Annex B
Include files

This annex shows the contents of the svc.h and svc_src.h include files.

B.1 Binary-level compatibility include file `svc.h`

```
/* canonical representation */

#define sv_0   0
#define sv_1   1
#define sv_z   2  /* representation of 4-st scalar z */
#define sv_x   3  /* representation of 4-st scalar x */

/* common type for 'bit' and 'logic' scalars. */
typedef unsigned char svScalar;

typedef svScalar svBit;    /* scalar */
typedef svScalar svLogic;  /* scalar */

/* Canonical representation of packed arrays */
/* 2-state and 4-state vectors, modelled upon PLI's avalue/bvalue */
#define SV_CANONICAL_SIZE(WIDTH) (((WIDTH)+31)>>5)

typedef unsigned int
        svBitVec32;/* (a chunk of) packed bit array */

typedef struct { unsigned int c; unsigned int d;} /* as in VCS */
        svLogicVec32; /* (a chunk of) packed logic array */

/* Since the contents of the unused bits is undetermined, the following macros
may be handy */
#define SV_MASK(N) (~(-1<<(N)))

#define SV_GET_UNSIGNED_BITS(VALUE,N)\
   ((N)==32?(VALUE):((VALUE)&SV_MASK(N)))

#define SV_GET_SIGNED_BITS(VALUE,N)\
   ((N)==32?(VALUE):\
    (((VALUE)&(1<<((N)1)))?((VALUE)|~SV_MASK(N)):((VALUE)&SV_MASK(N))))

/* implementation-dependent representation */
/* a handle to a generic object (actually, unsized array) */
typedef void* svHandle;

/* reference to a standalone packed array */
typedef void* svBitPackedArrRef;
typedef void* svLogicPackedArrRef;

/* total size in bytes of the simulator's representation of a packed array */
/* width in bits */
int svSizeOfLogicPackedArr(int width);
int svSizeOfBitPackedArr(int width);

/* Translation between the actual representation and the canonical
representation */
```



```
/* functions for translation between the representation actually used by
   simulator and the canonical representation */

/* s=source, d=destination, w=width */

/* actual <-- canonical  */
void svPutBitVec32   (svBitPackedArrRef   d, const svBitVec32*   s, int w);
void svPutLogicVec32 (svLogicPackedArrRef d, const svLogicVec32* s, int w);

/* canonical <-- actual  */
void svGetBitVec32   (svBitVec32*   d, const svBitPackedArrRef   s, int w);
void svGetLogicVec32 (svLogicVec32* d, const svLogicPackedArrRef s, int w);

/* Bit selects */

/* Packed arrays are assumed to be indexed n-1:0,
   where 0 is the index of least significant bit */

/* functions for bit select */

/* s=source, i=bit-index */
svBit svGetSelectBit(const svBitPackedArrRef s, int i);
svLogic svGetSelectLogic(const svLogicPackedArrRef s, int i);

/* d=destination, i=bit-index, s=scalar */
void svPutSelectBit(svBitPackedArrRef d, int i, svBit s);
void svPutSelectLogic(svLogicPackedArrRef d, int i, svLogic s);


/*
 * functions for part select
 *
 * a narrow (<=32 bits) part select is copied between
 * the implementation representation and a single chunk of
 * canonical representation
 * Normalized ranges and indexing [n-1:0] are used for both arrays:
 * the array in the implementation representation and the canonical array.
 *
 * s=source, d=destination, i=starting bit index, w=width
 * like for variable part selects; limitations: w <= 32
 */
/* canonical <-- actual */
void svGetPartSelectBit(svBitVec32* d, const svBitPackedArrRef s, int i,
                        int w);
void svGetPartSelectLogic(svLogicVec32* d, const svLogicPackedArrRef s, int i,
                          int w);

/* actual <-- canonical */
void svPutPartSelectBit(svBitPackedArrRef d, const svBitVec32 s, int i,
                        int w);
void svPutPartSelectLogic(svLogicPackedArrRef d, const svLogicVec32 s, int i,
                        int w);

/* Array querying functions */
/* These functions are modelled upon the SystemVerilog array querying
functions and use the same semantics*/
/* If the dimension is 0, then the query refers to the packed part (which is
one-dimensional) of an array, and dimensions > 0 refer to the unpacked part of
an array.*/
```

```
/* h= handle to open array, d=dimension */
int svLeft(const svHandle h, int d);
int svRight(const svHandle h, int d);
int svLow(const svHandle h, int d);
int svHigh(const svHandle h, int d);
int svIncrement(const svHandle h, int d);
int svLength(const svHandle h, int d);
int svDimensions(const svHandle h);

