DECEMBER 19, 2012

i.MX6 Gigabit Ethernet

We've recently been doing some digging into Gigabit Ethernet performance issues and questions for our i.MX6 boards and it's time to publish some of our results. We've discovered a number of settings and code updates that can dramatically improve network stability and throughput.

For the impatient

There are some architectural limitations on i.MX6 boards but some configuration options and driver issues are more likely to cause performance issues. We've identified a number of fixes that make the situation markedly better as shown below.

Before (TCP)

```
root@linaro-nano:~# cat /proc/cmdline
video=mxcfb0:dev=hdmi,1280x720M@60,if=RGB24 video=mxcfb1:off video=mxcfb2:off ...
root@linaro-nano:~# cat /proc/version
Linux version 3.0.35-2026-geaaf30e (b21710@bluemeany) ...
root@linaro-nano:~# while iperf -c 192.168.0.162 -r
                    | grep Mbits ; do echo -n ; done
[  5]  0.0-10.0 sec   474 MBytes   397 Mbits/sec
[  4]  0.0-10.1 sec  10.1 MBytes  8.47 Mbits/sec
[  5]  0.0-10.0 sec   474 MBytes   397 Mbits/sec
[  4]  0.0-10.0 sec  10.4 MBytes  8.72 Mbits/sec
[  5]  0.0-10.0 sec   472 MBytes   396 Mbits/sec
[  4]  0.0-10.0 sec  17.2 MBytes  14.4 Mbits/sec
```

After (TCP)

```
root@linaro-nano:~# cat /proc/cmdline
enable_wait_mode=off video=mxcfb0:dev=hdmi,1280x720M@60,if=RGB24 video=mxcfb1:off ...
root@linaro-nano:~# cat /proc/version
Linux version 3.0.35-2026-geaaf30e-02076-g68b5fa7 ...
root@linaro-nano:~# while iperf -c 192.168.0.162 -r
                    | grep Mbits ; do echo -n ; done
[  5]  0.0-10.0 sec   473 MBytes   397 Mbits/sec
[  4]  0.0-10.0 sec   509 MBytes   426 Mbits/sec
[  5]  0.0-10.0 sec   473 MBytes   397 Mbits/sec
[  4]  0.0-10.0 sec   508 MBytes   426 Mbits/sec
[  5]  0.0-10.0 sec   471 MBytes   395 Mbits/sec
[  4]  0.0-10.0 sec   510 MBytes   427 Mbits/sec
```

In the output from `iperf` above, each pair of lines indicate the transmit and receive bandwidth in that order. Note the horrible performance numbers for receive in the baseline. The UDP performance is markedly better, with transmit throughput of ~450 Mbits/s and receive speeds that can exceed 600 Mbits/s.

After (UDP)

```
[  4]  0.0- 1.0 sec  55.3 MBytes   462 Mbits/sec  0.084 ms   15/39459 (0.038%)
[  3]  0.0- 1.0 sec  72.8 MBytes   611 Mbits/sec  0.012 ms   -1/51843 (-0.0019%)
```

The details below will provide details of how we tested things, and describe a series of patches that lead to this improvement in both stability and speed.

Test environment

Four devices were used during the tests defined below:

- A Sony Vaio laptop with internal Gb Ethernet adapter,
- A Nitrogen6X board with i.MX6Quad TO 1.0
- A SABRE SD board with i.MX6Quad TO 1.1
- A SABRE Lite board with i.MX6Quad TO 1.2
- A Cisco Linksys SE2500 Gigabit Ethernet switch

The tests used a Linaro `nano` userspace. A tar-ball is available here (http://commondatastorage.googleapis.com/boundarydevices.com/imx6-iperf-test-20121214.tar.gz) that contains all of the kernel versions mentioned. Specific baseline kernel versions include:

- **Blue Meany** – This is the binary kernel (`uImage`) provided in the `images_L3.0.35_12.09.01-GA` release. We did not re-compile this kernel.
- **Boundary Before** – This is the first version we compiled, as a test to ensure that it matches *Blue Meany* and serves as the baseline for this series of tests.
- **Boundary Latest** – This is the latest release from the `boundary-L3.0.35_12.09.01-GA` branch of our Github kernel.

All of the testing was done with kernels based on Freescale's L3.0.35_12.09.01_GA release with various patches as described.

First change: enable_wait_mode=off

We've documented the first change made in this post (/wait-a-second-hopefully-less/) a week ago, but we didn't mention the throughput implications. In the output below, you can see around a 10% improvement in the *Blue Meany* kernel by just adding `enable_wait_mode=off` to the kernel command-line. This change made a huge difference on Tapeout 1.2. It increased the receive speed from on the order of 10 Mbits/s to ~200 Mbits/s. on Tapeout 1.0 devices, the difference was less dramatic, presumably because a number of the spots that use `enable_wait_mode` in the kernel are also conditional on the silicon revision. In any case, with just this change, both revisions of board have markedly increased receive performance as shown below.

```
root@linaro-nano:~# cat /proc/version
Linux version 3.0.35-2026-geaaf30e (b21710@bluemeany)...
root@linaro-nano:~# cat /proc/cmdline
enable_wait_mode=off ...
root@linaro-nano:~# while iperf -c 192.168.0.162 -r | grep Mbits ; do echo -n ; done
[  5]  0.0-10.0 sec   443 MBytes   372 Mbits/sec
[  4]  0.0-10.0 sec   250 MBytes   210 Mbits/sec
[  5]  0.0-10.0 sec   476 MBytes   399 Mbits/sec
[  4]  0.0-10.0 sec   252 MBytes   211 Mbits/sec
^C
```

As noted in the previous post, this environment update will also make ping times more consistent.

Measuring performance

In the summary above, we showed the output from the simplest invocation of `iperf`. When used as shown, the program will connect over TCP:

```
root@linaro-nano:~# iperf -c 192.168.0.162 -r
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 192.168.0.162, TCP port 5001
TCP window size: 58.4 KByte (default)
------------------------------------------------------------
[  5] local 192.168.0.119 port 52681 connected with 192.168.0.162 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec   475 MBytes   398 Mbits/sec
[  4] local 192.168.0.119 port 5001 connected with 192.168.0.162 port 42421
[  4]  0.0-10.0 sec   228 MBytes   191 Mbits/sec
```

Because of the use of TCP, flow control is imposed on the link by the upper layers, and the bandwidth is throttled to the slower of the speeds of the two ends. Using UDP removes this possible bottleneck and also allows a flag to set the target bandwidth (`-b SPEED`) and will show us the amount of packet loss. The `-t` flag allows us to override the default 10 second test for quicker results.

```
root@linaro-nano:~# iperf -c 192.168.0.162 -r -u -b 200M -t 2
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  106 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 192.168.0.162, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  106 KByte (default)
------------------------------------------------------------
[  4] local 192.168.0.119 port 51275 connected with 192.168.0.162 port 5001
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 2.0 sec  48.2 MBytes   202 Mbits/sec
[  4] Sent 34359 datagrams
[  4] Server Report:
[  4]  0.0- 2.0 sec  48.1 MBytes   202 Mbits/sec   0.063 ms   71/34358 (0.21%)
[  4]  0.0- 2.0 sec  1 datagrams received out-of-order
[  3] local 192.168.0.119 port 5001 connected with 192.168.0.162 port 53796
[  3]  0.0- 2.0 sec  48.3 MBytes   203 Mbits/sec   0.038 ms    0/34483 (0%)
[  3]  0.0- 2.0 sec  1 datagrams received out-of-order
```

