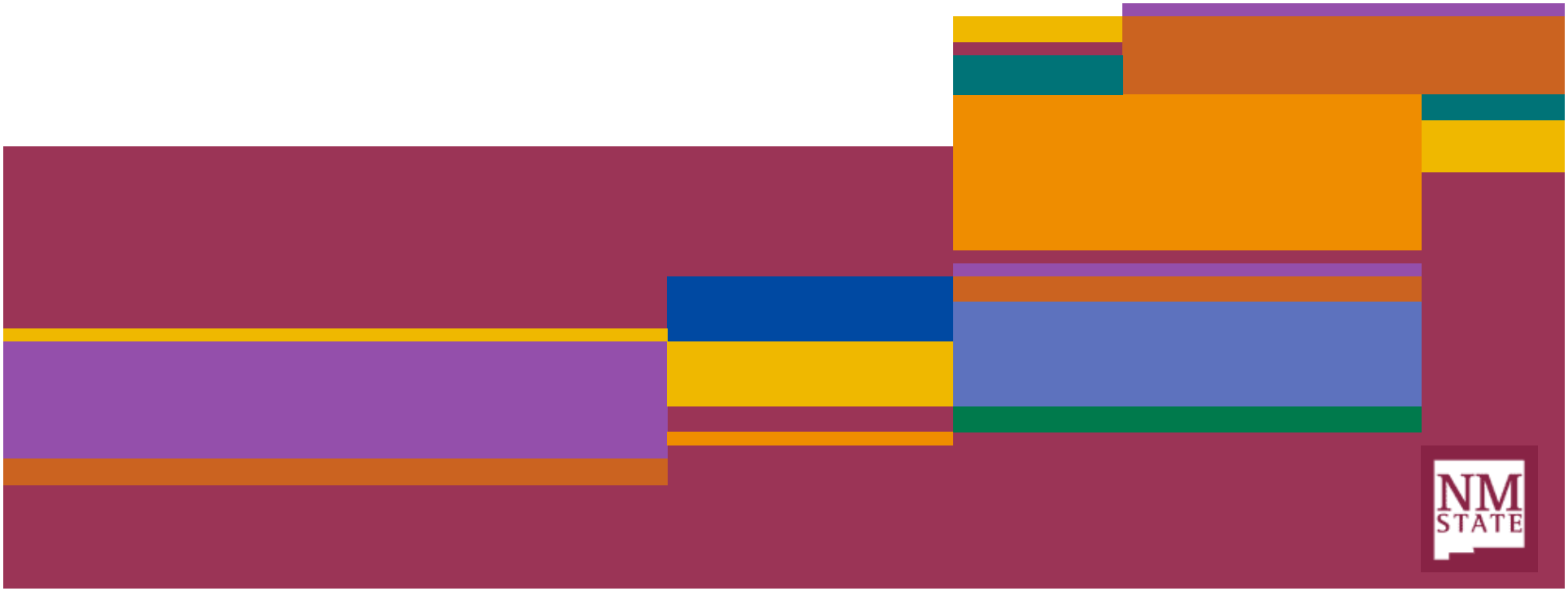
C Programming for Fixed-Point Digital Signal Processors



C Programming for Fixed-Point DSPs

- TMS320C6416 is a fixed-point DSP therefore all arithmetic is based on **short** (16 bit) variables
  - Multiplication of two 16 bit short numbers results in a 32 bit **int** value
  - Int value must ultimately be converted back to short

- Numerical mapping between fixed-point fraction and a short is simple
  - Multiply fraction by 2^{15} to get short
  - Divide short by 2^{15} (15 right shifts) to get a fixed-point fraction

