■ AXI/JavaRock

JavaRock で AXI にアクセスするコアを書きたいな，と思いながら
手がまわっていなかったのですが，ようやく着手.

とりあえず，ということで,

- 読み書き用それぞれのバッファを用意
- 読み書き制御用に，それぞれ start/busy を用意
- start の立ち上がりで処理を開始し，処理中は busy がアサートされる

という安直な方法で実装してみました.

こんなコード↓で AXI にアクセスできます.

```
import net.wasamon.javarock.rt.*;

@javarockhdl
public class AXITest{

  private final AXI_M axi = new AXI_M();

  private void sleep(int v){
    for(int i = 0; i < v; i++){ ; }
    return;
  }

  @auto
  public void run(){
    axi.write_addr = 0x20001000;
    axi.write_len = 128;
    axi.read_addr = 0x20000000;
    axi.read_len = 128;
    while(true){
      axi.read_start = false;
      axi.read_start = true;
      while(axi.read_busy == true){ ; }
      for(int i = 0; i < 128; i++){
  axi.out[i] = axi.in[i];
      }
      axi.write_start = false;
      axi.write_start = true;
      while(axi.write_busy == true){ ; }
      sleep(100);
    }
  }
}
```

ちなみに，AXI 空間の 0x20000000 は,
PCIe 越しでホスト PC のメインメモリの 512M 番地にマッピングしています.
つまり，上のコードは，512M 〜 512M+511 番地のデータ (128DWORD 分) を,
512M+0x1000 〜 512M+0x1000+511 番地にコピーするもの，です.

AXI な口は, EDK の AXI EXTERNAL MASTER CONTROLLER を使って引き出すことを想定して,

AXI トランザクションを行うプリミティブを VHDL で実装.

WREADY がずっと立ちっぱなしならブロック RAM からバーストでデータ読むんだけどな ...

ちなみに，AXI クロックとアプリクロックが違うことが考えられるので，
読み書きのクロックが独立したブロック RAM を使っています．

あわせて，これと JavaRock を接続するためのブリッジも記述．

今は 32bit 幅アクセスだけど，AXI 側は 128bit 幅アクセスくらいにしてもいいかも．

あと，本当は FIFO でデータの授受をしたいんだけど，
Java のコードとクロック単位でデータ送受信しなきゃいけない FIFO アクセスを
うまく組み合わせる方法を思いついていないんだよなあ．