Doing a quick smoke-test at a few key rates shows some interesting results. The following are slightly edited to make them more readable:

100Mbit/s UDP test

```
root@linaro-nano:~# iperf -c 192.168.0.162 -u -r -b 100M ;
[  4]  0.0- 2.0 sec  23.9 MBytes   100 Mbits/sec   0.048 ms    0/17082 (0%)
[  3]  0.0- 2.0 sec  24.0 MBytes   101 Mbits/sec   0.001 ms    3/17094 (0.018%)
```

400Mbit/s UDP test

```
root@linaro-nano:~# iperf -c 192.168.0.162 -u -r -b 400M -t 2;
[  4]  0.0- 2.0 sec  34.1 MBytes   143 Mbits/sec   0.091 ms    0/24338 (0%)
[  3]  0.0- 5.7 sec   205 MBytes   301 Mbits/sec   0.013 ms 198303/344825 (58%)
```

1Gbit/s UDP test

```
root@linaro-nano:~# iperf -c 192.168.0.162 -u -r -b 1000M -t 2;
[  4]  0.0- 2.0 sec   108 MBytes   453 Mbits/sec   0.036 ms   54/77241 (0.07%)
[  3]  0.0- 2.1 sec  64.9 MBytes   254 Mbits/sec  15.539 ms 95165/141465 (67%)
```

As you can see, there's no loss at 100M, very little loss at 400M and a huge amount of receiver loss at 1G (the second line reports the receiver numbers). Interestingly, the received bandwidth also decreased when going from 400M to 1Gbit/s. Using a script to be a bit more thorough, and convince ourselves that the pattern holds:

```
root@linaro-nano:~# cat > bwtest.sh << EOF
#!/bin/sh
bw=50;
while [ $bw -le 1000 ]; do
    echo "----------bandwidth $bw" ;
    iperf -c 192.168.0.162 -u -r -t 2 -b ${bw}M | grep % ;
    bw=`expr $bw + 50` ;
done
EOF
root@linaro-nano:~# chmod a+x bwtest.sh
root@linaro-nano:~# ./bwtest.sh
root@linaro-nano:~# ./bwtest.sh
----------bandwidth 50
[  4]  0.0- 2.0 sec  11.9 MBytes  50.0 Mbits/sec   0.010 ms    0/ 8510 (0%)
[  3]  0.0- 2.0 sec  11.9 MBytes  50.0 Mbits/sec   0.002 ms    0/ 8511 (0%)
----------bandwidth 100
[  4]  0.0- 2.0 sec  24.0 MBytes   100 Mbits/sec   0.048 ms    0/17085 (0%)
[  3]  0.0- 2.0 sec  24.0 MBytes   100 Mbits/sec   0.009 ms    4/17094 (0.023%)
----------bandwidth 150
[  4]  0.0- 2.0 sec  35.9 MBytes   150 Mbits/sec   0.063 ms    8/25601 (0.031%)
[  3]  0.0- 2.0 sec  35.9 MBytes   151 Mbits/sec   0.010 ms    0/25641 (0%)
----------bandwidth 200
[  4]  0.0- 2.0 sec  48.2 MBytes   202 Mbits/sec   0.066 ms    0/34413 (0%)
[  3]  0.0- 2.0 sec  48.3 MBytes   203 Mbits/sec   0.028 ms    0/34483 (0%)
----------bandwidth 250
[  4]  0.0- 2.0 sec  59.2 MBytes   248 Mbits/sec   0.056 ms   52/42246 (0.12%)
[  3]  0.0- 2.0 sec  59.7 MBytes   250 Mbits/sec   0.028 ms    0/42553 (0%)
----------bandwidth 300
[  4]  0.0- 2.0 sec  71.7 MBytes   301 Mbits/sec   0.030 ms   55/51222 (0.11%)
[  3]  0.0- 2.0 sec  71.8 MBytes   302 Mbits/sec   0.024 ms   33/51282 (0.064%)
----------bandwidth 350
[  4]  0.0- 2.0 sec  83.8 MBytes   352 Mbits/sec   0.040 ms   87/59888 (0.15%)
[  3]  0.0- 2.0 sec  83.7 MBytes   355 Mbits/sec   0.018 ms  868/60606 (1.4%)
----------bandwidth 400
[  4]  0.0- 2.0 sec  95.6 MBytes   401 Mbits/sec   0.043 ms    5/68180 (0.0073%)
[  3]  0.0- 2.0 sec  90.2 MBytes   379 Mbits/sec   0.012 ms 4601/68965 (6.7%)
----------bandwidth 450
[  4]  0.0- 2.0 sec   105 MBytes   440 Mbits/sec   0.036 ms  369/75113 (0.49%)
[  3]  0.0- 2.0 sec  98.9 MBytes   415 Mbits/sec   0.013 ms 6388/76922 (8.3%)
----------bandwidth 500
[  4]  0.0- 2.0 sec   110 MBytes   460 Mbits/sec   0.031 ms   36/78302 (0.046%)
[  3]  0.0- 2.0 sec  99.5 MBytes   420 Mbits/sec   0.010 ms 15956/86956 (18%)
----------bandwidth 550
[  4]  0.0- 2.0 sec   110 MBytes   459 Mbits/sec   0.031 ms   22/78186 (0.028%)
[  3]  0.0- 2.0 sec   105 MBytes   440 Mbits/sec   0.008 ms 20359/95236 (21%)
----------bandwidth 600
[  4]  0.0- 2.0 sec   109 MBytes   456 Mbits/sec   0.034 ms    0/77709 (0%)
[  3]  0.0- 2.0 sec  90.7 MBytes   381 Mbits/sec   0.009 ms 40526/105254 (39%)
----------bandwidth 650
[  4]  0.0- 2.0 sec   109 MBytes   458 Mbits/sec   0.035 ms    0/77991 (0%)
[  3]  0.0- 2.2 sec  91.2 MBytes   340 Mbits/sec  15.658 ms 46033/111110 (41%)
----------bandwidth 700
[  4]  0.0- 2.0 sec   109 MBytes   458 Mbits/sec   0.034 ms  111/78120 (0.14%)
[  3]  0.0- 1.9 sec  82.6 MBytes   358 Mbits/sec   0.009 ms 66049/124997 (53%)
----------bandwidth 750
[  4]  0.0- 2.0 sec   110 MBytes   463 Mbits/sec   0.031 ms   62/78837 (0.079%)
[  3]  0.0- 2.2 sec  82.0 MBytes   311 Mbits/sec  15.645 ms 74847/133328 (56%)
----------bandwidth 800
[  4]  0.0- 2.0 sec   109 MBytes   458 Mbits/sec   0.029 ms   11/78013 (0.014%)
[  3]  0.0- 2.0 sec  75.1 MBytes   315 Mbits/sec   0.006 ms 88480/142033 (62%)
----------bandwidth 850
[  4]  0.0- 2.0 sec   109 MBytes   456 Mbits/sec   0.056 ms   10/77684 (0.013%)
[  3]  0.0- 2.2 sec  70.2 MBytes   262 Mbits/sec  15.214 ms 99717/149777 (67%)
----------bandwidth 900
[  4]  0.0- 2.0 sec   109 MBytes   457 Mbits/sec   0.032 ms   85/77943 (0.11%)
[  3]  0.0- 2.0 sec  69.4 MBytes   290 Mbits/sec   0.009 ms 100431/149932 (67%)
----------bandwidth 950
[  4]  0.0- 2.0 sec   108 MBytes   451 Mbits/sec   0.075 ms    0/76778 (0%)
[  3]  0.0- 2.2 sec  71.4 MBytes   266 Mbits/sec  15.250 ms 91053/142012 (64%)
----------bandwidth 1000
[  4]  0.0- 2.0 sec   108 MBytes   453 Mbits/sec   0.076 ms    0/77143 (0%)
[  3]  0.0- 1.9 sec  71.2 MBytes   311 Mbits/sec   0.029 ms 90616/141376 (64%)
```