Fraction	–1.0	...	0.0	...	$+1.0 - 2^{-15}$
Short	–32768	...	0	...	+32767

C Programming for Fixed-Point DSPs

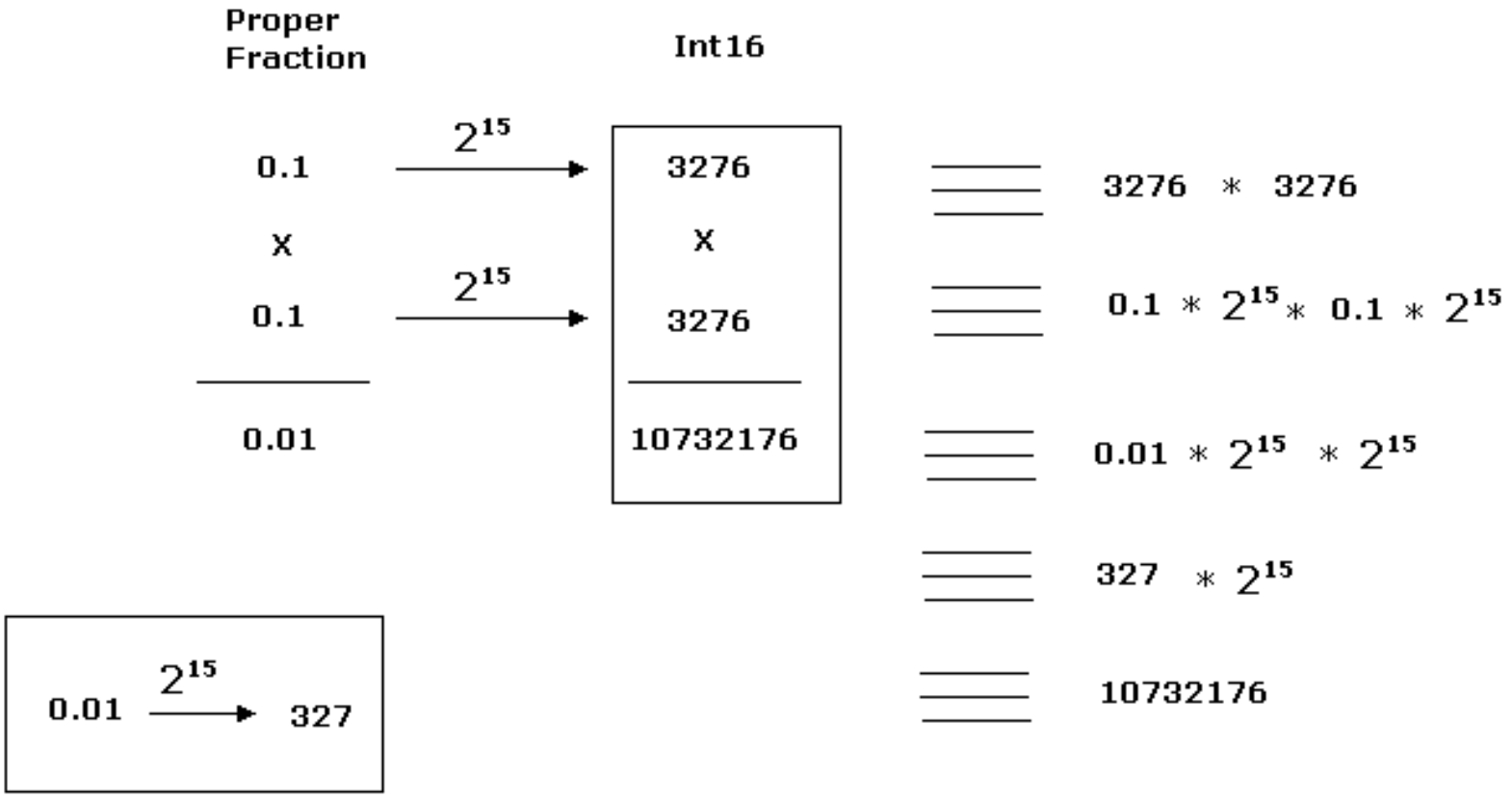
- Although it is possible to *typecast* a short to a float, perform all calculations in float, and typecast back to a short, there is significant overhead involved in performing floating-point computation on a fixed-point DSP

*All C codes for EE442/EE592 are **not** to use floats for any calculations*

Program 1

```
/* Addition of two shorts */

short a, b, c;
int c2;

int main (void)
{
   a = 30000;
   b = 10000;

   c = a + b;     /* c = -25536 = 40000 – 65536 */
   c2 = a + b;    /* c2 = 40000 */

   return 0;
}
```

Multiplication in Fixed-Point Hardware



must divide result of short x short by 2^{15} for correct answer

Program 2

```
/* Multiplication of two shorts */

short a, b, c;
int c2;

int main (void)
{
   a = 3277;   /* a = 0.1 */
   b = 3277;   /* b = 0.1 */
   c = a * b;  /* c = -9175 */

   c2 = a * b;     /* c2 = 10738729 */
   c2 = c2>>15;    /* c2 = 327 */
   c = (short)c2; /* c = 327 */
   return 0;
}
```

Program 2b

```
/* Demonstration of convergent rounding */
short a, b, c;
int       c2;

int main (void)
{
          a = 3277; /* a = 0.1 */
          b = 3277; /* b = 0.1 */

          c = a * b;          /* c = -9175 */
          c2 = a * b;         /* c2 = 10738729 */

          c = cround(c2)>>15; /* c = 328 */

          return 0;
}

int cround(int a)
{
          if (a & 0x3fff)               // If lower 14 bits are not all 0
                    a += 0x4000;        //   normal rounding
          else                          // else lower 15 bits are 0x4000
                    a += (a>>1) & 0x4000; //   convergent rounding: add half of lsb of upper ha
          return a & 0xffff8000;        // truncate lower 14 bits
}
```

Program 3

```
/* Division by short */

short a, c;
int c2;

int main (void)
{
   a = 2;
   c = 80000 / a;    /* c = -25536 = 40000 – 65536 */

   c2 = 80000 / a;    /* c2 = 40000 */

   return 0;
}
```