Asynchronous VME Bus Requestor

Mitch Bradley

ABSTRACT

The VME bus grant line is daisy-chained through the bus requestor circuit of every card on the bus. An asynchronous bus requestor is preferable to a synchronous one because synchronization delays increase the propagation time of the bus grant signal. However, asynchronous requestors must be harder to design, because every asynchronous design that I have seen to date is flawed.

Here is a design for an asynchronous ROR (Release on Request) VME requestor which I believe to be correct. It has a small, predictable delay and requires only a few components.

Overview

A VME requestor may be viewed as a "black box" as in Figure 1.

The simplest kind of requestor is called 'Release When Done' (RWD). An RWD requestor is fairly simple. It requests the bus, takes it when offered, uses it, then gives it up.

The next level of complexity is the 'Release on Request' (ROR) requestor. An ROR requestor requests the bus, takes it when offered, uses it, but doesn't give up the bus unless some other device actually asks for it.

The easy part of a requestor's job is to properly sequence the signals on the bus. The hard part is making a decision: do I take (or keep) the bus, or do I release (or not take) the bus. The outcome of this decision is reflected in the signals 'DMAGR', 'BBSY', and 'BGOUT'. If the bus is taken, 'DMAGR' an 'BBSY' are asserted, and 'BGOUT' is not asserted. If the bus is not taken, 'BGOUT' is asserted and 'DMAGR' and 'BBSY' are negated.

The problem lies in ensuring that the decision is made cleanly, without glitching any of the lines.

For an RWD requestor, the solution is simple. The only time that the decision has to be made is when the leading edge of 'BGIN' comes along. Use 'BGIN' to clock a flip-flop whose D input is connected to 'DMAREQ'. Wait for the flip-flop to settle, then either propagete BGIN to BGOUT, or assert 'DMAGR' and 'BBSY', leaving 'BGOUT' not asserted. The key point is to wait for the flip-flop to settle before doing anything.

For an ROR requestor, it's harder. There are three times when a decision has to be made:

1) When we don't have the bus and the leading edge of 'BGIN' comes along.

2) When we do have the bus, and we want to use it, as indicated by 'DMAREQ' becoming true.

3) When we do have the bus, and another device wants to use it, as indicated by one of the 'BR' signals becoming true.

The trick here is to cope with the possibility that 'DMAREQ' becomes true at the same time as one of the 'BR's. Event 1 is mutually exclusive with either 2 or 3, because a BGIN

(Bus Grant In) can't happen while somebody has the bus. However, 2 and 3 can occur at any time relative to each other. When either 2 or 3 occurs, the requestor has to decide whether to keep the bus or give it up. The decision has to be made cleanly, in case both 2 and 3 occur at nearly the same time. Furthermore, the loser in the decision can't be locked out forever; if the requestor decides to keep the bus, it must not lose track of the fact that someone wants the bus, so that the bus can be released as soon as the the on-board master is through.

See Figure 2 for a conceptual diagram of a solution. The important features are:

a) MyBus selects either BGIN or the 'or' of the bus requests.

b) The TakeBus line must not be sampled until DecisionValid is true.

c) If we have the bus and we decide to use it again (because of a DMAREQ), the Event line must be driven low (by DMAGR) so that pending external bus requests won't be locked out. If there is an external bus request pending, it will cause a new Event as soon as DMAGR is negated.

d) If we have the bus but it is taken away by an external bus request, and about the same time a DMAREQ arrives, the bus must be re-requested. Since MyBus is now false (the bus was taken away!), BGIN is the only signal that can cause an Event.

Full Implementation

The full implementation is shown in Fugure 3. Most of the logic fits in one PAL16L8. The only externals are a 74F74 flip-flop, one section of a delay line, and two sections of a 7438. The 74F74 is needed because the metastable time of flip-flops in PAL's is very long. The 7438's are needed in order to meet the drive specifications of the open-collector VME BRx- and BBSY- lines.

The section of the 74F74 that is connected to the output of the delay line is necessary because the delay line delays not only the leading edge of the Event signal, but also the trailing edge. If another Event were to occur very shortly after a cycle finished, it is conceivable that DecisionValid could still be true, due to the delay line. The flip flop ensures that the DecisionValid signal is negated when the Event signal is negated, instead of a delay time later.