From this, it's pretty clear that the transmit throughput rises pretty linearly to ~450 Mbits/s and stays there. The receiver bandwidth scales linearly to ~400 Mbits/s and then starts losing ground as the rate increases. Also note that we don't lose packets on the transmit side, only on the receiver side.

Cratering performance

Using UDP also exposed some issues on the `boundary` kernel. Using the `boundary-before` kernel, we see that the receive performance degrades as the bandwidth is increased past 400M. Note that this test has an updated `bwtest.sh` script that allows the test time to be set through the `tsecs` environment variable and the bandwidth increment to be set through `incr`.

```
root@linaro-nano:~# tsecs=2 incr=200 ./bwtest.sh
----------bandwidth 200
[  4]  0.0- 2.0 sec  48.1 MBytes   203 Mbits/sec   0.061 ms  164/34479 (0.48%)
[  3]  0.0- 2.0 sec  48.3 MBytes   203 Mbits/sec   0.034 ms    0/34483 (0%)
----------bandwidth 400
[  4]  0.0- 2.0 sec  96.5 MBytes   405 Mbits/sec   0.040 ms   67/68911 (0.097%)
[  3]  0.0- 1.9 sec  93.9 MBytes   406 Mbits/sec   0.035 ms 1990/68965 (2.9%)
----------bandwidth 600
[  4]  0.0- 2.0 sec   110 MBytes   460 Mbits/sec   0.030 ms  234/78615 (0.3%)
[  3]  0.0- 2.3 sec   110 MBytes   410 Mbits/sec  15.672 ms 26703/105262 (25%)
----------bandwidth 800
[  4]  0.0- 2.0 sec   110 MBytes   461 Mbits/sec   0.033 ms    0/78511 (0%)
[  3]  0.0- 2.2 sec  2.91 MBytes  11.1 Mbits/sec  101.865 ms 140266/142342 (99%)
----------bandwidth 1000
[  4]  0.0- 2.0 sec   110 MBytes   461 Mbits/sec   0.033 ms    0/78383 (0%)
[  3]  0.0- 0.2 sec  90.4 KBytes  3.18 Mbits/sec  110.420 ms 141295/141358 (1e+02%)
```

This one took a while to find because it turned out to not be a code change between Blue Meany and our source tree, but a configuration change to enable a new driver API (NAPI). This API is is described on this web page (http://www.linuxfoundation.org/collaborate/workgroups/networking/napi). It is an architecture to decrease the interrupt overhead on high-performance networks. When enabled, the interrupt handler in the FEC driver (https://github.com/boundarydevices/linux-imx6/blob/boundary-L3.0.35_12.09.01_GA/drivers/net/fec.c#L805) schedules but does not process incoming packets. Instead, those are handled out of interrupt context. The change to our config file is trivial (https://github.com/boundarydevices/linux-imx6/commit/38d622938f1352a6550a5e38c624b46b6929439f), but performance is much better at higher speeds as shown below:

```
----------bandwidth 200
[  5]  0.0- 2.0 sec  48.1 MBytes   203 Mbits/sec   0.063 ms  153/34482 (0.44%)
[  3]  0.0- 2.0 sec  48.3 MBytes   203 Mbits/sec   0.029 ms    0/34483 (0%)
----------bandwidth 400
[  4]  0.0- 2.0 sec  96.4 MBytes   404 Mbits/sec   0.052 ms  151/68888 (0.22%)
[  3]  0.0- 1.9 sec  85.5 MBytes   381 Mbits/sec   0.018 ms 7949/68965 (12%)
----------bandwidth 600
[  4]  0.0- 2.0 sec   109 MBytes   458 Mbits/sec   0.075 ms  269/78262 (0.34%)
[  3]  0.0- 1.9 sec   102 MBytes   447 Mbits/sec   0.007 ms 32747/105262 (31%)
----------bandwidth 800
[  4]  0.0- 2.0 sec   110 MBytes   461 Mbits/sec   0.090 ms    0/78464 (0%)
[  3]  0.0- 2.0 sec  82.2 MBytes   347 Mbits/sec   0.006 ms 84223/142847 (59%)
----------bandwidth 1000
[  4]  0.0- 2.0 sec   110 MBytes   461 Mbits/sec   0.072 ms  123/78698 (0.16%)
[  3]  0.0- 2.1 sec  70.6 MBytes   278 Mbits/sec  15.230 ms 91863/142251 (65%)
```

Note that there's still a lot of loss at rates of 400M and above.

How to improve this

Where is that loss coming from? If we look at `ifconfig` we can see that the network driver is aware of the dropped packets:

```
root@linaro-nano:~# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:19:b8:00:fa:9a
          inet addr:192.168.0.119  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3901387 errors:782502 dropped:0 overruns:782502 frame:782502
          TX packets:3775053 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4178248807 (4.1 GB)  TX bytes:853327178 (853.3 MB)
```

The condition that increments the `overrun` count is here (https://github.com/boundarydevices/linux-imx6/blob/0ece666e049cc26e2032737c27ea4c58cff2a0d6/drivers/net/fec.c#L570). Table 23-85 in the i.MX6DQ Reference Manual says that this means "A receive FIFO overrun occurred during frame reception". The i.MX6DQ Errata (http://cache.freescale.com/files/32bit/doc/errata/IMX6DQCE.pdf?fpsp=1&WT_TYPE=Errata&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=pdf&WT_ASSET=Documentation) document states in Errata `ERR004512` that the maximum performance "is limited to 470 Mbps (total for Tx and Rx)". The errata doesn't say what the precise symptom of exceeding that limit might be, but this sure looks like it. The numbers above are pretty close to the 400Mbit/s reported in the errata. It turns out that we can do something about this. The Ethernet spec calls for a form of flow control using something called "pause frames", which allows a receiver to tell a sender to back off for a quantum of time. That's what this patch (https://github.com/boundarydevices/linux-imx6/commit/23f8463da111a5d0f151cb233d5259d280e44a53) does. The very first part of the commit shows the addition of `SUPPORTED_Pause` to the `phy` device for i.MX6Quad and i.MX6DualLite processors. That part is key.

