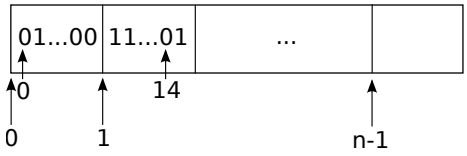
Informal Specification of Bitwalker

October 22, 2013

Introduction

We introduce some auxiliary concepts and formulate general assumptions:

- A *bit stream* is an array containing elements of type `uint8_t`.
- A bit stream can be indexed both by its array indices and its *bit indices*.
  As an example we represent a bit stream with array length `n`:



  The two bit indices, `0` and `14`, mark bit positions in the first and second array element, respectively.
- A bit stream is *valid* if the array is valid.
- A *bit sequence* is a consecutive sequence of bits within a bit stream. It is given by the position of its first bit within the bit stream and its *length*, that is, the number of bits it contains.
- We assume that the C-types `unsigned int` and `int` have a width of 32 bits.

The Function `Bitwalker_Peek`

The function `Bitwalker_Peek` reads a bit sequence from a bit stream and converts it to an integer.

The function signature reads as follows:

```
uint64_t    Bitwalker_Peek(unsigned int Startposition,
                           unsigned int Length,
                           uint8_t Bitstream[],
                           unsigned int BitstreamSizeInBytes);
```

Description

The function `Bitwalker_Peek` reads a bit sequence from a bit stream and converts it to a 64-bit unsigned integer.

- In this interpretation the left most bit of the sequence represents the most significant bit.
- The bit stream is given by the array `Bitstream` which has size `BitstreamSizeInBytes`.
- The bit sequence is defined by the bit index `Startposition` and `Length` which gives the number of bits in the sequence.

  If the bit sequence is not valid, that is, if it is not contained in the bit stream, then the function returns `0`. This increases the robustness of the function.

Preconditions

The following preconditions shall hold for the function arguments:

- `Bitstream` is a valid array of length `BitstreamSizeInBytes`
- `Length` $\leq$ `64` and
- `Startposition + Length` $\leq$ `UINT_MAX`.

Note that additional constraints are implicitly expressed by the use of *unsigned* integer types.

The Function `Bitwalker_Poke`

To be done.

Interaction of `Bitwalker_Peek` and `Bitwalker_Poke`

The functions `Bitwalker_Peek` and `Bitwalker_Poke` are inverse to each other.