AS, DTACK, and BERR are shown as inputs to the PAL, but they aren't strictly necessary for the requestor function. The VME spec requires that a master not start a cycle until the previous cycle has finished, as indicated by AS, DTACK, and BERR all being false. Strictly speaking, this is the responsibility of the 'master' circuit, instead of the requestor. However, it is sufficient (and convenient) to not assert DMAGR until AS, DTACK, and BERR are all false, thus making DMAGR a positive indication that it is okay to proceed with a bus cycle.

The pal equations are given. A number of the equations have been converted to their inverted (De-Morgan'ized) forms because the associated output needed to be active-high rather than active-low.

Interface to DMA Master Circuit

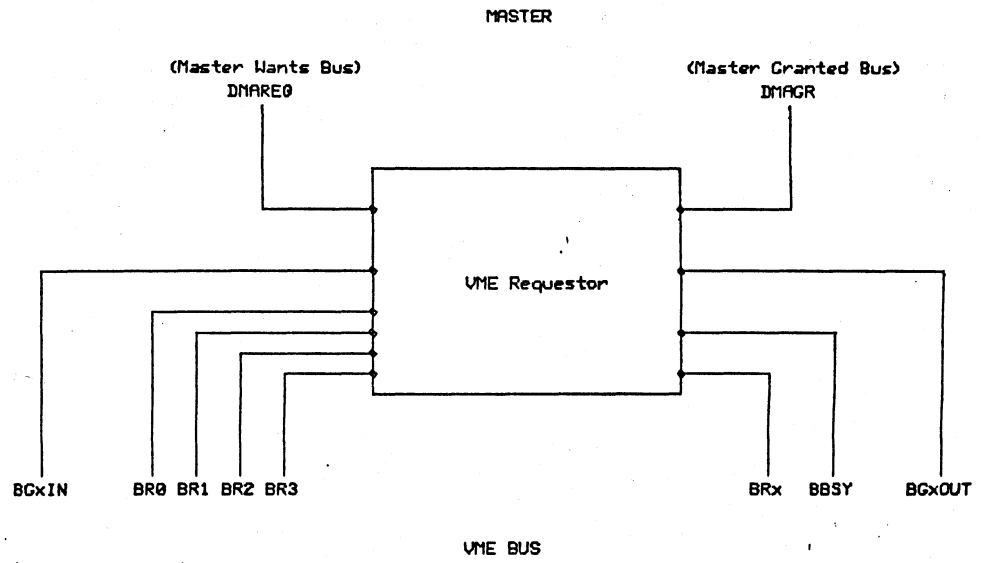
The interface between the requestor and the master circuit is a 2-wire handshake. When the master wants the bus, it asserts DMAREQ. When the requestor has obtained the bus, it asserts DMAGR. When the master has finished a cycle, it negates DMAREQ, then the requestor negates DMAGR. The master must negate DMAREQ at the end of a cycle if it wishes to allow the requestor to give up the bus. Negating DMAREQ does not force the requestor to release the bus; it only allows it to do so if someone else wants it. If the master wishes to lock onto the bus for an extended period of time, it may do so by not negating DMAREQ until it has finished with the bus.

Performance

Assuming a 25-nsec PAL and a 50-nsec delay line, the worst-case time to propagate BGOUT is 25 (to Event) + 50 (to DecisionValid) + 25 (through PAL again) + 20 (7438) =

120 nsec. This could be reduced to perhaps 80 nsec by using FAST SSI gates and reducing the delay time. A 16-Mhz synchronous design using the same 7438 as the BGOUT driver has a worst-case time of 145 nsec, totally neglecting any gate delays in the circuit, and requires synchronizing just about every signal in sight.

Figure 1



Inputs to Requestor on left   Outputs from Requestor on right

BGxIN BGxOUT BRx   Replace x with 0 1 2 or 3 depending on the level of the Requestor