/* a pointer to the actual representation of the whole array of any type */
/* NULL if not in C layout */
void *svGetArrayPtr(const svHandle);

int svSizeOfArray(const svHandle); /* total size in bytes or 0 if not in C
                                      layout */

/* Return a pointer to an element of the array
   or NULL if index outside the range or null pointer */

void *svGetArrElemPtr(const svHandle, int indx1, ...);

    /* specialized versions for 1-, 2- and 3-dimensional arrays: */
void *svGetArrElemPtr1(const svHandle, int indx1);
void *svGetArrElemPtr2(const svHandle, int indx1, int indx2);
void *svGetArrElemPtr3(const svHandle, int indx1, int indx2, int indx3);

/* Functions for translation between simulator and canonical representations*/
/* These functions copy the whole packed array in either direction. The user is
responsible for allocating an array in the canonical representation. */
/* s=source, d=destination */
/* actual <-- canonical  */
void svPutBitArrElemVec32 (const svHandle d, const svBitVec32* s,
                           int indx1, ...);
void svPutBitArrElem1Vec32(const svHandle d, const svBitVec32* s, int indx1);
void svPutBitArrElem2Vec32(const svHandle d, const svBitVec32* s, int indx1,
                           int indx2);
void svPutBitArrElem3Vec32(const svHandle d, const svBitVec32* s,
                           int indx1, int indx2, int indx3);

void svPutLogicArrElemVec32 (const svHandle d, const svLogicVec32* s,
                             int indx1, ...);
void svPutLogicArrElem1Vec32( const svHandle d, const svLogicVec32* s,
                              int indx1);
void svPutLogicArrElem2Vec32(const svHandle d, const svLogicVec32* s,
                             int indx1, int indx2);
void svPutLogicArrElem3Vec32(const svHandle d, const svLogicVec32* s,
                             int indx1, int indx2, int indx3);

/* canonical <-- actual  */
void svGetBitArrElemVec32 (svBitVec32* d, const svHandle s, int indx1, ...);
void svGetBitArrElem1Vec32(svBitVec32* d, const svHandle s, int indx1);
void svGetBitArrElem2Vec32(svBitVec32* d, const svHandle s, int indx1,
                           int indx2);
void svGetBitArrElem3Vec32(svBitVec32* d, const svHandle s,
                           int indx1, int indx2, int indx3);

void svGetLogicArrElemVec32 (svLogicVec32* d, const svHandle s, int indx1,
                             ...);
```



```
void svGetLogicArrElem1Vec32(svLogicVec32* d, const svHandle s, int indx1);
void svGetLogicArrElem2Vec32(svLogicVec32* d, const svHandle s, int indx1,
                             int indx2);
void svGetLogicArrElem3Vec32(svLogicVec32* d, const svHandle s,
                             int indx1, int indx2, int indx3);

svBit   svGetBitArrElem (const svHandle s, int indx1, ...);
svBit   svGetBitArrElem1(const svHandle s, int indx1);
svBit   svGetBitArrElem2(const svHandle s, int indx1, int indx2);
svBit   svGetBitArrElem3(const svHandle s, int indx1, int indx2, int indx3);

svLogic svGetLogicArrElem (const svHandle s, int indx1, ...);
svLogic svGetLogicArrElem1(const svHandle s, int indx1);
svLogic svGetLogicArrElem2(const svHandle s, int indx1, int indx2);
svLogic svGetLogicArrElem3(const svHandle s, int indx1, int indx2, int indx3);

void svPutLogicArrElem (const svHandle d, svLogic value, int indx1, ...);
void svPutLogicArrElem1(const svHandle d, svLogic value, int indx1);
void svPutLogicArrElem2(const svHandle d, svLogic value, int indx1,
                        int indx2);
void svPutLogicArrElem3(const svHandle d, svLogic value, int indx1, int indx2,
                        int indx3);

void svPutBitArrElem (const svHandle d, svBit value, int indx1, ...);
void svPutBitArrElem1(const svHandle d, svBit value, int indx1);
void svPutBitArrElem2(const svHandle d, svBit value, int indx1, int indx2);
void svPutBitArrElem3(const svHandle d, svBit value, int indx1, int indx2,
                      int indx3);
```

B.2 Source-level compatibility include file `svc_src.h`

```
/* macros for declaring variables to represent the SystemVerilog */
/* packed arrays of type bit or logic */
/* WIDTH= number of bits,NAME = name of a declared field/variable */
#define SV_BIT_PACKED_ARRAY(WIDTH,NAME)/* actual definition will go here */
#define SV_LOGIC_PACKED_ARRAY(WIDTH,NAME)/* actual definition will go here */
```