Sidebar: check out some other tools

Before we go too much further, we need to introduce a couple of key tools to understanding this. The first is a tool called `ethtool`. It is designed to allow you control the low-level functions of a network adapter. We'll use it to see the state of the link negotiation. The second is a tool we developed named `devregs`. It is designed to allow access to device registers through `/dev/mem`. You can find details in this post (http://boundarydevices.com/i-mx5x-device-register-access/). The post describes the use of the program on i.MX5x, but it's perfectly happy to run on i.MX6 and we have a lot of registers defined in devregs_imx6x.dat (https://github.com/boundarydevices/imx-utils/blob/master/devregs_imx6x.dat). Let's look at the output before and after the patch:

Before

```
root@linaro-nano:~# cat /proc/version
Linux version 3.0.35-2026-geaaf30e (b21710@bluemeany)
root@linaro-nano:~# ethtool eth0
Settings for eth0:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Supported pause frame use: No
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Speed: 1000Mb/s
        Duplex: Full
        Port: MII
        PHYAD: 6
        Transceiver: external
        Auto-negotiation: on
        Link detected: yes
root@linaro-nano:~# devregs ENET_RCR
ENET_RCR:0x02188084     =0x05ee0244
```

After

```
root@linaro-nano:~# cat /proc/version
root@linaro-nano:~# cat /proc/version
Linux version 3.0.35-2026-geaaf30e-02074-g92a9e1e ...
root@linaro-nano:~# ethtool eth0
Settings for eth0:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Supported pause frame use: Symmetric
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Advertised pause frame use: Symmetric
        Advertised auto-negotiation: Yes
        Speed: 1000Mb/s
        Duplex: Full
        Port: MII
        PHYAD: 6
        Transceiver: external
        Auto-negotiation: on
        Link detected: yes
root@linaro-nano:~# devregs ENET_RCR
ENET_RCR:0x02188084     =0x05ee0264
```

The key things to look at are the line that says “Supported pause frame use” and the line that shows the ENET_RCR register. Bit 5 of the ENET_RCR enables flow control (generation of pause frames) if set, and you can see that after the patch, flow control is enabled. Unfortunately, the situation is still much the same: The receive error numbers using bwtest.sh start rising and the bandwidth starts falling as we exceed 450 MBits/s.

Pause frames on TO1.0

The next patch in the series (https://github.com/boundarydevices/linux-imx6/commit/5b153d52c09d8cf35014d410bb7aec7462a6209c) fixes this. It turns out that tweaking the default *almost empty* threshold on Tapeout 1.0 helps the situation, as does increasing the *receive FIFO section full* register. After applying this patch, we can see stable results when we overload the ethernet receiver. Note that we’ve also added a couple of lines to bwtest.sh to read the values of the ENET_IEEE_T_FDXFC and ENET_IEEE_R_MACERR statistics registers. These tell us how many pause frames were transmitted and how many receive FIFO overruns are seen.

```
root@linaro-nano:~# tsecs=2 incr=200 ./bwtest.sh
----------bandwidth 200
[  4]  0.0- 1.7 sec  40.1 MBytes   203 Mbits/sec  50.557 ms  176/28804 (0.61%)
[  3]  0.0- 2.0 sec  48.3 MBytes   203 Mbits/sec   0.034 ms    0/34483 (0%)
ENET_IEEE_T_FDXFC:0x02188270    =0x00000d6c
ENET_IEEE_R_MACERR:0x021882d8   =0x00000000
----------bandwidth 400
[  4]  0.0- 2.0 sec  96.5 MBytes   405 Mbits/sec   0.043 ms  103/68952 (0.15%)
[  3]  0.0- 1.9 sec  90.0 MBytes   406 Mbits/sec   0.021 ms 4751/68965 (6.9%)
ENET_IEEE_T_FDXFC:0x02188270    =0x00001ad0
ENET_IEEE_R_MACERR:0x021882d8   =0x00000000
----------bandwidth 600
[  4]  0.0- 2.0 sec   110 MBytes   462 Mbits/sec   0.056 ms    0/78679 (0%)
[  3]  0.0- 1.9 sec   129 MBytes   583 Mbits/sec   0.061 ms 4750/96927 (4.9%)
ENET_IEEE_T_FDXFC:0x02188270    =0x0000f544
ENET_IEEE_R_MACERR:0x021882d8   =0x00000000
----------bandwidth 800
[  4]  0.0- 2.0 sec   110 MBytes   461 Mbits/sec   0.030 ms   92/78732 (0.12%)
[  3]  0.0- 2.0 sec   138 MBytes   580 Mbits/sec   0.062 ms   20/98693 (0.02%)
ENET_IEEE_T_FDXFC:0x02188270    =0x00000310
ENET_IEEE_R_MACERR:0x021882d8   =0x00000000
----------bandwidth 1000
[  4]  0.0- 2.0 sec   107 MBytes   449 Mbits/sec   0.060 ms  465/76969 (0.6%)
[  3]  0.0- 1.9 sec   129 MBytes   583 Mbits/sec   0.021 ms 4687/96830 (4.8%)
ENET_IEEE_T_FDXFC:0x02188270    =0x0000f482
ENET_IEEE_R_MACERR:0x021882d8   =0x00000000
```

Now that’s better! We’re seeing no FIFO overruns even up to 1G and a substantial increase in receive performance. Tapeout 1.2 shows even better performance, peaking at over 630 Mbit/s.

Final changes

The final two patches are really *belt and suspenders* updates. The first sets the *Frame truncation receive length* register (https://github.com/boundarydevices/linux-imx6/commit/f8626240dda0f1c0b517835356a2121a71600b97) so a FIFO error will not result in an extra long frame and spew error messages to the kernel log. The second treats frames with FIFO errors (https://github.com/boundarydevices/linux-imx6/commit/23e50de473fec3ba33505768b020d9966f5610f6) in the same way as framing errors and doesn't forward them to the network stack for processing. We found that this increased performance in the presence of FIFO overruns.