FIGURE 2

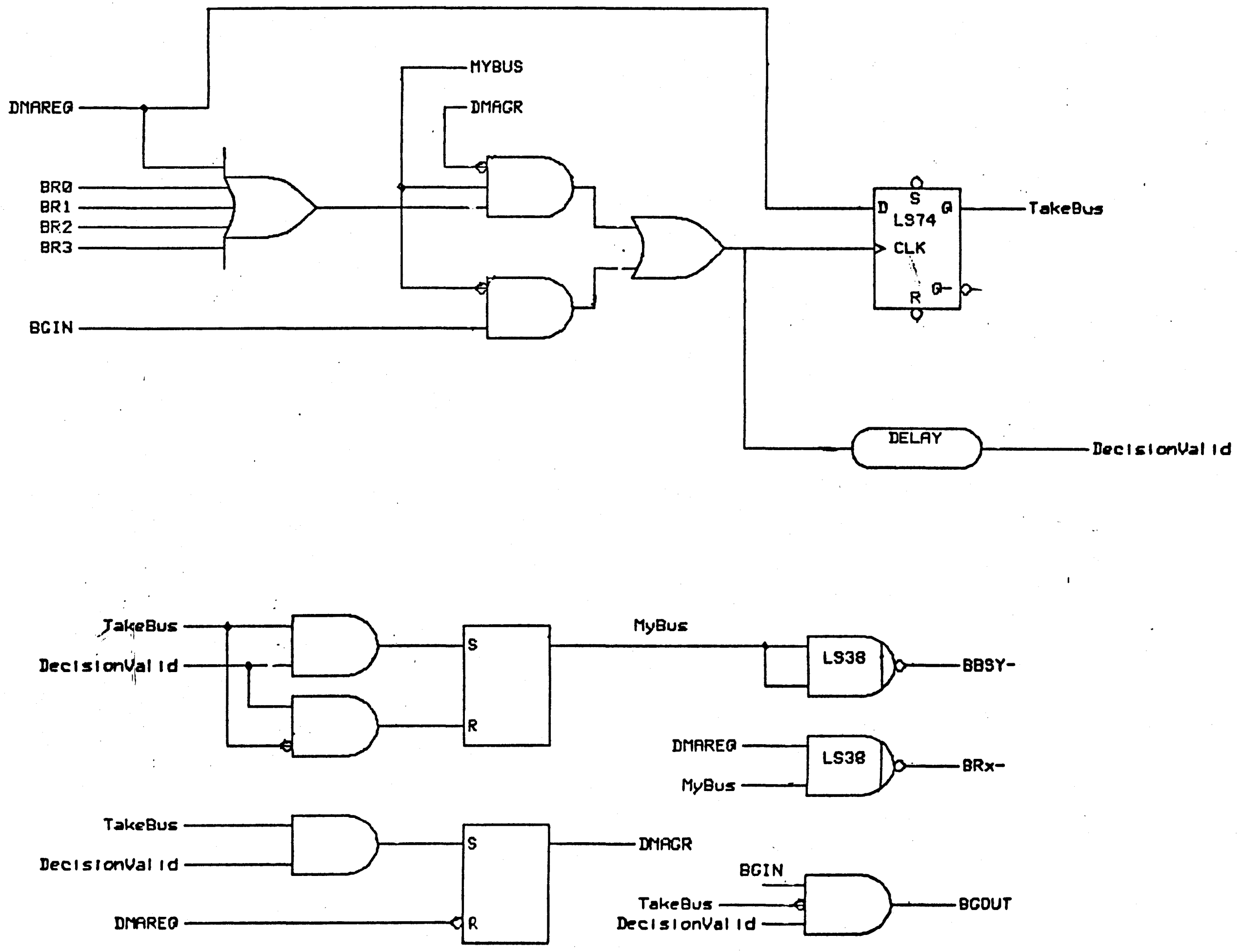
