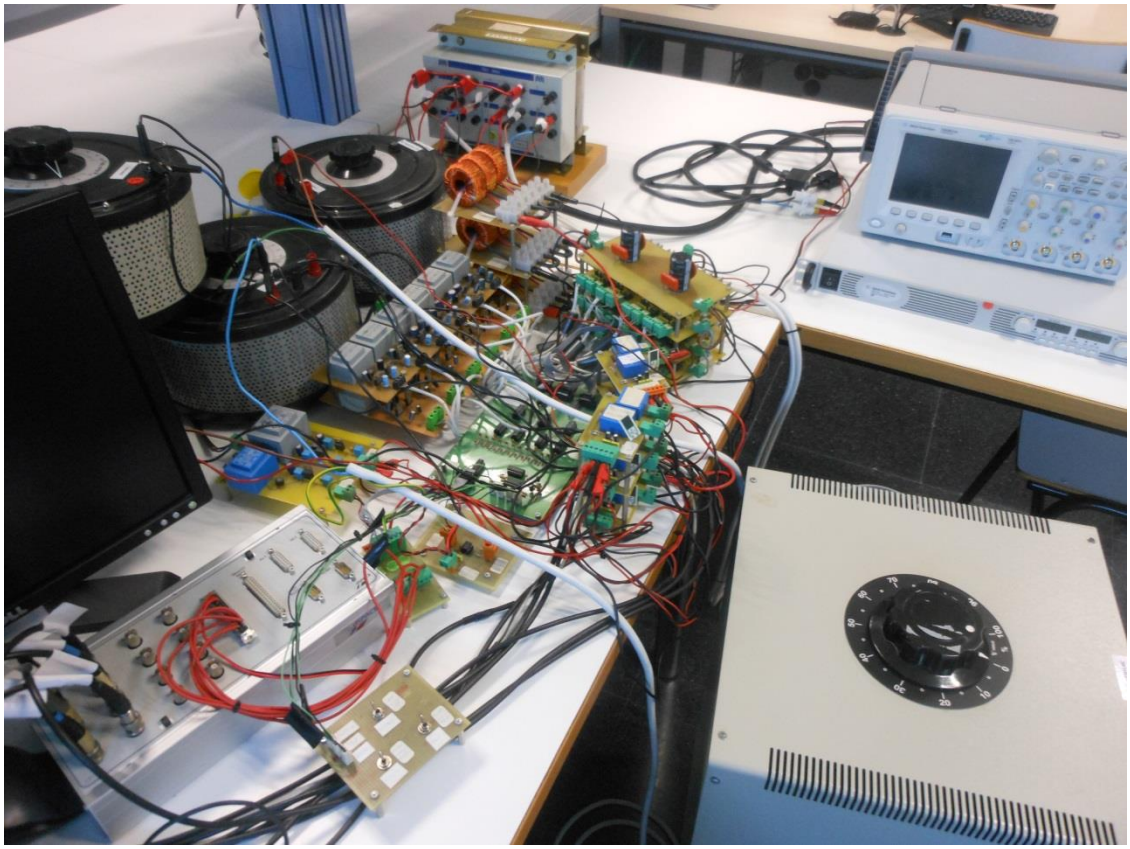
- **Sensors de tensió v_p i v_n**



- **Transformador trifàsic amb connexió Y-Y**

- **Regulador de tensió trifàsic**

- **Muntatge general**



Annex V. Continguts del CD-ROM.

1. Documentació del projecte (avantprojecte, memòria, estudi econòmic i annexos).
2. Programes de Matlab-Simulink dels tres models dissenyats.