Recap

We've uploaded the SD card image used in this testing so that you can repeat our results:

- imx6-iperf-test-20121214.tar.gz (http://commondatastorage.googleapis.com/boundarydevices.com/imx6-iperf-test-20121214.tar.gz)

If you format a single-partition SD card as `ext3`, you can extract it like so:

```
~/$ sudo mkfs.ext3 -L iperf /dev/sdc1
~/$ udisks --mount /dev/sdc1
... Assuming auto-mount as /media/iperf
~/$ sudo tar -C /media/iperf/ -zxvf imx6-iperf-test-20121214.tar.gz
~/$ sync && sudo umount /media/iperf
```

As mentioned earlier, this started off as a Linaro nano filesystem. We updated it to include a boot script, the `devregs` program and each of the kernels used in the tests above. The SD card image has each in the `/boot` directory. We encourage you to download the image, test it out on your boards and report back. Note that we haven't yet updated the Android kernel tree, but will do that shortly. We'll also be testing i.MX6 Solo, Dual-Lite, and the new SABRE SDB boards in upcoming days. Stay tuned to the blog for updates. If you're using Gigabit ethernet, you're likely to see improvements by adopting these updates.

Share this:

(https://boundarydevices.com/i-mx6-ethernet/?share=twitter&nb=1)

(https://boundarydevices.com/i-mx6-ethernet/?share=facebook&nb=1)

(https://boundarydevices.com/i-mx6-ethernet/?share=google-plus-1&nb=1)

i.MX6 (https://boundarydevices.com/tag/i-mx6/) linux (https://boundarydevices.com/tag/linux/) Freescale (https://boundarydevices.com/tag/freescale/) U-Boot (https://boundarydevices.com/tag/u-boot/) Android (https://boundarydevices.com/tag/android/) i.MX51 (https://boundarydevices.com/tag/imx51/) Nitrogen6X (https://boundarydevices.com/tag/nitrogen6x/) Ubuntu (https://boundarydevices.com/tag/ubuntu/) GPU (https://boundarydevices.com/tag/gpu/) imx6 (https://boundarydevices.com/tag/imx6/) x-windows (https://boundarydevices.com/tag/x-windows/) LTIB (https://boundarydevices.com/tag/ltib/) VPU (https://boundarydevices.com/tag/vpu/) gstreamer (https://boundarydevices.com/tag/gstreamer/) Yocto (https://boundarydevices.com/tag/yocto/) browser (https://boundarydevices.com/tag/browser/) camera (https://boundarydevices.com/tag/camera/) i.MX6 Single Board Computer (https://boundarydevices.com/tag/i-mx6-single-board-computer/) BD-SL-i.MX6 (https://boundarydevices.com/tag/bd-sl-i-mx6/) Debian (https://boundarydevices.com/tag/debian/) i.MX6 SOM (https://boundarydevices.com/tag/i-mx6-som/) i.MX7 (https://boundarydevices.com/tag/i-mx7/) WLAN (https://boundarydevices.com/tag/wlan/) Nitrogen6_MAX (https://boundarydevices.com/tag/nitrogen6_max/)

sabrelite (https://boundarydevices.com/tag/sabrelite/) Bluetooth (https://boundarydevices.com/tag/bluetooth/)
Qt (https://boundarydevices.com/tag/qt/)
i.MX6 System on Module (https://boundarydevices.com/tag/i-mx6-system-on-module/)
Freescale i.MX6 (https://boundarydevices.com/tag/freescale-i-mx6/)
timesys (https://boundarydevices.com/tag/timesys/) wifi (https://boundarydevices.com/tag/wifi/)
touch screen (https://boundarydevices.com/tag/touch-screen/)
nitrogen7 (https://boundarydevices.com/tag/nitrogen7/) ARM SBC (https://boundarydevices.com/tag/arm-sbc/)
i.MX6 SBC (https://boundarydevices.com/tag/i-mx6-sbc/) i.MX53 (https://boundarydevices.com/tag/i-mx53/)
Buildroot (https://boundarydevices.com/tag/buildroot/) webkit (https://boundarydevices.com/tag/webkit/)
GPIO (https://boundarydevices.com/tag/gpio/) Linaro (https://boundarydevices.com/tag/linaro/)

Comments 32

POST
AUTHOR

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

ericn (http://boundarydevices.com)

DECEMBER 21, 2012 AT 6:37 PM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1017)

The updates are also now up in the Android kernel git repository (https://github.com/boundarydevices/linux-imx6/commits/boundary-imx-android-r13.4-ga).

POST
AUTHOR

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

ericn (http://boundarydevices.com)

JANUARY 20, 2013 AT 9:30 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1018)

We’ve recently tested a PCIe Gigabit Ethernet controller:

Marvell Technology Group Ltd. 88E8057 PCI-E Gigabit Ethernet Controller (rev 10)

The performance was similar, with a peak 597 Mbits/s transmit and 507 Mbits/s receive throughput.

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

CS

MARCH 1, 2013 AT 8:56 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1019)

Hi Eric,

I use both the uImage-latest and also compiled the kernel from the boundary-imx-3.0.35_1.1.0 branch.
Using iperf i seems to be only getting ~400M bandwidth.

Anything thing else you would recommend?
Are there plans to get it to potentially full 1G range?

thanks

POST
AUTHOR

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

ericn (http://boundarydevices.com)

MARCH 1, 2013 AT 12:01 PM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1020)

Is this a Tapeout 1.0 part? You can see this at boot time right after the U-Boot banner:

```
U-Boot 2013.01-00058-g5957f6d (Feb 11 2013 - 08:42:43)
CPU:   Freescale i.MX6Q rev1.2 at 792 MHz
```

If so, then you might peak at around 400MBits/s. The Freescale specs have been updated to reflect this number, and there isn't anything we can do to address it.

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

CS

MARCH 1, 2013 AT 12:26 PM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1021)

U-Boot 2013.01-rc1-00130-g1e88922 (Feb 11 2013 – 13:46:35)
CPU: Freescale i.MX6Q rev1.2 at 792 MHz

I believe this means its TO 1.2 ?

if so what other tweaks do u recommend?

I did check to make sure the enable_wait_mode=off is set.

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

CS

MARCH 1, 2013 AT 1:13 PM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1022)

Some how my first reply didnt get thru.

anyways my TO seems to be 1.2

U-Boot 2013.01-rc1-00130-g1e88922 (Feb 11 2013 – 13:46:35)
CPU: Freescale i.MX6Q rev1.2 at 792 MHz

i checked to make sure my enable_wait_mode = off

iperf UDP i manage to get up to like 500M

but TCP is only 400M

any other recommendations you could think of?

thanks

ericn (http://boundarydevices.com)

POST AUTHOR

MARCH 1, 2013 AT 1:28 PM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1023)

TCP performance of ~400M is what you should expect. See here for details.

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

Paul

OCTOBER 15, 2013 AT 12:12 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1027)

Hi Eric,

Could you tell me how you evaluated the Marvell Technology Group Ltd. 88E8057 PCI-E Gigabit Ethernet Controller? Is there a PCI-E card available with this chip on it? If so where can I get one from?

POST
AUTHOR

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

ericn (http://boundarydevices.com)

OCTOBER 15, 2013 AT 6:35 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1028)

Hi Paul,

We received an eval board from our local Marvell sales rep, but this is a very common part, so I would expect adapters to be commercially available.

