

Re: How to develop a random number generation device

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2007-09/msg01455.html>

- *From:* MooseFET <kensmith@xxxxxxxxxx>
 - *Date:* Sat, 08 Sep 2007 17:52:11 -0700
-

On Sep 8, 3:16 pm, John Larkin
<jjlar...@xx> wrote:

On Sat, 08 Sep 2007 08:21:15 -0500, John Fields

<jfie...@xxxxxxxxxxxxxxxxxxxxxxxx> wrote:

On Fri, 07 Sep 2007 13:44:15 -0700, John Larkin
<jjlar...@xx> wrote:

On Fri, 07 Sep 2007 15:06:31 -0400, Spehro Pefhany
<speffS...@xxxxxxxxxxxxxxxxxxxxxxxx> wrote:

On Fri, 07 Sep 2007 10:21:50 -0500,
Vladimir Vassilevsky
<antispam_bo...@xxxxxxxxxx> wrote:

John Larkin wrote:

Hmmm,
seems
like
all
0's
must
be
the
lockup

Re: How to develop a random number generation device

for
an
all
XOR
feedback.

OK,
but
you
missed
my
point,
which
was
that
it's
possible
to
eliminate
the
lockup
state
by
forcing
it
to
be
part
of
the
sequence.

I don't
follow that.
One state,
all 0's
usually, is
the lockup.
How do
you force
that to be
part of the
sequence?

Re: How to develop a random number generation device

You can
add a
lockup state
detector, a
big OR gate
or
something,
and
jam in a "1"
if the whole
register ever
gets to the
all-0's state,
but
then the
all-0's state
is not part
of the
sequence,
because it
never
happens
again.

It can happen at the startup
though. You have to ensure
the nonzero
initial state.

VLV

If you care about fault tolerance you will
ensure recovery from a zero
state which occurs at any time.

Best regards,
Spehro Pefhany

That gets into the philosophical issue: should we attempt to
detect
and correct for transient hardware errors in digital systems?

Re: How to develop a random number generation device

That can
apply to config bits in FPGAs (do we check them on a
regular basis?),
registers in uPs (including PC, SP, etc), values in counters,
whatever.

We generally assume that if it's broke, it's broke.

The point here, though, is that the machine will get itself unbroke
if it ever accidentally gets into what would normally have been the
lock-up state.

And my point is that it shouldn't "accidentally" get into a broken
state, any more than the program counter of a CPU should accidentally
find itself in never-never land.

If a digital system is unreliable, the cause should be found and
fixed. The problem with kluges like this is the same problem with
watchdog timers: they hide the real problem, so keep it from getting
fixed.

I always turn off the watchdog timer on test units, and protos
delivered to customers. I only enable it after we're sure we don't
need it.

Watch dogs don't always recover the system from a glitch. If you are
storing data in battery back RAM or flash, you need to be sure that
wrong values don't cause things to hang in some non-recoverable way.