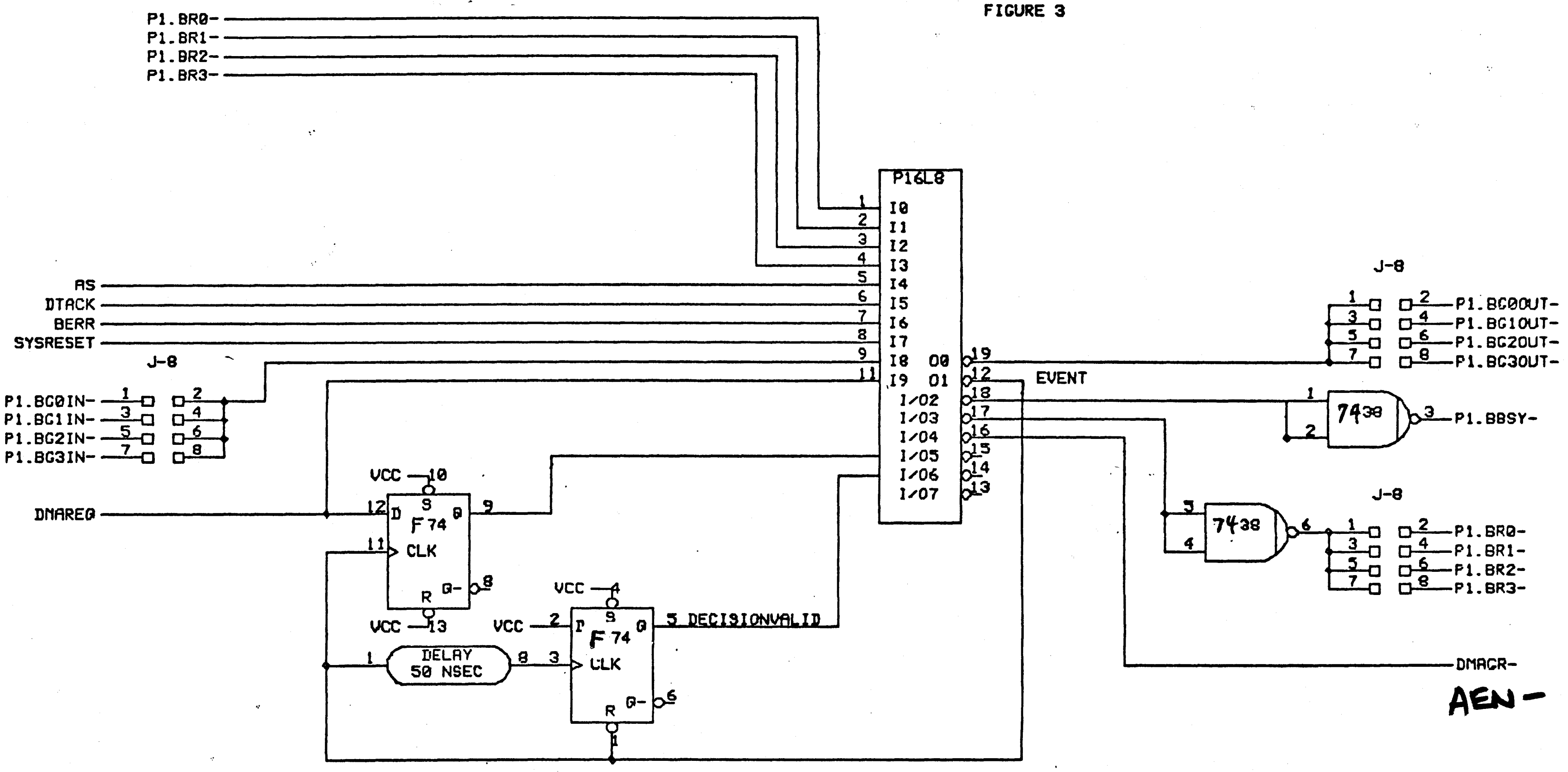