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

ARMHF

JULY 11, 2013 AT 3:43 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1024)

Did you see this:
https://community.freescale.com/thread/309344 (https://community.freescale.com/thread/309344)
https://community.freescale.com/message/336650 (https://community.freescale.com/message/336650)

?
I can also reproduce this (PC to Sabre Board).
A bit lower performance is not nice but acceptable – but hundreds of overruns (at just 200Mbit/s!) in a few seconds is ugly – and may even cause serious issues for some applications.

POST
AUTHOR

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

ericn (http://boundarydevices.com)

SEPTEMBER 17, 2013 AT 5:24 PM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1025)

Hi all,

While trying to get rid of some pesky <1% packet losses that appear even at low data rates for a customer, we bumped right into this fix (https://github.com/boundarydevices/u-boot-imx6/commit/fdf86c202c17adfc6f6313dc35f685b1d22b8125).

The lesson here is, upgrade to the latest U-Boot (http://commondatastorage.googleapis.com/boundarydevices.com/u-boot-production.tar.gz).

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

mikeu

OCTOBER 3, 2013 AT 4:47 PM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1026)

I have been doing Ethernet testing with a Nitrogen6X using a 10Base-T half duplex hub, which are used frequently for packet sniffing with tools like Wireshark, and it appears that collisions are not being detected and retried. From what I have been able to determine the KSZ9021RN PHY is not sending a collision indication to the imx6 FEC. Over RGMII, a collision is detected by the FEC by when the PHY asserts RXDV while TXEN is asserted. When the FEC detects a collision it should automatically resend the packet after a random back-off delay. The Sabre AI board, which uses an AR8031 PHY seems to work.

Has anyone else seen this problem?

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

WANGLEE

MARCH 29, 2016 AT 8:42 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-372778)

how to get the latest- uboot for packet loss?

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

Gary Bisson (http://boundarydevices.com)

MARCH 29, 2016 AT 9:22 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-372783)

Hi,

What do you mean by “uboot for packet loss”?

Anyway, here is the latest uboot blog post which contains an archive:
https://boundarydevices.com/compiling-latest-u-boot-for-i-mx6-2015-edition/ (https://boundarydevices.com/compiling-latest-u-boot-for-i-mx6-2015-edition/)

Regards,
Gary

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

IMX64LINUX

OCTOBER 27, 2013 AT 4:11 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1029)

Dear Ericn,

Looking in detail at the different tests in this excellent report, we can see some strange results (assuming the same test bed, except for the script/no script usage):

In section: measuring performance,
* using the command: iperf -c 192.168.0.162 -u -r -b 400M -t 2; we can see a 58% loss
* while just below using the script, for the 400Mb/s line, we can see 6.7% loss

Do you have some ideas on the source of the different results between the 2 test cases? (different boards or?)

In section Pause frames on TO1.0, we can see again some strange behavior:
* using the script, for the 400Mb/s line, we can see 6.9% loss
* using the script, for above the 600Mb/s and next lines, we can see the drop in the frame loss (compared with the 400Mb/s, while we would expect an increase in the loss)

Do you have any ideas on what could case this frame loss rate decrease on your experiments?

(I am concerned on the ERR005783 that says implicitly we cannot avoid frame loss due to small frames, even using the ethernet flow control especially when for example we have on the Ethernet an SSH connection and an Iperf traffic directed to the IMX6)

Many thanks in advance for your reply.

POST
AUTHOR

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

ericn (http://boundarydevices.com)

OCTOBER 27, 2013 AT 7:52 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1030)

I wouldn’t put too much stock in any one line of output. Most of the tests above are short (two second) and though they yield pretty reproducible results, they’re not comprehensive.

The test results are all from the same board under test, same switch, and same PC, but there were other devices connected to the switch, and there may have been other traffic on the network.

Also note that this post pre-dates our knowledge of ERR005783 and doesn’t address it.

Finally, note that some of the small packet loss numbers at low data rates is addressed by a U-Boot patch as mentioned in the comment above (http://boundarydevices.com/i-mx6-ethernet/#comment-59282).

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

IMX64LINUX

NOVEMBER 1, 2013 AT 4:58 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1031)

Dear Ericn,

Thanks for clarifications especially in the environment set up.
In our side we have applied all the patches, including the DDR interface set up in uboot you mention (due to the documentation error from freescale).

We have done several trials and found issues with the ethernet while using the GPU:

In an environment with a freescale SabreSD, (IMX6Q 1GHz, TO 1.2) connected directly to the gigabit ethernet of a PC directly (running unbutoo for example) and an SSH for a remote control of the sabre,

performed tests:
———————
* test1: udp iperf running (in addition to the SSH connection)
* test2 = test1+ running a GPU application (e.g. /opt/viv_samples/vdk/tutorial7))

Results:

* test1: results as in "For the impatient"
* test2: datagrams loss

Test 6: On SabreSD, with GPU example

```
xxxxx:~# iperf -u -c 192.168.1.1 -b1000M -t20
————————————————————
Client connecting to 192.168.1.1, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 104 KByte (default)
————————————————————
[ 3] local 192.168.1.2 port 34216 connected with 192.168.1.1 port 5001
[ ID] Interval Transfer Bandwidth
[ 3] 0.0-20.0 sec 1.04 GBytes 445 Mbits/sec
[ 3] Sent 756257 datagrams
[ 3] Server Report:
[ 3] 0.0-20.0 sec 1006 MBytes 422 Mbits/sec 0.068 ms 38526/756256 (5.1%)
[ 3] 0.0-20.0 sec 1 datagrams received out-of-order
```

(the ifconfig eht0 does not report overruns during this test)

The % loss is random (here around 230fps loss) , we may have significant variations (in terms of packet loss) each time we run iperf.

Has someone seen this problem and more important an idea on a potential work around?

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

Rick

NOVEMBER 5, 2013 AT 8:53 PM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1032)

Hi,
The SDK 4.1.0 from freescale seems improve the network performance, I've tried to patch the frame pause code piece into 4.1.0 source.

The patches applied:
https://github.com/boundarydevices/linux-imx6/commit/0629c43b668328caa53e509c1118e45af4fb8ec7 (https://github.com/boundarydevices/linux-imx6/commit/0629c43b668328caa53e509c1118e45af4fb8ec7)

https://github.com/boundarydevices/linux-imx6/commit/49fe4550243cd04f4f734dcc56bd262628cdc9dc (https://github.com/boundarydevices/linux-imx6/commit/49fe4550243cd04f4f734dcc56bd262628cdc9dc)

https://github.com/boundarydevices/linux-imx6/commit/05e94b34e3921c2b5d2ca848c4624827378f034f (https://github.com/boundarydevices/linux-imx6/commit/05e94b34e3921c2b5d2ca848c4624827378f034f)

Do you have benchmark of SDK 4.1.0 and got the same result?

Thanks

POST
AUTHOR

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

ericn (http://boundarydevices.com)

NOVEMBER 5, 2013 AT 8:57 PM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1033)

Thanks Rick,

We are using the 4.1.0 kernel as the best *production ready* kernel, but had to apply some of the same patches to get the same results. You can see the full change log here (https://github.com/boundarydevices/linux-imx6/commits/boundary-imx_3.0.35_4.1.0/drivers/net/fec.c).

