■ Xillybus のドライバを読む

xillybus_core.c から．まずは，ざっと，どんな感じのものがあるのか見てみる．
これは FPGA 内の IP コアの FIFO とデバイスファイルのパイプを作る．
キャラクタデバイスで，構造体は↓な感じ．

```
static const struct file_operations xillybus_fops = {
        .owner      = THIS_MODULE,
        .read       = xillybus_read,
        .write      = xillybus_write,
        .open       = xillybus_open,
        .flush      = xillybus_flush,
        .release    = xillybus_release,
        .llseek     = xillybus_llseek,
        .poll       = xillybus_poll,
};
```

■初期化関連

xillybus_init/xillybus_exit

ドライバロード時にやってるのは,

- class_create
- alloc_workqueue

アンロード時には,

- destroy_workqueue
- class_destroy

xillybus_init_endpoint

```
EXPORT_SYMBOL(xillybus_init_endpoint);
```

でエクスポートされている関数.
xillybus_pcie.c の xilly_probe(← pci ドライバの .porbe にマッピングされてる) か,
xillybus_of.c の xilly_drv_probe(platform_driver の .probe にマッピングされてる) から呼ばれる．
デバドラ内部で使うデータの初期化

xillybus_endpoint_discovery

```
EXPORT_SYMBOL(xillybus_endpoint_discovery);
```

でエクスポートされている関数.
xillybus_pcie.c の xilly_probe(← pci ドライバの .porbe にマッピングされてる) か,
xillybus_of.c の xilly_drv_probe(platform_driver の .probe にマッピングされてる) から呼ばれる．
ハードウェアから情報をスキャンしてハンドラを作る．ハンドラの定義は，xillybus.h にある

```
struct xilly_idt_handle {
        unsigned char *chandesc;
```

```
        unsigned char *idt;
        int entries;
};
```

- xilly_setupchannels チャネルのセットアップ.
- xilly_obtain_idt ハードウェアのバージョンチェックなど
- xilly_scan_idt

ハンドラが用意できたら，xillybus_init_chrdev を呼ぶ

xillybus_init_chrdev

キャラクタデバイスとしての初期化，デバイスファイルを作るのも，ここ.

■アクセス関連

xillybus_open

モードに応じたチャネルやロックの設定など

- mutex_lock_interruptible
- iowrite32 でデバイスにオープンされたことを通知

xillybus_read

データの読み出し．こんな感じ.

```
while(1)
  bytes_to_do = count - bytes_done; 残りの読むべきバイト数をセット

  if バッファが empty でない
    if バッファにたまっている個数 (howmany) が byte_to_do より多い
      bufferdone = 0;
    else
      bufferdone = 1;
    end
  end

  if empty でない
    if (bufops == 0) // バッファが空
      channel->endpoint->ephw->hw_sync_sgl_for_cpu で DMA する (DMA_FROM_DEVICE)
    end
    copy_to_user でバッファのデータをユーザ領域にデータをコピー
    userbuf += howmany
    bytes_done += howmany
    if bufferdone が 0 でない then
      channel->endpoint->ephw->hw_sync_sgl_for_cpu で DMA する (DMA_FROM_DEVICE)
      // FPGA に bufferdone を通知
      iowrite32(1 | (channel->chan_num << 1) | (bufidx << 12),
                &channel->endpoint->registers[fpga_buf_ctrl_reg]);
    end
  end

  // ループの終了判定
  bytes_done が count 以上か eof に到達 => ループから抜ける
  exhausted でない => ループ頭に戻る
  bytes_done が 0 より大きい
   かつ
  no_time_left か (channel->wr_synchronous && channel->wr_allow_partial) => ループから抜ける
  no_time_left でない かつ O_NONBLOCK が指定されてた
   bytes_done が 0 より大きい => ループから抜ける
   ready が 0 でない => desperate へ
```

```
        それ以外 => bytes_done を -EAGAIN( エラー ) にしてループから抜ける

      !no_time_left または bytes_done > 0
       アライメント調整

      if !channel->wr_allow_partial || (no_time_left && (bytes_done == 0))
        do{
          if (wait_event_interruptible(channel->wr_wait, (!channel->wr_sleepy)))
            => interrupted へ
          if (mutex_lock_interruptible(&channel->wr_mutex))
            => interrupted へ
        }while(channel->wr_sleep);
        ループの先頭に戻る

        interrupted:
          bytes_done が 0 でなければ，return bytes_done
          あるいは エラー
      end

      if left_to_sleep > 0 then
         left_to_sleep = wait_event_interruptible_timeout(channel->wr_wait,
                                                          (!channel->wr_sleepy),
                                                          left_to_sleep);
         (!channel->wr_sleepy) => ループの先頭に戻る
         if left_to_sleep < 0 then
           bytes_done が 0 でなければ，return bytes_done
           あるいは エラー
         end
      end

      desparate:
        no_time_left = 1
    end
```

続き

xillybus_write, xillybus_flush, xillybus_release, xillybus_llseek, xillybus_poll は，また今度.