DNAREQ
BR0
BR1
BR2
BR3
BGIN
MYBUS
DMAGR
D
S
LS74
Q
CLK
R
Q
TakeBus
DELAY
DecisionValid
TakeBus
DecisionValid
S
R
MyBus
LS38
BBSY-
DMAREQ
LS38
BRx-
MyBus
TakeBus
DecisionValid
S
R
DMAGR
DMAREQ
BGIN
TakeBus
DecisionValid
BGOUT

FIGURE 3



P1.BR0-
P1.BR1-
P1.BR2-
P1.BR3-
AS
DTACK
BERR
SYSRESET
J-8
P1.BG0IN-
P1.BG1IN-
P1.BG2IN-
P1.BG3IN-
DMAREQ
P16L8
I0
I1
I2
I3
I4
I5
I6
I7
I8
I9
O0
O1
I/O2
I/O3
I/O4
I/O5
I/O6
I/O7
EVENT
VCC
F74
CLK
DELAY 50 NSEC
DECISIONVALID
J-8
P1.BG0OUT-
P1.BG1OUT-
P1.BG2OUT-
P1.BG3OUT-
7438
P1.BBSY-
J-8
P1.BR0-
P1.BR1-
P1.BR2-
P1.BR3-
DMACR-
AEN-

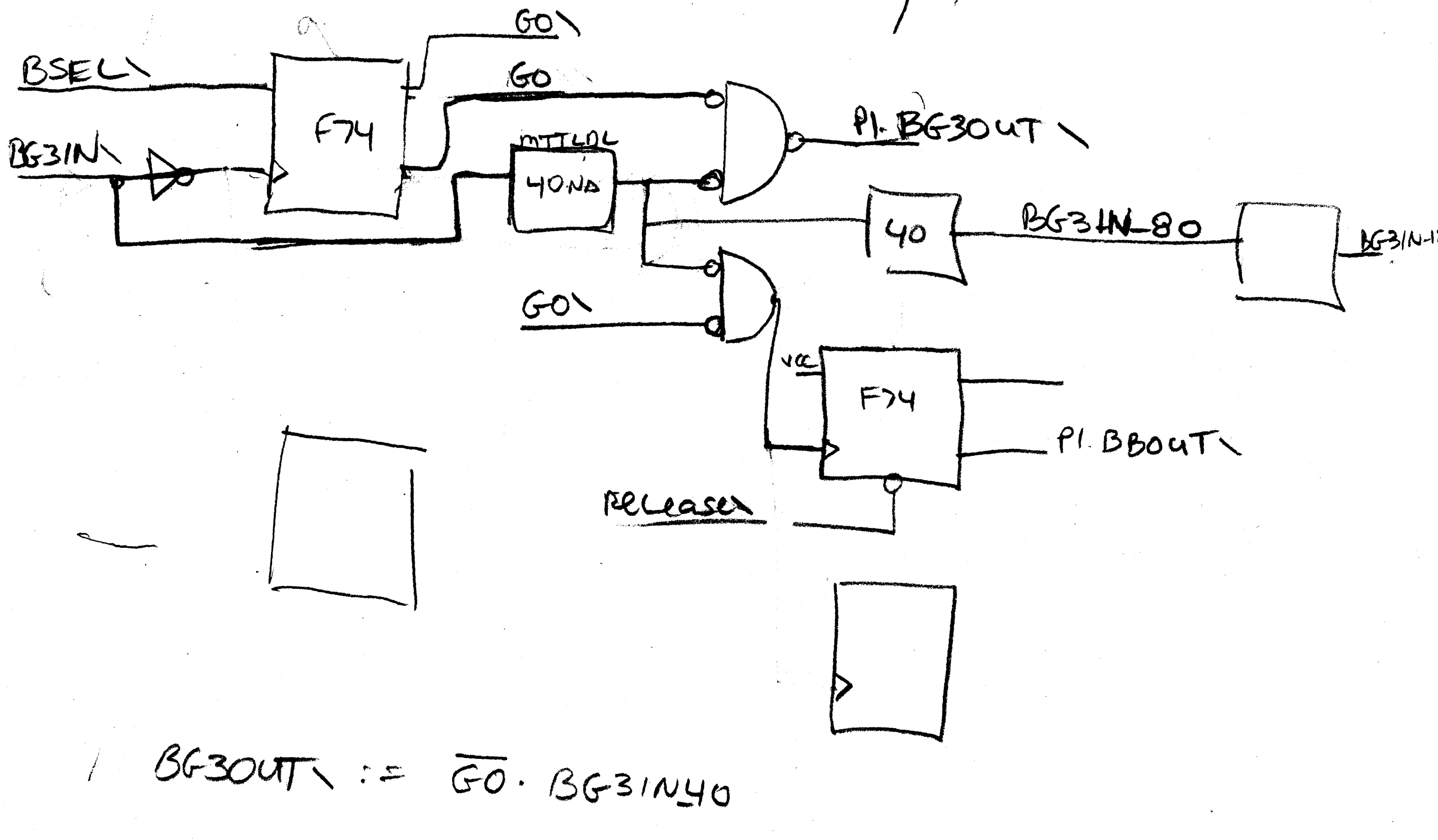
Arbiter

$\overline{BG3IN} := \overline{Reset} \cdot ARB \cdot BR3 \cdot \overline{BBSY}$

CPU Requestor

$\overline{BROUT} := BSEL \cdot \overline{BBOUT} \cdot \overline{AEN}$

$BROUT1 := BROUT$ /* Small delay */

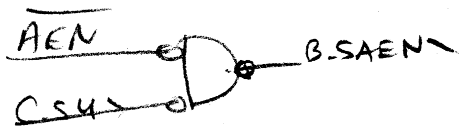


$BG3OUT\backslash := \overline{GO} \cdot BG3IN\_40$

$BBOUT\backslash := GO \cdot BG3IN\_40$
$\quad + ~~GO \cdot BG3IN\_80~~$
$\quad + BBOUT \cdot /Release$

$Release\backslash := \cdot \overline{BROUT} \cdot BR \cdot BBOUT \cdot P1\_AS \cdot Bsel$
$\quad + \overline{BROUT} \cdot BR \cdot BBOUT \cdot \overline{CS4} \cdot \overline{BSEL}$
$\quad + \overline{BROUT} \cdot BR \cdot BBOUT \cdot CS6 \cdot \overline{BSEL}$