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

Rick

NOVEMBER 6, 2013 AT 4:56 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1034)

Hi Ericn,

I noticed you have patch about TO1.0, does this means you use TO1.0 chip?
I am trying TO1.2 chip, maybe I need to check the changes between these version.

I will try all the patches you mentioned, thank you very much.

POST AUTHOR

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

ericn (http://boundarydevices.com)

NOVEMBER 6, 2013 AT 6:34 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1035)

Hi Rick,

We have lots of early silicon out in the field, and we're trying to support those boards with each kernel release, though that is becoming harder as time goes by.

All of our patches should be usable on either version.

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

Pev

FEBRUARY 3, 2014 AT 3:41 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-1036)

Hi,

Have you guys ever tested PACKET_MMAP on the sabre boards? (Test code available from http://wiki.ipxwarzone.com/index.php5?title=Linux_packet_mmap (http://wiki.ipxwarzone.com/index.php5?title=Linux_packet_mmap)) – this normally works excellently as a way to reduce CPU load for high throughput ethernet but on the board Ive tested on it seems to drastically reduce performance down to 200Mbit or so…!

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

CODYTUDOR

NOVEMBER 20, 2015 AT 10:35 PM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-370550)

After racking my brain against a wall trying to figure out why the i.MX6 Ethernet controller on the Nitrogen6X SOM was unable to use the PHY chip we specified I finally figured it out today and thought to post a blog about it since there are numerous people in the same situation as me.

The Nitrogen6X SOM only breaks out the necessary i.MX6 BALLS to utilize a Reduced Gigabit Media Independent Interface (RGMII).

In order to use a Reduced Media Independent Interface (RMII) BD would need to break out i.MX6 BALLS U21, W22, U20, & W20 (ENET_CRS_DV, ENET_RXD1, ENET_TXD0, & ENET_TXD1).

In order to use a Media Independent Interface (MII) BD would need to break out the above BALLS for RMII & i.MX6 BALLS W5 & V6 (ENET_RX_DATA3 & ENET_TX_DATA3). These latter two are in a different voltage domain so either a fast (>25Mbps) voltage translator would need to be used to achieve the +2.5V of the ENET domain OR change the ENET domain to be the same as the NVCC_GPIO domain. Furthermore, many of the KEY_COLx & KEY_ROWx PADs device tree bindings would have to be changed from their current configurations to support the MII interface.

In short, if using the Nitrogen6X SOM in a custom board design and you intend to add an Ethernet PHY controller you MUST USE an IC that uses the RGMII interface for your MAC-PHY OR MAC-MAC connection.

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

Gary Bisson (http://boundarydevices.com)

NOVEMBER 23, 2015 AT 9:42 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-370562)

Thanks for sharing your experience!

Regards,
Gary

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

Jayakumar.n

NOVEMBER 28, 2015 AT 5:32 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-370612)

Hi,

I have a iMx6 Evaluation Kit with WinCE7 OS, Does the fix or workaround is available for WinCE7 OS ?

Setup:
Evaluation Kit : iMx6 Quad – Boundary devices Saber Lite rev-2
OS : WinCE7
Tools used : NetIo Network throughput Analyzer.
Configuration :
– Phy speed is set to 1Gbps
– Full duplex
– RGMII Interface
– NDIS 6.0

Please let me know any fix or workaround has been identified for the issue specified for WinCE7.

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

Jayakumar.n

NOVEMBER 28, 2015 AT 6:19 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-370613)

From the analysis what we could see is, when Imx6 device is receiving data packets, Flow Control Pause Frames Transmitted Statistic Register (ENET_IEEE_T_FDXFC) is not incremented when overrun occurred, which means that Pause frames are not send from from Device MAC to sender to avoid Overrun.

We had checked the Phy configuration & Mac Configuration to check any Pause frames related settings are missing, couldn't find any Configuration problems.

Please let me know any Phy Configuration or Mac Configurations needed to be set to instruct transmitting Path send Pause Frames when thresholds are reached. Note : Receive FIFO Almost Full Threshold is set to 4 by default.

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

Gary Bisson (http://boundarydevices.com)

DECEMBER 8, 2015 AT 9:47 PM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-370702)

Hi,

I've replied to your question on Freescale's forum (now NXP):
https://community.freescale.com/message/595246#595246
(https://community.freescale.com/message/595246#595246)

Basically testing done on SabreLite proved that the PAUSE frames are properly sent and ENET_IEE_T_FDXFC is properly incremented. Please make sure to specify the image you are using with the Linux kernel version associated to it.

Regards,
Gary

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

aleo

NOVEMBER 30, 2015 AT 8:35 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-370620)

Hello,
We have bought several imx6 sabrelite rev-2 boards (6-20-13) which we planned to use in our project replacing imx53 boards. But we have troubles (no link) to connect this boards to our ethernet devices. There also troubles to connect board to commonly used equipment . Some ethernet switches can be connected to sabrelite fine in almost any cases with any cable, some switches work only with certain cables, some devices do not bring link up with any cable. Some gigabit switches can be connected succesfully to sabrelite only with disabled gigabit feature or autonegotiation on imx board but this configuration does not help to connect to other devices. And so on – no system was found yet. There was no troubles with ethernet connections on imx53 board…
Has anyone else seen such behavior?

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

Ismail

DECEMBER 3, 2015 AT 7:21 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-370651)

We are getting following errors for TCP Test on SABRE Lite rev-1.2; 6-20-13 with uImage-latest:
Client Side – Waiting for servers threads to complete. Interrupt again to force quit.
Server Side – connect failed: Connection refused

Board used: Boundary Devices; SABRE Lite rev-1.2; 6-20-13

uImage used: imx6-iperf-test-20121214\boot\uImage-latest (downloaded from a link on this page: A tar-ball “is available here”).

TCP Test:
TCP Server – root@linaro-nano:~# iperf -s
TCP Client – root@pc:~# iperf -c 192.168.38.100 -r

Log @Client Side

```
——————————————————————
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
——————————————————————
——————————————————————
Client connecting to 192.168.38.100, TCP port 5001
TCP window size: 16.0 KByte (default)
——————————————————————
```

```
[ 5] local 192.168.38.182 port 35290 connected with 192.168.38.100 port 5001
[ ID] Interval Transfer Bandwidth
[ 5] 0.0-14.2 sec 55.4 MBytes 32.7 Mbits/sec
Waiting for servers threads to complete. Interrupt again to force quit.

root@pc:~# ifconfig eth2
eth2 Link encap:Ethernet HWaddr 00:1d:09:1a:32:02
inet addr:192.168.38.182 Bcast:192.168.39.255 Mask:255.255.248.0
inet6 addr: fe80::21d:9ff:fe1a:3202/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2875076 errors:2 dropped:0 overruns:0 frame:2
TX packets:3448518 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:34181879 (34.1 MB) TX bytes:860945882 (860.9 MB)
Interrupt:16
```

Log @Server Side

```
——————————————————————
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
——————————————————————
[ 4] local 192.168.38.100 port 5001 connected with 192.168.38.182 port 35290
[ ID] Interval Transfer Bandwidth
[ 4] 0.0-14.2 sec 55.4 MBytes 32.6 Mbits/sec
connect failed: Connection refused
```

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

Ismail

DECEMBER 3, 2015 AT 7:40 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-370652)

We are getting following errors for UDP Test on SABRE Lite rev-1.2; 6-20-13 with uImage-latest:
Client Side:
– read failed: Connection refused (for 700Mbps test)
– [4] WARNING: did not receive ack of last datagramafter 10 tries. (for 800Mbps and 1000Mbps test)
Server Side – connect failed: Connection refused

Board used: Boundary Devices; SABRE Lite rev-1.2; 6-20-13

uImage used: imx6-iperf-test-20121214\boot\uImage-latest (downloaded from a link on this page: A tar-ball “is available here”).