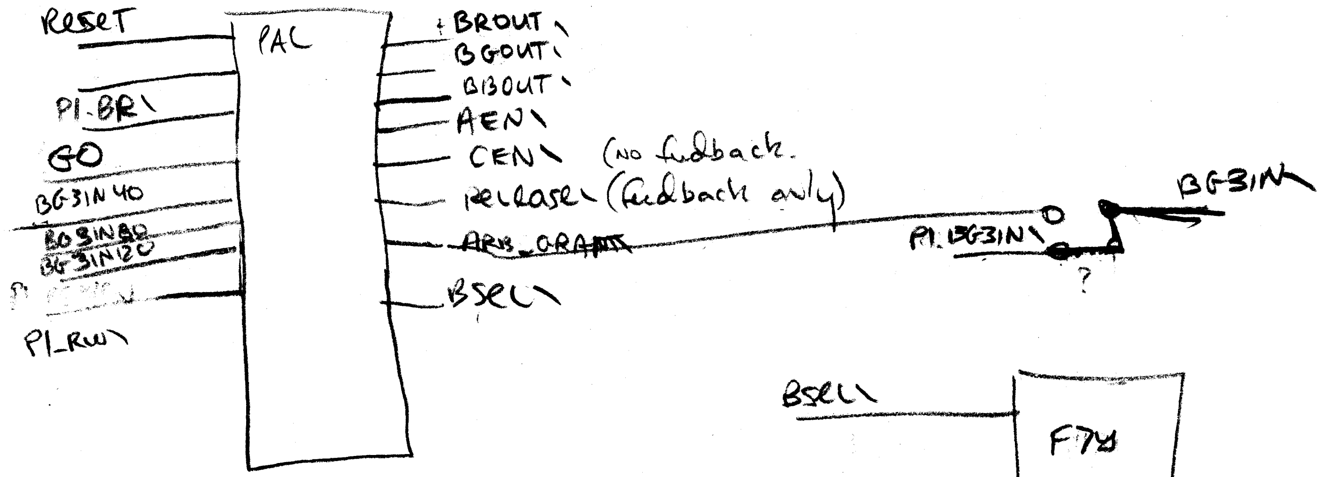
$$\text{AEN} := \text{GO\$B3IN\_40}$$
$$+\ \text{AEN} \cdot \text{BBOUT}$$
$$+\ \text{AEN} \cdot \text{P1.AS}$$

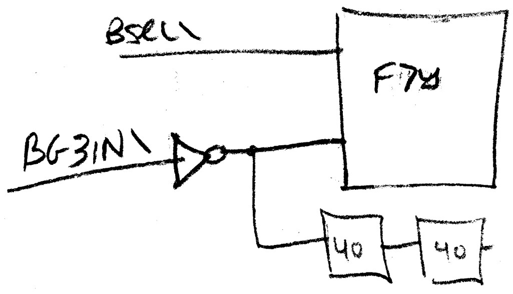


$$\text{BEN} := \text{AEN} \cdot \overline{\text{BGIN40}} \cdot \overline{\text{BGIN80}} \cdot \overline{\text{BGIN120}}$$

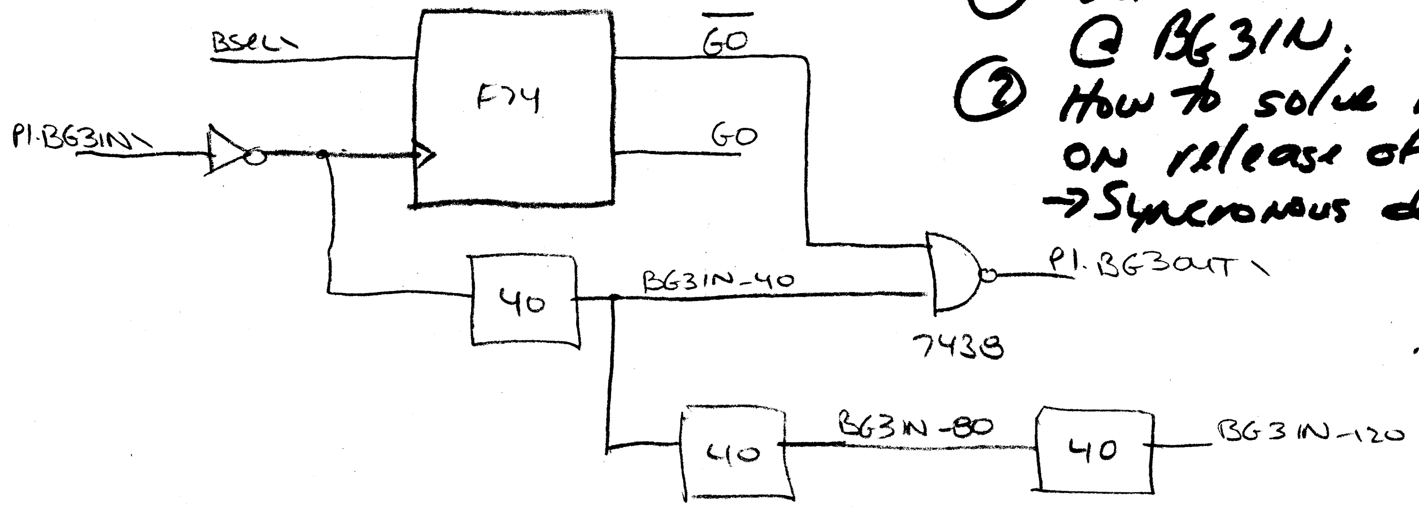
Adds 80 ns to 1st access.

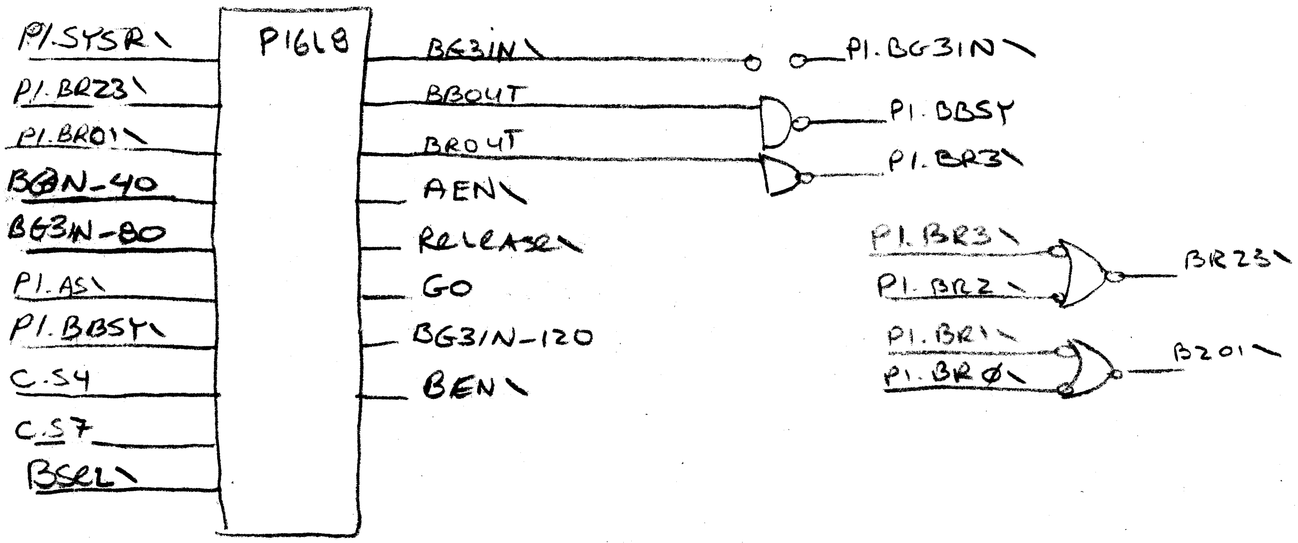
$$\text{CEN} := \text{BSEL} \cdot \text{BEN}$$





① Solve MetaStable @ BG3IN.
② How to solve metastable on release of BBSY?
→ Syncronous design.





BG3IN\ := $\overline{SYSR} \cdot BR23 \cdot \overline{BBSY}$
$+ SYSR \cdot BR01 \cdot \overline{BBSY}$

BBOUT := $\overline{GO} \cdot \overline{BBOUT}$
$+ \overline{GO} \cdot RELEASE$
$+ \overline{BGIN\text{-}40} \cdot \overline{BBOUT}$
$+ \overline{BGIN\text{-}40} \cdot \overline{BBOUT}$

% Demorganize: $GO \cdot BGIN\text{-}40$
$+ BBOUT \cdot /Release$

BROUT := $\overline{BSEL} + BBOUT + AEN$ % Demorganized

AEN\ := $GO \cdot BG3IN40$
$+ AEN \cdot BBOUT$
$+ AEN \cdot PLAS$

BEN\ := $AEN \cdot \overline{BGIN\text{-}40} \cdot \overline{BGIN\text{-}80} \cdot \overline{BGIN\text{-}120}$