UDP Test:
UDP Server:
– root@linaro-nano:~# iperf -u -s
UDP Client:
– root@pc:~# iperf -c 192.168.38.100 -u -r -b 100M
– root@pc:~# iperf -c 192.168.38.100 -u -r -b 200M
in increments of 100M till
– root@pc:~# iperf -c 192.168.38.100 -u -r -b 1000M

Log @Client Side

700Mbits/sec:

```
—————————————————————
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 110 KByte (default)
—————————————————————
—————————————————————
Client connecting to 192.168.38.100, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 110 KByte (default)
—————————————————————
[ 4] local 192.168.38.182 port 38402 connected with 192.168.38.100 port 5001
[ ID] Interval Transfer Bandwidth
[ 4] 0.0-10.0 sec 799 MBytes 671 Mbits/sec
[ 4] Sent 570173 datagrams
[ 4] Server Report:
[ 4] 0.0-10.0 sec 413 MBytes 347 Mbits/sec 0.023 ms 275289/570172 (48%)
[ 4] 0.0-10.0 sec 1 datagrams received out-of-order
[ 3] local 192.168.38.182 port 5001 connected with 192.168.38.100 port 44635
[ 3] 0.0-10.0 sec 472 MBytes 396 Mbits/sec 0.295 ms 3/336799 (0.00089%)
[ 3] 0.0-10.0 sec 1 datagrams received out-of-order
```

```
root@pc:~# ifconfig eth2
eth2 Link encap:Ethernet HWaddr 00:1d:09:1a:32:02
inet addr:192.168.38.182 Bcast:192.168.39.255 Mask:255.255.248.0
inet6 addr: fe80::21d:9ff:fe1a:3202/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2098296 errors:0 dropped:0 overruns:0 frame:0
TX packets:2120926 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:3063791103 (3.0 GB) TX bytes:3143342723 (3.1 GB)
Interrupt:16
```

1000Mbits/sec:

```
—————————————————————
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 110 KByte (default)
—————————————————————
—————————————————————
Client connecting to 192.168.38.100, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 110 KByte (default)
—————————————————————
[ 4] local 192.168.38.182 port 39118 connected with 192.168.38.100 port 5001
[ ID] Interval Transfer Bandwidth
[ 4] 0.0-10.0 sec 841 MBytes 706 Mbits/sec
[ 4] Sent 600008 datagrams
[ 3] local 192.168.38.182 port 5001 connected with 192.168.38.100 port 60162
[ 3] 0.0- 6.6 sec 300 MBytes 384 Mbits/sec 0.254 ms 184126/398238 (46%)
[ 3] 0.0- 6.6 sec 1 datagrams received out-of-order
```

```
root@pc:~# ifconfig eth2
eth2 Link encap:Ethernet HWaddr 00:1d:09:1a:32:02
inet addr:192.168.38.182 Bcast:192.168.39.255 Mask:255.255.248.0
inet6 addr: fe80::21d:9ff:fe1a:3202/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2854986 errors:2 dropped:0 overruns:0 frame:2
TX packets:3408310 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:27854543 (27.8 MB) TX bytes:800040860 (800.0 MB)
Interrupt:16
```

Log in to Reply (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F)

Gary Bisson (http://boundarydevices.com)

DECEMBER 8, 2015 AT 10:01 AM (HTTPS://BOUNDARYDEVICES.COM/I-MX6-ETHERNET/#COMMENT-370699)

Hi,

Please try with a current image (Yocto Jethro, Ubuntu Trusty dec.2015 or Buildroot v2015.11):
https://boundarydevices.com/imx6-builds/ (https://boundarydevices.com/imx6-builds/)

Currently on my SabreLite rev-2 (6-20-13), using a TP-Link TL-SG105 gigabit switch, I obtain the following results:

```
# iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.1.50, port 43132
[ 5] local 192.168.1.34 port 5201 connected to 192.168.1.50 port 43134
[ ID] Interval Transfer Bandwidth
[ 5] 0.00-1.00 sec 56.6 MBytes 473 Mbits/sec
[ 5] 1.00-2.00 sec 59.4 MBytes 498 Mbits/sec
[ 5] 2.00-3.00 sec 59.6 MBytes 499 Mbits/sec
[ 5] 3.00-4.00 sec 59.6 MBytes 500 Mbits/sec
[ 5] 4.00-5.00 sec 59.4 MBytes 499 Mbits/sec
[ 5] 5.00-6.02 sec 62.5 MBytes 517 Mbits/sec
[ 5] 6.02-7.00 sec 58.5 MBytes 497 Mbits/sec
[ 5] 7.00-8.00 sec 59.5 MBytes 500 Mbits/sec
[ 5] 8.00-9.00 sec 60.4 MBytes 507 Mbits/sec
[ 5] 9.00-10.00 sec 59.5 MBytes 499 Mbits/sec
[ 5] 10.00-10.01 sec 640 KBytes 454 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval Transfer Bandwidth Retr
[ 5] 0.00-10.01 sec 602 MBytes 504 Mbits/sec 0 sender
[ 5] 0.00-10.01 sec 596 MBytes 499 Mbits/sec receiver
```

Regards,
Gary

Leave a Reply

You must be logged in (https://boundarydevices.com/login/?redirect_to=https%3A%2F%2Fboundarydevices.com%2Fi-mx6-ethernet%2F) to post a comment.

Products (http://boundarydevices.com/products/)
Custom (http://boundarydevices.com/custom/)
Video Reviews (http://boundarydevices.com/video-reviews/)
Where To Buy (http://boundarydevices.com/where-else-to-buy/)
Support (http://boundarydevices.com/support/)
Contact (http://boundarydevices.com/contact/)
Blog (http://boundarydevices.com/category/blog/)
About Us (http://boundarydevices.com/company/)
Careers (http://boundarydevices.com/careers/)
Privacy Policy (http://boundarydevices.com/privacy-policy/)
Return Policy (http://boundarydevices.com/return-policy/)
Terms and Conditions (http://boundarydevices.com/terms-and-conditions/)

Register for Downloads

Go! (http://boundarydevices.com/register/)

Product Selector Guide

Still not sure about exactly what you need? Check out our informative guide!

Go! (http://boundarydevices.com/product-selector-guide)