RELEASE := $\overline{BROUT} \cdot (BR01 + BR23) \cdot BBOUT \cdot PLAS \cdot \overline{BSEL}$
$+ \overline{BROUT} \cdot (BR01 + BR23) \cdot BBOUT \cdot \overline{CS4} \cdot \overline{BSEL}$
$+ \overline{BROUT} \cdot (BR01 + BR23) \cdot BBOUT \cdot CS7 \cdot \overline{BSEL}$

```
% VME Bus Requestor

paltype pal1618
palname requestor
palid %I% %E%

PALBEGIN

% Inputs

 1 INPUT p1.br0-
 2 INPUT p1.br1-
 3 INPUT p1.br2-
 4 INPUT p1.br3-
 5 INPUT as
 6 INPUT dtack
 7 INPUT berr
 8 INPUT sysreset
 9 INPUT p1.bgin-
11 INPUT dmareq
14 INPUT DecisionValid
15 INPUT dmareq.s

% outputs

19 OUTPUT p1.bgout-
18 OUTPUT bbsy
17 OUTPUT brout
16 OUTPUT dmagr-
12 OUTPUT event

10 GND
20 VCC

EQUATIONS

% Event means that the requestor has to make a decision.
% Three things can cause an event:
%
% 1) If we aren't asserting bbsy, meaning we don't have control of the
%    bus, only p1.bgin- is an event.
% 2) If we are asserting bbsy (we have the bus), dmareq is an event
%    signifying that we want to use the bus.
% 3) If we are asserting bbsy (we have the bus), any p1.brx- is an event
%    signifying that someone else wants to use the bus.
%
% For any event, there are two possible outcomes:
% Either we get the bus, or we don't get it.
%
% An event clocks the external dmareq synchronizer flip flop, which
% decides whether or not dmareq was asserted.  The event signal is also
% fed through an external delay line and back into the pal as DecisionValid.
% The delay provides settling time for the synchronizer.
%
% After the delay, the pal makes the decision whether to take or
% release the bus.
%
% Event has to be deasserted after the bus is acquired, in case an
% external request comes in during our cycle.  This guarantees that the second
% event will eventually cause a rising edge on the event line, and it will
% not be ignored.

% event = bbsy * / dmagr * ( dmareq + p1.br0 + p1.br1 + p1.br2 + p1.br3 )
%       + / bbsy * p1.bgin

ASSERT event  % Conditions to turn OFF event (De-Morganized)
```

```
OR  / bbsy  / p1.bgin % idle state - we don't own the bus, no bus grant
OR    bbsy    dmagr   % After we've granted ourself the bus, turn off event

OR  / p1.bgin  dmagr  % Probably redundant - if we've granted ourself the
                      % bus, bbsy must be true, so p1.bgin must be false

OR    bbsy  / dmagr  / p1.br0  / p1.br1  / p1.br2  / p1.br3
        % no event if we own the bus but nobody wants it

OR  / p1.bgin  / dmagr  / p1.br0  / p1.br1  / p1.br2  / p1.br3
        % no event if nobody wants the bus and no grant

% bbsy = / sysreset *
%        (
%           ( DecisionValid * dmareq.s )
%          + ( bbsy * ( / DecisionValid + dmareq.s ) )
%        )

ASSERT bbsy         ( De-Morganized )
ENABLE  ALWAYS
    % turn off on system reset
OR      sysreset

    % stay off until a bus grant has clocked-in our request
OR      / dmareq.s / bbsy

    % and until a flip-flop settling delay after the bus grant
OR      / DecisionValid    / bbsy

    % once on, don't turn off until the synchronized dmareq has
    % gone away and it has been validated
OR      / dmareq.s   DecisionValid


% br = / sysreset * dmareq * / bbsy

ASSERT brout    % De-Morganized !
ENABLE  ALWAYS
OR      sysreset          % turn off on system reset
OR      / dmareq          % no bus req. if we don't need the bus
OR        bbsy            % no bus req. if we already have the bus
                          % or after we have captured it

ASSERT dmagr-
ENABLE  ALWAYS
    % on when our request has been granted and flip-flop has settled
    % and everybody is off the bus
OR      dmareq  dmareq.s  DecisionValid / sysreset  / as / dtack / berr
    % stay on until request goes away
OR      dmareq  dmagr / sysreset

ASSERT p1.bgout-                  % daisy-chained bus grant out
ENABLE  ALWAYS
   % pass grant if we don't want the bus, but first wait for f/f to settle
OR      p1.bgin DecisionValid / dmareq.s

PALEND
```